



НТО ЮНИОРЫ 2026/27 · СФЕРА

Игротех

Задачник-справочник для подготовки к
отборочному этапу

5–7 класс

Автор-составитель

Артём Осинowski, методист и разработчик детского технопарка «Кванториум»

Детский технопарк «Кванториум», Новосибирск · 2026

Игротех

Задачник-справочник для подготовки к отборочному этапу НТО Юниоры. 5–7 класс

Автор-составитель – **Артём Осиновский**, методист и разработчик детского технопарка «Кванториум».

Нашли ошибку или неточность? Напишите на osinovsky.aa@kvantorium54.ru (рабочая почта): сфера, версия книги (1.0) и номер страницы – исправим в следующей версии. Актуальная версия и список изменений – nto.kvantorium54.ru/knigi. Личная почта для других вопросов – artem081033@gmail.com.

Важно. Это не официальное пособие олимпиады, а книга методиста «Кванториума». Каких заданий ждать на отборе 2026/27, мы не знаем: организаторы их заранее не показывают, а задания разных лет не повторяются. Книга готовит к темам из программы сферы и приёмам решения, а не к конкретным задачам.

Книга помогает школьнику самостоятельно подготовиться к отборочному этапу сферы «Игротех» Национальной технологической олимпиады «НТО Юниоры»: теория с нуля, подробные разборы задач и задачи для самопроверки с ответами.

Большинство задач составлены автором по образовательной программе сферы, часть – по мотивам открытых заданий прошлых лет; источник указан у каждой задачи. Уровни сложности – шкала этой книги, а не баллы олимпиады. Правила, сроки и формат отбора – на сайте junior.ntcontest.ru.

Версия 1.0 от 02.10.2026, сборка 02.10.2026. © Артём Осиновский, 2026.

Содержание

Как устроена эта книга	4
Для кого эта книга	4
Как устроен отбор	4
Что пригодится на отборе и где это в книге	5
Как пользоваться книгой	6
Как вводить ответы на отборе	6
Что понадобится	7
План на месяц до отбора	17
Родителям и наставникам	17
Честно об этой книге	18
Глава 1 Игры и жанры: язык геймдизайнера	19
1.1. Что такое игра	20
1.2. Жанры: как узнать игру по описанию	22
1.3. Из чего собрана игра: механики и элементы	24
1.4. Квест – главный жанр сферы	26
1.5. Как записывать ответ. Словарь геймдизайнера	28
1.6. Разбираем задачи	30
Глава 2 Геймдизайн: механики, баланс, документ игры	42
2.1. Игровой цикл: действие – награда – новая цель	43
2.2. Баланс: чтобы не было скучно и обидно	44
2.3. Прогрессия: как растут цены и опыт	46
2.4. Сколько минут и ходов до цели	48
2.5. ГДД – паспорт игры	49
2.6. Плейтест и обратная связь	50
2.7. Разбираем задачи	51
Глава 3 Левел-дизайн: карты и пути	64
3.1. Карта в клетку и координаты	65
3.2. Исполнитель-робот: команды движения	66
3.3. Кратчайший путь по клеткам: волна	67
3.4. Карта как граф: комнаты, проходы, ключи	68
3.5. Сколько путей из угла в угол	70
3.6. Внимание игрока: ориентиры, свет, «хлебные крошки»	72
3.7. Разбираем задачи	73
Глава 4 Логика, выборы и случайность	87
4.1. Кто где: таблица с крестиками	87
4.2. Правда и ложь: рыцари и лжецы	89
4.3. И, ИЛИ, НЕ: условия в игре	90
4.4. Дерево выборов: сколько концовок	92
4.5. Шанс: доля, кубик и «1 из 5»	94
4.6. Монета не помнит: серия неудач	96
4.7. Разбираем задачи	97

Глава 5 Машины состояний: как думает NPC	108
5.1. Состояние и переход	109
5.2. Подпись на стрелке: событие [условие] / действие	110
5.3. Как думает NPC: стражник и торговец	112
5.4. Трассировка: куда приведут события	115
5.5. Тупики, лишние и недостижимые переходы	117
5.6. Вложенные состояния: одна стрелка вместо многих	118
5.7. Разбираем задачи	120
Глава 6 Переменные, условия и поиск ошибок	130
6.1. Переменная – коробка с именем	131
6.2. Условие: если – иначе	133
6.3. Порядок проверок	135
6.4. Трассировочная таблица: следим за переменными	136
6.5. Как найти ошибку в чужой программе	139
6.6. Электронные таблицы: формулы и ссылки	140
6.7. Функции и модули: делим программу на части	142
6.8. Разбираем задачи	144
Глава 7 Сюжет и диалоги: язык Ink	160
7.1. Ветвящийся сюжет: узлы, выборы, последствия	161
7.2. Граф сюжета: считаем прохождения и концовки	163
7.3. Первая история на Ink	165
7.4. Переменные: история запоминает выбор	168
7.5. Недостижимые концовки и другие ошибки	171
7.6. Хороший диалог	172
7.7. Разбираем задачи	174
Глава 8 3D и визуал: Blender	196
8.1. Из чего сделана 3D-модель: вершины, рёбра, грани	197
8.2. Примитивы: с чего начинается модель	200
8.3. Зеркало: модификатор Mirror	201
8.4. Массив: модификатор Array	203
8.5. Масштаб, единицы и экспорт в игру	205
8.6. Растровая картинка: пиксели и разрешение	206
8.7. Эскиз и силуэт персонажа	208
8.8. Разбираем задачи	209
Ответы и решения	218
Справочник	226
Формулы и правила	226
Мини-шпаргалки по инструментам	230
Словарик	237
Шпаргалка: всё «Запомни» из глав	240
Где что найти	246

Как устроена эта книга

Эта книга – тренажёр для отборочного этапа сферы «Игротех» олимпиады «НТО Юниоры». Здесь есть всё, чтобы готовиться самостоятельно: теория с нуля, подробные разборы задач и похожие задачи для самопроверки с ответами в конце книги. Мощный игровой компьютер не нужен: многие задачи решаются в тетради в клетку, а остальные – в браузере и в бесплатном Blender.

Для кого эта книга

- Для школьников **5–7 классов**, которые выбрали (или думают выбрать) сферу «Игротех».
- Для тех, кто любит играть, но ни разу не делал игру сам и не программировал. Каждая тема начинается с самого начала: что такое механика, чем состояние отличается от события, как устроена развилка в сюжете.
- Для **наставников и родителей**, которые хотят помочь. Для них есть отдельный раздел в конце введения.

Как устроен отбор

Таблица В.1

Когда	Что происходит
до 9 ноября	регистрация на олимпиаду (наш совет – успеть до 20 октября, тогда будут обе попытки)
20–30 октября	отбор, попытка 1 (3 часа)
31 октября – 10 ноября	отбор, попытка 2 (засчитывают лучшую из двух)
15 декабря	финал сферы «Игротех» – в один день во всех регионах

- Отбор проходит **дома, онлайн, с компьютера**. Задания открываются в личном кабинете на my.ntcontest.ru и решаются на платформе «Орбита» (orbita.education). Каждый решает сам за себя.
- Задания – это задачи с ответом, который проверяет компьютер: ввести число, выбрать верные варианты и записать их номера подряд, сопоставить, расставить шаги по порядку. В программе сферы среди проверочных заданий прямо названы тест на знание терминов и поиск ошибок в чужой программе – к обоим в книге много задач.
- Свой игровой проект для отбора не нужен. На вебинаре для наставников разработчики сферы сказали: на отборе и в финале участник работает с готовыми шаблонами. А ещё – что многое из того, что будет происходить на олимпиаде, связано с редактором игр «Урсула». Поэтому зайти в «Урсулу» заранее и пощёлкай по ней, пока есть время.
- За тест бесплатного курса своей сферы дают **+5 баллов**. Курс «Игротеха» – на платформе «Мастерская игр «Урсула»» (games.kruzhok.org), вход через «Талант». Бонус

нужно включить самому в личном кабинете («Мои привилегии») до конца отбора. Учти: после этого сферу уже не поменять. И бонус засчитают, только если за задания ты набрал больше нуля.

- У сферы есть проектные конкурсы «Идея и визуал» и «Геймдизайн и прототип». Для отбора они не обязательны, но это хорошая тренировка: придумать персонажа, собрать маленький прототип. Сроки и условия – на games.kruzhok.org, в разделе «Лаборатория».
- В октябре идёт фестиваль НТО Юниоры (junior.ntcontest.ru/fest). На вебинаре сферы обещали, что там будут пробные задания, – не пропусти.
- Финал – **командный** и очный. Команде выдают проект в редакторе игр: там уже есть сцена и объекты, а нужно добавить персонажей, запрограммировать их и добиться, чтобы игра в жанре «квест» заработала. Команду можно собрать заранее, а можно прямо на месте – организаторы финала распределяют участников сами.

Совет. Сроки и правила могут уточняться. Главный источник – сайт олимпиады junior.ntcontest.ru и памятка для Новосибирской области nto.kvantorium54.ru.

Что пригодится на отборе и где это в книге

На странице сферы перечислено, что пригодится на отборе. Вот где это в книге:

Таблица В.2

Что пригодится на отборе	Глава
Общие знания о том, какие бывают игровые механики, жанры, элементы игр; опыт игры в квесты	1. Игры и жанры: язык геймдизайнера
Продумывание механик, баланс ресурсов и сложности; вычисления по формулам, дроби, пропорции, проценты	2. Геймдизайн: механики, баланс, документ игры
Левел-дизайн: карты уровней и внимание игрока; эскизы и визуализация идей	3. Левел-дизайн: карты и пути
Базовые элементы логики и комбинаторики	4. Логика, выборы и случайность
Алгоритмы и исполнители; программирование поведения персонажей (ПРИМС из программы сферы)	5. Машины состояний: как думает NPC
Переменные и условия; отладка программ и поиск ошибок в логике; модульное программирование	6. Переменные, условия и поиск ошибок
Придумывать истории про персонажей; разветвлённый сюжет с выборами и последствиями, диалоги	7. Сюжет и диалоги: язык Ink
Опыт работы в 2D- и 3D-редакторах; Blender: модели, текстуры, экспорт в игру	8. 3D и визуал: Blender

Начни с глав 1 и 2: в них появляются слова, на которых говорит вся остальная книга. Главы 3 и 4 можно читать в любом порядке, для них хватит тетради в клетку и карандаша. Главу 5 читай раньше главы 6: в шестой мы добавляем к машинам состояний переменные и ищем в них ошибки. Главе 7 пригодится глава 4 – там разобраны развилки и выборы. Для главы 8 нужен компьютер с Blender, её можно начать в любой момент.

Как пользоваться книгой

В каждой главе – теория и разборы задач:

1. **Теория.** Коротко и по делу, с рисунками. Каждая новая тема начинается с большой картинкой «на пальцах», а самое важное собрано в рамках «Запомни». Все «Запомни» из всех глав повторяются в конце книги, в шпаргалке.
2. **Разбираем задачи.** Задачи такие же, как на отборе, и к каждой – решение по шагам: что дано, что делаем и почему, где обычно ошибаются, какой ответ получился и похож ли он на правду. Сразу после разбора – две похожие задачи **«Реши сам»**: сначала посмотри, как решают, потом реши сам похожую.
3. **Ответы** к «Реши сам» – в конце книги, в разделе «Ответы и решения». Там только ответ, а если в задаче есть подвох – ещё короткая подсказка.

Хочешь пробежаться по всей программе быстро – есть отдельная короткая книжка **«Быстрый старт»** (около 60 страниц): из каждой главы главное, один разбор и несколько задач «Реши сам».

Обычная задача – без пометки. У трудных задач рядом с номером стоит звёздочка (*): в них нужно несколько идей сразу, они труднее, чем на отборе. Лёгкие задачи, в одно действие, помечены словом «разминка».

Это уровни нашей книги, а не баллы олимпиады. Сколько задач будет на отборе и сколько баллов стоит каждая, заранее не объявляют.

Совет. Разбор читай с листком бумаги. Закрой решение, попробуй решить сам, а потом сверяйся по шагам. Так ты запомнишь в три раза больше, чем если просто прочитаешь.

У каждой задачи внизу написано, откуда она взялась. «По мотивам задачи» – значит, похожая задача была на олимпиаде прошлых лет: у школьников постарше, на профиле НТО «Разработка компьютерных игр» для 8–11 классов, или в прежней игровой сфере НТО Junior. Мы пересказали её своими словами, упростили и поменяли числа. «Своя задача по программе сферы» – значит, мы придумали её по теме из программы 2026/27 года.

Как вводить ответы на отборе

Так было на отборочных этапах прошлых лет. На «Орбите» перед каждой задачей читай инструкцию: там сказано, в каком виде нужен ответ.

- **Только число**, без единиц измерения и без знака процента, если не сказано иначе.

- **Номера вариантов – подряд, без пробелов и запятых.** Если в трёх вопросах ты выбрал варианты 4, 1 и 6, ответ – 416. Если в одном вопросе верных вариантов два, например 2 и 4, их обычно пишут по возрастанию: 24.
- **Дробную часть** на сайте обычно отделяют **точкой**: 0.25, а не 0,25. В книге мы пишем по-русски, через запятую.
- **Доля или проценты?** Шанс 25 % можно записать как 25 и как 0.25. Смотри, что просит условие.
- **Удары, ходы и копии – целые.** Если врагу нужно 3,3 удара, ударов будет 4: округляем вверх. Промежуточные результаты не округляй – только ответ.
- **У каждого могут быть свои числа.** В некоторых задачах условие генерируется для каждого участника отдельно, так что списать ответ не выйдет.
- **Попытки ограничены.** На задачу обычно даётся от 3 до 5 попыток, и каждая неверная снимает 30 % балла. Сначала проверь ответ в тетради или в редакторе, и только потом вводи.

Что понадобится

- Тетрадь в клетку, карандаш, ластик – для карт уровней, схем состояний и деревьев выборов. Ластик нужен всерьёз: схемы перерисовывают.
- Калькулятор. Хватит обычного на телефоне: проценты, дроби, умножение.
- Компьютер с интернетом. Планшета мало: Blender ставится на компьютер (Windows 10 или 11, или Linux; в программе сферы рекомендуют от 8 ГБ оперативной памяти).
- **Браузер** (например, Яндекс Браузер или Google Chrome) – для «Мастерской игр «Урсула»» (games.kruzhok.org). Там же курсы сферы, редактор игр и твои проекты.
- **Редактор диаграмм «Кибериада Graph Editor»** – в нём рисуют поведение персонажей на языке ПРИМС. Как его открыть, рассказано в курсах «Урсулы».
- **Blender** (blender.org) – бесплатная программа для 3D-моделирования, подойдёт версия 3.0 или новее.
- **Inky** – бесплатный редактор для языка Ink от его авторов (inklestudios.com/ink). Пишешь историю слева – и сразу играешь в неё справа.

Inky: редактор историй на Ink

Inky – бесплатный редактор языка Ink от его авторов, студии inkle. Слева пишешь историю, справа сразу в неё играешь. Интерфейс на английском, но слов в нём немного – ниже все нужные. Как устроен сам язык, рассказано в главе 7; здесь – как работать в программе и три задачи, чтобы руки привыкли.

Шаг 1. Скачать. Открой страницу github.com/inkle/inky/releases (ссылка есть и на inklestudios.com/ink) и в последней версии скачай `Inky_windows_64.zip` (около 225 МБ; для Mac – `Inky.dmg`). Устанавливать ничего не нужно: распакуй архив в любую папку и запусти `Inky.exe`. Если Windows спросит, можно ли запускать программу из интернета, – нажми «Подробнее» → «Выполнить в любом случае» вместе со взрослым.

Шаг 2. Окно Inky. При первом запуске в нём лежит маленький пример. Сотри его и пиши своё.

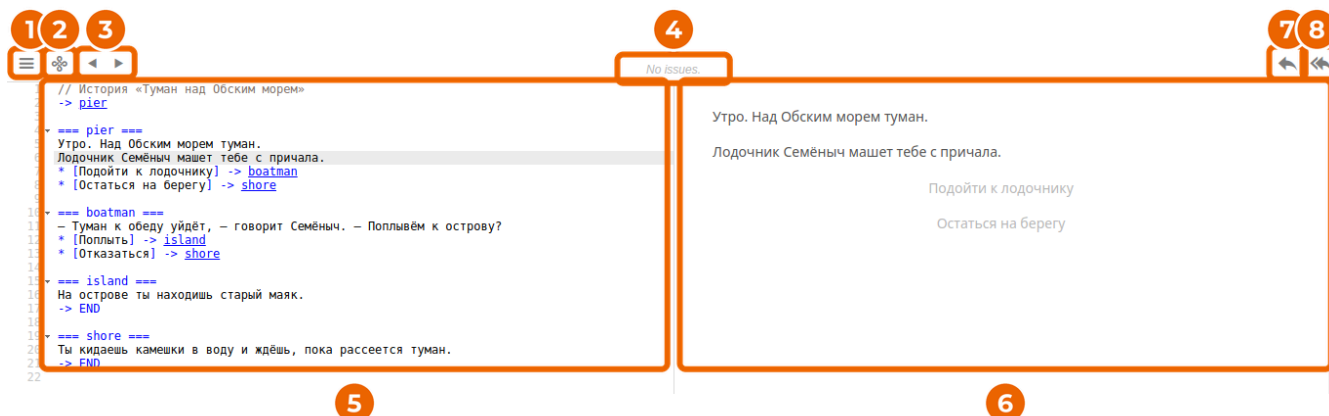


Рисунок В.1 – Окно Inky

1. **Toggle file browser** – список файлов проекта (нужен, когда история из нескольких файлов).
2. **Toggle knot browser** – список узлов истории: щёлкнул по имени – попал в узел.
3. **Navigate back / forward** – назад и вперёд по местам в коде, где ты был.
4. **Строка ошибок.** «No issues» – ошибок нет. Если есть – тут появится число и красный кружок.
5. **Редактор** – здесь пишут историю. Номера строк слева помогают найти ошибку.
6. **Игра** – история, как её увидит игрок. Обновляется сама, пока ты печатаешь.
7. **Rewind a single choice** – отменить последний выбор.
8. **Restart story** – начать историю сначала.

Шаг 3. Сохранить. «File → Save Project» (Ctrl + S). История сохранится файлом `.ink`. Открыть снова – «File → Open...». «File → Export for web...» соберёт из истории папку с веб-страницей: её можно открыть в браузере и дать поиграть друзьям.

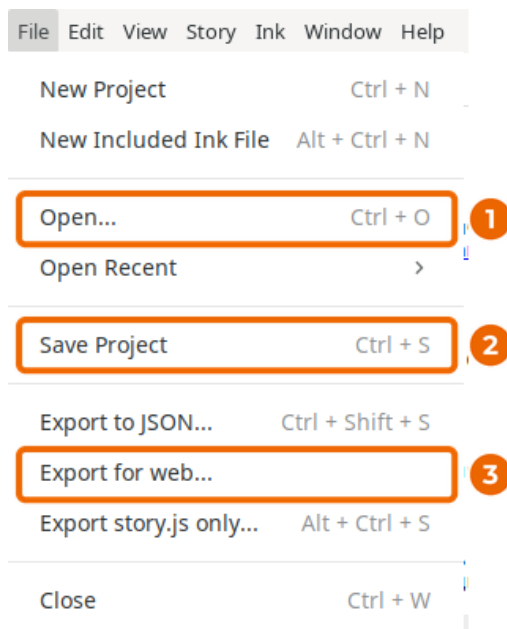


Рисунок В.2 – Меню File: открыть (1), сохранить (2), выгрузить в веб-страницу (3)

Задача П.1. Сыграй все пути. Набери в редакторе историю «Туман над Обским морем» из раздела 7.3. Справа нажми «Подойти к лодочнику» (1), потом «Поплыть». Начни заново кнопкой Restart story (8 на карте окна) и пройди остальные пути.

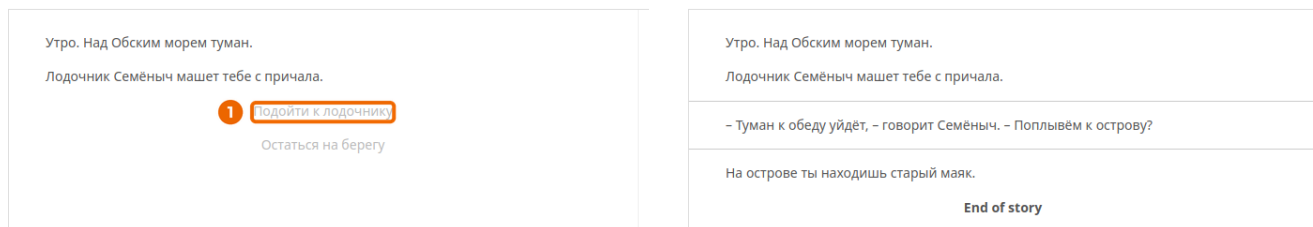


Рисунок В.3 – Слева – первый выбор, справа – концовка «остров»

Что должно получиться: три разных прохождения, две концовки («остров» и «берег»), в конце каждого Inky пишет «End of story». Сравни со схемой истории в разделе 7.3.

Задача П.2. Найди ошибку. В строке * [Отказаться] → shore сотри последнюю букву, чтобы получилось → shor . Посмотри, что делает Inky.

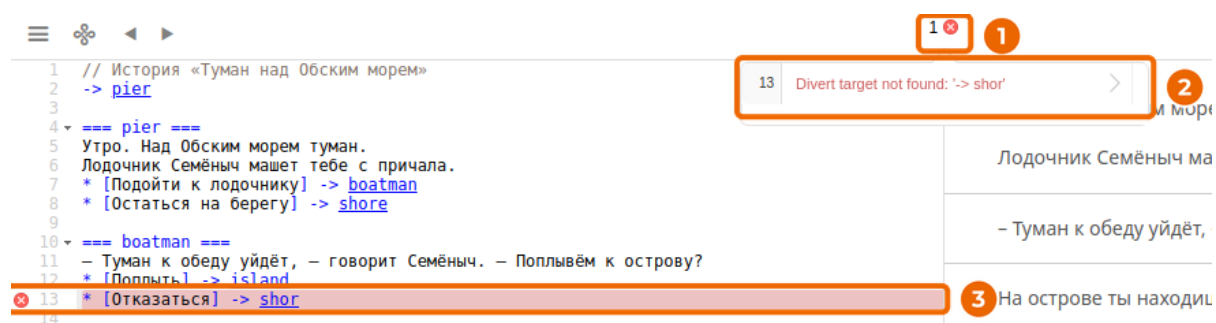


Рисунок В.4 – Ошибка: число ошибок (1), сообщение (2), строка с ошибкой (3)

Что должно получиться: вверху вместо «No issues» – «1» и красный кружок (1). Щёлкни по нему – появится список: номер строки 13 и сообщение «Divert target not found: '-> shor'» (2) – «узел для перехода не найден». Сама строка подсвечена красным (3). Верни букву – ошибка пропадёт. Так Inky ловит главную ошибку новичков: опечатку в имени узла.

Открой список узлов кнопкой Toggle knot browser (1). Там видно все четыре узла (2) – удобно проверять, что каждый узел, куда ведут стрелки, действительно есть.

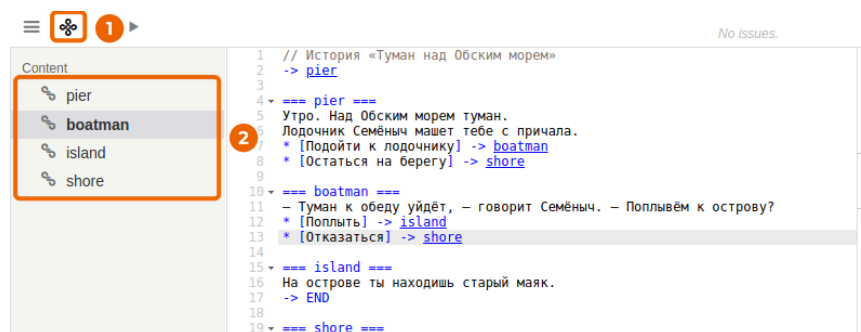


Рисунок В.5 – Список узлов истории

Задача П.3. История запоминает выбор. Сделай, чтобы Семёныч помнил, говорил ли ты с ним. Добавь в начало VAR talked = false , первой строкой узла boatman – ~ talked = true , а в узел shore перед → END – строку {talked: Семёныч машет тебе с лодки на прощание. |Лодка так и стоит у причала.} .

```

1  VAR talked = false
2  -> pier
3
4  ▾ === pier ===
5  Утро. Над Обским морем туман.
6  Лодочник Семёныч машет тебе с причала.
7  * [Подойти к лодочнику] -> boatman
8  * [Остаться на берегу] -> shore
9
10 ▾ === boatman ===
11 ~ talked = true
12 - Туман к обеду уйдёт, – говорит Семёныч. – Поплывём к острову?
13 * [Поплыть] -> island
14 * [Отказаться] -> shore
15
16 ▾ === island ===
17 На острове ты находишь старый маяк.
18 -> END
19
20 ▾ === shore ===
21 Ты кидаешь камешки в воду и ждёшь, пока рассеется туман.
22 {talked: Семёныч машет тебе с лодки на прощание. | Лодка так и стоит у причала.}
23 -> END
24 |

```

Рисунок В.6 – История с переменной talked

Пройди два пути, которые ведут на берег: сразу «Остаться на берегу» и через лодочника – «Отказаться».

Утро. Над Обским морем туман. Лодочник Семёныч машет тебе с причала. Ты кидаешь камешки в воду и ждёшь, пока рассеется туман. Лодка так и стоит у причала. End of story	Утро. Над Обским морем туман. Лодочник Семёныч машет тебе с причала. – Туман к обеду уйдёт, – говорит Семёныч. – Поплывём к острову? Ты кидаешь камешки в воду и ждёшь, пока рассеется туман. Семёныч машет тебе с лодки на прощание. End of story
---	---

Рисунок В.7 – Слева – сразу на берег, справа – через лодочника: концовки разные

Что должно получиться: в первом пути – «Лодка так и стоит у причала.», во втором – «Семёныч машет тебе с лодки на прощание.». Узел один и тот же, а текст разный – его решила переменная. Подробнее о переменных и условиях – раздел 7.4.

Blender: установка и первые шаги

Blender – бесплатная программа для 3D-моделей, в ней делают модели для игр «Урсулы». Ниже – как её поставить, где что в окне и четыре задачи, которые повторяют главу 8 руками: масштаб, массив, зеркало и экспорт в игру. Кадры сняты в Blender 5.2 LTS (октябрь 2026); в других версиях кнопки те же, могут немного съехать.

Шаг 1. Скачать и поставить. Открой blender.org/download и скачай версию для Windows: установщик `blender-...-windows-x64.msi` (ставится как обычная программа) или архив `.zip` (распаковал – и запускай `blender.exe`, установка не нужна; удобно в школьном классе). Видеокарта должна поддерживать OpenGL 4.3 – на очень старых компьютерах Blender может не запускаться.

Шаг 2. Первый запуск. Blender спросит настройки. Язык (1) оставь английским: в книге и на вебинарах сферы команды называют по-английски – Scale, Array, Mirror. Нажми «Continue» (2), а в следующем окне – «General»: откроется сцена с кубом, камерой и лампой.

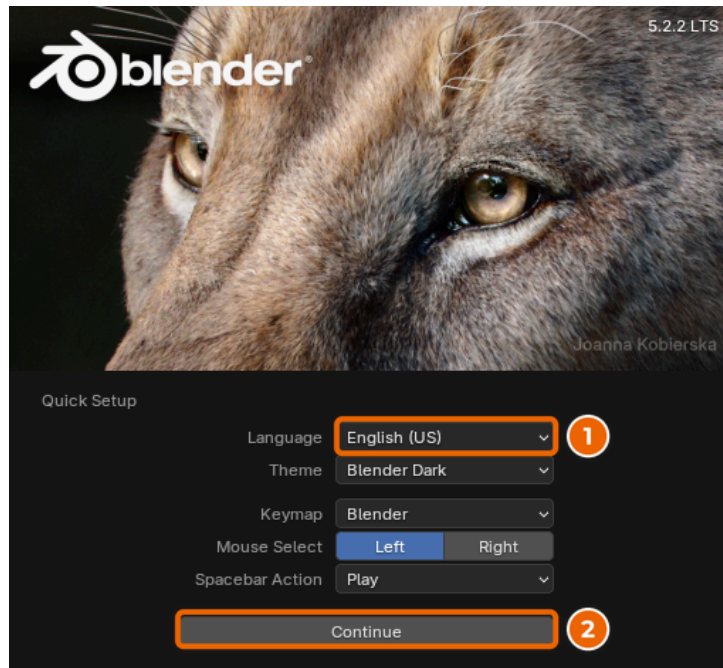


Рисунок В.8 – Первый запуск: язык (1) и «Continue» (2)

Шаг 3. Окно Blender.

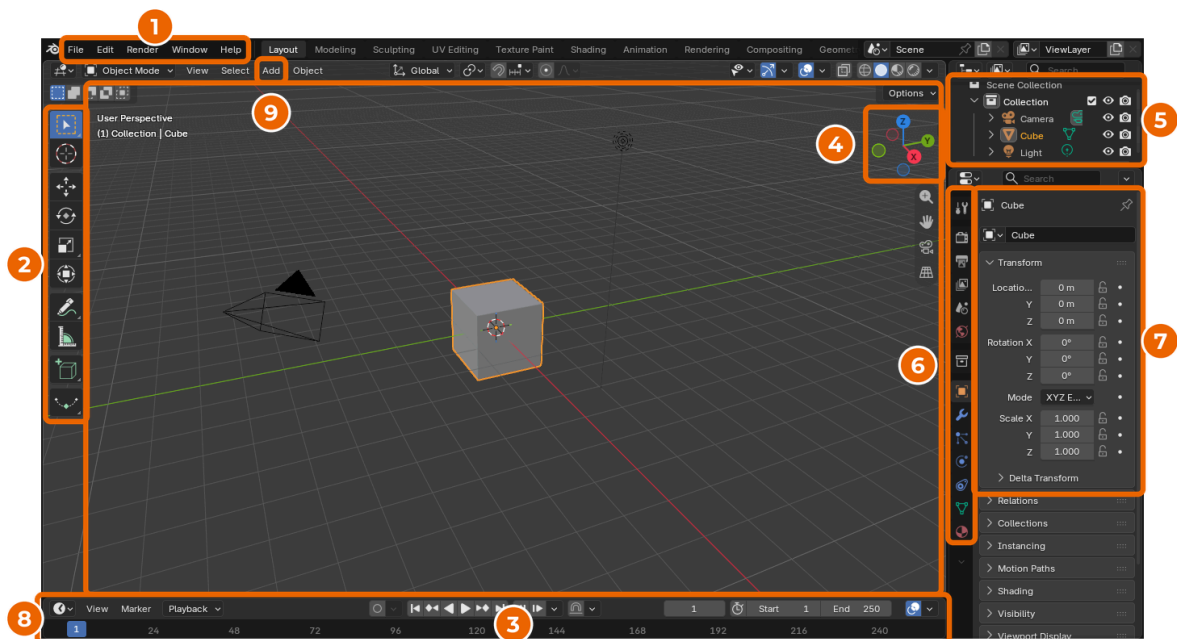


Рисунок В.9 – Окно Blender

1. **Меню** – File (сохранить, экспорт), Edit (отменить – Ctrl + Z) и другие.
2. **Инструменты** – выделить, двигать, вращать, масштабировать.
3. **3D-вид** – сцена. Колесо мыши – приблизить и отдалить; зажатое колесо – повернуть вид; Shift + зажатое колесо – сдвинуть вид; клавиша Home – показать всю сцену.
4. **Оси** – X красная, Y зелёная, Z синяя (вверх). Щёлкни по кружку оси – посмотришь на сцену с этой стороны.
5. **Список объектов сцены** (Outliner): Camera, Cube, Light. Щёлкнул по имени – выделил объект.

6. **Вкладки свойств.** Гаечный ключ – модификаторы, оранжевый квадрат – свойства объекта.
7. **Свойства** выделенного объекта на выбранной вкладке.
8. **Шкала времени** – для анимации, в задачах книги не нужна.
9. **Add** – добавить объект (то же, что Shift + A в 3D-виде).

Клавиши работают там, где стоит мышь: G, R, S, Tab и N жми, когда курсор над 3D-видом.

Задача П.1. Ящик высотой 0,5 м. Выдели куб, наведи мышь на 3D-вид и нажми N – справа откроется панель с числами: Scale (1) – во сколько раз растянут объект, Dimensions (2) – его настоящие размеры. У куба по умолчанию 2 м.

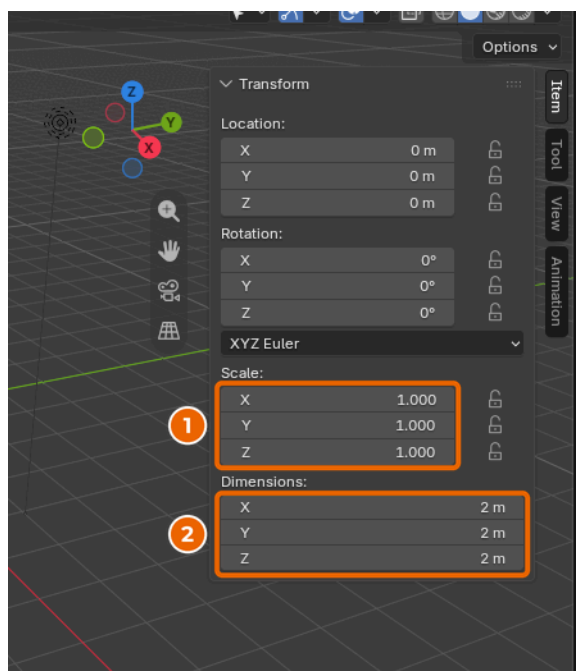


Рисунок В.10 – Панель N: масштаб (1) и размеры (2)

Дважды щёлкни по Scale Z (1), впиши 0.25 и нажми Enter. Посмотри на Dimensions Z (2).

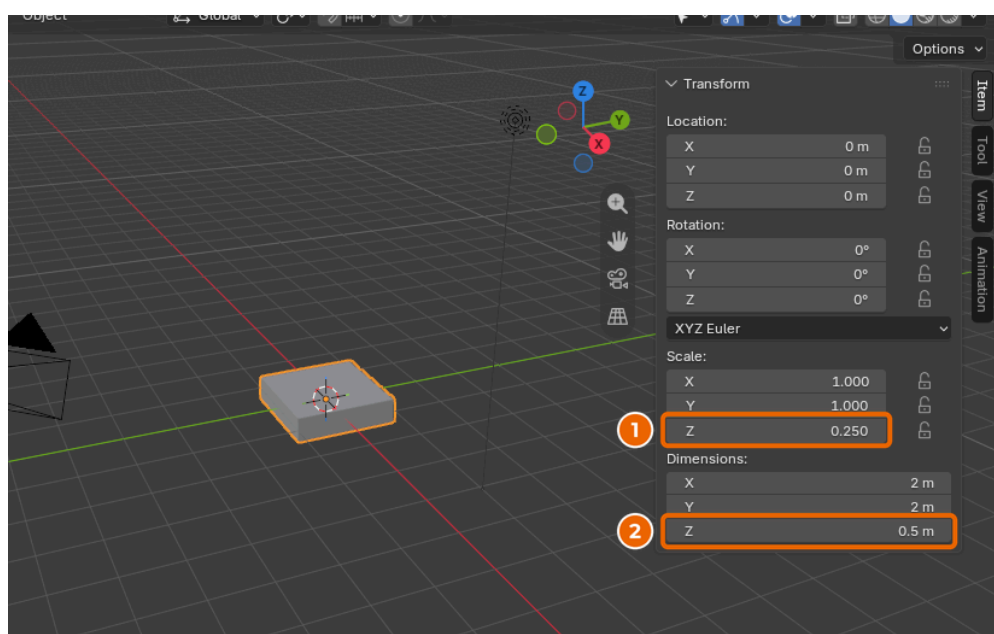


Рисунок В.11 – Scale Z = 0,25 – высота 0,5 м

Теперь примени масштаб: мышь над 3D-видом, Ctrl + A, в меню Apply – Scale (1).

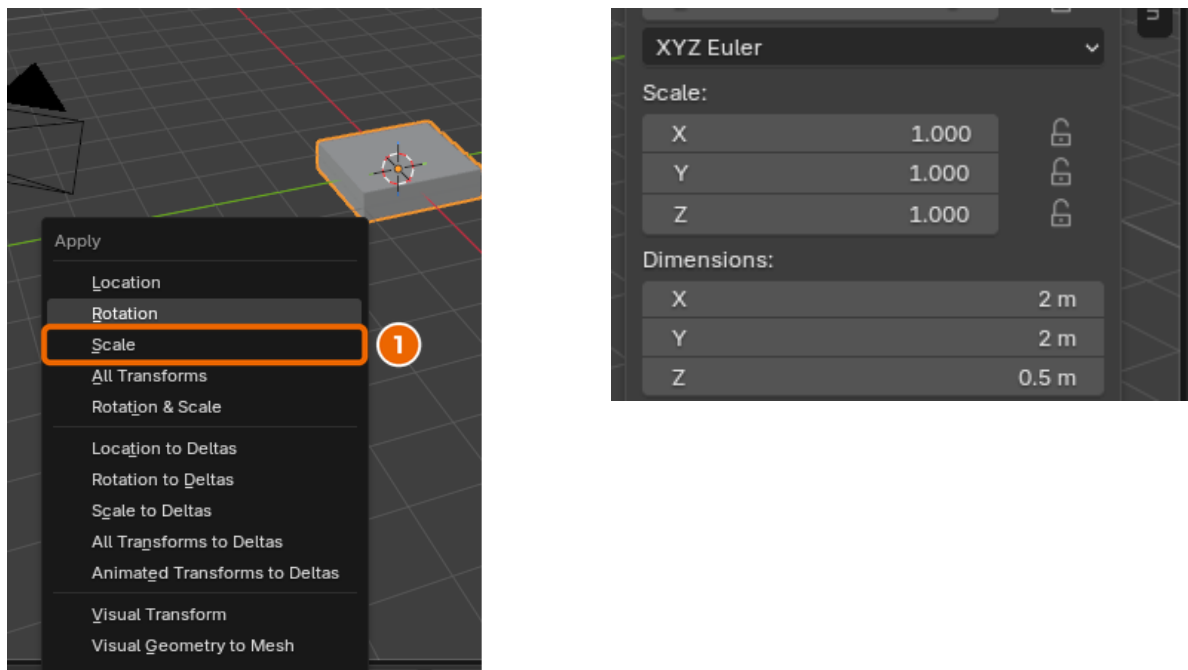


Рисунок В.12 – Слева – меню Apply, справа – числа после применения масштаба

Что должно получиться: после ввода 0.25 размер по Z – 0,5 м (сетка куба 2 м · 0,25). После Ctrl + A → Scale масштаб снова 1.000 по всем осям, а Dimensions Z остаётся 0,5 м. Так и готовят модель для игры (раздел 8.5).

Задача П.2. Забор из одной доски. Открой вкладку модификаторов (гаечный ключ, 6 на карте окна) и нажми «Add Modifier» (1).

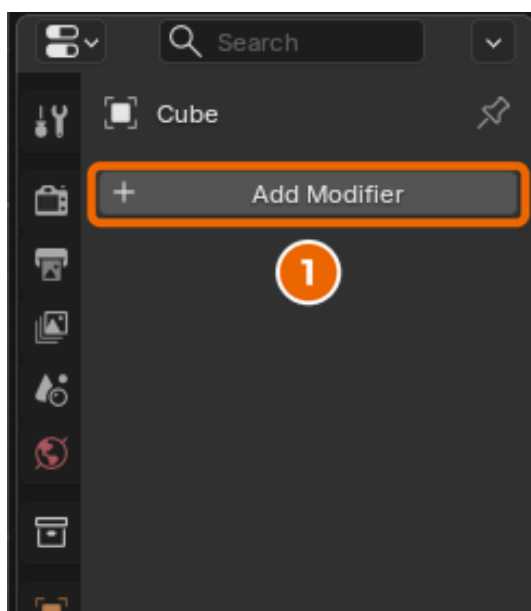


Рисунок В.13 – Вкладка модификаторов

В меню наведи на «Generate» (1) и выбери «Array» (2). В Blender 5 есть и «Array (Legacy)» – старый массив; у нового те же главные настройки.

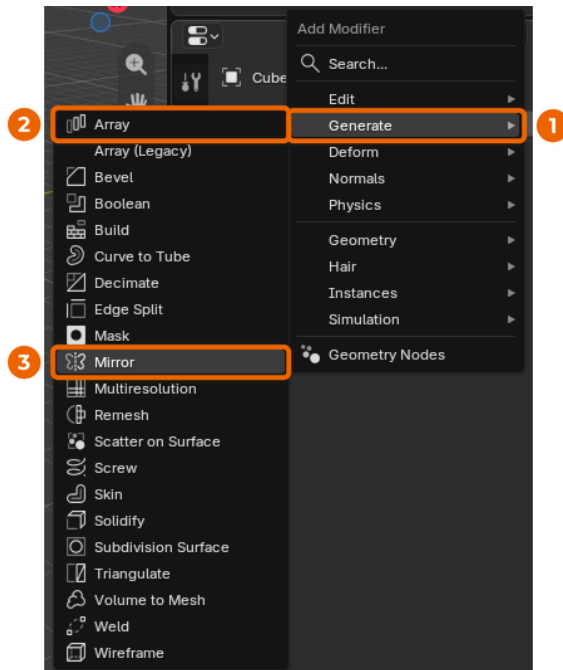


Рисунок В.14 – Меню Add Modifier → Generate: Array (2) и Mirror (3)

Поставь Count (1) – 6, Offset X (2) – 1.5. Чтобы посчитать вершины, включи статистику: стрелка рядом со значком наложений (Overlays) над 3D-видом (1), галочка Statistics (2).

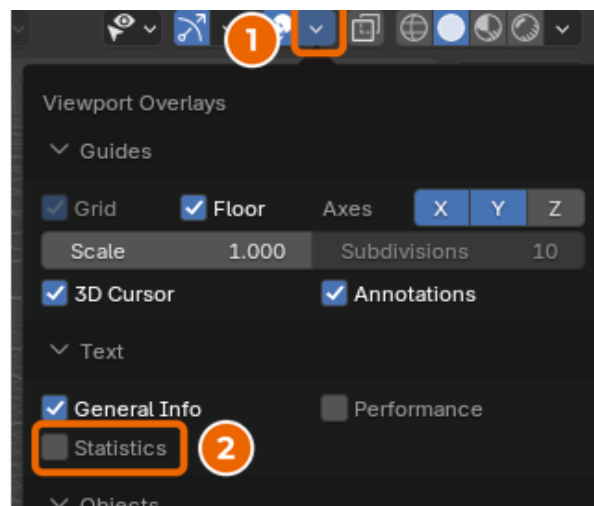


Рисунок В.15 – Включаем статистику

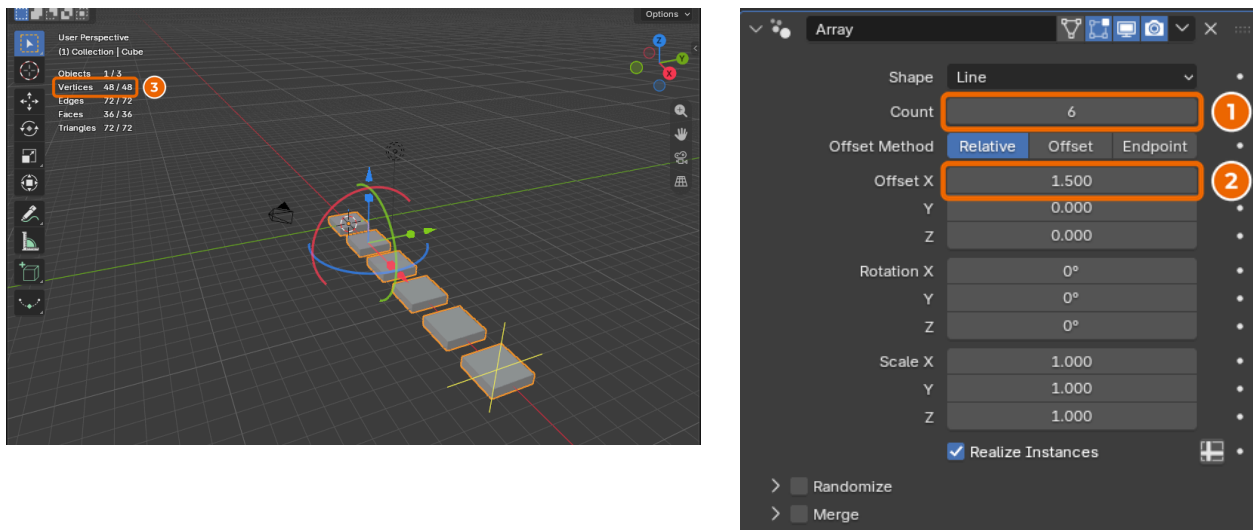


Рисунок В.16 – Массив из 6 досок: слева – 3D-вид со статистикой (3), справа – Count (1) и Offset X (2)

Что должно получиться: шесть досок в ряд с промежутками (сдвиг 1,5 ширины доски). В статистике Objects 1/3 – это всё ещё один объект, а Vertices 48 – у бруска 8 вершин, и Array умножил их на Count: $8 \cdot 6 = 48$ (раздел 8.4). Поставь Offset X = 1 – доски сомкнутся в одну длинную.

Задача П.3. Зеркало отражает от центра объекта. Удали Array (крестик в его заголовке). Наведи мышь на 3D-вид, нажми Tab (режим редактирования), A (выделить всё), потом по очереди G, X, 3 и Enter – сетка сдвинется на 3 м вправо, а центр объекта (оранжевая точка) останется на месте. Tab – обратно. Добавь модификатор Mirror (3 на рисунке с меню).

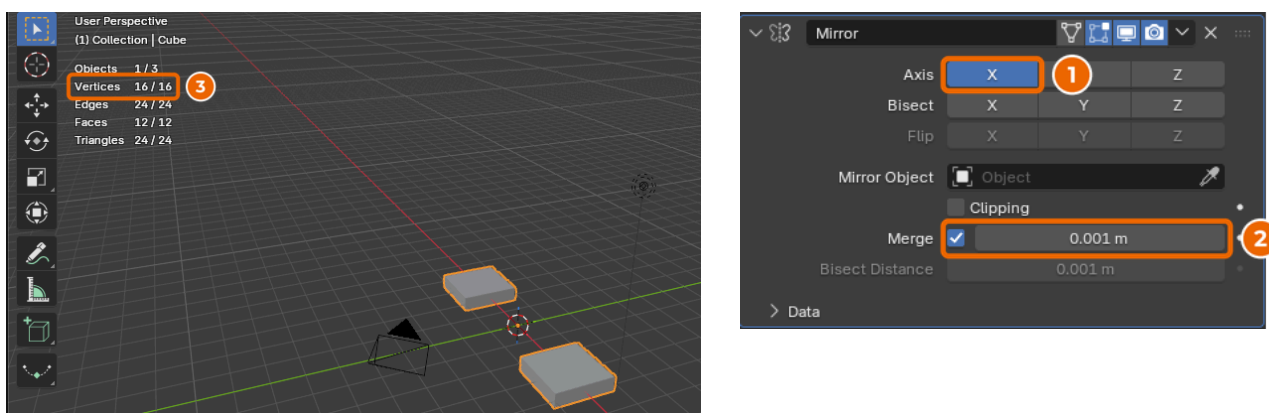


Рисунок В.17 – Mirror: копия появилась по другую сторону центра объекта; ось X (1), Merge (2), вершины (3)

Что должно получиться: копия бруска встала на 3 м **влево** от центра объекта – зеркало стоит в центре объекта, а не посередине бруска. Ось – X (1), галочка Merge (2) включена сама. Вершин 16 (3): по формуле $2V - k$ из раздела 8.3 $2 \cdot 8 - 0 = 16$ – на зеркале нет ни одной общей вершины, склеивать нечего.

Задача П.4. Экспорт в игру. File → Export (1) → glTF 2.0 (.glb/.gltf) (2).

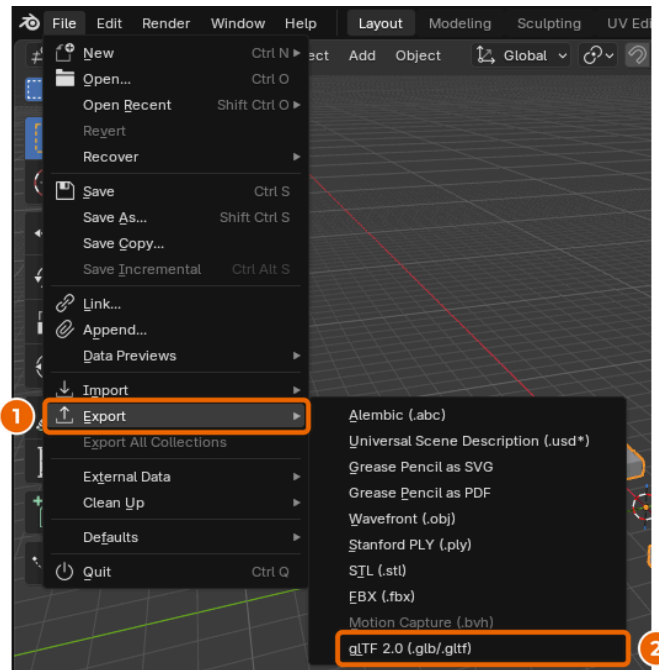


Рисунок В.18 – Меню экспорта

Справа в окне экспорта проверь: формат – glTF Binary (.glb) (1); в разделе Include – галочка Selected Objects (2), чтобы в файл не попали камера и лампа; в разделе Data → Mesh – галочка **Apply Modifiers** (3). Она по умолчанию **выключена**, без неё в файл уйдёт брусок без зеркальной копии. Нажми «Export glTF 2.0» (4).

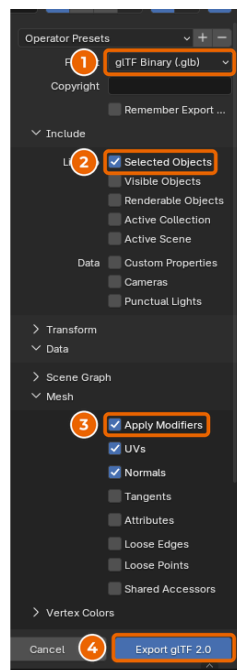


Рисунок В.19 – Настройки экспорта

Что должно получиться: файл `.glb` (несколько килобайт) с одним объектом. Мы проверили: с галочкой Apply Modifiers в файле обе половины – 24 треугольника, без неё – только 12. Именно такой файл загружают в редактор игры.

План на месяц до отбора

Если начинаешь в конце сентября, вот спокойный план: примерно по часу в день.

Таблица В.3

Неделя	Что читаем	Что делаем
1	главы 1–2	каждый день решать «Реши сам» по свежим разборам; войти на games.kruzhok.org и начать курс сферы
2	главы 3–4	нарисовать в клетку карту своего уровня; собрать похожую локацию в «Урсуле»
3	главы 5–6	повторить в «Кибериаде» машины состояний из разборов, загрузить в «Урсулу» и проверить в игре
4	главы 7–8 и шпаргалка	написать в Inku историю с двумя концовками; сделать в Blender простую модель по главе 8; заново решить задачи, где были ошибки; пройти тест курса сферы (+5 баллов)

Отбор длится до 10 ноября, и попыток две. Если первая попытка прошла неудачно, вернись к главам, где потерял баллы, и иди на вторую.

Родителям и наставникам

- **Помогайте разбираться, но не решайте за ребёнка.** На командном финале сразу видно, кто разобрался сам, а за кого решили. Лучше пусть ребёнок ошибётся на отборе и научится, чем попадёт в финал и растеряется там.
- **Играйте вместе – это тоже подготовка.** В списке того, что пригодится на отборе, первым стоит опыт игры в квесты. Пройдите вдвоём короткую игру-головоломку и спросите: где ты застрял? Как игра подсказала, что делать? Почему было интересно? Смотрите на возрастную маркировку. В книге мы старались упоминать только игры, которые подходят детям 11–14 лет.
- **Садитесь рядом на первых шагах в «Урсуле».** Детей чаще всего останавливает не сложность, а «проект не сохранился», «объект не программируется» или «непонятно, куда нажать». Разработчики сферы честно говорят, что освоиться в редакторе поначалу непросто.
- **Всё про регистрацию, бонусы и финал** для Новосибирской области – на nto.kvantorium54.ru (вкладки «Я наставник» и «Я участник или родитель»). Для наставников у сферы есть свой раздел с вебинарами и материалами для кружка: games.kruzhok.org/laboratory/mentor (вход через «Талант»).

- **Региональный координатор НТО Юниоры в Новосибирской области** – детский технопарк «Кванториум», info@kvantorium54.ru.

Честно об этой книге

Главное, что нужно знать про эту книгу: заданий отбора 2026/27 мы не видели – организаторы их заранее не показывают. Угадывать их бесполезно: наставники, которые следят за олимпиадой не первый год, говорят, что задания из года в год не повторяются. Поэтому книга учит темам из программы сферы и приёмам решения, а не готовит к конкретным задачам. Когда организаторы выложат пробные задания, мы сверим с ними книгу и выпустим новую версию.

Это не официальное издание НТО. Книгу составил методист «Кванториума» по открытым материалам олимпиады: программе сферы «Основы геймдизайна и разработки игр» 2026/27, записям вебинаров сферы для наставников (сентябрь 2026 года) и сборнику задач профиля «Разработка компьютерных игр» НТО для 8–11 классов (сезон 2025/26).

Задач отбора «Игротеха» 2026/27 года заранее никто не видел: их откроют только 20 октября. Разобранных задач прошлых лет по нынешним программам сферы («Урсула», ПРИМС, Ink) в открытом доступе мы тоже не нашли. Сборник 2022/23 года для прежней сферы «Технологии и компьютерные игры» был про Roblox и язык Lua – из него мы взяли только то, как выглядят сами задания. Задачи старшего трека сложнее, чем нужно тебе: мы брали из них идею, упрощали и пересчитывали. Поэтому эта книга учит понимать темы программы сферы, а не угадывать задания отбора.

«Урсула» и редактор диаграмм обновляются. Поэтому кнопки и окна мы описываем в общих чертах, а смысл – подробно: смысл от версии к версии не меняется. Точную инструкцию к нынешней версии смотри в курсах на games.kruzhok.org.

Каждое число в ответах мы пересчитали на компьютере, а каждую историю на Ink запустили. Если ты всё-таки нашёл ошибку или что-то осталось непонятным, напиши автору: osinovsky.aa@kvantorium54.ru. Следующие версии книги станут лучше благодаря тебе.

Игры и жанры: язык геймдизайнера



Прежде чем делать игру, надо договориться о словах. Что такое механика и чем она отличается от сюжета? Кто такой NPC? Почему «квест» – это и жанр, и задание? В этой главе мы соберём словарь геймдизайнера и научимся узнавать жанр игры по короткому описанию. На отборе такие задачи выглядят как тест: выбрать верные варианты, сопоставить слова и описания, найти лишнее в ряду. Ответ – строка из цифр, например «3142». Считать почти не нужно, а вот условие читай по слову: одна пропущенная частица «не» – и верный ход мысли даёт неверный ответ.

1.1. Что такое игра

НА ПАЛЬЦАХ

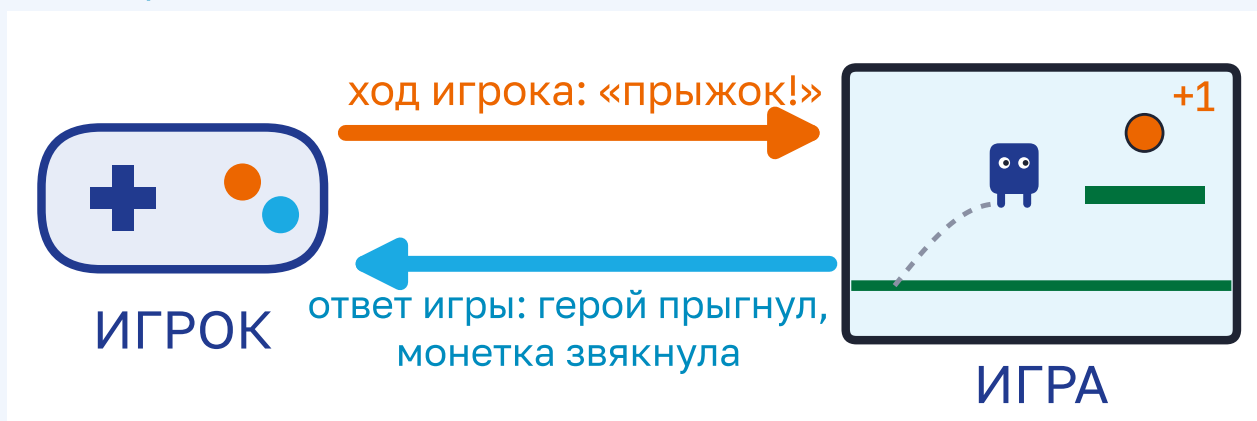


Рисунок 1.1 – Игра – это разговор: ты делаешь ход, игра отвечает

Мультфильм можно смотреть, сложив руки на коленях. Игру – нельзя: пока ты ничего не нажал, она ждёт. Нажал «прыжок» – герой прыгнул, монетка звякнула, счёт вырос на 1.

Книга и фильм рассказывают историю сами: от тебя ничего не зависит, конец будет тем же, даже если ты узнёшь. В игре ты участник. Ты действуешь, игра отвечает, ты смотришь, что вышло, и решаешь, что делать дальше. Этот круг «ход – ответ – новый ход» называют **петлёй обратной связи**. В быстрой игре он проворачивается по несколько раз в секунду.

Любая игра – компьютерная, настольная или во дворе – держится на четырёх опорах.

- **Цель** – чего добивается игрок. Дойти до флажка, собрать 100 кристаллов, первым потопить все корабли.
- **Правила** – что можно делать и что за это будет. «Ходим по очереди», «коснулся воды – минус одно сердечко».
- **Препятствия** – что мешает сразу добиться цели. Яма, враг, запертая дверь, таймер, который тикает.
- **Обратная связь** – как игра показывает, что произошло. Счёт, звук, мигание экрана, реплика персонажа.

Проверим на догонялках во дворе. Цель водящего – осалить кого-нибудь. Правила: коснулся рукой – осалил, теперь водит другой, за гаражи не убегать. Препятствия: все разбегаются, а водящий один. Обратная связь: крик «Осалил!», и тот, кого коснулись, начинает водить. Убери любую опору – и игра развалится. Без цели все просто бегает. Без правил начинается спор. Без препятствия цель достигается за секунду, и становится скучно. А без обратной связи никто не знает, осалили его или нет.

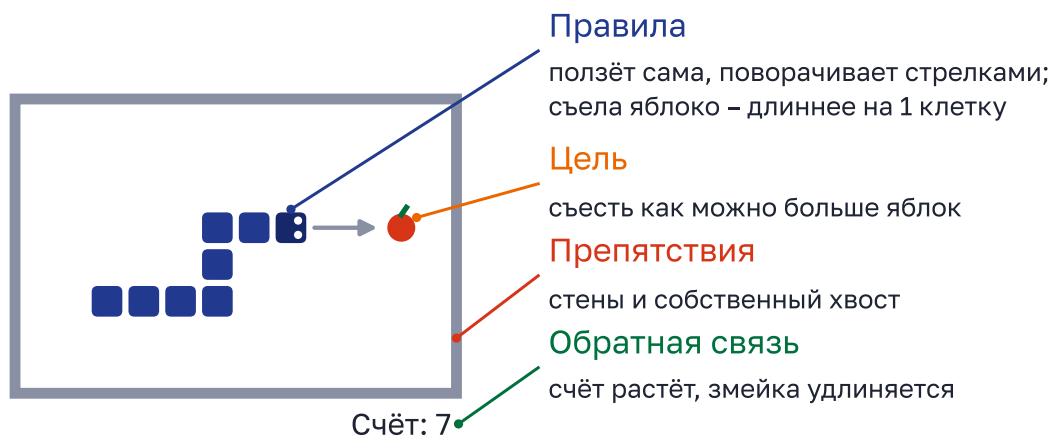


Рисунок 1.2 – Четыре опоры игры на примере «Змейки»

Посмотри на «Змейку» – в неё наверняка играли твои родители на кнопочном телефоне. Цель – съесть как можно больше яблок. Правила: змейка сама ползёт вперёд, ты только поворачиваешь её стрелками, а за каждое съеденное яблоко она становится длиннее на клетку. Препятствия – стены и собственный хвост: врежешься – игра окончена. Обратная связь – счёт в углу и то, как змейка растёт на глазах. Заметь хитрость: награда (змейка длиннее) одновременно усложняет игру – хвост всё больше мешает. Такие находки и делают игру интересной, о них поговорим в главе 2.

У большинства игр есть ещё и **условие победы** – момент, когда игра говорит: «Ты справился». Флажок в конце уровня, последний кусочек пазла, табличка «Уровень пройден». У «Змейки» и «Тетриса» такого момента нет, они идут, пока ты не проиграешь. Там победу заменяет рекорд: цель – набрать больше, чем в прошлый раз.

Запомни. Цель – чего хочешь. Правила – что можно и что за это будет. Препятствие – что мешает. Обратная связь – как игра показывает результат. Если в задаче просят «сопоставить часть игры с описанием», спроси себя про каждую фразу: «Она про то, чего добиться, как устроен мир, что мешает или как игра ответила?»

Совет. Препятствие тоже работает по правилам («сосулька падает раз в 5 секунд»). Если сомневаешься, правило перед тобой или препятствие, спроси: «Зачем это в игре?» Если чтобы мешать – это препятствие. Если чтобы объяснить, как всё устроено, – правило.

1.2. Жанры: как узнать игру по описанию

НА ПАЛЬЦАХ

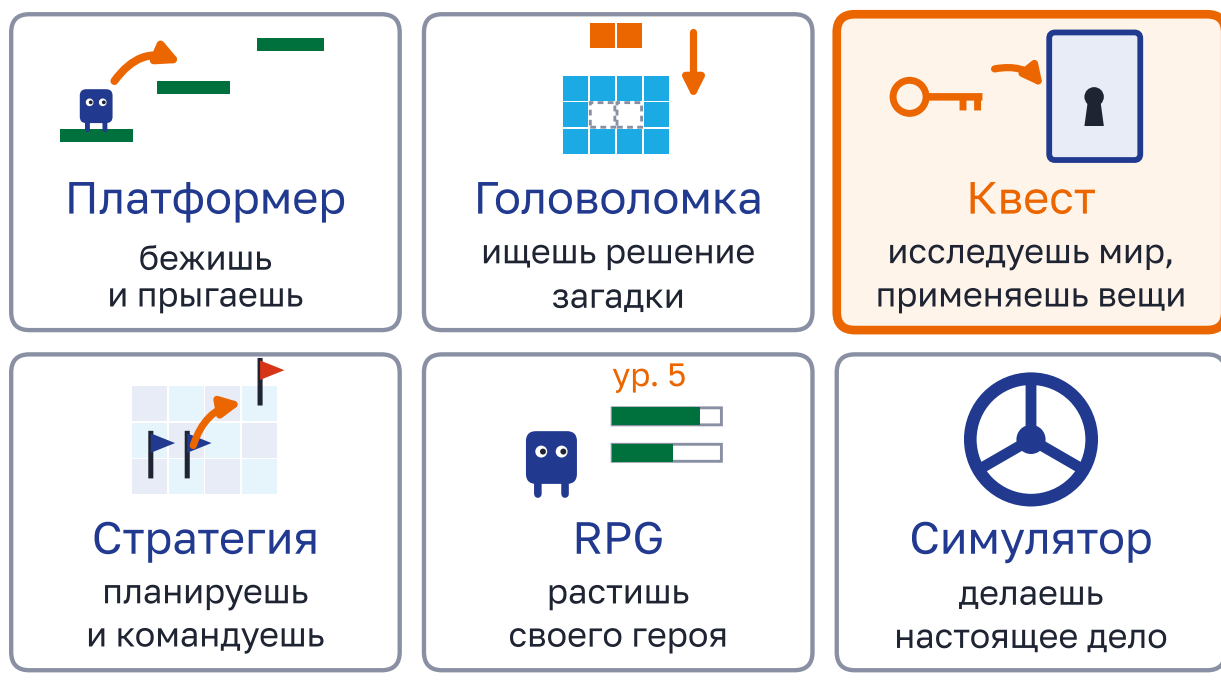


Рисунок 1.3 – Жанр – это то, что ты делаешь в игре большую часть времени

Названия жанров запоминать не нужно – нужно запомнить глаголы. Прыгаешь – платформер. Разгадываешь – головоломка. Исследуешь мир и применяешь найденное – квест.

Жанр – это группа игр, в которых игрок занят похожим делом. В книжном магазине книги стоят по полкам: детективы, фантастика, стихи. С играми так же: жанр – это «полка», по которой игрок заранее понимает, что его ждёт. Узнавать жанр надо по тому, **что игрок делает большую часть времени**, а не по картинке и не по сюжету. Игра про космос может оказаться и платформером, и стратегией, и симулятором.

Вот шесть жанров, которые встречаются в задачах чаще всего. Примеры игр мы сверили с Википедией.

Квест (он же **приключенческая игра**, по-английски adventure). Есть главный герой и история. Игрок исследует мир: осматривает комнаты, говорит с персонажами, собирает предметы и применяет их – ключ к двери, верёвку к колодцу. Главное – загадки и догадки, а драки, спешка и счёт денег сведены к минимуму или их нет совсем. Примеры: Machinarium и Samorost чешской студии Amanita Design, российская серия «Братья Пилоты».

Платформер. Основа игры – бег и прыжки по платформам: с уступа на уступ, через ямы, по лестницам, со сбором предметов по пути. Примеры: Super Mario Bros., Celeste, Rayman Legends.

Головоломка. Игрок решает задачи на сообразительность: как уложить фигуры, в каком порядке нажать, куда повернуть. У уровня обычно есть правильное решение, история не глав-

ное или её нет вовсе. Примеры: «Тетрис» (его придумал советский программист Алексей Па- житнов), Cut the Rope, Monument Valley.

Стратегия. Игрок управляет не одним героем, а многими: армией, городом, базой, целой страной. Главное – планировать: что построить, куда послать отряд, на что потратить ре- сурсы. Бывают **пошаговые** стратегии, где ходят по очереди, как в шахматах (Sid Meier’s Civilization VI), и стратегии **в реальном времени**, где все действуют одновременно (Clash Royale). «Защита башен» (tower defense) – разновидность стратегии: ставишь башни, чтобы остановить волны врагов.

RPG – ролевая игра (от английского role-playing game). У героя есть числовые **характери- стики**: очки здоровья, сила, ловкость, умения. За победы и задания герой получает **опыт**, растёт в **уровне** и становится сильнее. Часто игрок ещё и выбирает, как поступить в истории. Пример: первые игры серии Pokémon – Pokémon Red и Blue.

Симулятор. Игра повторяет настоящее дело, по возможности так, как оно устроено в жизни: вести грузовик, пахать поле, управлять самолётом, жить обычной жизнью. Примеры: Euro Truck Simulator 2, Farming Simulator, Microsoft Flight Simulator. The Sims 4 – «симулятор жизни».

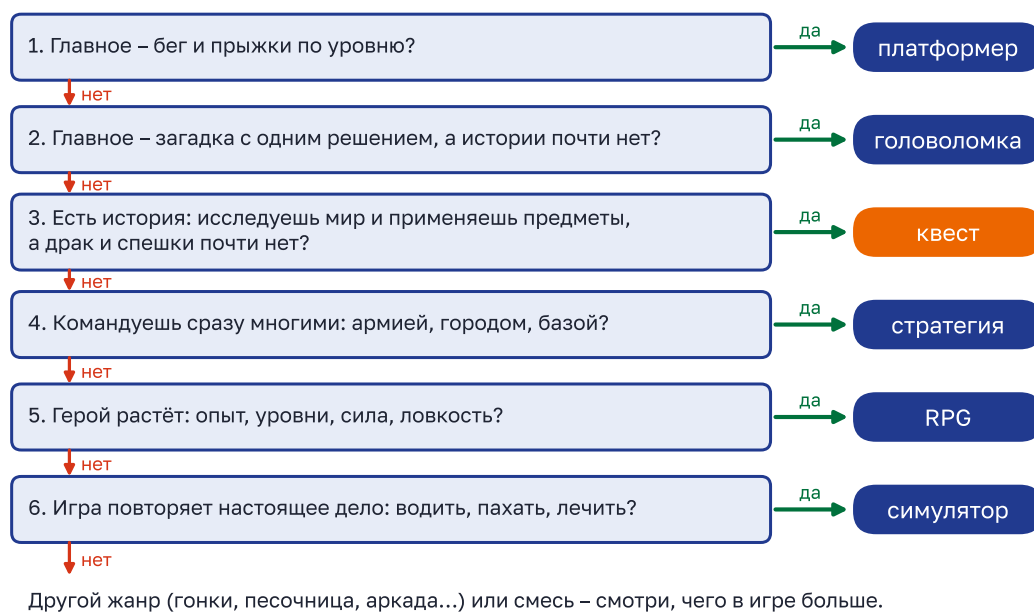


Рисунок 1.4 – Определитель жанра: задавай вопросы сверху вниз, пока не услышишь «да»

Таблица 1.1

Жанр	Что делает игрок	Слова-подсказки в описании
Квест	исследует мир, применяет предметы	«осматривает», «находит ключ», «без драк»
Платформер	бежит и прыгает по уровню	«с уступа на уступ», «через яму»
Головоломка	решает задачу на смекалку	«поверни», «одно решение»
Стратегия	планирует, командуем многими	«строит», «отряды», «ресурсы»
RPG	растит своего героя	«опыт», «уровень героя»

Жанр	Что делает игрок	Слова-подсказки в описании
Симулятор	повторяет настоящее дело	«как настоящий», «рычаги как в кабине»

Жанр – ярлык условный. На вебинаре сферы об этом сказали прямо: большинство современных игр – **смесь**. В Stardew Valley ты ведёшь ферму (симулятор жизни), а герой ещё и получает опыт и растёт в навыках (ролевая игра). Про Minecraft вообще говорят «песочница»: одной обязательной цели там нет, и каждый играет по-своему – один строит замок, другой роет шахту. Поэтому в задачах обычно берут игру, где один жанр явно главный, или прямо просят отметить **все** подходящие жанры.

Есть и другие жанры: гонки, файтинги, ритм-игры, аркады, визуальные новеллы (почти книга с картинками, где игрок выбирает, как пойдёт история). Если в задаче встретится незнакомое название, ищи в описании тот же главный глагол.

Запомни. Жанр определяют по главному действию игрока. Тема и картинка не в счёт: игра про ферму бывает и симулятором, и стратегией. Если признаков двух жанров поровну, перед тобой смесь – и тогда в задаче обычно просят отметить все подходящие.

Совет. В описании подчеркни глаголы, которые повторяются, и слова «главное», «большую часть времени», «нет». Фраза «сюжета нет, у уровня одно решение» почти наверняка про головоломку, а «драться не нужно, герой находит ключ» – про квест. Название игры, если оно знакомо, проверь в Википедии: в карточке справа есть строка «Жанр».

1.3. Из чего собрана игра: механики и элементы

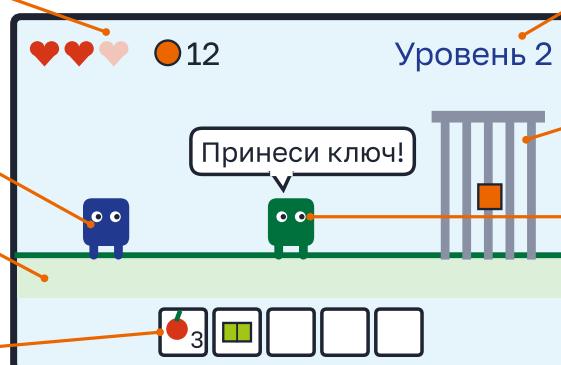
НА ПАЛЬЦАХ

ресурсы: здоровье
и монеты

персонаж игрока

локация –
двор у замка

инвентарь –
вещи с собой



уровень

препятствие:
закрытые ворота

NPC – персонаж,
которым управляет игра

Рисунок 1.5 – Обычный экран игры, а на нём – половина словаря геймдизайнера

Три сердечка и монеты – ресурсы. Зелёный персонаж – NPC, им управляет игра.
Внизу – инвентарь, всё, что герой несёт с собой.

Геймплей (игровой процесс) – это то, что игрок делает в игре и как игра ему отвечает. Когда друг спрашивает «а какой там геймплей?», он хочет услышать не сюжет, а «бегаешь по лесу, собираешь грибы и отбиваешься от ос».

Механика – одно правило, по которому игра отвечает на действие игрока. Геймдизайнеры записывают механику как цепочку из трёх звеньев: **событие → условие → изменение мира**. Событие – что произошло: игрок нажал кнопку, подошёл к двери, кончилось время. Условие – что игра проверяет: есть ли ключ, хватает ли монет. Изменение – что стало с миром: дверь открылась, монеты убавились, появилась надпись. Например: «игрок нажал „Открыть“ у ворот (событие); если в инвентаре есть ключ (условие), ворота открываются, а ключ исчезает (изменение)».

Самая частая путаница – принять за механику тему игры или её историю. Сравни:

Таблица 1.2

Это НЕ механика	Это механика
«Игра про выживание в тайге» – это тема, сеттинг	«Каждую минуту сытость падает на 1, а съеденная ягода добавляет 5»
«Героиня мечтает найти брата» – это сюжет	«Если поговорить с лесником дважды, он даст карту»
«Музыка спокойная, немного грустная» – это атмосфера	«Когда здоровья меньше 20, экран краснеет»

Сеттинг – место, время и мир, где всё происходит: средневековый замок, колония на Луне, деревня на берегу Обского моря. **Сюжет** – история: что случилось с героями и в каком порядке. Без них игра была бы скучной, но ни сеттинг, ни сюжет не говорят, что игроку делать и что ему за это будет.

Теперь детали, из которых собирается почти любая игра.

- **Персонаж игрока** (герой) – тот, кем управляешь ты.
- **NPC** (читается «эн-пи-си», от английского non-player character – неигровой персонаж) – персонаж, которым управляет не игрок, а сама игра: торговец, стражник, бабушка у калитки, ёж, который не пускает дальше. Поведение NPC программируют заранее. В сфере «Игротех» это делают машинами состояний на языке ПРИМС, об этом – глава 5.
- **Локация** – место в игровом мире: двор, пещера, школьный коридор. Из локаций складывается карта игры.
- **Уровень** – у этого слова два смысла. Первый – кусок игры с началом и концом: «я прошёл третий уровень». Второй – в RPG: «мой герой пятого уровня», то есть насколько он вырос. Какой из двух смыслов нужен, подскажет сама фраза.
- **Ресурс** – то, что можно накопить и потратить и чего вечно не хватает: здоровье, монеты, энергия, древесина, запас снежков, время. Если чего-то бесконечно много, это уже не ресурс.
- **Предмет** – вещь, которую можно взять: ключ, яблоко, фонарь. **Квестовый предмет** нужен для задания: без ключа от сарая удочку не достать.
- **Инвентарь** – «рюкзак» героя: список всех предметов, которые он несёт. Часто инвентарь – это сетка **ячеек**.
- **Интерфейс** – всё, что нарисовано поверх мира: сердечки, счёт, мини-карта, кнопки.

- **Чекпоинт** – точка сохранения: если герой проиграет, игра начнётся отсюда, а не с самого начала.
- **Босс** – особенно сильный противник, обычно в конце уровня.
- **Игровой цикл** – действия, которые игрок повторяет снова и снова: «собрал ресурсы → построил → защитил → собрал больше».

Как устроен инвентарь с ячейками. Во многих играх одинаковые предметы складываются в одну ячейку **стопкой**, но у стопки есть предел. Пусть в ячейку помещается 20 шишек. Тогда 47 шишек займут не 2 ячейки, а 3: две полные по 20 и третью, где лежат оставшиеся $47 - 40 = 7$. Словами: «делим число предметов на размер стопки и округляем **вверх**» – неполная стопка всё равно занимает целую ячейку. Бывают и предметы, которые не складываются вовсе: каждый ключ, каждая карта – в своей ячейке. Что складывается, а что нет, в задаче всегда сказано в условии.

Запомни. Проверка на механику: разложи фразу на «событие → условие → изменение мира». Разложилась – механика. Не разложилась – перед тобой сеттинг, сюжет или атмосфера. А в инвентаре неполная стопка занимает целую ячейку, поэтому число ячеек – это деление с округлением вверх.

1.4. Квест – главный жанр сферы

НА ПАЛЬЦАХ



Рисунок 1.6 – Квест в пять шагов – от просьбы бабушки до ключа от калитки

Кто-то просит, ты понимаешь цель, что-то мешает, ты находишь способ – и мир меняется. Так устроено любое игровое задание, от самого короткого до самого длинного.

В сфере «Игротех» 2026/27 квест – жанр номер один. На установочном вебинаре организаторы рассказали: на финале команда получит сцену в редакторе «Мастерская игр «Урсула»» и должна будет собрать на ней игру в жанре квест – добавить персонажей и запрограммировать их. Поэтому про квесты надо знать больше, чем про остальные жанры.

У слова «квест» два значения.

1. **Квест – жанр игры** (приключенческая игра), мы описали его в разделе 1.2: история, исследование мира, предметы и загадки.

2. **Квест – задание внутри игры:** «принеси», «найди», «почини». Такие задания есть почти в любом жанре – в RPG, в стратегиях, в песочницах. Их список игра часто держит в **журнале заданий**.

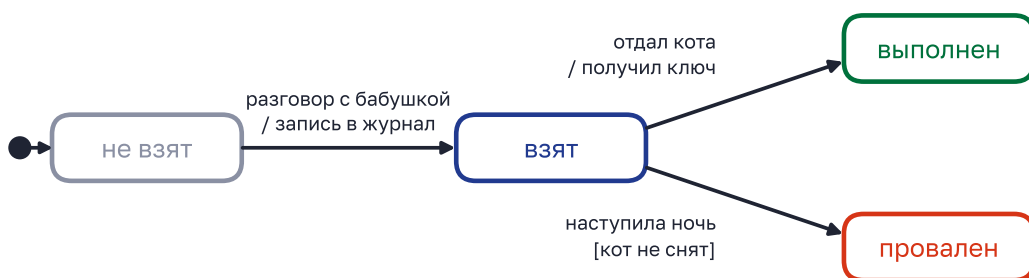
Поэтому из фразы «NPC дал герою квест» ещё не следует, что игра – квест по жанру. Путать эти два значения – одна из самых частых ошибок в тестах по геймдизайну.

Из чего состоит одно задание. Геймдизайнеры раскладывают его на пять шагов (посмотри на картинку выше):

- **Выдача** – как игрок узнаёт, что надо что-то сделать. Бабушка кричит: «Мой кот залез на сосну!»
- **Цель** – чего надо добиться. Снять кота с дерева.
- **Препятствие** – почему нельзя сделать это сразу. Высоко, а лестницы нет. Без препятствия это не задание, а просто действие: «подошёл и взял».
- **Условие** – что должно быть выполнено, чтобы цель засчиталась. В инвентаре есть лестница (её, например, дают у соседа за помощь с дровами).
- **Исход** – что изменилось в мире. Кот дома, бабушка дарит ключ от калитки, и открывается новая локация.

Выдать задание можно в лоб: NPC прямо говорит «Принеси мне три доски». Так надёжно – игрок точно поймёт, что делать. А можно, чтобы задание подсказал сам мир: ты видишь запертые ворота и догадываешься, что нужен ключ. Так приятнее: игрок радуется, что догадался сам.

Состояния задания. В каждый момент квест находится ровно в одном состоянии: ещё **не взят** (игрок о нём не знает), **взят** (игрок знает цель и работает над ней), **выполнен** или **провален** (сделать уже нельзя: наступила ночь, NPC уехал). Быть «немножко выполненным» квест не может. Переходы между состояниями рисуют так же, как поведение NPC, – в главе 5 ты увидишь, что это одна и та же машина состояний. Подпись у стрелки пишут в виде «событие [условие] / действие».



В каждый момент квест ровно в одном состоянии.

Рисунок 1.7 – Состояния квеста про кота. Подпись у стрелки: событие, в квадратных скобках – условие, после косой черты – действие

Порядок действий. В квесте одни дела можно делать только после других: удочку не достать из сарая, пока нет ключа. Такие связи рисуют стрелками «сначала → потом» (такую схему ты увидишь в задаче 1.7). А дело, которое ни от чего не зависит, можно сделать когда угодно, – и прохождений становится несколько.

Пример. «Взять ключ» и «открыть сарай» идут строго друг за другом. Накопать червей можно в любой момент: до ключа, между ключом и сараем или после сарая. Выходит три разных прохождения. Свободное дело встаёт в любой **промежуток** ряда, а промежутков всегда на один больше, чем дел: у двух дел их три – перед, между и после.

Запомни. Квест – и жанр, и задание. Шаги задания: выдача → цель → препятствие → условие → исход. Состояний у задания четыре: не взят, взят, выполнен, провален, – и в каждый момент только одно. Свободное дело можно вставить в ряд из трёх дел четырьмя способами, в ряд из четырёх – пятью.

Совет. Когда сопоставляешь фразы с шагами квеста, ищи подсказки в словах. «Говорит», «просит», «кричит» – выдача. «Нужно, чтобы...» – цель. «Но», «сломан», «заперт», «нет» – препятствие. «Если...» – условие. «Теперь», «в награду», «открылось» – исход.

1.5. Как записывать ответ. Словарь геймдизайнера

НА ПАЛЬЦАХ



Рисунок 1.8 – Ответ – это строка цифр, собранная по правилу из условия

Слева – «сопоставь»: под каждой буквой пишешь её номер и читаешь слева направо. Справа – «выбери все верные»: номера отмеченных вариантов идут по возрастанию.

Ответ на отборе проверяет компьютер. Человек понял бы и «А – 3, Б – 1», а компьютер сравнивает строку символ за символом, и лишняя запятая для него – уже другой ответ. В этой книге мы пишем ответы так (в задачах отбора обычно так же, но **правило из условия всегда главнее**):

- **«Выбери все верные»** – номера всех верных вариантов **по возрастанию**, без пробелов и запятых: «2468».
- **«Сопоставь»** – для каждой буквы по порядку (А, Б, В...) пишешь номер, который ей подходит: «3142» значит А – 3, Б – 1, В – 4, Г – 2.
- **«Лишнее в ряду»** – из каждого ряда по одной цифре, ряды по порядку: «31422».
- **Несколько вопросов в одной задаче** – ответы подряд, в том порядке, в каком идут вопросы. Так было, например, в задаче старшего трека НТО 2025/26 про жанры: ответ «416» – это три ответа на три вопроса.

На отборе на каждую задачу дают от 3 до 5 попыток ввода, но каждая неверная попытка снижает 30 % балла. Две ошибки в записи – и от верного решения остаётся меньше половины. Поэтому сначала проверь строку, потом жми «Отправить».

Запомни. Прочитай, в каком виде нужен ответ, **до** того, как начнёшь решать. «Все верные» – по возрастанию. «Сопоставь» – по порядку букв. Лишний пробел или запятая могут сделать верный ответ неверным.

Совет. Перед отправкой посчитай цифры в строке. Сопоставляешь 5 букв – цифр должно быть 5. Лишнее в 4 рядах – 4 цифры. Если цифр больше или меньше, ты что-то пропустил.

Словарь. В программе сферы «Игротех» есть «тест на знание терминов» (геймплей, NPC, ПРИМС, модификаторы), и засчитывается он, если верно не меньше 70 % ответов. Какими будут вопросы на отборе, мы не знаем, но слова ниже встретятся тебе во всех главах книги. Проверь себя: закрой ладонью объяснение после тире и расскажи термин своими словами.

Из этой главы (в скобках – раздел):

- **Игра (1.1)** – действие по правилам, где у игрока есть цель, препятствия и обратная связь.
- **Обратная связь (1.1)** – как игра показывает результат действия: звук, счёт, реплика.
- **Условие победы (1.1)** – момент, когда игра говорит «ты справился».
- **Жанр (1.2)** – группа игр с похожим главным действием игрока.
- **Квест (1.2, 1.4)** – 1) жанр: исследовать мир и применять предметы; 2) задание в игре.
- **Геймплей (1.3)** – игровой процесс: что игрок делает и как игра отвечает.
- **Механика (1.3)** – правило «событие → условие → изменение мира».
- **Сеттинг (1.3)** – где и когда происходит игра, какой в ней мир.
- **Сюжет (1.3)** – история: что происходит с героями и в каком порядке.
- **NPC (1.3)** – неигровой персонаж, им управляет игра.
- **Локация (1.3)** – место в игровом мире.
- **Уровень (1.3)** – 1) этап игры; 2) насколько вырос герой в RPG.
- **Ресурс (1.3)** – то, что копят и тратят: здоровье, монеты, время.
- **Инвентарь (1.3)** – предметы, которые герой несёт с собой.
- **Чекпоинт (1.3)** – точка, с которой игра продолжится после проигрыша.
- **Игровой цикл (1.3)** – действия, которые игрок повторяет снова и снова.

Встретятся дальше (в скобках – глава):

- **Геймдизайнер** (2) – человек, который придумывает правила и механики игры.
- **ГДД** (2) – документ игры: цели, механики, сюжет, персонажи.
- **Прототип** (2) – быстрая черновая версия игры для проверки идеи.
- **Плейтест** (2) – проверка игры живыми игроками.
- **Машина состояний** (5) – схема: в каком состоянии объект и что переводит его в другое.
- **ПРИМС** (5) – язык, на котором в «Урсуле» программируют поведение персонажей машинами состояний.
- **«Урсула»** (3, 5) – «Мастерская игр «Урсула»», российская онлайн-платформа и редактор игр сферы.
- **Баг** (6) – ошибка, из-за которой игра работает не так, как задумано.
- **Ink** (7) – язык для ветвящихся историй и диалогов.
- **Blender** (8) – бесплатная программа для 3D-моделирования.
- **Модификатор** (8) – в Blender операция, которая меняет модель целиком: Mirror отражает, Array размножает.

1.6. Разбираем задачи

Задача 1.1 Дворник против сугробов

разминка · цель, правила, препятствия, обратная связь, сопоставление

Описание мобильной игры «Чистая дорожка» разбили на четыре фразы.

- А.** С крыши то и дело падают сосульки, а снегоуборочная машина сгребаёт снег обратно на дорожку.
- Б.** Нужно за 2 минуты расчистить дорожку от подъезда до школьного автобуса.
- В.** Каждая расчищенная клетка вспыхивает зелёным и звенит, а таймер в углу показывает, сколько секунд осталось.
- Г.** Одно нажатие лопатой убирает одну клетку снега, а сугроб выше колена – за два нажатия.

Сопоставь каждую фразу с частью игры: **1** – цель, **2** – правило, **3** – препятствие, **4** – обратная связь. В ответ запиши четыре цифры для фраз А, Б, В, Г по порядку.

Дано: четыре фразы (А–Г) и четыре части игры (1–4).

Найти: для каждой фразы – номер части игры.

Решение

Шаг 1. Вспомним вопросы-ключи. Цель – «чего добиться?». Правило – «как устроен мир, что будет за действие?». Препятствие – «что мешает?». Обратная связь – «как игра показала результат?».

Шаг 2. Разберём каждую фразу.

- **А.** Сосульки и машина, которая возвращает снег, не дают дворнику спокойно чистить – они **мешают**. Это препятствие, **3**.
- **Б.** «Нужно за 2 минуты расчистить дорожку» – это то, **чего надо добиться**. Цель, **1**.
- **В.** Вспышка, звон и таймер – игра **показывает**, что происходит. Обратная связь, **4**.
- **Г.** Сколько нажатий нужно на клетку – так **устроен мир**, так работает лопата. Правило, **2**.

Шаг 3. Собираем строку. А – 3, Б – 1, В – 4, Г – 2: получаем «3142».

Шаг 4. Проверим строку. Каждая цифра от 1 до 4 встретилась по одному разу: частей игры четыре, и фраз тоже четыре. Если какая-то цифра повторилась, значит, две фразы попали в одну «корзину» и одну из них надо перечитать.

 **Типичная ошибка.** Назвать фразу Г препятствием: «сугроб выше колена – это же трудно». Но фраза описывает, **как работает** лопата, а не что мешает. Вторая ошибка – записать ответ в порядке номеров частей («цель – Б, правило – Г...») и получить «БГАВ» или «2413». Порядок в ответе – по буквам фраз.

Ответ: 3142.

Своя задача по программе сферы, модуль 1

Реши сам

1.1. Правила «Морского боя» разбили на четыре фразы. **А.** Соперник отвечает: «мимо», «ранил» или «убил». **Б.** Нужно первым потопить все корабли соперника. **В.** Ходите по очереди; выстрел – это название клетки, например «Б-5». **Г.** Корабли соперника спрятаны: его поле тебе не видно. Сопоставь фразы с частями игры: 1 – цель, 2 – правило, 3 – препятствие, 4 – обратная связь. Ответ – четыре цифры для А–Г.

1.2. Сопоставь термины и определения. Термины: **А** – NPC, **Б** – инвентарь, **В** – локация, **Г** – геймплей, **Д** – ресурс. Определения: 1) место в игровом мире: лес, пещера, двор замка; 2) то, что игрок копит и тратит и чего бывает мало; 3) персонаж, которым управляет не игрок, а игра; 4) что игрок делает в игре и как игра ему отвечает; 5) список вещей, которые герой несёт с собой. Ответ – пять цифр для А–Д.

Задача 1.2 Шесть игр без названий

жанры игр, определение жанра по описанию

Издатель прислал описания шести новых игр, но забыл написать жанры. Определи жанр каждой игры.

А. Ты работаешь на ферме: пашешь поле трактором, следишь за топливом, вовремя поливаешь и продаёшь урожай. Кнопки повторяют рычаги настоящей машины.

Б. Сыщик бродит по старому дому, разговаривает с жильцами, находит ржавый ключ и прищепку, а потом соображает, что прищепкой можно вытащить ключ из щели в полу. Драться и спешить не нужно.

В. На каждом уровне – поле из трубок. Поворачивай их, чтобы вода дошла от крана до цветка. Сюжета нет, у каждого уровня одно решение.

Г. Ты строишь деревни, нанимаешь отряды, решаешь, куда послать разведчиков, и ходишь по очереди с соперником.

Д. Колобок катится по лесу, прыгает с пенёка на пенёк, перепрыгивает ямы и собирает зёрнышки, пока не доберётся до флажка.

Е. Героиня получает опыт за каждое задание, растёт в уровне, распределяет очки силы и ловкости и выбирает, кому из жителей помочь.

Жанры: **1** – квест, **2** – платформер, **3** – головоломка, **4** – стратегия, **5** – RPG, **6** – симулятор. Каждый жанр подходит ровно к одной игре. В ответ запиши шесть цифр для игр А–Е по порядку.

Дано: шесть описаний (А–Е), шесть жанров (1–6), каждый жанр – ровно один раз.

Найти: жанр каждой игры.

Решение

Шаг 1. Ищем в каждом описании главный глагол – что игрок делает большую часть времени – и слова-подсказки из таблицы в разделе 1.2.

- **А.** «Пашешь трактором», «сведишь за топливом», «кнопки повторяют рычаги настоящей машины» – повторяем настоящее дело, как в жизни. **Симулятор, 6.**
- **Б.** «Бродит», «разговаривает», «находит ключ», «соображает, как применить», «драться и спешить не нужно» – исследуем мир и применяем предметы. **Квест, 1.**
- **В.** «Поворачивай трубки», «сюжета нет», «одно решение» – загадка на сообразительность. **Головоломка, 3.**
- **Г.** «Строишь деревни», «нанимаешь отряды», «ходишь по очереди» – командуем многими и планируем. **Стратегия (пошаговая), 4.**
- **Д.** «Прыгает с пенька на пенёк», «перепрыгивает ямы», «до флажка» – бег и прыжки по уровню. **Платформер, 2.**
- **Е.** «Опыт», «растёт в уровне», «очки силы и ловкости» – растим героя. **RPG, 5.**

Шаг 2. Проверяем, что каждый жанр использован один раз. Цифры 6, 1, 3, 4, 2, 5 – все от 1 до 6, без повторов.

Шаг 3. Собираем строку по буквам А–Е: «613425».

Шаг 4. А не головоломка ли Б? Загадка там есть, и история тоже. Но загадка решается предметами, которые герой нашёл, исследуя дом, – это визитная карточка квеста. У головоломки В нет ни героя, ни истории.

 **Типичная ошибка.** Определять жанр по теме: «ферма – значит, что-то про животных и деревню, наверное, стратегия». Тема ничего не решает, решает главное действие: на ферме из А ты водишь трактор «как настоящий» – это симулятор. Ещё легко перепутать Б и В: загадка есть и там, и там. Спроси себя: есть ли герой, который ходит по миру и применяет найденное? Если есть – квест.

Ответ: 613425.

По мотивам задачи НТО 2025/26, профиль «Разработка компьютерных игр» (старший трек, 8–11 классы), первый отборочный этап, инженерный тур, «Общий. Жанры игр»

Реши сам

1.3. Определи жанр по описанию: 1 – квест, 2 – платформер, 3 – головоломка, 4 – стратегия, 5 – RPG, 6 – симулятор. **А.** Ты водишь городской автобус: останавливаешься на остановках, открываешь двери, следишь за расписанием и скоростью. **Б.** Лиса бежит по лесу, прыгает по веткам и пенькам, переправляется через реку по кочкам. **В.** Расставь на поле зеркала так, чтобы луч попал в мишень; у каждого уровня одно решение, истории нет. **Г.** Девочка попадает в заброшенный музей, осматривает залы, говорит со зрителем, находит в вазе ключ и открывает им витрину. Ответ – четыре цифры для А–Г.

1.4.* В каждой игре отметь **все** жанры, признаки которых есть в описании: 1 – квест, 2 – платформер, 3 – головоломка, 4 – стратегия, 5 – RPG, 6 – симулятор. **Игра А.** Ты управляешь пекарней как настоящей: замешиваешь тесто, ставишь противни в печь, следишь за температурой. За каждый рабочий день пекарь получает опыт, растёт в уровне и учит новые рецепты. **Игра Б.** Робот бежит и прыгает по платформам. Иногда путь перегорожен, и надо переставить ящики так, чтобы открыть проход; у каждой такой загадки одно решение. **Игра В.** Сыщик в старой библиотеке осматривает полки, говорит с библиотекарем, собирает предметы и применяет их; драк и спешки нет. Запиши для каждой игры её цифры по возрастанию, игры – по порядку А, Б, В, всё подряд без пробелов.

Задача 1.3 Найди лишнее

термины: жанры, ресурсы, NPC, обратная связь, «лишнее в ряду»

Из каждого ряда выбери одно лишнее слово. В ответ запиши номера выбранных слов по порядку рядов.

Ряд 1: 1) платформер; 2) головоломка; 3) инвентарь; 4) стратегия.

Ряд 2: 1) сюжет; 2) здоровье; 3) монеты; 4) энергия.

Ряд 3: 1) Super Mario Bros.; 2) Celeste; 3) Rayman Legends; 4) «Тетрис».

Ряд 4: 1) торговец в лавке; 2) герой, которым управляешь ты; 3) стражник у ворот; 4) лодочник на пристани.

Ряд 5: 1) звон монетки при сборе; 2) название игры на коробке; 3) мигание героя при ударе; 4) счёт в углу экрана.

Дано: 5 рядов по 4 слова.

Найти: в каждом ряду – слово, которое не подходит к трём остальным.

Решение

Шаг 1. Для каждого ряда сначала находим, что объединяет большинство, и только потом – кто выпадает.

- **Ряд 1.** Платформер, головоломка, стратегия – жанры. Инвентарь – элемент игры, а не жанр. Лишнее – **3**.
- **Ряд 2.** Здоровье, монеты, энергия – ресурсы: их копят и тратят. Сюжет – история, его не потратишь. Лишнее – **1**.
- **Ряд 3.** Super Mario Bros., Celeste, Rayman Legends – платформеры (проверено по карточкам игр в Википедии). «Тетрис» – головоломка. Лишнее – **4**.

- **Ряд 4.** Торговец, стражник, лодочник – персонажи, которыми управляет игра, то есть NPC. Герой, которым управляешь ты, – персонаж игрока, не NPC. Лишнее – **2**.
- **Ряд 5.** Звон, мигание и счёт – обратная связь: игра показывает результат твоих действий прямо во время игры. Название на коробке ни на какое действие не отвечает. Лишнее – **2**.

Шаг 2. Собираем строку: 3, 1, 4, 2, 2 – «31422».

Шаг 3. Нет ли второго лишнего? Рядов пять – и цифр пять. Общий признак трёх оставшихся в каждом ряду называется одним словом: «жанры», «ресурсы», «платформеры», «NPC», «обратная связь». Второго кандидата ни в одном ряду нет.

 **Типичная ошибка.** В ряду 3 выбрать незнакомую игру: «Про Celeste я не слышал – значит, она лишняя». Лишнее ищут по признаку, а не по тому, что знакомо. Если название не знаешь, проверь жанр в Википедии: умение искать информацию сфера сама называет среди навыков, которые пригодятся на отборе. А в ряду 4 кто-нибудь обязательно решит, что лодочник лишний: «он же не в городе». Признак здесь другой: кто управляет персонажем.

Ответ: 31422.

По мотивам задачи НТО 2025/26, профиль «Разработка компьютерных игр» (старший трек, 8–11 классы), первый отборочный этап, инженерный тур, «Арт. Лишнее значение»

Реши сам

1.5. Из каждого ряда выбери одно лишнее, ответ – номера по порядку рядов. Ряд 1: 1) квест; 2) стратегия; 3) NPC; 4) симулятор. Ряд 2: 1) здоровье; 2) локация; 3) золото; 4) древесина. Ряд 3: 1) «Тетрис»; 2) Euro Truck Simulator 2; 3) Cut the Rope; 4) Monument Valley. Ряд 4: 1) выдача; 2) препятствие; 3) инвентарь; 4) исход.

1.6. Ответь на три вопроса, ответ – три цифры подряд. (1) Какая игра **не** головоломка: 1) «Тетрис»; 2) Cut the Rope; 3) Monument Valley; 4) Pokémon Red и Blue? (2) Какая игра **не** симулятор: 1) The Sims 4; 2) Euro Truck Simulator 2; 3) Celeste; 4) Farming Simulator? (3) Какая игра – квест: 1) Super Mario Bros.; 2) Machinarium; 3) Sid Meier’s Civilization VI; 4) «Тетрис»?

Задача 1.4 Что из этого механика

механика, сеттинг, сюжет, «выбери все верные»

Геймдизайнер описал будущую игру восемью фразами. Выбери **все** фразы, которые описывают механику – правило, по которому игра отвечает на действие игрока или на событие в мире. В ответ запиши номера по возрастанию.

1. Игра про колонию на Луне.
2. Если персонаж выйдет из купола без скафандра, у него каждую секунду убывает 5 единиц кислорода.
3. Главная героиня – инженер Вера, она ищет пропавшего брата.

4. Нажми «Е» рядом с ящиком – ящик откроется, если у тебя есть лом.
5. Музыка в игре спокойная и немного грустная.
6. За каждые 10 собранных кристаллов открывается новая шахта.
7. Действие происходит в 2150 году.
8. Когда кислорода меньше 20 единиц, экран краснеет и слышно тяжёлое дыхание.

Дано: 8 фраз.

Найти: номера всех фраз-механик.

Решение

Шаг 1. Проверка на механику. Попробуем разложить каждую фразу на три звена «событие → условие → изменение мира». Получилось – механика. Не получилось – это сеттинг, сюжет или атмосфера.

Шаг 2. Разбираем.

- **1.** Нет события и нет изменения – только тема и место. **Сеттинг**, не механика.
- **2.** Событие: персонаж вышел из купола. Условие: без скафандра. Изменение: кислород убывает на 5 каждую секунду. **Механика**.
- **3.** Кто героиня и чего она хочет – **сюжет**.
- **4.** Событие: нажал «Е» рядом с ящиком. Условие: есть лом. Изменение: ящик открылся. **Механика**.
- **5.** Никакого действия игрока, никакого изменения – описание **атмосферы**.
- **6.** Событие: собран десятый кристалл. Изменение: открылась шахта. **Механика** (условие «набралось 10» спрятано в самой фразе).
- **7.** Когда происходит действие – **сеттинг**.
- **8.** Событие: кислород упал ниже 20. Изменение: экран краснеет, звучит дыхание. **Механика обратной связи** – это тоже механика.

Шаг 3. Собираем номера по возрастанию: 2, 4, 6, 8 – «2468».

Шаг 4. Посмотри на слова. Во всех выбранных фразах стоит «если», «когда» или «за каждые» – верный признак правила. В остальных четырёх таких слов нет.

 **Типичная ошибка.** Пропустить фразу 8: «это же про красивые эффекты, а не про правила». Но эффект появляется по правилу – при событии «кислород ниже 20» – и помогает игроку понять, что он в опасности. Обратная связь тоже задаётся механиками. Обратная ошибка – выбрать фразу 1, потому что «колония на Луне – это же и есть игра». Тема – ещё не игра.

Ответ: 2468.

Своя задача по программе сферы, модуль 1

Реши сам

1.7. Выбери все фразы, которые описывают механику, ответ – номера по возрастанию. 1) Игра о подводной станции в океане. 2) Если водолаз дотронется до медузы, он теряет одно сердечко. 3) Главная героиня – биолог Аня. 4) Когда воздух в баллоне кончается, водолаз всплывает, и уровень начинается заново. 5) Станция выкрашена в жёлтый цвет. 6) Нажми «Пробел» рядом с раковиной – в инвентаре появится жемчужина. 7) За 5 жемчужин торговец продаёт новый фонарь.

1.8. Выбери все верные утверждения, ответ – номера по возрастанию. 1) В квесте как жанре главное – история, исследование мира и загадки. 2) Квест – это жанр, где главное – быстро стрелять по врагам. 3) В квесте предметы из инвентаря часто применяют к чему-то в мире или соединяют друг с другом. 4) Словом «квест» называют и жанр, и отдельное задание в игре. 5) Если NPC выдаёт герою задание, игра обязательно относится к жанру квест. 6) В квесте не бывает картинки – только текст.

Задача 1.5 Мост через овраг

структура квеста: выдача, цель, препятствие, условие, исход, сопоставление

Сценарист разбил задание из игры на пять фраз, но перепутал их порядок.

- А.** Мост через овраг сломан – доски сгнили.
- Б.** Лесник у костра говорит: «Помоги, мне надо на ту сторону оврага, там мой дом».
- В.** Если принести три доски и молоток, мост можно починить.
- Г.** Лесник переходит мост, а на прощание дарит фонарь – теперь можно спуститься в пещеру.
- Д.** Нужно, чтобы лесник попал домой.

Сопоставь каждую фразу с шагом квеста: **1** – выдача, **2** – цель, **3** – препятствие, **4** – условие, **5** – исход. В ответ запиши пять цифр для фраз А–Д по порядку.

Дано: пять фраз (А–Д), пять шагов квеста (1–5).

Найти: шаг для каждой фразы.

Решение

Шаг 1. Вспоминаем слова-подсказки из совета в разделе 1.4: «говорит, просит» – выдача; «нужно, чтобы» – цель; «сломан, заперт, нет» – препятствие; «если» – условие; «теперь, в награду» – исход.

Шаг 2. Разбираем фразы.

- **А.** «Сломан», «сгнили» – то, что мешает. **Препятствие, 3.**
- **Б.** NPC говорит, что ему нужно, – так игрок узнаёт о задании. **Выдача, 1.**
- **В.** «Если принести... можно починить» – что должно быть выполнено. **Условие, 4.**
- **Г.** Лесник ушёл, игрок получил фонарь, открылась пещера – мир изменился. **Исход, 5.**
- **Д.** «Нужно, чтобы...» – чего добиться. **Цель, 2.**

Шаг 3. Собираем строку по буквам: А – 3, Б – 1, В – 4, Г – 5, Д – 2: «31452».

Шаг 4. Прочитаем по порядку. Расставим фразы по номерам шагов: Б (выдача) → Д (цель) → А (препятствие) → В (условие) → Г (исход). Получилась связная история: лесник попросил, мы

поняли цель, увидели сломанный мост, узнали, что нужно для починки, и получили награду. Значит, сопоставили верно.

❗ Типичная ошибка. Перепутать цель и выдачу: фраза Б тоже говорит, что леснику нужно домой. Разница в том, **кто и как** это сообщает. Выдача – момент, когда игрок узнаёт о задании (реплика лесника). Цель – само «что нужно сделать», без того, кто сказал. Ещё путают условие и препятствие: препятствие – что мешает («мост сломан»), условие – как это преодолеть («если принести доски»).

Ответ: 31452.

Своя задача по программе сферы, модуль 1

Реши сам

1.9. Сопоставь фразы с шагами квеста: 1 – выдача, 2 – цель, 3 – препятствие, 4 – условие, 5 – исход. **А.** Если вставить в робота новую батарейку, он снова заработает. **Б.** Робот-почтальон мигает и пищит: «Заряд кончился, помоги!» **В.** Батарейки продают только в лавке, а лавка откроется, когда кто-нибудь починит её вывеску. **Г.** Нужно, чтобы робот снова развозил письма. **Д.** Робот оживает и отдаёт тебе письмо, которое искал весь город. Ответ – пять цифр для А–Д.

1.10. Квест бывает в одном из четырёх состояний: 1 – не взят, 2 – взят, 3 – выполнен, 4 – провален. Рыбак просит принести ему 3 рыбы до заката. Определи состояние квеста. **А.** Рыбак уже попросил, у тебя пока одна рыба. **Б.** Солнце село, у тебя только 2 рыбы, рыбак ушёл домой, и задание больше не выполнить. **В.** Ты ещё не подходил к рыбаку и не знаешь, что ему нужно. **Г.** Ты отдал рыбаку 3 рыбы, и он подарил тебе лодку. Ответ – четыре цифры для А–Г.

Задача 1.6 Сколько шишек влезет в рюкзак

инвентарь, стопки, деление с округлением вверх

Рюкзак героя – это 12 ячеек. Одинаковые предметы складываются в одну ячейку стопкой, но не больше 20 штук в стопке. Ключи и карты не складываются: одна вещь – одна ячейка.

Класть разные предметы в одну ячейку нельзя.

У героя 47 шишек, 18 досок, 3 ключа и 1 карта. Сколько ещё шишек он сможет подобрать, пока рюкзак не заполнится полностью? Ответ – целое число.

Дано: 12 ячеек; стопка – не больше 20; шишек 47, досок 18, ключей 3, карт 1.

Найти: сколько ещё шишек поместится.

Решение

Шаг 1. Сколько ячеек занимают шишки. $47 : 20 = 2,35$. Две ячейки по 20 шишек вмещают только 40, оставшиеся $47 - 40 = 7$ шишек лягут в третью ячейку. Неполная стопка тоже занимает целую ячейку, поэтому округляем **вверх: 3 ячейки**.

Шаг 2. Остальные вещи. 18 досок – меньше 20, они помещаются в **1 ячейку**. 3 ключа – **3 ячейки**, карта – **1 ячейка**: они не складываются.

Шаг 3. Сколько ячеек свободно. Занято $3 + 1 + 3 + 1 = 8$ ячеек, свободно $12 - 8 = 4$ ячейки.



Рисунок 1.9 – Рюкзак героя из задачи 1.6 до того, как он пошёл собирать шишки

Шаг 4. Куда можно добавить шишки. Есть два места:

- в неполную третью стопку, где лежит 7 шишек: туда влезет ещё $20 - 7 = 13$;
- в 4 пустые ячейки – по 20 в каждую: $4 \cdot 20 = 80$.

Итого $13 + 80 = 93$ шишки.

Шаг 5. Проверка. Всего шишек станет $47 + 93 = 140$. Это $140 : 20 = 7$ полных ячеек. Вместе с досками, ключами и картой: $7 + 1 + 3 + 1 = 12$ – рюкзак забит под завязку. Сходится.

⚠ Типичная ошибка. Округлить 2,35 вниз и решить, что шишки занимают 2 ячейки. Тогда свободных ячеек выйдет 5, а ответ – больше сотни. Но 7 «лишних» шишек никуда не делись, они лежат в третьей ячейке. Бывает и наоборот: про неполную стопку забывают и отвечают $4 \cdot 20 = 80$, а в третью ячейку влезет ещё 13 шишек.

Ответ: 93 шишки.

Своя задача по программе сферы, модуль 1

Реши сам

1.11. Сумка героя – 10 ячеек, одинаковые предметы складываются стопкой не больше 16 штук, разные в одну ячейку класть нельзя, ведра и фонари не складываются. У героя 35 яблок, 16 морковок, 2 ведра и 1 фонарь. Сколько ещё яблок он сможет подобрать, пока сумка не заполнится полностью?

1.12. В тесте 18 вопросов, засчитывают, если верно не меньше 70 %. Сколько ошибок **самое большее** можно допустить, чтобы тест всё-таки засчитали?

Задача 1.7* Сколько способов пройти квест

порядок действий в квесте, комбинаторика, подсчёт вариантов

Дед просит внука наловить рыбы. В квесте семь действий:

- А** – поговорить с дедом (без этого квест не начнётся);
- Б** – взять у бабушки ключ от сарая;

В – достать из сарая удочку;

Г – накопать червей (лопата стоит у забора, ключ не нужен);

Д – взять в сарае насос и накачать колесо велосипеда;

Е – доехать на велосипеде до озера (нужны удочка, черви и накачанное колесо);

Ж – поймать рыбу.

Сарай заперт, пока нет ключа. Каждое действие выполняется ровно один раз. Сколько существует разных порядков, в которых можно пройти этот квест? Ответ – целое число.

Дано: 7 действий и связи «сначала – потом»: А раньше Б и Г; Б раньше В и Д; В, Г и Д раньше Е; Е раньше Ж.

Найти: число разных порядков.

Решение

Шаг 1. Рисуем связи. Стрелка « $X \rightarrow Y$ » значит «X нужно сделать раньше Y».



Стрелка « $X \rightarrow Y$ »: X нужно сделать раньше, чем Y.

Рисунок 1.10 – Что для чего нужно в квесте про рыбалку

Шаг 2. Что стоит на своих местах намертво. А – самое первое: без разговора с дедом квест не взят. Ж – последнее: рыбу ловят на озере, а Е (доехать) требует всего остального. Значит, Е – шестое, Ж – седьмое. Свободно переставляются только четыре действия в середине: Б, В, Г, Д.

Шаг 3. Какие правила действуют в середине. Б раньше В и раньше Д (без ключа в сарай не попасть). Г ни от чего, кроме А, не зависит и может стоять где угодно среди этих четырёх.

Шаг 4. Сначала расставим Б, В, Д. Б обязано быть первым из них, а В и Д после него – в любом порядке: «Б, В, Д» или «Б, Д, В». Это 2 варианта.

Шаг 5. Вставим Г. Это то самое «свободное дело» из раздела 1.4. В ряду из трёх действий четыре промежутка: перед первым, между первым и вторым, между вторым и третьим, после третьего. Г можно поставить в любой из них. Для каждого из 2 рядов – 4 места: $2 \cdot 4 = 8$ порядков.

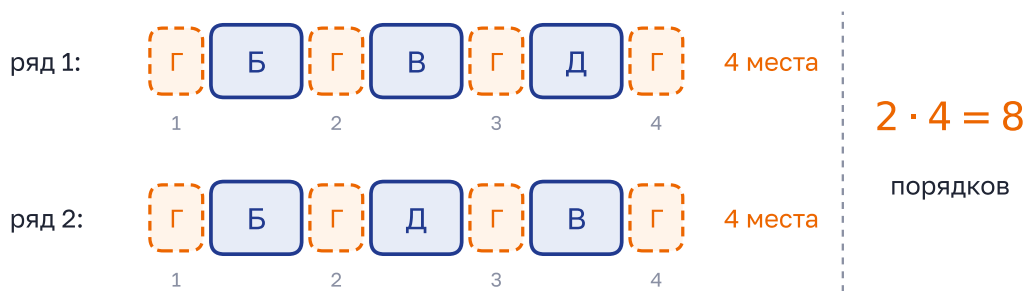


Рисунок 1.11 – Два ряда из шага 4 и четыре промежутка в каждом, куда можно поставить Г

Шаг 6. Проверим, выписав все порядки. Ряд «Б, В, Д»: ГБВД, БГВД, БВГД, БВДГ. Ряд «Б, Д, В»: ГБДВ, БГДВ, БДГВ, БДВГ. Добавим А в начало, Е и Ж в конец – получим 8 прохождений, например АГБВДЕЖ или АБДВГЕЖ. Повторов нет, и каждый порядок соблюдает все стрелки.

Много это или мало? Без всяких правил семь действий можно выстроить 5040 способами. Но правила жёсткие: первое, шестое и седьмое места заняты намертво, В и Д ждут ключа. От пяти тысяч должна остаться горстка – 8 на неё похоже.

Это самая трудная задача главы. Не улеглись шаги 4 и 5 с первого раза – выпиши порядки на листке, как в шаге 6, и промежутки станут видны своими глазами.

❗ **Типичная ошибка.** Переставлять четыре средних действия как угодно и получить $4 \cdot 3 \cdot 2 \cdot 1 = 24$. Но В и Д не могут стоять раньше Б – сарай ещё заперт. Другая ошибка – забыть, что Г может идти и до Б: «сначала же ключ!». Червей копают лопатой у забора, ключ для этого не нужен, поэтому Г может стоять даже вторым.

Для любопытных. Откуда 5040: на первое место можно поставить любое из 7 действий, на второе – любое из 6 оставшихся, и так далее: $7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5040$. Если ты знаешь Python, проверь ответ программой: она перебирает все 5040 порядков и считает те, где каждое «X раньше Y» выполнено.

```
from itertools import permutations

# пары (X, Y): X нужно сделать раньше Y
before = [("А", "Б"), ("Б", "В"), ("Б", "Д"), ("А", "Г"),
          ("В", "Е"), ("Д", "Е"), ("Г", "Е"), ("Е", "Ж")]

count = 0
for p in permutations("АБВГДЕЖ"):
    place = {x: i for i, x in enumerate(p)} # на каком месте каждое действие
    if all(place[x] < place[y] for x, y in before):
        count += 1
print(count) # 8
```

Ответ: 8.

По мотивам задачи НТО 2025/26, профиль «Разработка компьютерных игр» (старший трек, 8–11 классы), первый отборочный этап, инженерный тур, «Геймдизайн. Путь игрока»

Реши сам

1.13.* Дедушка просит сделать кормушку для синиц. Действия: **А** – поговорить с бабушкой (без этого квест не начнётся); **Б** – взять у папы ключ от гаража; **В** – взять в гараже доски; **Г** – взять в гараже молоток и гвозди; **Д** – сколотить кормушку (нужны доски, молоток и гвозди); **Е** – купить в магазине семечки (ключ для этого не нужен); **Ж** – повесить кормушку и насыпать в неё семечки. Гараж заперт, пока нет ключа, каждое действие – ровно один раз. Сколько существует разных порядков прохождения квеста?

1.14.* Младшая сестра просит слепить снеговика. Действия: **А** – поговорить с сестрой (квест начинается только после этого); **Б** – скатать нижний ком; **В** – скатать средний ком и поставить его на нижний; **Г** – скатать голову и поставить её на средний ком; **Д** – найти в сарае старое ведро; **Е** – взять в холодильнике морковь; **Ж** – надеть снеговика ведро и приделать нос-морковь. Ком ставят только на уже готовый ком, сарай не заперт, каждое действие – ровно один раз. Сколько существует разных порядков прохождения квеста?

Геймдизайн: механики, баланс, документ игры



Сколько здоровья у слизня? Сколько стоит новая кирка? Через сколько минут игрок накопит на замок? Каждое такое число кто-то придумал и проверил на калькуляторе. Это работа геймдизайнера. В главе ты научишься считать баланс, проверять документ игры и разбирать итоги плейтеста. На отборе такая задача – короткая история с правилами игры, а ответ – одно число или строка из номеров вариантов.

2.1. Игровой цикл: действие – награда – новая цель

НА ПАЛЬЦАХ



Рисунок 2.1 – Игровой цикл – колесо, которое крутит сам игрок

Стукнул киркой по камню – выпала руда. Продал руду – купил кирку покрепче. Новой киркой можно разбить камень потвёрже – и колесо пошло на новый круг.

Вспомни игру, от которой трудно оторваться. Ты делаешь в ней одно и то же десятки раз – и не скучаешь. Почему? Каждое действие приносит **награду**, а награда открывает **новую цель**. Такая повторяющаяся цепочка называется **игровым циклом**. В задачах старшего трека НТО его называют по-английски – core loop («кор-луп», главный цикл).

У цикла три звена:

- **Действие** – что игрок делает руками: прыгает, копает, сажает, стреляет, отвечает на вопрос.
- **Награда** – что он за это получает: монеты, опыт, новый предмет, очки, кусочек истории.
- **Новая цель** – куда награду потратить и куда двигаться дальше: купить улучшение, открыть уровень, дойти до ворот.

Пример – «Змейка» из главы 1. Действие: довести голову до яблока. Награда: очко, и змейка стала длиннее на клетку. Новая цель: следующее яблоко, а длинный хвост делает дорогу к нему опаснее. Цикл замкнулся, и каждый круг чуть труднее предыдущего. На этом «чуть» игра и держит.

Разорванный цикл. Если одно звено выпало, колесо останавливается. Монеты сыплются, а тратить их не на что – минут через десять игрок перестанет их подбирать. Или наоборот: в лавке лежит меч за 5000 монет, а монеты нигде не падают. Хуже всего, когда игра вообще не отвечает на действие: кликаешь – и ничего не происходит. На вебинаре сферы эту ошибку называли одной из самых частых среди тех, что убивают интерес.

Циклы вложены друг в друга, как шестерёнки в часах. Короткий длится секунды: прыгнул – монетка звякнула. Средний занимает один уровень: прошёл – получил звёзды – открыл следующий. Длинный тянется всю игру: прокачал героя – открыл новый мир. Игрок возвращается, если награда ждёт его и через секунду, и через неделю.

Из чего сделано колесо. Каждое звено цикла – это механика, правило «событие → условие → изменение мира» из главы 1. «Ударил камень киркой – выпала руда» – одна механика. «Отдал руду торговцу – получил монеты» – вторая. «Хватает монет – кирка стала крепче» – третья. Сцепи их в кольцо, чтобы изменение одной запускало следующую, – получится цикл.

Запомни. Цикл – кольцо: действие → награда → новая цель → снова действие. Выпало любое звено – кольцо рвётся, и игрок уходит. Короткие циклы (секунды) вложены в длинные (уровень, вся игра).

Совет. Чтобы найти цикл незнакомой игры, задай себе три вопроса подряд. Что я делаю чаще всего? Что я за это получаю? Зачем мне это нужно? Ответ на третий вопрос обычно возвращает к первому – значит, цикл найден.

2.2. Баланс: чтобы не было скучно и обидно

НА ПАЛЬЦАХ

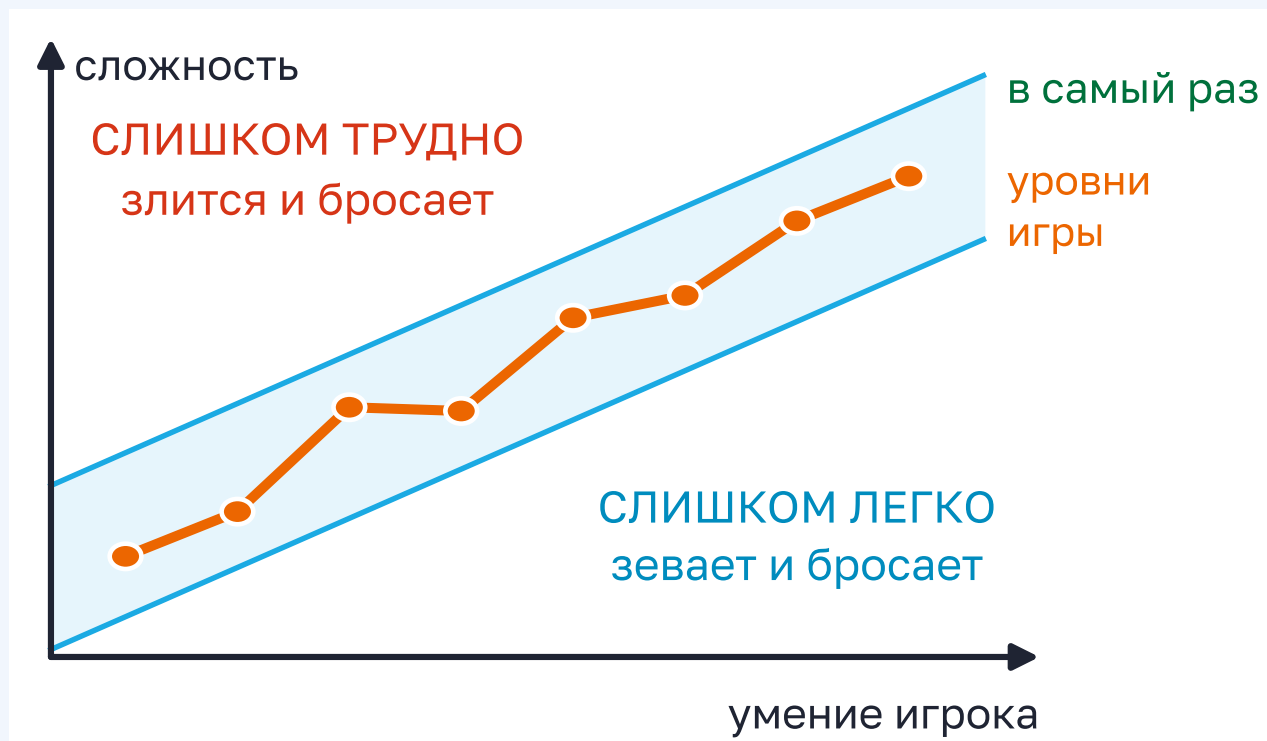


Рисунок 2.2 – Коридор интереса: сложность растёт вместе с умением игрока

Выше коридора – слишком трудно, игрок злится. Ниже – слишком легко, игрок зевает. Баланс держит уровни внутри коридора.

Баланс – это настройка чисел в игре: здоровья, урона, цен, наград. Правила могут быть отличными, но если у первого же слизня 1000 здоровья, а меч снимает 5, игрок уйдёт на втором уровне. На вебинаре сферы одним из источников интереса назвали растущее умение игрока: сначала трудно, потом получается, уровни сложнее – а ты всё круче. Значит, сложность должна расти, но не быстрее, чем игрок учится.

Числа баланса считают по простым формулам. Но сначала перечитай правило в условии. Законы в игре такие, какие написал автор: если сказано «броня снимает 5 очков», это очки, а не проценты.

Сколько нужно ударов. Словами: здоровье врага делим на урон одного удара и округляем вверх.

$$n = \frac{HP}{d} - \text{округлить вверх,}$$

где HP – здоровье врага (от английского hit points – «очки жизни»), d – **урон** одного удара, то есть сколько здоровья он снимает, n – число ударов. Почему вверх? У волка 100 здоровья, удар снимает 30: $100:30 = 3,33$. После трёх ударов у волка останется $100 - 90 = 10$ – он ещё жив. Нужен четвёртый удар. Треть удара нанести нельзя.



Рисунок 2.3 – У волка 100 здоровья, удар снимает 30. Трёх ударов не хватает, четвёртый добивает

Броня и проценты. Броня часто снижает урон на какой-то процент. **Процент** – сотая часть: 25 % – это 25 частей из 100, то есть 0,25. Снизить на 25 % – значит оставить $100 - 25 = 75$ % удара, то есть умножить урон на 0,75:

$$d_{\text{после брони}} = d \cdot \left(1 - \frac{p}{100}\right),$$

где p – сколько процентов забирает броня. Удар 40, броня 25 %: $40 \cdot 0,75 = 30$. Броня в 25 % снимает четверть удара, а вовсе не 25 очков урона. Запись $40 - 25 = 15$ – ошибка.

Если процентов несколько подряд, их не складывают. Каждый следующий процент берут от того, что получилось после предыдущего. Зелье даёт +50 % к урону, а проклятие отнимает 50 %: $40 \cdot 1,5 \cdot 0,5 = 30$. Урон не вернулся к 40, а стал меньше.

Урон в секунду. Одно оружие бьёт сильно, но редко, другое – слабо, зато часто. Чтобы их сравнить, считают урон за одну секунду: урон удара умножают на число ударов в секунду. Кинжал бьёт на 10 три раза в секунду – это $10 \cdot 3 = 30$ урона в секунду. Молот бьёт на 50 один раз в 2 секунды – это $50:2 = 25$ урона в секунду. У кинжала урон в секунду больше, хотя каждый его удар слабее.

Пропорции: обмен и рецепты. Во многих играх одно превращают в другое: из 3 брёвен – 5 досок, за 4 монеты – 1 кристалл. Это **пропорция**: соотношение остаётся тем же, сколько раз ни повторяй обмен. Из 3 брёвен выходит 5 досок, значит, из 6 – 10 досок, из 9 – 15. Одна ловушка: обмен идёт целыми порциями. Нужно 12 досок – это $12:5 = 2,4$ порции. Двух порций

(10 досок) мало, значит, пилим три порции, тратим $3 \cdot 3 = 9$ брёвен, и 3 доски останутся лишними.

Польза за монету. Чтобы сравнить две покупки, дели пользу на цену. Меч прибавляет 12 урона и стоит 300 монет: $12 : 300 = 0,04$ урона за монету. Топор прибавляет 20 урона за 600 монет: $20 : 600 \approx 0,033$. Каждая монета, потраченная на меч, приносит больше урона, хотя топор сильнее. Топор стоит вдвое дороже меча, а прибавляет меньше чем вдвое больше.

Запомни. Удары, ходы и порции обмена бывают только целыми – округляй вверх, чтобы хватило. Минус $p\%$ – умножить на $(1 - p/100)$, плюс $p\%$ – на $(1 + p/100)$. Проценты подряд перемножают, а не складывают.

Совет. Проверь ответ «на один меньше». Хватило бы $n - 1$ ударов? Если да – ты округлил лишнее. Если нет, а n хватает – ответ верный.

2.3. Прогрессия: как растут цены и опыт

НА ПАЛЬЦАХ

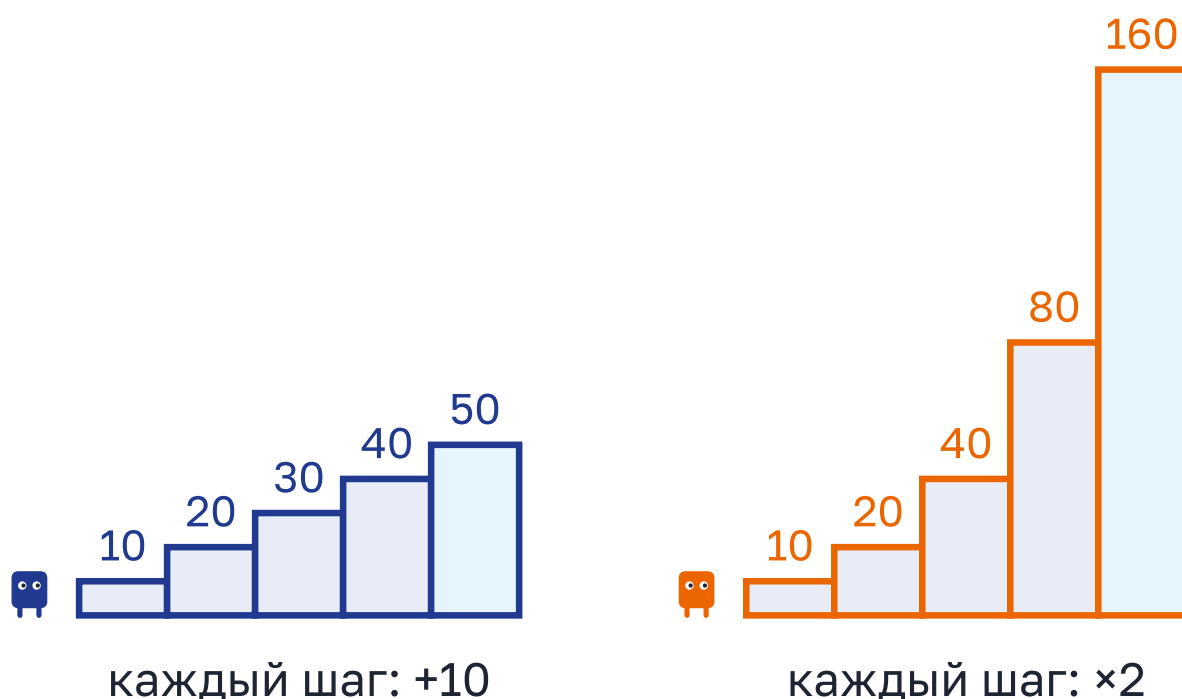


Рисунок 2.4 – Две лестницы: у левой ступеньки одинаковые, у правой – каждая выше предыдущей

Слева каждый шаг прибавляет одно и то же число. Справа каждый шаг умножает на одно и то же число – и лестница быстро уходит вверх.

Герой растёт, и игра хочет, чтобы каждый следующий уровень давался чуть труднее. Для этого числа выстраивают в ряд по правилу. Такой ряд называют **прогрессией**. В играх обычно

встречаются два правила.

Каждый раз прибавляем одно и то же число. Чтобы перейти с 1-го уровня на 2-й, нужно 100 очков опыта, а каждый следующий переход требует на 40 больше: 100, 140, 180, 220... Чтобы найти 10-е число ряда, не нужно выписывать все десять. От первого числа до десятого – 9 шагов, и каждый шаг прибавляет 40. Словами: число под номером n равно первому числу плюс $n - 1$ прибавок.

$$\text{число № } n = \text{первое} + (n - 1) \cdot \text{прибавка}.$$

Для 10-го: $100 + 9 \cdot 40 = 460$. Почему $n - 1$, а не n ? Вспомни забор: между первым и десятым столбиком 9 промежутков.

Каждый раз умножаем на одно и то же число. Первое улучшение рюкзака стоит 20 монет, каждое следующее – в 3 раза дороже: 20, 60, 180, 540... Шагов снова $n - 1$, только каждый шаг теперь умножение на 3. Для 5-го улучшения шагов 4:

$$20 \cdot 3 \cdot 3 \cdot 3 \cdot 3 = 20 \cdot 3^4 = 20 \cdot 81 = 1620.$$

Запись 3^4 – это **степень**: тройку умножают саму на себя 4 раза. Словами: число под номером n равно первому числу, умноженному на множитель $n - 1$ раз. В инженерном калькуляторе (в телефоне он появляется, если повернуть экран набок) для степени есть своя кнопка, обычно x^y .

Умножение растёт гораздо быстрее, чем прибавление. Геймдизайнеры этим пользуются: первые улучшения дешёвые и радуют сразу, а последние требуют терпения. Но если перестараться, десятое улучшение будет стоить миллион, и игрок бросит копить.

Сколько всего. Часто спрашивают не одну цену, а сумму: сколько опыта нужно, чтобы дойти с 1-го уровня до 5-го, или сколько улучшений купишь на 1000 монет. Тут надёжнее всего таблица: цена каждого шага и строка «всего с начала».

Таблица 2.1

Переход	1 → 2	2 → 3	3 → 4	4 → 5
Опыт за переход	100	140	180	220
Всего с начала	100	240	420	640

С 1-го уровня до 5-го – четыре перехода, а не пять. Всего нужно 640 очков опыта.

В электронной таблице такой ряд строится за минуту. Первую цену пишем в ячейку B2, а в B3 – формулу `=B2+40` (для умножения – `=B2*3`). Потом тянем B3 за уголок вниз, и таблица сама считает следующие цены. Столбец «всего» устроен так же: в C2 пишем `=B2`, в C3 – `=C2+B3` и тоже тянем вниз.

Запомни. От первого числа ряда до n -го – $n - 1$ шаг, как и от 1-го уровня до n -го – $n - 1$ переход. Шаг «плюс»: первое + $(n - 1) \cdot$ прибавка. Шаг «умножить»: первое $\cdot$ множитель в степени $n - 1$. Сумму надёжнее считать таблицей со строкой «всего».

Для любопытных. В 9 классе такие ряды называют **арифметической** (шаг «плюс») и **геометрической** (шаг «умножить») прогрессией и пишут буквами: $a_n = a_1 + (n - 1) \cdot d$ и $b_n = b_1 \cdot q^{n-1}$. Здесь a_1 и b_1 – первые числа, d – прибавка, q – множитель, маленькая буква n внизу – номер числа в ряду. Это те же самые фразы, только короче.

2.4. Сколько минут и ходов до цели

НА ПАЛЬЦАХ

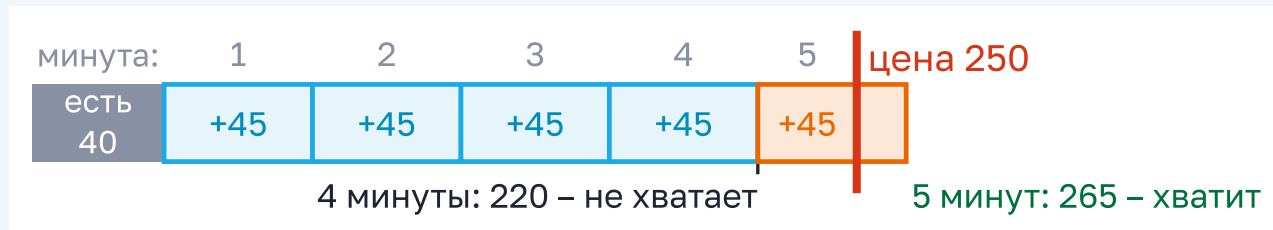


Рисунок 2.5 – Копим на улучшение за 250 монет: 40 уже есть, каждую минуту приходит ещё 45

За четыре минуты набирается 220 – мало. Пятая минута перекрывает цену.
Ждать 5 минут, хотя деление даёт 4,67.

Любимый вопрос игрока: «Сколько мне ещё копить?» Геймдизайнер задаёт его себе заранее, чтобы игроку не пришлось ждать час там, где терпения хватит на пять минут. Считать просто: из цены вычитаем то, что уже лежит в кармане, и делим остаток на доход за одну минуту (или за ход, за забег, за пойманную рыбу). Округляем **вверх**: за неполную минуту монеты ещё не пришли.

$$t = \frac{\text{цена} - \text{уже есть}}{\text{доход за минуту}} - \text{округлить вверх}.$$

Пример с картинки: $(250 - 40) : 45 = 210 : 45 = 4,67$. Ждать придётся 5 минут. Если доход дан не за минуту, а за действие (за рыбу, за забег), сначала найди число действий, округли его вверх и только потом умножай на время одного действия.

Окупаемость. Иногда выгоднее сначала потратить монеты на улучшение, которое ускорит доход. Шахта даёт 20 монет в минуту, а улучшение за 120 монет добавляет ещё 10 монет в минуту. Каждую минуту улучшение приносит лишние 10 монет, поэтому свои 120 монет оно вернёт за $120 : 10 = 12$ минут. Это **срок окупаемости**. Если впереди ещё час игры, улучшение выгодно. Если до конца события осталось 10 минут, покупать его не стоит: оно не успеет вернуть свою цену.

$$\text{срок окупаемости} = \frac{\text{цена улучшения}}{\text{прибавка к доходу за минуту}}.$$

Когда путей к цели несколько, не гадай. Посчитай время для каждого пути отдельно и сравни – в задаче 2.7 так и сделаем, путь за путём.

Запомни. Время = недостающее : доход, округлить вверх. Если доход дан «за действие», сначала находим число действий (вверх), потом время. Улучшение окупается за «цена : прибавка» минут.

2.5. ГДД – паспорт игры

НА ПАЛЬЦАХ



Рисунок 2.6 – Один документ – одна игра для всей команды

Художник, программист и сценарист читают один и тот же документ. Если в нём дыра, каждый заполнит её по-своему – и выйдут три разные игры.

ГДД – геймдизайн-документ, описание игры, по которому её делает команда. Даже если игру делаешь один, через неделю забудешь, что придумал про ворота и сколько морковок нужно снеговика. ГДД помнит за тебя.

В программе сферы готовый ГДД проверяют так: есть ли в нём **цели, механики, сюжет и персонажи** и **связаны ли разделы между собой**. Обычно в ГДД такие разделы (подробная таблица – в справочнике в конце книги):

- **Концепция** – название, жанр, для кого игра, о чём она в одной-двух фразах.
- **Цель** – как победить и как можно проиграть.
- **Механики** – что делает игрок и как игра ему отвечает.
- **Игровой цикл** – что игрок повторяет снова и снова.
- **Сюжет** – завязка, главные события, концовка.
- **Персонажи** – кто они и как себя ведут.
- **Уровни и локации** – карта, где что лежит.
- **Управление** – какие клавиши нажимать, что видно на экране.

Связность – на ней чаще всего и ловят в задачах. Все разделы должны говорить об одной и той же игре. Если в «Цели» сказано «собери три ключа», то в «Механиках» или «Уровнях» должно быть написано, где взять каждый из трёх. Если в «Персонажах» есть сторож, должно

быть сказано, что он делает. Любое упоминание – как ниточка, у которой где-то в документе должен найтись второй конец.

Чек-лист проверки ГДД. По этому списку решаются задача 2.5 и задачи 2.9 и 2.10 из «Реши сам». Номера пунктов пишут в ответ.

1. Концепция: указаны жанр и для кого игра.
2. Цель: сказано, как победить.
3. Цель: сказано, как можно проиграть.
4. Всё, что нужно для победы (предметы, их количество), в игре можно получить, и числа в разных разделах совпадают.
5. Описан игровой цикл: что игрок повторяет.
6. Сюжет: есть завязка и концовка.
7. У каждого персонажа сказано, как он себя ведёт: что делает и когда.
8. Управление: сказано, какие клавиши или кнопки нажимать.

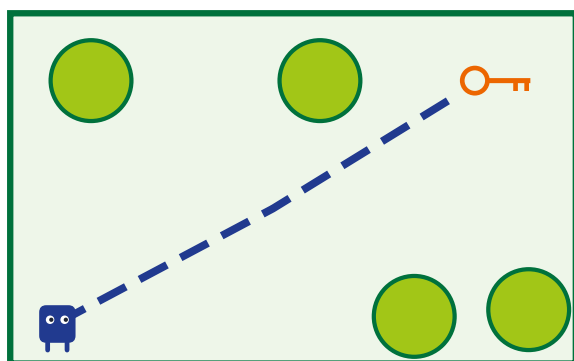
Как проверять: иди по списку сверху вниз и для каждого пункта ищи в тексте конкретную фразу. Нашёл – ставь галочку. Не нашёл – это пропуск. Пункт 4 проверяй отдельно: выпиши всё, что нужно для победы, и напротив каждого предмета напиши, в каком разделе сказано, где его взять.

Совет. Красивые слова не заменяют правила. «Сторож – строгий, но справедливый» – это характер, а не поведение. Поведение звучит так: «если игрок подошёл к складу ближе чем на три клетки, сторож идёт к нему и отводит к воротам».

2.6. Плейтест и обратная связь

НА ПАЛЬЦАХ

автор: знает, где ключ



новичок: кружит по саду

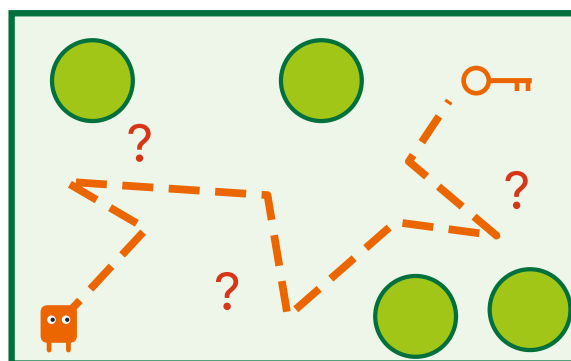


Рисунок 2.7 – Автор знает дорогу к ключу, новичок – нет

Автор проходит уровень за минуту: он сам прятал ключ. Новичок кружит по саду. Плейтест показывает игру глазами того, кто видит её впервые.

Плейтест – проверка игры на живых игроках. Разработчик знает свою игру наизусть и уже не замечает, где непонятно. Об этом прямо говорили и на вебинаре сферы: самому играть в свою игру полезно, но обязательно нужны и внешние люди – у них найдутся другие трудности и другие ошибки. Поэтому работа над игрой идёт по кругу: идея → прототип → плейтест → доработка → снова плейтест. **Прототип** – быстрая черновая версия игры, на которой проверяют, работает ли идея.

Как провести плейтест, чтобы от него была польза:

- Дай поиграть **без объяснений**. Когда игру выложат, тебя рядом с игроком не будет.
- **Молчи и записывай**: где игрок застрял и на сколько минут, где засмеялся, где начал зевать, куда кликал, а ничего не произошло.
- Не спорь и не подсказывай. Если хочется сказать «да там же видно!» – значит, не видно.
- Вопросы задавай **после** игры: «Что было непонятно? Что хотелось сделать, но не получилось?»

Обратная связь – то, что игроки говорят и делают. Отделяй факт от совета. «Я пять минут искал ключ» – это факт, и его надо чинить. «Сделай ключ красным» – это совет, один из способов починить. Каким способом чинить, решаешь ты.

Считаем итоги. Один игрок застрял – может быть, случайность. Застряли 7 из 10 – виновата игра. Поэтому итоги плейтеста переводят в проценты: какая доля игроков нашла ключ, какая дошла до конца уровня. Удобно записать, сколько игроков прошли каждый этап. Потом для каждого этапа считают, какая часть **дошедших до него** на нём ушла:

$$\text{ушли на этапе} = \frac{\text{дошли до этапа} - \text{прошли этап}}{\text{дошли до этапа}} \cdot 100 \%$$

Делить нужно именно на тех, кто до этапа дошёл, а не на всех игроков. До последних уровней добираются немногие, и даже небольшое число ушедших может оказаться огромной долей.

После правки проводят новый плейтест. Меняй одну вещь за раз: если исправил сразу три, не узнаешь, какая из них помогла.

Запомни. Плейтест проводят на тех, кто игру ещё не видел. Смотри и записывай, а не объясняй. Долю ушедших считают от дошедших до этапа, а не от всех.

2.7. Разбираем задачи

Задача 2.1 Каменный слизень

разминка · урон, броня в процентах, округление вверх

У каменного слизня 200 очков здоровья и броня, которая снижает любой урон на 25 %. Меч героя наносит 40 урона за удар. Сколько ударов мечом нужно, чтобы победить слизня? Ответ – целое число.

Дано: $HP = 200$; $d = 40$; броня $p = 25\%$.

Найти: n – число ударов.

Решение

Шаг 1. Что происходит. Броня забирает часть каждого удара. Сначала узнаем, сколько урона доходит до слизня, потом – сколько таких ударов нужно.

Шаг 2. Урон после брони. Броня снимает 25% , значит, до слизня доходит $100 - 25 = 75\%$ удара: $40 \cdot 0,75 = 30$ урона.

Шаг 3. Число ударов. $200 : 30 = 6,67$.

Шаг 4. Округляем вверх. Проверим оба соседних целых числа. После 6 ударов: $6 \cdot 30 = 180$ – это меньше 200, у слизня осталось 20 здоровья, он жив. После 7 ударов: $7 \cdot 30 = 210$ – больше 200, слизень побеждён. Нужно 7 ударов.

Прикинь без брони. Хватило бы $200 : 40 = 5$ ударов. Броня ослабила каждый удар на четверть – ударов прибавилось, но не вдвое: было 5, стало 7.

⚠ Типичная ошибка. Вычесть проценты из урона как очки: $40 - 25 = 15$, и тогда $200 : 15 = 13,3$ – «14 ударов». Броня дана в процентах, а не в очках. Ещё одна ловушка – округлять «по правилам». Здесь $6,67$ по правилам тоже даёт 7, но если бы вышло $6,2$, округление по правилам дало бы 6, и слизень остался бы жив. Удары округляем только вверх.

Ответ: 7.

Своя задача по программе сферы, модуль 1

Реши сам

2.1. У робота-охранника 250 очков здоровья и щит, который снижает любой урон на 40% . Лазер героя наносит 55 урона за выстрел. Сколько выстрелов нужно, чтобы победить робота? Ответ – целое число.

2.2. Удар героя – 80 урона. Зелье силы увеличило урон на 25% , а потом проклятие уменьшило получившийся урон на 25% . Каким стал урон? На сколько процентов он меньше исходных 80?

Задача 2.2 Чей это цикл

разминка · игровой цикл, жанры

Сопоставь каждой игре её игровой цикл. Один цикл лишний. В ответ запиши номера циклов по порядку игр А, Б, В, Г, Д – пять цифр подряд, без пробелов.

Игры: **А** – Tetris; **Б** – Minecraft; **В** – Plants vs. Zombies; **Г** – Angry Birds; **Д** – Stardew Valley.

Циклы:

1. Запустить птицу из рогатки → разрушить постройку и сбить всех свиней → получить звёзды за уровень → открыть следующий уровень.
2. Посадить и полить семена → собрать урожай → продать его → купить новые семена и улучшить ферму.
3. Разогнаться на трассе → прийти к финишу первым → получить кубок → открыть новую трассу.
4. Повернуть и поставить падающую фигуру → заполнить ряд целиком → ряд исчезает, начисляются очки → фигуры падают всё быстрее.
5. Добыть блоки → сделать из них инструмент получше → добыть блоки более редкие и прочные.
6. Собрать солнце → посадить растения на лужайке → отбить волну зомби → получить новое растение.

Дано: 5 игр, 6 циклов, один лишний.

Найти: номера циклов для игр А–Д.

Решение

Шаг 1. Главное действие каждой игры. Вспомним, что игрок делает в игре чаще всего. В Tetris – ставит падающие фигуры. В Minecraft – добывает блоки и мастерит из них вещи. В Plants vs. Zombies – сажает растения, которые защищают дом от зомби. В Angry Birds – стреляет птицами из рогатки. В Stardew Valley – работает на ферме.

Шаг 2. Ищем цикл, который начинается с этого действия. Помогают слова-приметы. «Рогатка» и «свины» – только в цикле 1, это Angry Birds (Г). «Зомби» – в цикле 6, это Plants vs. Zombies (В). «Фигура» и «ряд» – цикл 4, Tetris (А). «Урожай» и «ферма» – цикл 2, Stardew Valley (Д). «Блоки» и «инструмент» – цикл 5, Minecraft (Б).

Шаг 3. Лишний цикл. Остался цикл 3 – про гонки. Ни одна из пяти игр не гонка, значит, он и есть лишний.

Шаг 4. Записываем по порядку игр. А – 4, Б – 5, В – 6, Г – 1, Д – 2.

Прочитай пары вслух. «Tetris – поставить фигуру и заполнить ряд», «Minecraft – добыть блоки и сделать инструмент». Каждая звучит как рассказ про знакомую игру, каждой игре достался свой цикл, а лишний – про гонки, которых в списке нет.

 **Типичная ошибка.** Спутать Minecraft и Stardew Valley: в обеих играх что-то добывают, а потом тратят. Смотри на действие и награду. В Stardew Valley действие – растить урожай, а награда – деньги за него. В Minecraft действие – добывать блоки, а награда – новый инструмент. И совсем обидно: сопоставил всё верно, но выписал цифры по порядку циклов, а не игр.

Ответ: 45612.

По мотивам задачи первого отборочного этапа НТО 2025/26, профиль «Разработка компьютерных игр» (8–11 классы), «Геймдизайн. Игровой цикл»

Реши сам

2.3. Сопоставь играм их игровые циклы. Один цикл лишний. В ответ запиши номера циклов по порядку игр А, Б, В, Г – четыре цифры подряд.

Игры: **А** – Pokémon; **Б** – Super Mario Bros.; **В** – Just Dance; **Г** – Machinarium.

1. Повторять движения танцора на экране в такт музыке → получить очки за точность → выбрать новую песню.
2. Осмотреть место → найти и подобрать предметы → решить с их помощью головоломку → попасть в новое место.
3. Собрать команду футболистов → сыграть матч → получить деньги → купить игроков посильнее.
4. Встретить дикое существо → сразиться с ним → твоя команда получает опыт и становится сильнее → идти к соперникам посильнее.
5. Бежать и прыгать через ямы и врагов, собирая монеты → дойти до флагштока в конце уровня → открыть следующий уровень.

2.4. Геймдизайнер набросал циклы для четырёх игр. Про каждый набросок реши: 1 – цикл целый; 2 – за действие игрок ничего не получает; 3 – награду некуда потратить. В ответ запиши четыре цифры по порядку набросков А, Б, В, Г.

- **А.** Игрок ловит рыбу и продаёт её в лавке, а на монеты покупает удочку, которой можно поймать рыбу крупнее.
- **Б.** Игрок собирает на уровне звёзды. Счётчик звёзд растёт, но звёзды ничего не открывают, и купить за них нечего.
- **В.** Игрок поливает цветы в саду. Цветы от этого не растут, монет и очков за полив нет. А в лавке продаются новые лейки за монеты.
- **Г.** Игрок побеждает слизней и получает опыт. С каждым новым уровнем героя открывается пещера, где слизи сильнее.

Задача 2.3 Удочка мечты

время до цели, округление вверх, перевод секунд

В игре про рыбалку на Обском море удочка «Щучья» стоит 530 монет. У тебя в кармане 70 монет. Один заброс длится 40 секунд, за каждый заброс ловится ровно одна рыба, а лавка сразу покупает её за 18 монет. Сколько секунд придётся рыбачить, чтобы купить удочку? Ответ – целое число секунд.

Дано: цена 530 монет; есть 70 монет; 18 монет за рыбу; 1 заброс = 1 рыба = 40 с.

Найти: время, с.

Решение

Шаг 1. Что происходит. Доход здесь дан не за минуту, а за одно действие – за рыбу. Поэтому сначала найдём, сколько нужно забросов, округлим, и только потом переведём забросы во время.

Шаг 2. Сколько не хватает. $530 - 70 = 460$ монет.

Шаг 3. Сколько забросов. $460 : 18 = 25,56$. Проверим соседние целые. 25 забросов принесут $25 \cdot 18 = 450$ монет – не хватает 10. 26 забросов принесут $26 \cdot 18 = 468$ монет – хватает. Нужно **26 забросов**.

Шаг 4. Время. $26 \cdot 40 = 1040$ с. В минутах: $1040 : 60 = 17$ мин и ещё $1040 - 17 \cdot 60 = 20$ с, то есть 17 мин 20 с.

Проверка. После 26 забросов в кармане $70 + 468 = 538$ монет, удочка куплена, и 8 монет осталось. А 17 минут рыбалки ради хорошей удочки – честная цена: столько игрок подождёт.

 **Типичная ошибка.** Про 70 монет в кармане легко забыть и посчитать $530 : 18 = 29,4$, то есть 30 забросов и 1200 с. Бывает, что дробное число забросов умножают на время: $25,56 \cdot 40 = 1022$ с. Но рыба ловится только за целый заброс. И последняя ловушка – записать $1040 : 60 = 17,33$ как «17 мин 33 с». 0,33 минуты – это 20 секунд, а не 33.

Ответ: 1040 с (17 мин 20 с).

Своя задача по программе сферы, модуль 1

Реши сам

2.5. Для портала нужно 300 кристаллов, у тебя уже есть 45. Одно подземелье проходится за 4 минуты и приносит 12 кристаллов. Сколько минут нужно ходить по подземельям, чтобы открыть портал?

2.6. На лесопилке из 3 брёвен получается 5 досок, распилить можно только целую порцию из 3 брёвен. Для моста нужно 42 доски и ещё 6 брёвен на опоры. Сколько всего брёвен нужно?

Задача 2.4 Кирка всё дороже

прогрессия с умножением, сумма, таблица

Первое улучшение кирки стоит 32 монеты, каждое следующее – в 1,5 раза дороже предыдущего. Улучшения покупают только по порядку. а) Сколько стоит 5-е улучшение? б) У игрока 400 монет. Сколько улучшений подряд, начиная с первого, он сможет купить и сколько монет у него останется?

Дано: $b_1 = 32$ монеты; $q = 1,5$; в кармане 400 монет.

Найти: b_5 ; число улучшений на 400 монет и остаток.

Решение

Шаг 1. Что происходит. Цена каждый раз умножается на одно и то же число – это прогрессия с умножением. Для пункта а) хватит формулы, для пункта б) нужна таблица со строкой «всего».

Шаг 2. Цена 5-го улучшения. От 1-го до 5-го – 4 шага: $b_5 = 32 \cdot 1,5^4 = 32 \cdot 5,0625 = 162$ монеты. Проверим цепочкой: $32 \rightarrow 48 \rightarrow 72 \rightarrow 108 \rightarrow 162$ – сходится.

Шаг 3. Таблица. Выписываем цены и складываем нарастающим итогом.

Таблица 2.2

Улучшение	1	2	3	4	5
Цена	32	48	72	108	162
Всего потрачено	32	80	152	260	422

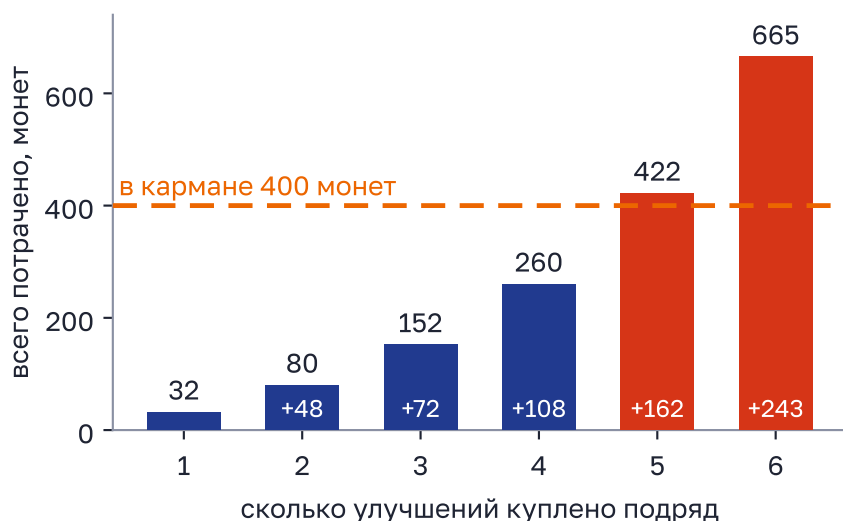


Рисунок 2.8 – Сколько всего потрачено, если купить улучшения подряд. После пятого сумма превышает 400

Шаг 4. Сколько купим. Четыре улучшения стоят вместе 260 монет – это не больше 400. С пятым набирается 422 – больше 400. Значит, игрок купит 4 улучшения, и у него останется $400 - 260 = 140$ монет.

Проверка. Пятое стоит 162, а в кармане 140 – на него не хватает.

Первые улучшения стоят копейки, а пятое уже дороже первых трёх вместе. Геймдизайнер на это и рассчитывал: чем дальше, тем дольше копить.

⚠ **Типичная ошибка.** Возвести в пятую степень вместо четвёртой: $32 \cdot 1,5^5 = 243$ – это цена 6-го улучшения. Шагов всегда на один меньше, чем номер. Не лучше поделить 400 на 32 и решить, что хватит на 12 улучшений: так вышло бы, если бы все они стоили по 32, а цены растут.

Ответ: а) 162 монеты; б) 4 улучшения, останется 140 монет.

Своя задача по программе сферы, модуль 1

Реши сам

2.7. Первое улучшение рюкзака стоит 25 монет, каждое следующее – вдвое дороже предыдущего. Улучшения покупают по порядку. Сколько улучшений подряд, начиная с первого, можно купить на 1000 монет?

2.8. Чтобы перейти с 1-го уровня на 2-й, нужно 100 очков опыта, а каждый следующий переход требует на 50 очков больше предыдущего. а) Сколько очков опыта нужно для перехода с 9-го уровня на 10-й? б) Сколько всего очков опыта нужно набрать, чтобы с 1-го уровня дойти до 6-го?

Задача 2.5 Смотритель маяка

ГДД, чек-лист, связность разделов

Прочитай фрагмент ГДД и проверь его по чек-листу из раздела 2.5. Какие пункты чек-листа нарушены? Номера запиши по возрастанию, без пробелов.

Смотритель маяка.

Концепция. Квест для игроков 10–13 лет. Внук смотрителя маяка должен зажечь старый маяк, пока на Обское море не пришёл шторм.

Цель. Победа – фонарь маяка зажжён. Для этого нужно принести к фонарю керосин, фитиль и спички. Поражение – шторм начался (через 10 минут после старта), а фонарь не горит.

Механики. Керосин стоит в сарае. Дверь сарая открывается, если у героя есть ключ; ключ висит на гвозде у причала. На причале лежит удочка: если герой взял её и стоит у воды, он ловит рыбу. Рыбак меняет рыбу на фитиль.

Игровой цикл. Осмотреть берег → найти предмет → применить его или обменять → открыть новый путь.

Сюжет. Смотритель заболел и попросил внука присмотреть за маяком. Вечером по радио объявляют шторм. Концовка: маяк загорается, и лодка рыбака успевает вернуться к берегу.

Персонажи. Внук смотрителя – главный герой, им управляет игрок. Рыбак дядя Толя сидит на причале; если герой отдаёт ему рыбу, рыбак отдаёт фитиль. Чайка Гоша – хитрая и очень шумная птица.

Управление. W, A, S, D – ходьба, E – подобрать или отдать предмет, Esc – пауза.

Дано: фрагмент ГДД; чек-лист из 8 пунктов (раздел 2.5).

Найти: номера нарушенных пунктов.

Решение

Шаг 1. Идём по чек-листу сверху вниз.

- Пункт 1. Жанр – квест, для кого – 10–13 лет. Есть.
- Пункт 2. Победа – «фонарь маяка зажжён». Есть.
- Пункт 3. Поражение – «шторм начался, а фонарь не горит». Есть.
- Пункт 4 проверим отдельно, в шаге 2.
- Пункт 5. Игровой цикл описан. Есть.

- Пункт 6. Завязка (зритель заболел, объявили шторм) и концовка (маяк загорается) есть.
- Пункт 7. Персонажи: у внука – управляет игрок; у рыбака – «если отдать рыбу, отдаёт фитиль», это поведение. У чайки – только характер: «хитрая и шумная». Что она делает и когда – не сказано. **Нарушен.**
- Пункт 8. Клавиши названы. Есть.

Шаг 2. Пункт 4: всё ли для победы можно получить. Для победы нужны три предмета. Для каждого ищем, где его взять.

Таблица 2.3

Что нужно	Где взять	Что для этого нужно
Керосин	в сарае	ключ – на гвозде у причала
Фитиль	у рыбака	рыба – ловится удочкой с причала
Спички	нигде не сказано	–

Про спички в ГДД нет ни слова, кроме «Цели». Ниточка без второго конца – **пункт 4 нарушен.**

Шаг 3. Ответ. Нарушены пункты 4 и 7, по возрастанию – 47.

Глазами программиста. Представь, что игру делают по этому ГДД. Программист сразу спросит: «Где лежат спички?» и «Что делает чайка?» Ровно два вопроса – ровно два пропуска.

 **Типичная ошибка.** Посчитать пункт 7 выполненным, раз про чайку «что-то написано». Характер – не поведение: по словам «хитрая и шумная» программист не сможет запрограммировать чайку. С пунктом 4 другая беда: его проверяют «на глаз». Спички упомянуты в «Цели», и кажется, что всё на месте. Выручает только таблица «что нужно – где взять».

Ответ: 47.

Своя задача по программе сферы, модуль 1

Реши сам

2.9. Проверь фрагмент ГДД по чек-листу из раздела 2.5. Какие пункты нарушены? Номера – по возрастанию, без пробелов.

Дачный переполох.

Концепция. Головоломка для детей 8–11 лет. Внучка Соня осталась одна на даче и должна успеть спасти огород.

Цель. Победа – до заката полить все 6 грядок. Поражение – солнце село, а хотя бы одна грядка сухая.

Механики. Воду берут из колодца, колодец заперт на замок. Полной лейки хватает на 2 грядки, воды в колодце сколько угодно.

Игровой цикл. Набрать воду → донести до грядки → полить → вернуться за водой.

Сюжет. Бабушка звонит из города и просит полить огород до заката: вечером приедут гости.

Персонажи. Соня – героиня, ей управляет игрок. Кот Барсик лежит на крыльце. Если Соня пробегает мимо крыльца с полной лейкой, кот бросается под ноги, и половина воды проливается.

Управление. Щелчок мышью по месту – Соня идёт туда, щелчок по грядке – полить её.

2.10. Проверь ещё один фрагмент ГДД по тому же чек-листу. Какие пункты нарушены? Номера – по возрастанию, без пробелов.

Снежная крепость.

Концепция. Аркада для детей 9–12 лет: игрок строит во дворе снежную крепость.

Цель. Победа – на крепость уложены все 5 снежных кирпичей и воткнут флажок.

Механики. Снежный кирпич получается, если прокатить снежный ком по сугробу 3 секунды. Флажок лежит на скамейке у подъезда.

Уровни. Во дворе 4 сугроба. Каждого сугроба хватает на один кирпич, потом он исчезает.

Игровой цикл. Найти сугроб → скатать ком → донести кирпич до крепости → искать следующий сугроб.

Сюжет. Завязка: в первый снежный день ребята из соседнего двора вызвали героя на снежную битву. Концовка: крепость достроена, и над ней поднят флажок.

Персонажи. Пёс Барон бежит по двору. Если игрок несёт кирпич и Барон оказался рядом, пёс прыгает, и кирпич разбивается.

Управление. Стрелки – ходьба, пробел – поднять или положить предмет.

Задача 2.6 Где уходят игроки

плейтест, проценты, доля от дошедших

Команда провела плейтест своего платформера на 60 школьников. Начали игру все 60. Команда записала, сколько игроков прошли каждый этап (кто не прошёл этап, дальше не играл).

Таблица 2.4

Этап	1. Обучение	2. Уровень 1	3. Уровень 2	4. Уровень 3	5. Уровень 4
Прошли этап	48	40	34	24	21

а) На каком этапе уходит наибольшая доля игроков из тех, кто до этого этапа дошёл? Ответ – номер этапа. б) Сколько процентов дошедших до этого этапа на нём ушли? Округли до целых.

Дано: начали 60 игроков; прошли этапы: 48, 40, 34, 24, 21.

Найти: номер этапа с наибольшей долей ушедших; эту долю в %.

Решение

Шаг 1. Сколько игроков дошли до каждого этапа. До этапа доходят те, кто прошёл предыдущий. До обучения – все 60, до уровня 1 – 48, до уровня 2 – 40, до уровня 3 – 34, до уровня 4 – 24.

Шаг 2. Сколько ушли на каждом этапе. Дошли минус прошли: $60 - 48 = 12$; $48 - 40 = 8$; $40 - 34 = 6$; $34 - 24 = 10$; $24 - 21 = 3$.

Шаг 3. Доля ушедших от дошедших.

- Этап 1: $12:60 \cdot 100 = 20\%$.
- Этап 2: $8:48 \cdot 100 = 16,7\%$.
- Этап 3: $6:40 \cdot 100 = 15\%$.
- Этап 4: $10:34 \cdot 100 = 29,4\%$.
- Этап 5: $3:24 \cdot 100 = 12,5\%$.

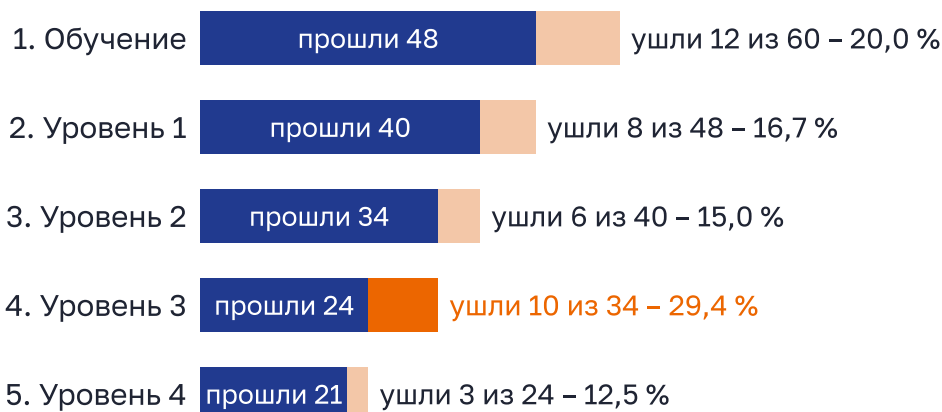


Рисунок 2.9 – Итоги плейтеста: синее – прошли этап, оранжевое – ушли на нём

Шаг 4. Выбираем наибольшую. Больше всех – этап 4 (уровень 3): 29,4 %, после округления до целых – 29 %.

Проверка. $34 \cdot 0,294 = 10,0$ – как раз 10 ушедших, сходится.

Что это значит для команды. На уровне 3 ушёл почти каждый третий из дошедших – примерно вдвое больше, чем на соседних уровнях. Похоже, на уровне 3 что-то непонятное или слишком трудное. Команде стоит пересмотреть записи плейтеста и найти, где там застревали.

❗ Типичная ошибка. Многие выбирают этап 1: там ушло больше всего людей – 12. Но вопрос про долю, а 12 из 60 – это только 20 %. Другой промах – делить на всех 60 игроков: $10:60 = 16,7\%$, и тогда «победит» обучение. Делить нужно на тех, кто до этапа дошёл.

Ответ: а) 4; б) 29 %.

По мотивам задачи заключительного этапа НТО 2025/26, профиль «Разработка компьютерных игр» (8–11 классы), «UX и метрики»

Реши сам

2.11. В плейтесте участвовали 25 человек, и только 8 из них нашли ключ без подсказки. а) Какой процент игроков нашёл ключ? б) Ключ перенесли ближе к тропинке и провели новый плейтест: ключ нашли 21 игрок из 28. Какой процент нашёл ключ теперь?

2.12. Кружок выложил в школьный чат онлайн-квест про Академгородок. Начали его 80 человек. Кто не прошёл этап, дальше не играл.

Таблица 2.5

Этап	1. Музей	2. Парк	3. Библиотека	4. Обсерватория
Прошли этап	60	52	45	30

а) На каком этапе ушла наибольшая доля игроков из тех, кто до него дошёл? Ответ – номер этапа. б) Сколько процентов дошедших до этого этапа на нём ушли? Округли до целых.

Задача 2.7* Улучшать шахту или копать?

время до цели, окупаемость, сравнение путей

В стратегии нужно построить замок за 900 монет. Шахта приносит 30 монет в конце каждой минуты. Шахту можно улучшать, но только по порядку. Первое улучшение стоит 240 монет и поднимает доход до 50 монет в минуту. Второе стоит 400 монет и поднимает доход до 80 монет в минуту. Улучшение можно купить в любой момент, когда хватает монет, и оно работает уже со следующей минуты. В начале у игрока 0 монет. За какое наименьшее число минут можно построить замок?

Дано: замок – 900 монет; доход 30 монет/мин; улучшение 1 – 240 монет, доход 50 монет/мин; улучшение 2 – 400 монет, доход 80 монет/мин; в начале 0 монет.

Найти: наименьшее время до замка, мин.

Решение

Шаг 1. Что происходит. Есть три пути: А – не улучшать шахту совсем, Б – купить только первое улучшение, В – купить оба. Считаем время для каждого пути отдельно и сравниваем. Покупать улучшение выгодно сразу, как только хватает монет: чем раньше оно работает, тем больше приносит.

Шаг 2. Путь А. $900:30 = 30$ мин.

Шаг 3. Путь Б. Копим на улучшение: $240:30 = 8$ мин. Покупаем – в кармане снова 0. Копим на замок с новым доходом: $900:50 = 18$ мин. Всего $8 + 18 = 26$ мин.

Шаг 4. Путь В. Первые 8 минут – как в пути Б. Копим на второе улучшение по 50 монет: $400:50 = 8$ мин, всего уже 16 мин, в кармане 0. Копим на замок по 80 монет: $900:80 = 11,25$. Монеты приходят только в конце минуты: после 11 минут будет $11 \cdot 80 = 880$ – мало, после 12 минут – 960, хватает. Значит, 12 мин. Всего $16 + 12 = 28$ мин.

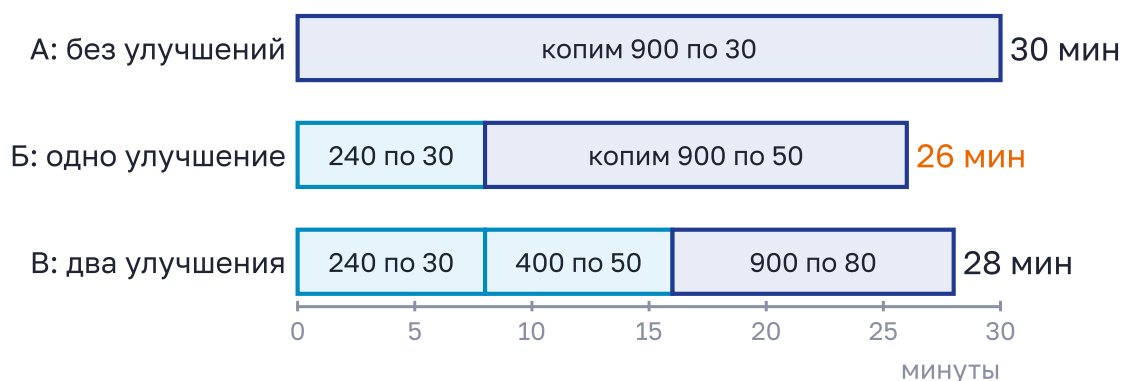


Рисунок 2.10 – Три пути к замку. Светлые куски – копим на улучшение, тёмные – копим на замок

Шаг 5. Сравниваем. А – 30 мин, Б – 26 мин, В – 28 мин. Быстрее всего путь Б: 26 минут.

Шаг 6. Почему второе улучшение не помогло. Посчитаем окупаемость. Первое улучшение прибавляет $50 - 30 = 20$ монет в минуту и окупается за $240:20 = 12$ мин. После его покупки игра идёт ещё 18 мин – улучшение успевает вернуть свою цену с запасом. Второе прибавляет $80 - 50 = 30$ монет в минуту и окупается за $400:30 = 13,3$ мин. А после его покупки до замка остаётся всего 12 минут – меньше срока окупаемости. Улучшение не успевает вернуть свою цену, и путь выходит длиннее.

Прикинем на глаз. Первое улучшение почти вчетверо дешевле замка, а доход поднимает в $50:30 = 1,7$ раза – понятно, что его стоит купить. Второе стоит почти половину замка: такая дорогая покупка выгодна, только если после неё ещё долго играть.

⚠ Типичная ошибка. Решить, что чем больше улучшений, тем быстрее, и ответить 28, не проверив путь Б. Ещё часто забывают вычитать потраченные монеты: «к замку уже есть 240», и тогда $(900 - 240):50 = 13,2$, итого 22 минуты. Но 240 монет ушли на улучшение, в кармане 0. А кто не округлил 11,25 вверх, получит 27,25 минуты, хотя монеты приходят только в конце минуты.

Ответ: 26 мин.

Своя задача по программе сферы, модуль 1

Реши сам

2.13.* На ярмарке грядки приносят 12 монет в конце каждой минуты. Теплица стоит 300 монет и добавляет ещё 20 монет в минуту, работать она начинает сразу. У игрока ровно 300 монет. а) До конца ярмарки 20 минут. На сколько монет больше будет у игрока к концу, если купить теплицу сразу, чем если не покупать её вовсе? б) А если бы до конца оставалось 10 минут?

2.14.* У великана 300 здоровья. Лук наносит 18 урона за выстрел и стреляет каждые 0,5 с. Меч наносит 45 урона за удар и бьёт каждые 1,5 с. И с луком, и с мечом первая атака случается сразу, в момент 0. С каким оружием великан будет побеждён раньше и на сколько секунд?

Левел-дизайн: карты и пути



Левел-дизайнер – это человек, который строит уровень: где стены, где проходы, где лежит ключ и откуда игрок увидит выход. Прежде чем собрать локацию в редакторе, её рисуют на бумаге в клетку. Здесь ты научишься читать такую карту: находить клетку по адресу, вести по ней робота, искать самую короткую дорогу и считать, сколько всего дорог ведёт к замку. В списке «что пригодится на отборе» это пункты про алгоритмы, исполнителей и комбинаторику. Ответ в таких задачах – обычно одно число: шаги, метры, число путей или номер верной программы.

3.1. Карта в клетку и координаты

НА ПАЛЬЦАХ



Рисунок 3.1 – Адрес клетки – два числа: столбец и строка

Чтобы сказать другу, где сундук, не нужно рисовать – достаточно двух чисел.
Первое – столбец, второе – строка.

Любой уровень – даже трёхмерный – сначала рисуют сверху, как план квартиры. Удобнее всего в клетку: одна клетка – один шаг героя или один метр мира. На такой карте у каждой клетки есть **адрес – координаты**. Их записывают в скобках через точку с запятой: (x; y).

- x – номер **столбца**, считаем слева направо;
- y – номер **строки**, считаем снизу вверх, как на координатной плоскости в учебнике математики.

В этой главе клетки нумеруются с единицы: левая нижняя клетка – (1; 1). Сундук на рисунке стоит в клетке (5; 3): пятый столбец слева, третья строка снизу. Порядок важен. (3; 5) – совсем другая клетка, третий столбец и пятая строка.

Совет. В разных программах ноль бывает в разных местах. На экране компьютера строки часто считают **сверху вниз**, а в 3D-редакторах есть третья координата – высота. Прежде чем решать, найди в условии, где клетка (1; 1) или где ноль. Перепутанное направление оси – одна из самых обидных ошибок.

Сколько шагов от клетки до клетки. Пусть герой ходит только по сторонам клеток – вправо, влево, вверх, вниз, без диагоналей – и стен на пути нет. Тогда ему нужно пройти разницу по столбцам плюс разницу по строкам. От героя в (2; 1) до сундука в (5; 3): по столбцам $5 - 2 = 3$ шага, по строкам $3 - 1 = 2$ шага, всего $3 + 2 = 5$ шагов. Идти можно и «лесенкой», и сначала вверх, потом вправо – если стен нет, выйдет те же 5 шагов.

Формулой: **число шагов = разница по x + разница по y**. Разницу берём «без минуса»: из большего вычитаем меньшее. Математики пишут это вертикальными чёрточками: $|2 - 5| = 3$.

$$\text{шагов} = |x_2 - x_1| + |y_2 - y_1|.$$

Если герою **можно ходить и по диагонали**, и диагональный шаг занимает столько же времени, сколько обычный, считать надо иначе. Каждый диагональный шаг сдвигает героя сразу и по x , и по y . Поэтому шагов нужно столько, сколько **большая** из двух разниц. От $(2; 1)$ до $(5; 3)$ – всего 3 шага: два по диагонали и один вправо.

Масштаб. Клетка на плане – это кусок настоящего (игрового) мира. Если клетка – 2 метра, то 5 шагов по клеткам – это $5 \cdot 2 = 10$ метров. А если герой бежит со скоростью 4 метра в секунду, на дорогу уйдёт $10 : 4 = 2,5$ секунды.

Запомни. Координаты $(x; y)$: сначала столбец, потом строка. Без диагоналей шагов столько, сколько разница по x плюс разница по y . С диагоналями – сколько большая из двух разниц. Клетки в метры – умножением на размер клетки.

3.2. Исполнитель-робот: команды движения

НА ПАЛЬЦАХ

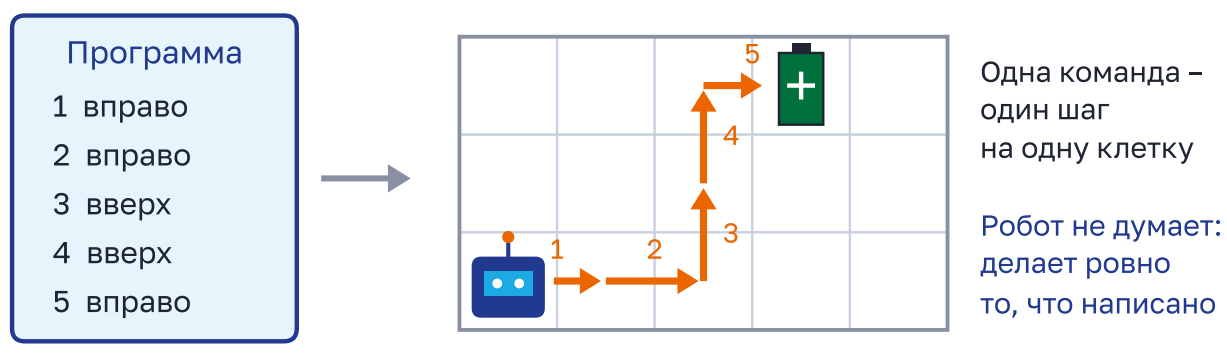


Рисунок 3.2 – Программа – это список команд. Робот выполняет их по порядку, ровно одну за другой

Робот не видит батарейку и не знает, куда ему надо. Он просто делает шаг за шагом то, что написано в карточке.

Исполнитель – это тот, кто выполняет команды: робот на клетчатом поле, персонаж игры, NPC. У исполнителя есть короткий список команд, которые он понимает. **Программа** (или **алгоритм**) – это список таких команд по порядку.

У нашего робота четыре команды: **вверх**, **вниз**, **влево**, **вправо**. Каждая сдвигает его ровно на одну клетку. Команда **вправо** прибавляет к x единицу, **влево** – отнимает. Команда **вверх** прибавляет единицу к y , **вниз** – отнимает.

Ещё есть команда **повторения**, или **цикл**: **повтори 3 [вправо, вверх]**. Всё, что в квадратных скобках, робот выполнит 3 раза подряд. Такая запись заменяет шесть команд: **вправо, вверх, вправо, вверх, вправо, вверх**. Короче писать и меньше шансов ошибиться.

Главное правило исполнителя: **он не думает**. Если команда ведёт в стену, робот врежется и останавливается – программа ломается. Он не обойдёт стену сам и не догадается, что «имелось в виду». Поэтому проверять программу надо так же послушно: пальцем по клеткам, по одной команде.

В «Урсуле» персонажей тоже программируют командами. На вебинаре сферы показывали модуль движения: персонажу можно сказать, на сколько метров пройти вперёд или вправо, или выбрать случайную точку в заданном радиусе, чтобы он «патрулировал». Там мир трёхмерный и шаги в метрах, но смысл тот же, что у нашего робота: команда – сдвиг, программа – цепочка сдвигов.

Запомни. Каждая команда движения меняет одну координату на единицу. Чтобы узнать, где окажется робот, проще всего сложить сдвиги: сколько всего раз вправо минус сколько раз влево – это изменение x ; вверх минус вниз – изменение y . А вот врежется ли он в стену, по сумме не узнать: тут придётся пройти путь клетка за клеткой.

Совет. Цикл раскрывай на черновике полностью: выпиши команды столбиком и пронумеруй. Так легко найти, на какой по счёту команде что-то случилось, и не забыть повторить тело цикла нужное число раз.

3.3. Кратчайший путь по клеткам: волна

НА ПАЛЬЦАХ

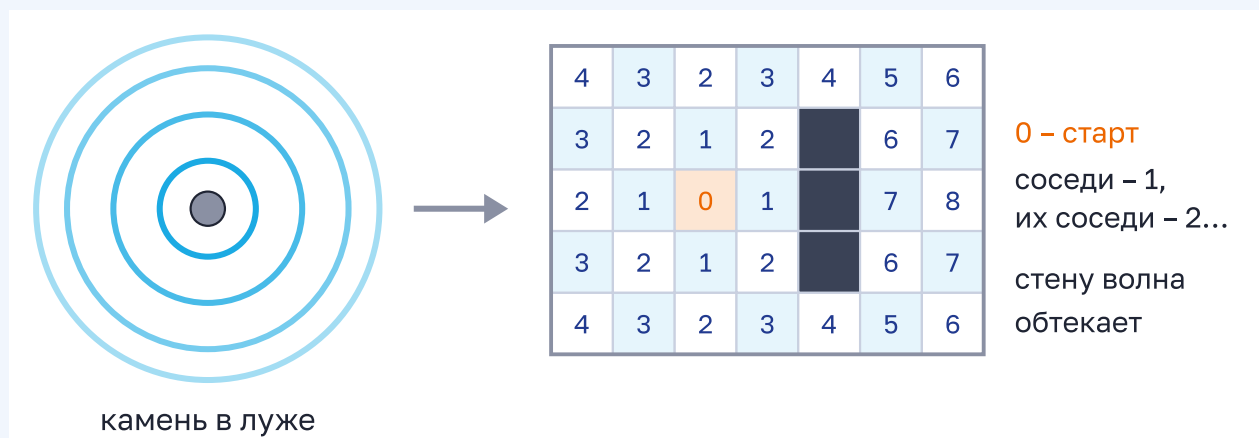


Рисунок 3.3 – Брось камень в лужу – круги расходятся во все стороны. Так же расходятся по клеткам числа

Число в клетке – сколько шагов до неё от старта. Стена волну не пропускает, и она обходит её сбоку.

Когда на карте есть стены, формула «разница по x + разница по y » уже не работает: путь приходится огибать. Как найти самый короткий, не перебирая все дороги подряд? Есть надёжный способ – **волновой алгоритм**. Его придумали для печатных плат – чтобы прокладывать дорожки между деталями. В играх похожим способом враг находит дорогу к герою в обход стен.

1. В клетку старта пиши 0.
2. Во все свободные соседние клетки (по сторонам, не по диагонали) пиши 1.

3. В ещё пустых соседей клеток с единицей пиши 2. Дальше 3, 4, 5... Каждый раз – на единицу больше.
4. Как только волна дошла до финиша, число в нём – это **длина кратчайшего пути** в шагах.
5. Сам путь находим с конца: от финиша идём в соседнюю клетку с числом на 1 меньше, и так до нуля.

Волна честная: она расходится одновременно во все стороны и потому первой приходит в любую клетку именно по кратчайшей дороге. Если в клетке уже есть число, второй раз её не трогаем – туда уже пришли быстрее.

Запомни. Волна: старт – 0, соседи – 1, их соседи – 2... Число в финише – длина кратчайшего пути. Путь – обратным ходом, по числам на 1 меньше. Клетку, где число уже стоит, не переписываем.

Совет. Пиши числа карандашом прямо на карте. Если волна зашла в тупик, это видно сразу: дальше писать некуда. Тупики не стирай – они нужны, чтобы не заходить туда второй раз.

3.4. Карта как граф: комнаты, проходы, ключи

НА ПАЛЬЦАХ

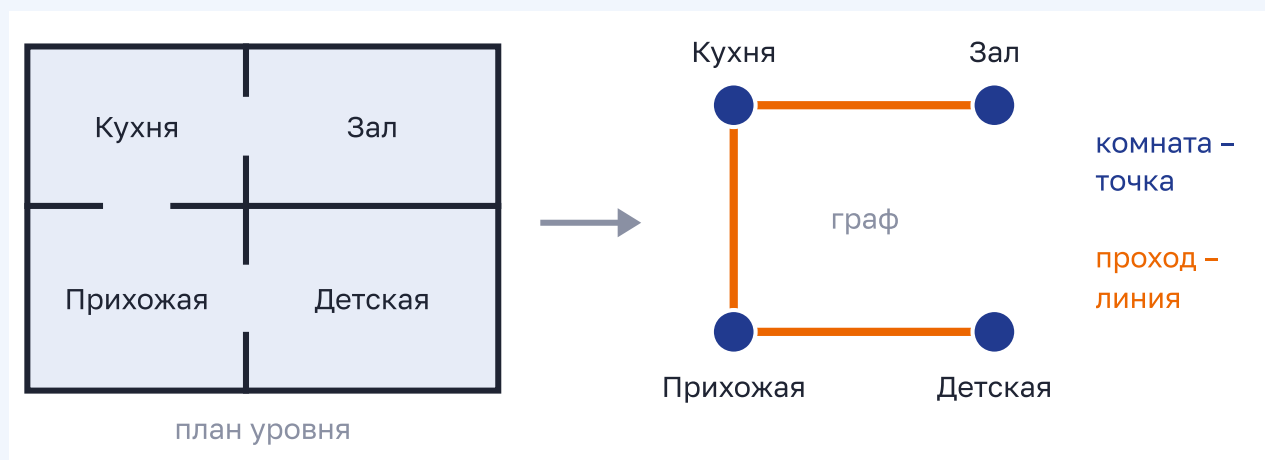


Рисунок 3.4 – Точные размеры комнат не важны, важно, откуда куда можно пройти

Комната превращается в точку, дверной проём – в линию между точками. Получается граф.

Посмотри на схему новосибирского метро. На ней нет ни домов, ни улиц – только кружки-станции и линии между ними. Зато сразу понятно, как доехать и где пересесть: с «Красного проспекта» на «Сибирскую». Такая схема называется **граф**. Точки графа – **вершины**, линии – **рёбра**.

Геймдизайнеры делают так же. Большой уровень – замок, школа, подземелье – рисуют как граф: вершина – комната, ребро – проход. Над ребром можно подписать длину прохода в мет-

рах или время в секундах. Тогда граф отвечает на вопросы: можно ли вообще дойти, какой путь короче, куда игрок не попадёт никогда.

Кратчайший путь по графу с длинами ищут метками. Это та же волна, только шаги разной длины.

1. У старта пиши 0. У соседей старта пиши длину прохода до них.
2. Выбери среди **ещё не закреплённых** комнат ту, где число меньше всех, и **закрепи** её: лучше этого числа уже не найдётся.
3. От закреплённой комнаты посчитай пути до её соседей: число в ней плюс длина прохода. Если вышло меньше, чем уже написано у соседа, зачеркни старое и напиши новое.
4. Повторяй шаги 2–3, пока не закрепишь финиш.

Осторожно: самый короткий путь не обязательно проходит через меньше всего дверей. Три длинных коридора могут оказаться дольше пяти коротких.

Ключи и двери. В квестах часть рёбер заперта. Красную дверь открывает только красный ключ, а он лежит где-то в другой комнате. Получается порядок прохождения: сначала дойти до ключа, потом – к двери. Найти этот порядок несложно. Отметь все комнаты, куда можно попасть **без ключей**, и посмотри, какие ключи в них лежат: они открывают новые двери. За дверями – новые комнаты и новые ключи. Снова открывай двери – и так, пока не дойдёшь до выхода.

Если на каком-то шаге новых ключей не прибавилось, а выход не отмечен, – уровень не пройти. Самый частый такой баг – ключ лежит **за той самой дверью, которую открывает**. Хороший левел-дизайнер проверяет это до того, как отдать уровень игрокам.

Запомни. Граф: комната – вершина, проход – ребро, над ребром длина. Кратчайший путь – метками: закрепляй комнату с самым маленьким числом и обновляй соседей. Ключи – по кругам: куда можно дойти → какие ключи там → какие двери они открывают.

3.5. Сколько путей из угла в угол

НА ПАЛЬЦАХ

		финиш
1	3	6
1	2	3
1	1	1
старт		

Ходить можно только
вправо и вверх

В клетку приходят слева или снизу:

$$3 + 3 = 6 \text{ путей}$$

У нижнего и левого края – по 1 пути

Рисунок 3.5 – Каждая клетка складывает то, что пришло к ней слева и снизу

Ходим только вправо и вверх. В клетку можно попасть двумя способами – значит, путей к ней столько, сколько к двум её соседям вместе.

Иногда геймдизайнеру нужен не самый короткий путь, а **все** пути. Сколько разных маршрутов у игрока? Не слишком ли уровень «коридорный»? Сколько вариантов перебрать тестировщику? Это задача **комбинаторики** – науки о том, как считать варианты, не выписывая каждый.

Возьмём поле в клетку и разрешим ходить **только вправо и вверх** – например, герой всё время поднимается в гору и идёт к востоку. Как попасть в какую-то клетку? Только из клетки слева или из клетки снизу. Отсюда правило:

Число путей в клетку = число путей в клетку слева + число путей в клетку снизу.

- В стартовой клетке пишем 1: там мы уже стоим, это один «путь».
- В клетках нижней строки – по 1: туда можно прийти только идя вправо, другого пути нет.
- В клетках левого столбца – тоже по 1: туда только вверх.
- Остальные клетки заполняем по правилу, строка за строкой снизу вверх, в каждой строке слева направо.

На рисунке «на пальцах» поле 3×3 : в правой верхней клетке $3 + 3 = 6$ путей. Для поля 4×3 (4 столбца, 3 строки) выйдет 10. Проверь сам, заполнив табличку.

Если клетка запрещена – там лава, яма, стена, – в ней пишем 0: через неё не пройдёт ни один путь. Дальше считаем по тому же правилу. Но следи за краем поля: если лава стоит в нижней строке, то все клетки этой строки правее лавы тоже получают 0 – попасть в них можно было только через лаву. С левым столбцом так же: выше лавы – нули.

у				
3	1	1	2	4
2	1	0	1	2
1	1	1	1	1
	1	2	3	4
	х			

Лавы в клетке (2; 2): через неё
путей нет – пишем 0

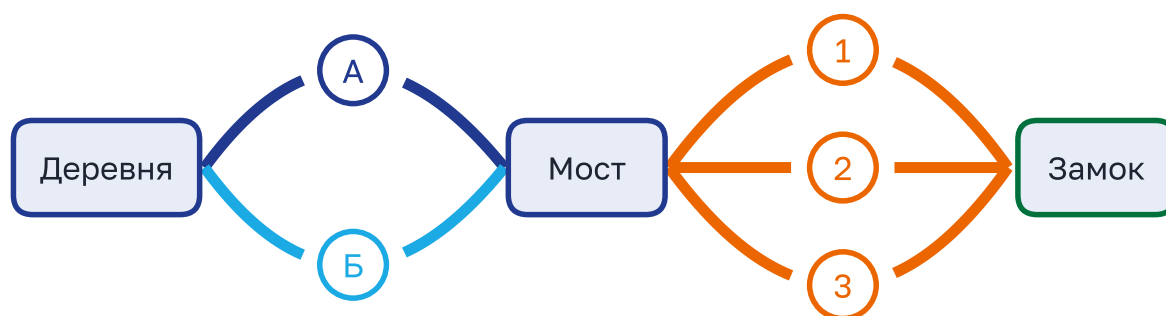
Без лавы было бы 10 путей,
с лавой – только 4

Рисунок 3.6 – Поле 4 × 3 с лавой в клетке (2; 2). Ходим только вправо и вверх

Одна клетка лавы в поле 4 × 3 уничтожила 6 путей из 10. Вот почему одна яма в нужном месте так сильно меняет уровень.

Правило умножения.

НА ПАЛЬЦАХ



2 дороги · 3 дороги = 6 маршрутов

A1, A2, A3, B1, B2, B3

Рисунок 3.7 – Из деревни к мосту ведут 2 дороги, от моста к замку – 3. Всего маршрутов $2 \cdot 3 = 6$

К каждой дороге до моста подходит любая из трёх дорог после него. Поэтому варианты не складывают, а умножают.

Маршрут на рисунке выбирают в два приёма: сначала дорогу до моста, потом дорогу после. Первую можно выбрать двумя способами, и у каждой есть три продолжения: $2 \cdot 3 = 6$. Проверь: A1, A2, A3, B1, B2, B3 – шесть. Так работает **правило умножения**: если выбор делают по шагам и число вариантов на шаге не зависит от того, что выбрали раньше, числа вариантов перемножают. С одеждой то же самое: 3 футболки и 2 пары джинсов дают $3 \cdot 2 = 6$ разных нарядов.

Если путь обязан пройти через клетку (там рычаг, сундук, контрольная точка), эта клетка работает как мост. Считаем пути от старта до неё, потом пути от неё до финиша – как будто она новый старт – и перемножаем. Для поля 5 × 4 без препятствий путей 35. Через клетку (3; 2): до неё 3 пути, от неё до финиша 6, всего $3 \cdot 6 = 18$.

Запомни. Только вправо и вверх: клетка = слева + снизу. На нижнем и левом краю – по 1, но только до первой запрещённой клетки. Запрещённая клетка – 0. Если выбор идёт по шагам, варианты шагов перемножают: так считают путь через обязательную клетку.

Совет. Заполни табличку без препятствий и посмотри на неё по косым линиям – увидишь ряды 1, 2, 1, потом 1, 3, 3, 1, потом 1, 4, 6, 4, 1. Это **треугольник Паскаля**, его называли по имени французского учёного Блеза Паскаля. Если у тебя ряды вышли другими, где-то ошибка сложения.

3.6. Внимание игрока: ориентиры, свет, «хлебные крошки»

НА ПАЛЬЦАХ



Рисунок 3.8 – Уровень ведёт игрока к выходу без единого слова: ориентир, свет, дорожка из монет

Куда ты пойдёшь, если впереди развилка, а справа видна башня и горит свет?
Туда и пойдёшь. Это и есть работа левел-дизайнера.

На установочном вебинаре сферы левел-дизайн объясняли так: это расстановка объектов на сцене, чтобы они **проводили героя сквозь себя**, а не были просто препятствиями. Хороший уровень понятен без подсказок: игрок сам догадывается, куда идти, и ему кажется, что он всё нашёл сам. Для этого у левел-дизайнера есть простые приёмы.

- **Ориентир** – заметный издали объект: башня, огромное дерево, гора, мельница с красной крышей. Игрок видит его с разных мест и понимает, где он и куда идти. В Новосибирске такой ориентир – купол Оперного театра.
- **Свет** – глаз сам тянется к светлomu. Освещённый проход, фонарь у нужной двери, костёр у выхода – а всё второстепенное в полутени.
- **Цвет-подсказка** – всё, с чем можно что-то сделать, окрашено одинаково: например, уступы, за которые можно зацепиться, – жёлтые. Один раз понял – дальше узнаёшь такие места сразу.

- **«Хлебные крошки»** – цепочка мелких предметов вдоль пути: монеты, кристаллы, следы, светлячки. Название из сказки, где дети бросали крошки, чтобы найти дорогу домой. Игрок собирает их – и сам не замечает, как пришёл куда надо.
- **Линии** – дорога, река, забор, рельсы тянут взгляд вдоль себя.
- **Безопасное обучение** – новое препятствие сначала встречается там, где ошибиться не страшно: первый враг – на ровной широкой площадке, первая яма – неглубокая.

А ещё левел-дизайнер следит, чтобы приёмы не ввали. Монетки, которые ведут к обрыву, или лестница того же серого цвета, что и стена, – это ловушка не для героя, а для игрока, и он элится на игру.

Запомни. Игрок идёт туда, где свет, где виден ориентир и куда ведут «хлебные крошки». Одинаковые коридоры, темнота везде и подсказки текстом каждые десять секунд – признаки слабого уровня.

3.7. Разбираем задачи

Задача 3.1 Тропинка к колодцу

разминка · координаты, шаги по клеткам, масштаб

На плане деревни каждая клетка – 2 м на 2 м. Герой стоит в клетке (2; 3), колодец – в клетке (9; 7). Ходить можно только по сторонам клеток, без диагоналей, стен и заборов на пути нет. Сколько метров пройдёт герой до колодца по кратчайшему пути? Ответ – целое число.

Дано: герой (2; 3); колодец (9; 7); клетка – 2 м; ходы только по сторонам.

Найти: длину пути в метрах.

Решение

Шаг 1. Разница по столбцам. x меняется от 2 до 9: $9 - 2 = 7$ шагов вправо.

Шаг 2. Разница по строкам. y меняется от 3 до 7: $7 - 3 = 4$ шага вверх.

Шаг 3. Всего шагов. Без диагоналей шаги складываются: $7 + 4 = 11$ шагов. В каком порядке их делать – неважно, стен нет.

Шаг 4. Шаги в метры. Один шаг – одна клетка, то есть 2 м: $11 \cdot 2 = 22$ м.

22 м – это много? Колодец на 7 клеток правее и на 4 выше – это 14 м вбок и 8 м вперёд. Без диагоналей путь равен сумме этих отрезков, 22 м. Примерно столько от подъезда до детской площадки.

 **Типичная ошибка.** Считать клетки, а не шаги, и учесть стартовую: «клетки со 2-й по 9-ю – это 8 клеток». Так получится $8 + 5 = 13$ клеток и 26 м. Но в стартовой клетке герой уже стоит, её проходить не надо. Проверка на пальцах: из клетки 2 в клетку 3 – это один шаг, а не два.

Ответ: 22 м.

Реши сам

3.1. На карте лагеря одна клетка – 1,5 м. Вожатый стоит в клетке (3; 8), флагшток – в клетке (10; 1). Ходить можно только по сторонам клеток, препятствий нет. Сколько метров до флагштока по кратчайшему пути? Ответ – число.

3.2. Рыцарь стоит в клетке (2; 2), портал – в клетке (9; 5). Каждый шаг на соседнюю клетку занимает 0,5 с. Сначала рыцарь умел ходить только по сторонам клеток, а после улучшения научился ходить и по диагонали (диагональный шаг – тоже 0,5 с). На сколько секунд быстрее он теперь доберётся до портала? Препятствий нет. Ответ – десятичная дробь.

Задача 3.2 Какая программа доведёт робота

исполнитель, команды движения, цикл «повтори»

Робот стоит в клетке (1; 1) и должен добраться до батарейки в клетке (6; 4). Тёмные клетки – стены: если команда ведёт в стену, робот врежется и останавливается. За край поля выходить тоже нельзя.

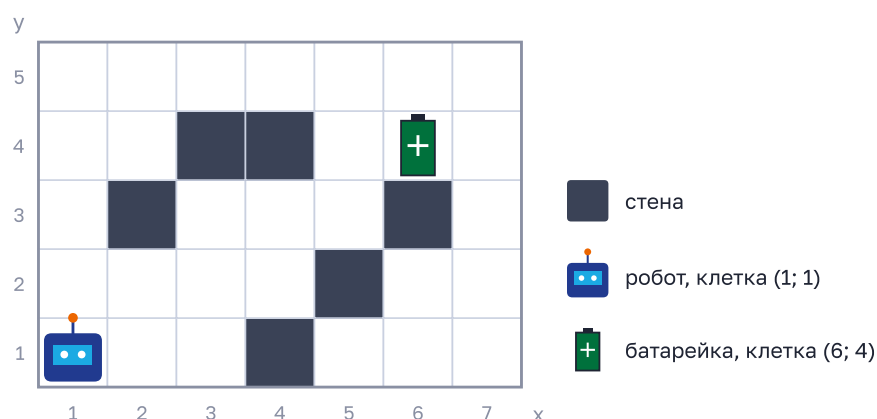


Рисунок 3.9 – Поле робота к задаче 3.2

Какая из четырёх программ приведёт робота к батарейке? Ответ – номер программы.

- Программа 1: повтори 5 [вправо]
повтори 3 [вверх]
- Программа 2: повтори 3 [вверх]
повтори 5 [вправо]
- Программа 3: вверх
повтори 2 [вправо, вправо, вверх]
вправо
- Программа 4: вверх
повтори 2 [вправо, вправо, вверх]

Дано: старт (1; 1); цель (6; 4); стены – по рисунку; четыре программы.

Найти: номер программы, которая приводит робота к цели.

Решение

Шаг 1. Быстрая проверка сдвигом. Чтобы попасть из (1; 1) в (6; 4), всего нужно $6 - 1 = 5$ шагов вправо и $4 - 1 = 3$ шага вверх. Считаем в каждой программе:

- программа 1: 5 вправо, 3 вверх – годится по сумме;
- программа 2: 3 вверх, 5 вправо – годится по сумме;
- программа 3: $1 + 2 \cdot 1 = 3$ вверх, $2 \cdot 2 + 1 = 5$ вправо – годится;
- программа 4: $1 + 2 = 3$ вверх, $2 \cdot 2 = 4$ вправо – одного шага вправо не хватает. Робот закончит в (5; 4), рядом с батареей, но не на ней. Программа 4 не подходит.

Сумма сдвигов показывает только, где робот **окажется**, если ни во что не врежется. Стены проверяем дальше.

Шаг 2. Программа 1 – по клеткам. Команды вправо: (2; 1), (3; 1), а следующая клетка (4; 1) – стена. Робот врежется уже на третьей команде. Не подходит.

Шаг 3. Программа 2 – по клеткам. Вверх: (1; 2), (1; 3), (1; 4) – свободно. Вправо: (2; 4), а дальше (3; 4) – стена. Врезается на пятой команде. Не подходит.

Шаг 4. Программа 3 – по клеткам. Раскроем цикл и пронумеруем команды:

Таблица 3.1

№	Команда	Клетка после неё
1	вверх	(1; 2)
2	вправо	(2; 2)
3	вправо	(3; 2)
4	вверх	(3; 3)
5	вправо	(4; 3)
6	вправо	(5; 3)
7	вверх	(5; 4)
8	вправо	(6; 4)

Сверяемся с рисунком: ни одна из этих клеток не стена. После восьмой команды робот в (6; 4) – на батарее.

 **Типичная ошибка.** Проверить только сумму сдвигов и выбрать программу 1 – она самая «прямая» и сумма у неё верная. Но робот не обходит стены, он в них врежется. Бывает и так: тело цикла выполнили один раз вместо двух. Тогда программа 3 приведёт робота в (4; 3), и покажется, что верной программы нет вовсе.

Ответ: 3.

Своя задача по программе сферы, модуль 2

Реши сам

3.3. На поле нет стен. Робот стоит в клетке (2; 3) и выполняет программу:

```
вправо  
повтори 4 [вверх, вправо, вправо]  
вниз  
вниз
```

В какой клетке он окажется? Ответ запиши двумя числами: сначала x , потом y .

3.4. Карта: # – стена, . – свободная клетка, Р – робот, Б – батарейка.

```
#####  
#...#...#  
#.#.#.#.  
#.#...#.#  
#Р..#..Б#  
#####
```

Программист написал для робота программу:

```
повтори 2 [вправо]  
вверх  
повтори 2 [вправо]  
повтори 2 [вверх]  
повтори 2 [вправо, вниз]  
вниз
```

Робот врезался в стену. На какой по счёту команде это случилось, если считать каждую команду внутри цикла отдельно? Ответ – номер команды.

Задача 3.3 Самый короткий путь из подземелья

лабиринт, кратчайший путь, волновой алгоритм

Герой стоит в клетке С, выход – клетка Ф. Ходить можно на соседнюю клетку по стороне, тёмные клетки – стены. Какое наименьшее число шагов нужно, чтобы дойти от С до Ф? Ответ – целое число.

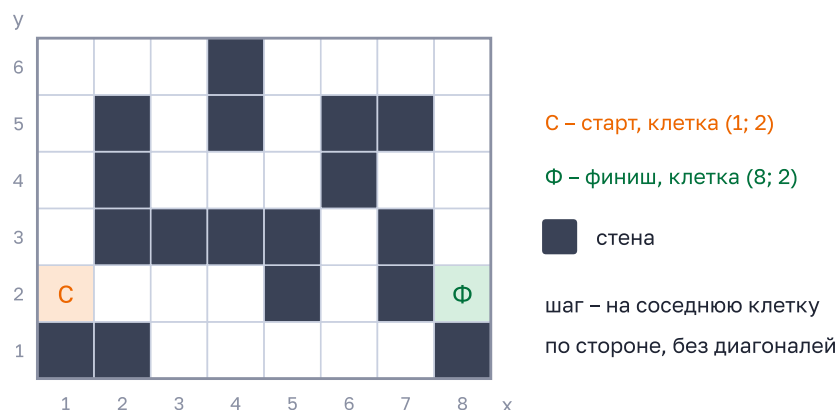


Рисунок 3.10 – Лабиринт к задаче 3.3

Дано: карта 8×6 ; старт (1; 2); финиш (8; 2); ходы по сторонам.

Найти: длину кратчайшего пути в шагах.

Решение

Шаг 1. Прикинем на глаз. С и Ф стоят в одной строке, между ними $8 - 1 = 7$ столбцов. Если бы стен не было, хватило бы 7 шагов. Но в строках 2 и 3 стоят стены, прямой дороги нет. Считаем волной.

Шаг 2. Запускаем волну. В С пишем 0. Соседи С – клетки (2; 2) и (1; 3), в них 1. Дальше волна идёт двумя рукавами.

- **Нижний рукав:** (3; 2) – 2, (4; 2) и (3; 1) – 3, по нижней строке 4, 5, 6, 7 до клетки (7; 1). Из (6; 1) волна поднимается в (6; 2) – 7 и (6; 3) – 8. Дальше писать некуда: все соседи – стены. Это **тупик**, хотя он ближе всех к финишу.
- **Верхний рукав:** по левому краю вверх 1, 2, 3, 4 до угла (1; 6), вправо 5, 6, вниз в (3; 5) – 7, (3; 4) – 8, вправо 9, 10 до (5; 4), вверх 11, 12 до (5; 6), вправо по верхней строке 13, 14, 15, вниз по правому краю 16, 17, 18 и в Ф – **19**.

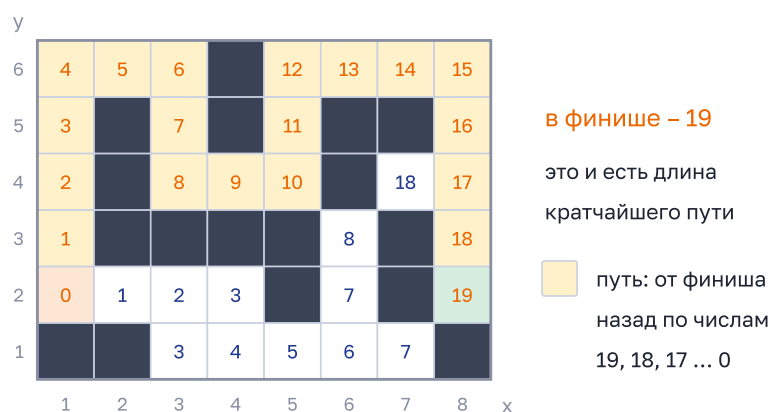


Рисунок 3.11 – Волна в лабиринте задачи 3.3. Жёлтым – кратчайший путь

Шаг 3. Читаем ответ. В клетке Ф стоит 19 – значит, кратчайший путь занимает 19 шагов. Обратным ходом (19, 18, 17 ... 0) находим сам путь: он показан жёлтым.

Почему так много? Прямо было бы 7 шагов, но героя заставляют обойти почти весь лабиринт по краю: подняться на 4 строки, пройти поверху и спуститься обратно. Вот и выходит 19 – по-

что втрое больше: главный проход в этом лабиринте идёт кругом.

❗ **Типичная ошибка.** Поверить «короткой» дороге низом: она ведёт к финишу ближе всех, клетка (7; 1) в двух шагах от Ф. Но (7; 2) и (8; 1) – стены, а по диагонали ходить нельзя. И ещё промах: переписать клетку, куда уже пришло число из другого рукава волны. Клетка получает только **первое**, самое маленькое число.

Ответ: 19.

Своя задача по программе сферы, модуль 1

Реши сам

3.5. Карта: # – стена, . – свободная клетка, С – старт, Ф – финиш. Ходить можно по сторонам клеток. Какое наименьшее число шагов от С до Ф? Ответ – целое число.

```
..#.....  
..#.##..  
С.#..#.Ф  
..##.##..  
.....
```

3.6. Выход Ф открывается только ключом К. Сначала нужно дойти до ключа, а потом – до выхода. Ходить можно по сторонам клеток. Какое наименьшее число шагов нужно на весь путь С → К → Ф? Ответ – целое число.

```
С...#....  
.##..#.#.  
.#К#...#.  
.#.###.#.  
.....Ф
```

Задача 3.4 Короткая дорога в сокровищницу

граф, кратчайший путь по рёбрам с длинами

План замка нарисован как граф: кружки – комнаты, линии – коридоры, над линиями – длины коридоров в метрах. Какова длина самого короткого пути от Входа до Сокровищницы? Ответ – целое число метров.

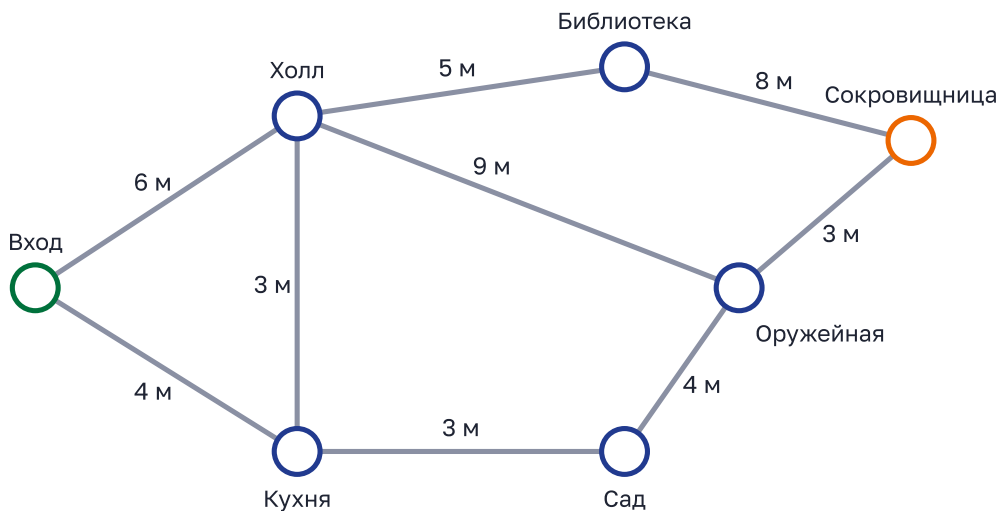


Рисунок 3.12 – План замка к задаче 3.4

Дано: 7 комнат и 9 коридоров с длинами (по рисунку).

Найти: длину кратчайшего пути Вход → Сокровищница, м.

Решение

Шаг 1. Старт. У Входа пишем 0 и закрепляем. Соседи: Холл – $0 + 6 = 6$, Кухня – $0 + 4 = 4$.

Шаг 2. Самое маленькое незакреплённое число – 4, Кухня. Закрепляем. От Кухни: до Холла $4 + 3 = 7$ – больше, чем уже написанные 6, оставляем 6. До Сада $4 + 3 = 7$ – пишем 7.

Шаг 3. Дальше самое маленькое – 6, Холл. Закрепляем. От Холла: Библиотека $6 + 5 = 11$, Оружейная $6 + 9 = 15$.

Шаг 4. Самое маленькое – 7, Сад. Закрепляем. От Сада до Оружейной $7 + 4 = 11$ – это меньше, чем 15. Зачёркиваем 15, пишем 11.

Шаг 5. Библиотека (11) и Оружейная (11). От Библиотеки до Сокровищницы $11 + 8 = 19$. От Оружейной $11 + 3 = 14$ – меньше. У Сокровищницы остаётся 14, и меньше этого числа незакреплённых комнат нет. Закрепляем.

Шаг 6. Сам путь. Идём с конца: 14 пришло из Оружейной, 11 в Оружейной – из Сада, 7 в Саду – из Кухни, 4 в Кухне – от Входа. Путь: Вход → Кухня → Сад → Оружейная → Сокровищница, $4 + 3 + 4 + 3 = 14$ м.

Для верности сравним с дорогами через Холл: через Оружейную $6 + 9 + 3 = 18$ м, через Библиотеку $6 + 5 + 8 = 19$ м. Обе длиннее 14.

⚠ **Типичная ошибка.** Выбрать путь с наименьшим числом коридоров: Вход → Холл → Оружейная → Сокровищница – всего три коридора. Но один из них длинный, 9 м, и выходит 18 м. Граф считает метры, а не двери.

Ответ: 14 м.

Своя задача по программе сферы, модуль 1

Реши сам

3.7. В лесу шесть полян: А, Б, В, Г, Д, Е. Между ними тропинки (длины в метрах): А–Б 5, А–В 2, В–Б 2, Б–Г 6, В–Д 7, Б–Д 3, Г–Е 3, Д–Е 4. Каков самый короткий путь от поляны А до поляны Е? Ответ – число метров.

3.8. Герой живёт в шалаше Ш. Проходы (в метрах): Ш–А 5, А–Б 4, Ш–Б 10, Б–Ф 3 – по ним можно ходить в обе стороны. Ещё есть обрыв из А прямо к фонтану Ф длиной 2 м, но с него можно только спрыгнуть: из Ф в А по нему не забраться. Герой идёт от шалаша к фонтану и возвращается обратно. Какова наименьшая длина всей прогулки туда и обратно? Ответ – число метров.

Задача 3.5 Три двери и три ключа

ключи и двери, порядок прохождения квеста

На плане уровня прямоугольники – комнаты, линии – проходы. На трёх проходах двери, каждая открывается ключом своего цвета. Ключи лежат в комнатах Г, Д, Е. Двери, открытые один раз, остаются открытыми.

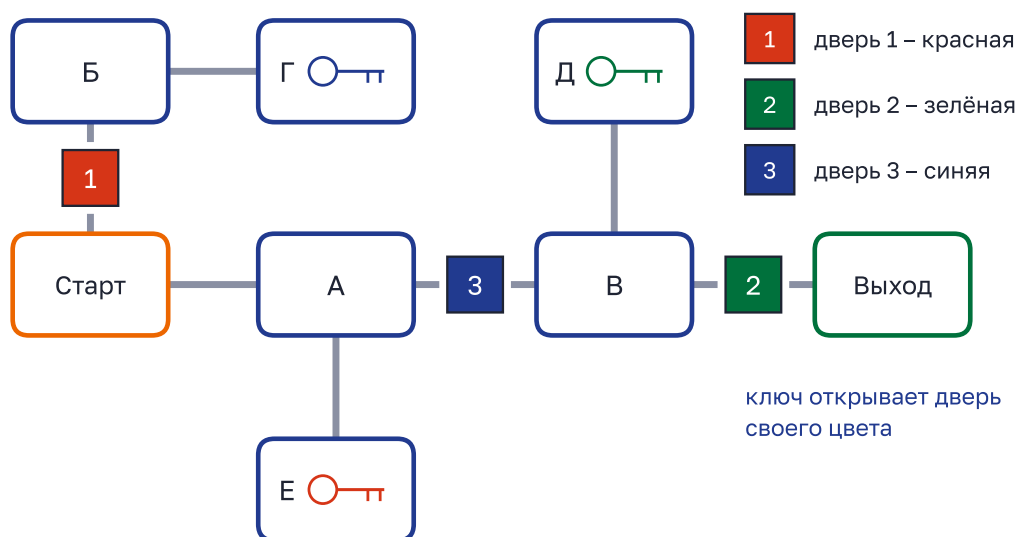


Рисунок 3.13 – План уровня к задаче 3.5

Герой начинает в комнате «Старт». В каком порядке он будет открывать двери, чтобы добраться до выхода? Ответ – номера дверей в порядке открытия, без пробелов.

Дано: двери 1 (красная, Старт – Б), 2 (зелёная, В – Выход), 3 (синяя, А – В); красный ключ в Е, синий в Г, зелёный в Д.

Найти: порядок открытия дверей.

Решение

Шаг 1. Куда можно пойти без ключей. От Старта открыт проход в А, от А – в Е. Комнаты: Старт, А, Е. Двери 1 и 3 пока закрыты.

Шаг 2. Какие ключи там лежат. В Е – красный ключ. Он открывает дверь **1** (Старт – Б).

Шаг 3. Новый круг. За дверью 1 – комнаты Б и Г. В Г – синий ключ, он открывает дверь **3** (А – В).

Шаг 4. Ещё круг. За дверью 3 – комнаты В и Д. В Д – зелёный ключ, он открывает дверь **2** (В – Выход). Выход достигнут.

Шаг 5. Проверка маршрутом. Пройдём уровень целиком: Старт → А → Е (красный ключ) → А → Старт → Б (дверь 1) → Г (синий ключ) → Б → Старт → А → В (дверь 3) → Д (зелёный ключ) → В → Выход (дверь 2). Двери открылись в порядке 1, 3, 2. Переходов из комнаты в комнату – 13.

Заметь: каждый ключ лежит за предыдущей дверью. Так квесты и строят – игрок раскрывает уровень кусочками. Другого порядка здесь быть не может: синий ключ спрятан за красной дверью, а зелёный – за синей.

 **Типичная ошибка.** Записать двери по порядку номеров – «123». Номер двери ничего не говорит о порядке, его задают только ключи: где лежит ключ и какая дверь его прячет. Вторая ошибка – выписать только двери, которые стоят на прямой дороге Старт → А → В → Выход, и ответить «32». Но синий ключ для двери 3 лежит за дверью 1, без неё не обойтись.

Ответ: 132.

Своя задача по программе сферы, модуль 1

Реши сам

3.9. В уровне семь комнат: Старт, А, Б, В, Г, Д и Выход. Открытые проходы: Старт – А, Старт – Б, В – Д. Двери: жёлтая (А – В), фиолетовая (Б – Г), красная (Г – Выход). Ключи: жёлтый – в Б, фиолетовый – в Д, красный – в В. Каждый переход через проход в соседнюю комнату – один ход. За какое наименьшее число ходов герой доберётся от Старта до Выхода? Ответ – целое число.

3.10. Тестировщик жалуется: уровень не пройти. Проходы: Старт – А и Г – Выход открыты. Двери: 1 – красная (А – Б), 2 – синяя (Старт – В), 3 – жёлтая (В – Г). Ключи: красный – в А, синий – в Б, жёлтый – в Г. Какую дверь игрок не сможет открыть никогда? Ответ – номер двери.

Задача 3.6 Сколько дорог ведёт к замку

комбинаторика, число путей по клеткам, запрещённые клетки

Герой стоит в левой нижней клетке поля 5×4 , замок – в правой верхней. Шагать можно только вправо или вверх на соседнюю клетку. В двух клетках лава, туда наступать нельзя. Сколько существует разных путей от героя до замка? Ответ – целое число.

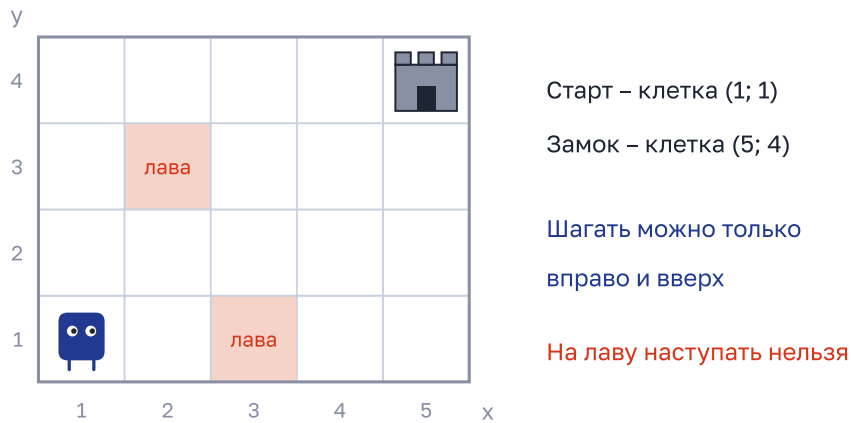


Рисунок 3.14 – Поле к задаче 3.6

Дано: поле 5×4 ; старт (1; 1); замок (5; 4); лава в клетках (3; 1) и (2; 3); ходы только вправо и вверх.

Найти: число путей от старта до замка.

Решение

Шаг 1. Нижняя строка. В старт (1; 1) пишем 1. В (2; 1) приходят только слева – тоже 1. В (3; 1) лава – 0. Дальше ловушка. В (4; 1) можно прийти только слева, а слева лава; снизу у нижней строки ничего нет. Значит, в (4; 1) пишем 0, и в (5; 1) – тоже 0.

Шаг 2. Строка 2, слева направо. (1; 2) – левый край, 1. (2; 2): слева 1, снизу 1, всего 2. (3; 2): слева 2, снизу 0 – получается 2. (4; 2): $2 + 0 = 2$. (5; 2): $2 + 0 = 2$.

Шаг 3. Строка 3. (1; 3) – 1. (2; 3) – лава, 0. (3; 3): слева 0, снизу 2 – это 2. (4; 3): $2 + 2 = 4$. (5; 3): $4 + 2 = 6$.

Шаг 4. Строка 4. (1; 4) – 1. (2; 4): слева 1, снизу лава 0 – это 1. (3; 4): $1 + 2 = 3$. (4; 4): $3 + 4 = 7$. В замке (5; 4): $7 + 6 = 13$.

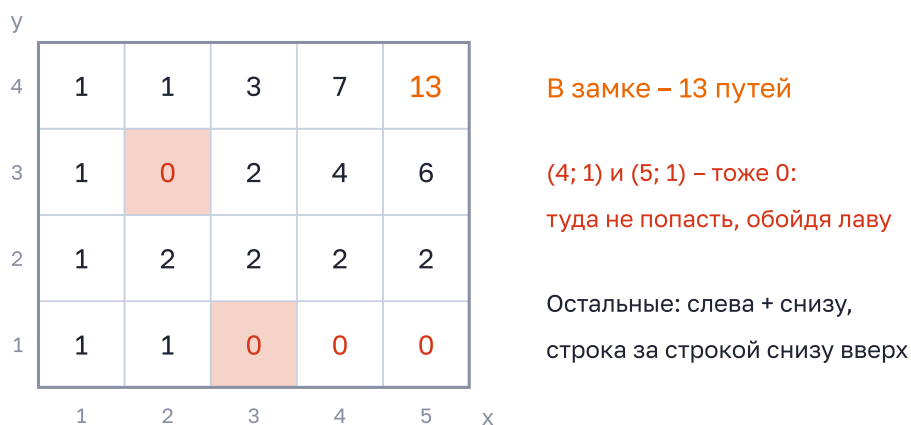


Рисунок 3.15 – Число путей в каждой клетке к задаче 3.6

Сравни с пустым полем. Без лавы на нём 35 путей – это число было в разделе 3.5. Две клетки лавы оставили 13, чуть больше трети: лава в нижней строке отрезала целый кусок края, вот и потеря большая.

❗ **Типичная ошибка.** По привычке заполнить всю нижнюю строку единицами: «на краю всегда по 1». Тогда в (4; 1) и (5; 1) окажутся единицы, и в замке выйдет 17. Но в эти клетки можно попасть только слева, через лаву. Правило «на краю по 1» работает только до первой запрещённой клетки.

Ответ: 13.

Своя задача по программе сферы, модуль 1

Реши сам

3.11. Поле – 5 столбцов и 4 строки. Герой стоит в левой нижней клетке (1; 1) и идёт в правую верхнюю (5; 4), шагая только вправо или вверх. В клетках (2; 2) и (4; 3) – проруби, туда наступать нельзя. Сколько существует разных путей? Ответ – целое число.

3.12. Поле – 6 столбцов и 4 строки, препятствий нет. Робот-уборщик едет из левой нижней клетки (1; 1) в правую верхнюю (6; 4), только вправо или вверх. По дороге он обязательно заезжает на зарядку в клетке (3; 2). Сколько существует таких путей? Ответ – целое число.

Задача 3.7 Как вывести игрока из пещеры

внимание игрока, ориентиры, свет, «хлебные крошки», выбор всех верных

Плейтест показал: игроки блуждают по тёмной пещере и не находят выход. Левел-дизайнер хочет, чтобы уровень сам вёл игрока к выходу – светом, ориентирами, предметами на пути, **без надписей и стрелок на экране**. Какие изменения ему подходят? Ответ – номера всех верных вариантов в порядке возрастания, без пробелов.

1. У выхода развести яркий костёр, а остальные тоннели оставить тёмными.
2. Разложить вдоль нужного тоннеля светящиеся кристаллы, которые можно собирать.
3. Сделать все тоннели одинаковыми по цвету и ширине, чтобы ни один не выделялся.
4. Сделать в потолке над выходом трещину, из которой бьёт столб света, видный из любой части пещеры.
5. Каждые 10 секунд показывать надпись «Выход – на востоке».
6. Поставить светящийся сундук с монетами – самую заметную вещь уровня – в дальний тупик.

Дано: 6 изменений.

Найти: те, что ведут к выходу средствами самого уровня.

Решение

Шаг 1. Что требует условие. Вариант должен тянуть игрока к **выходу** и обходиться **без текста**. Каждый прогоняем через оба вопроса.

Шаг 2. По вариантам.

- 1 – свет у выхода, темнота в остальном: глаз тянется к светлomu. Подходит.

- 2 – «хлебные крошки»: игрок собирает кристаллы и идёт по ним. Подходит.
- 3 – наоборот, убирает все различия. Игрок не может выбрать тоннель. Не подходит.
- 4 – столб света виден отовсюду: это ориентир. Подходит.
- 5 – ведёт к выходу, но надписью. Условие запрещает. Не подходит.
- 6 – самая заметная вещь стоит в тупике и уводит игрока от выхода. Не подходит.

Шаг 3. Записываем по возрастанию: 1, 2, 4.

Все три верных варианта – знакомые приёмы из раздела 3.6: свет, «хлебные крошки», ориентир.

 **Типичная ошибка.** Выбрать вариант 5: «он же помогает найти выход!». Помогает, но словами, а условие просит средства уровня. В задачах на выбор ловушка часто сидит в одном слове условия – «без надписей», «к выходу», «все верные». Перед тем как вводить ответ, перечитай условие.

Ответ: 124.

Своя задача по программе сферы, модуль 1

Реши сам

3.13. Сопоставь описание и приём левел-дизайна. Приёмы: 1 – ориентир, 2 – свет, 3 – «хлебные крошки», 4 – цвет-подсказка.

- А. Вдоль обрыва к мосту растут светящиеся грибы, и игрок идёт от гриба к грибу.
- Б. Над деревней стоит мельница с красной крышей, её видно из любого места карты.
- В. У нужной двери горит фонарь, а все остальные двери в тени.
- Г. Все ящики, которые можно двигать, – с жёлтой полосой, а неподвижные – без неё.

Ответ – четыре цифры: номера приёмов для А, Б, В, Г по порядку.

3.14. Какие решения левел-дизайнера запутают игрока? Ответ – номера всех таких вариантов в порядке возрастания.

1. Цепочка монет ведёт к обрыву, за которым ничего нет.
2. Дверь, которую нужно открыть, подсвечена, а декоративные двери тёмные.
3. Лестница, по которой можно залезть, того же серого цвета, что и стена, по которой нельзя.
4. Башня-ориентир стоит у выхода и видна со старта.
5. Все пять коридоров лабиринта одинаковые, без единой приметы.
6. Первый враг появляется на широкой ровной площадке, где легко увернуться.

Задача 3.8* Снегоход на трассе в две полосы

комбинаторика, подсчёт путей, правило умножения

Гонка на снегоходе по льду Обского моря. Трасса – 7 столбцов и 2 полосы, в некоторых клетках лежат монеты. Снегоход въезжает в столбец 1 в любую полосу. За каждый шаг он переходит в следующий столбец: либо прямо по своей полосе, либо в соседнюю полосу. Монету из клетки, куда он въехал, он забирает. Назад и вбок в том же столбце ехать нельзя.

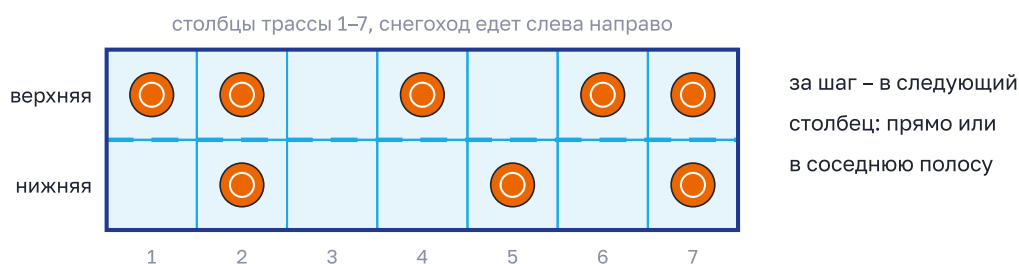


Рисунок 3.16 – Трасса к задаче 3.8. Вид сверху

Сколько существует разных маршрутов, на которых снегоход соберёт **ровно 4** монеты? Маршруты разные, если хотя бы в одном столбце снегоход едет по разным полосам. Ответ – целое число.

Дано: верхняя полоса по столбцам 1–7: монета, монета, пусто, монета, пусто, монета, монета; нижняя: пусто, монета, пусто, пусто, монета, пусто, монета.

Найти: число маршрутов ровно с 4 монетами.

Решение

Шаг 1. Что такое маршрут. В каждом столбце снегоход бывает ровно один раз, и в каждом столбце выбирает: верх или низ. Из любой полосы можно перейти в любую полосу следующего столбца. Значит, маршрут – это просто 7 выборов «верх/низ» подряд. По правилу умножения из раздела 3.5 всех маршрутов $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^7 = 128$. Перебирать их по одному долго, поэтому дальше пойдём хитрее.

Шаг 2. Разложим столбцы на три сорта. В этом вся соль задачи.

Таблица 3.2

Сорт	Столбцы	Сколько монет даёт
монеты в обеих полосах	2, 7	всегда 1
пусто в обеих полосах	3	всегда 0
монета только в одной полосе	1, 4, 5, 6	1 или 0 – как выберешь

В столбцах 2 и 7 снегоход возьмёт монету при любом выборе – это 2 монеты «даром». В столбце 3 не возьмёт ничего.

Шаг 3. Сколько монет добрать. Нужно ровно 4, две уже есть. Значит, в двух из четырёх «выбирающих» столбцов 1, 4, 5, 6 монету берём, а в двух других проезжаем мимо.

Шаг 4. Сколькими способами выбрать эти два столбца. Выпишем все пары: 1 и 4, 1 и 5, 1 и 6, 4 и 5, 4 и 6, 5 и 6 – **6 пар**. Для каждой пары полосы в столбцах 1, 4, 5, 6 определены однозначно: где берём монету – едем по полосе с монетой, где не берём – по пустой.

Шаг 5. Свободные столбцы. В столбцах 2, 3, 7 выбор полосы на число монет не влияет, но маршрут от него меняется. Там по 2 варианта: $2 \cdot 2 \cdot 2 = 8$.

Шаг 6. Перемножаем. $6 \cdot 8 = 48$ маршрутов.

Шаг 7. Проверка программой. Этот шаг – для тех, кто знает Python; остальные могут его пропустить. Компьютер перебирает все 128 маршрутов и считает, в скольких из них ровно 4 монеты:

```
from itertools import product

top = "1101011"      # верхняя полоса, столбцы 1-7: 1 - монета
bot = "0100101"      # нижняя полоса
count = 0
for lanes in product("вн", repeat=7):      # все  $2^7 = 128$  маршрутов
    coins = 0
    for i, lane in enumerate(lanes):
        row = top if lane == "в" else bot
        coins += int(row[i])                # монета в клетке, куда въехали
    if coins == 4:
        count += 1
print(count)                                # 48
```

Много это или мало? 48 из 128 – чуть больше трети. Меньше 2 монет не собрать никак: столбцы 2 и 7 дают их всегда. Больше 6 – тоже. А 4 – середина между 2 и 6, и середину обычно можно набрать больше всего способами. Вот ей и достался самый большой кусок.

⚠ Типичная ошибка. Посчитать только 6 пар и забыть про свободные столбцы 2, 3, 7. Число монет там от выбора не зависит, но маршруты-то разные: по верху или по низу – это два разных маршрута. А кто-то пытается в столбцах 2 и 7 «не брать» монету – не выйдет: там монеты в обеих полосах, мимо не проехать.

Ответ: 48.

По мотивам задачи НТО 2025/26, профиль «Разработка компьютерных игр» (старший трек, 8–11 классы), заключительный этап, предметный тур по информатике, «Двухэтажный сайд-скроллер»

Реши сам

3.15.* Снова гонка по правилам задачи 3.8, но трасса другая. Верхняя полоса по столбцам 1–7: монета, монета, пусто, пусто, монета, монета, пусто. Нижняя: пусто, монета, монета, пусто, пусто, монета, монета. Сколько существует маршрутов, на которых снегоход соберёт **ровно 3** монеты? Ответ – целое число.

3.16.* Та же гонка, что в задаче 3.8, но другая трасса. Верхняя полоса по столбцам 1–7: монета, пусто, монета, монета, пусто, монета, пусто. Нижняя: пусто, монета, пусто, пусто, пусто, монета, монета. Сколько существует маршрутов, на которых снегоход соберёт **хотя бы 5** монет? Ответ – целое число.

Логика, выборы и случайность



Эта глава – про математику, которая прячется внутри квеста. Подсказки вида «кузнец живёт не у колодца» распутывают таблицей с крестиками. Лжеца ловят на противоречии. Дальше – условия с И, ИЛИ, НЕ, подсчёт концовок в ветвистой истории и шанс, что из босса выпадет редкий питомец. На отборе такая задача – короткая история, а ответ – число, доля или строка цифр вроде «213». Формул почти нет, зато много аккуратного перебора: пропустил один случай – ответ неверный.

4.1. Кто где: таблица с крестиками

НА ПАЛЬЦАХ

1. Кузнец живёт не у колодца и не у мельницы.
2. Травница живёт не у колодца.

	колодец	мельница	мост
Кузнец	✕	✕	●
Травница	✕	●	✕
Стражник	●	✕	✕

одна пустая клетка – кузнец живёт здесь

✕ крестик из подсказки
✕ дом уже занят

Рисунок 4.1 – Три жителя, три дома и две подсказки

Каждая подсказка с «не» – это крестик. Когда в строке осталась одна пустая клетка, житель живёт там.

В квестах то и дело попадают загадки «кто где живёт», «кто что потерял», «какой ключ от какой двери». Подсказок несколько, и ни одна сама по себе ответа не даёт. Держать всё в голове трудно: на третьей подсказке забываешь первую. Поэтому такие загадки решают на бумаге, в **таблице исключений**. Строки – персонажи, столбцы – места или предметы.

Правила заполнения простые:

- подсказка «кузнец живёт не у колодца» – крестик в клетке «кузнец – колодец»;
- в каждой строке и в каждом столбце ровно одна точка: у каждого жителя один дом, в каждом доме один житель;
- если в строке крестики стоят везде, кроме одной клетки, – в неё точку;
- поставил точку – зачеркни остальные клетки в её столбце: этот дом уже занят.

Разберём картинку сверху. Жители – кузнец, травница и стражник, дома – у колодца, у мельницы и у моста. Подсказки: «Кузнец живёт не у колодца и не у мельницы», «Травница живёт не у колодца». Первая подсказка даёт два крестика в строке кузнеца – осталась одна клетка, мост. Ставим точку и зачёркиваем мост у остальных. У травницы теперь закрыты колодец (по подсказке) и мост (он занят) – она живёт у мельницы. Стражнику остался колодец.

На отборе ответ такой задачи обычно вводят строкой цифр. Пусть дома пронумерованы: 1 – колодец, 2 – мельница, 3 – мост, а спрашивают, где живут кузнец, травница и стражник, по порядку. Ответ: **321**. Порядок в ответе задаёт вопрос, а не таблица, – перечитай его перед вводом.

Запомни. Строки – кто, столбцы – где. Подсказка с «не» – крестик. Одна свободная клетка в строке – точка. Точка закрывает свой столбец для всех остальных.

Совет. Подсказки вида «рядом», «левее», «через один дом» говорят сразу про двух жителей, и одним крестиком их не запишешь. Отложи их и сначала выжми всё из подсказок с «не». Потом вернись к отложенным: вариантов к этому моменту останется два-три, их легко проверить по очереди.

4.2. Правда и ложь: рыцари и лжецы

НА ПАЛЬЦАХ

«Какого цвета снег?»



Рисунок 4.2 – Рыцарь и лжец отвечают на один и тот же вопрос

Рыцарь говорит правду всегда. Лжец лжёт всегда – даже про цвет снега.

В квестах часто стоят два стражника у двух дверей, и один из них обманщик. В математике таких персонажей называют **рыцарями** и **лжецами**. Рыцарь каждой своей фразой говорит правду, лжец каждой фразой лжёт. Лжец не «иногда путает»: если он скажет, что сейчас зима, – значит, сейчас не зима.

Решают такие задачи одним приёмом: **предположи и проверь**. Берём одного персонажа и говорим: «Пусть он рыцарь». Тогда его слова верны. Смотрим, что из них следует для остальных, и проверяем их слова тоже. Если где-то вышло, что кто-то одновременно рыцарь и лжец, или слова рыцаря оказались ложью, – это **противоречие**. Значит, наше «пусть» было неверным, и персонаж – лжец. Если противоречия нет и всё сходится, – вариант подходит. Проверь оба варианта, «пусть рыцарь» и «пусть лжец»: в правильно составленной задаче подойдёт только один.

Три наблюдения экономят кучу времени.

- **«Я лжец» не может сказать никто.** Рыцарь этим соврал бы, а лжец сказал бы правду.
- **«Я рыцарь» может сказать любой.** Рыцарь говорит правду, лжец врёт. Такая фраза ничего не сообщает – пропускай её.
- **Если А говорит «Б – лжец», то А и Б разные.** Если А рыцарь, Б правда лжец. Если А лжец, всё наоборот, и Б рыцарь. А фраза «Б – рыцарь» значит, что А и Б одинаковые.

Пример. Стражник А у левой двери говорит: «Мы с ним оба лжецы». Пусть А рыцарь. Тогда фраза верна, и А – лжец. Но мы только что сказали, что он рыцарь, – противоречие. Значит, А лжец. Тогда его фраза ложна: неверно, что оба лжецы. Раз сам А лжец, то второй стражник –

рыцарь. Ответ: А – лжец, Б – рыцарь. Второй стражник вообще молчал, а мы всё равно про него узнали.

Запомни. Предположи – проверь – ищи противоречие. Противоречие значит, что предположение неверно. «А говорит, что Б лжец» – они разные. «А говорит, что Б рыцарь» – они одинаковые.

4.3. И, ИЛИ, НЕ: условия в игре

НА ПАЛЬЦАХ

И: нужны оба ключа

ИЛИ: хватит любой двери

Рисунок 4.3 – Дверь с двумя замками и комната с двумя дверями

Два замка на одной двери – нужны оба ключа: это И. Две двери в одну комнату – хватит любой: это ИЛИ.

Высказывание – фраза, про которую можно сказать «верно» или «неверно». «У героя есть ключ» – высказывание: ключ либо есть, либо нет. «Сколько у тебя монет?» – вопрос, «Открой двери!» – приказ, это не высказывания. Всё, что игра проверяет в условиях, – высказывания: «здоровье больше нуля», «квест выполнен», «стражник рядом».

Из простых высказываний собирают сложные тремя словами.

- **И.** «Есть ключ И горит факел» верно, только когда верны **обе** части. Нет факела – всё высказывание неверно.
- **ИЛИ.** «Есть ключ ИЛИ есть отмычка» верно, когда верна **хотя бы одна** часть. Есть и ключ, и отмычка – тоже верно. Неверно только тогда, когда нет ни того, ни другого.
- **НЕ.** Переворачивает высказывание. «НЕ стражник рядом» верно, когда стражника рядом нет.

Таблица 4.1

А	Б	А И Б	А ИЛИ Б
---	---	-------	---------

А	Б	А И Б	А ИЛИ Б
да	да	да	да
да	нет	нет	да
нет	да	нет	да
нет	нет	нет	нет

Самая частая ошибка здесь – житейское «или». Мама спрашивает «чай или компот?», и это значит «что-то одно». В логике и в программах ИЛИ значит «хотя бы одно»: если выполнены обе части, условие тоже выполнено.

Порядок действий как в арифметике. Там умножают раньше, чем складывают, а здесь сначала НЕ, потом И, потом ИЛИ. Скобки – самыми первыми. Условие «есть ключ ИЛИ есть отмычка И нет ловушки» без скобок читается как «есть ключ ИЛИ (есть отмычка И нет ловушки)». То есть с ключом можно открыть сундук даже с ловушкой. Если автор имел в виду другое, он обязан поставить скобки.

В программах эти слова пишут по-английски: `and`, `or`, `not`. Вот условие сундука на Python. В Python можно называть переменные русскими словами, а `True` значит «да», `False` – «нет».

```
есть_ключ = True
есть_отмычка = False
ловушка_взведена = True
можно_открыть = (есть_ключ or есть_отмычка) and not ловушка_взведена
print(можно_открыть)    # False
```

Первые три строки записывают, что есть у героя: ключ есть, отмычки нет, ловушка взведена. Четвёртая считает условие. В скобках: ключ ИЛИ отмычка – «да», ведь ключ есть. НЕ ловушка – «нет», ведь ловушка взведена. «Да» И «нет» – «нет». Пятая строка печатает `False`: сундук не откроется, пока ловушку не обезвредят. В ПРИМС и в Ink тоже есть такие условия – мы встретим их в главах 5–7.

Запомни. И – нужны обе части. ИЛИ – хватит одной, обе тоже годятся. НЕ – наоборот. Порядок: скобки, НЕ, И, ИЛИ.

Совет. «Неверно, что у героя есть и меч, и щит» – это не «нет ни меча, ни щита». Это «чего-то одного не хватает, а может, и обоих». Если условие запутанное, выпиши все случаи (есть оба; только меч; только щит; ничего) и проверь каждый. Случаев всего четыре – это быстрее, чем гадать.

4.4. Дерево выборов: сколько концовок

НА ПАЛЬЦАХ

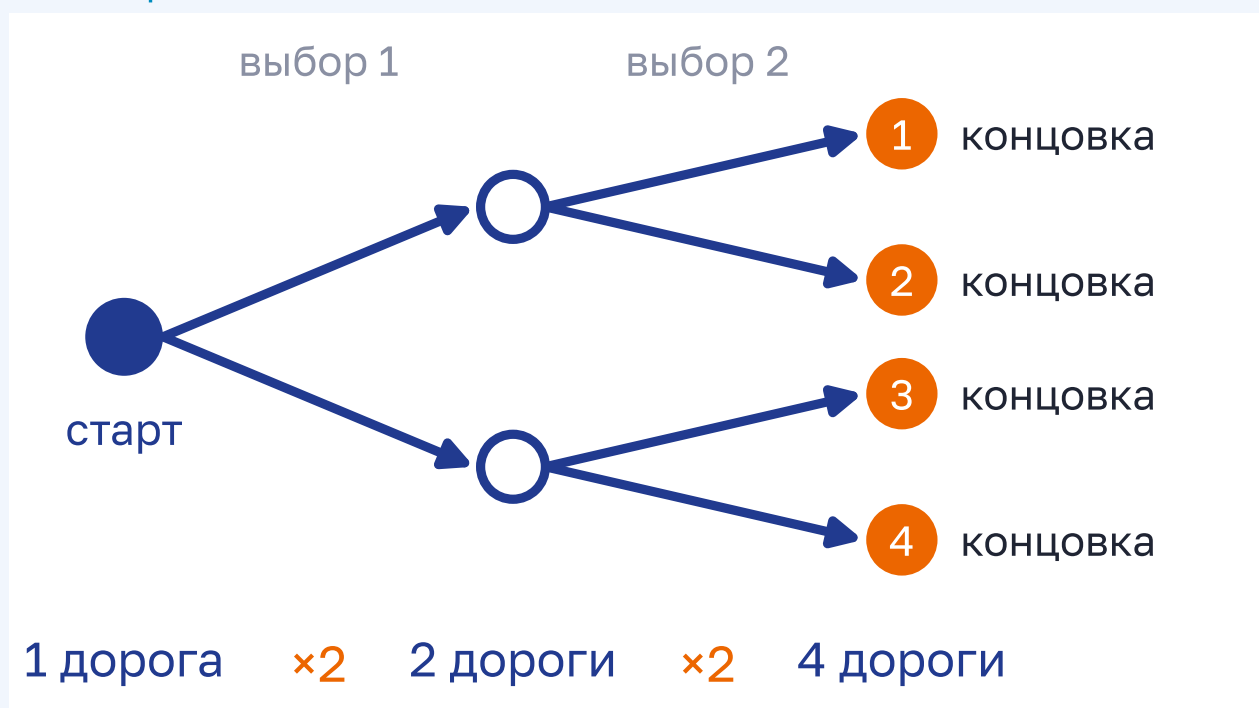


Рисунок 4.4 – Две развилки по два варианта

Каждая развилка удваивает число дорог: одна дорога, две, четыре.

Когда в игре выбор, сценарист рисует **дерево выборов**. Каждая **развилка** – место, где игрок выбирает. От неё расходятся **ветки** – по одной на каждый вариант. Там, где ветка закончилась, стоит **концовка**. Сколько концовок у истории – столько «листьев» у дерева.

Правило умножения. Если на первой развилке 3 варианта, а после любого из них – ещё 2, то путей $3 \cdot 2 = 6$: каждая из трёх дорог раздваивается, и дорог становится вдвое больше. Так же считают варианты в редакторе персонажа: 3 шлема, 4 плаща и 2 вида оружия дают $3 \cdot 4 \cdot 2 = 24$ разных героя. Любой шлем можно надеть с любым плащом, поэтому числа перемножаются.

Если развилок k и на каждой по 2 варианта, а ветки нигде не сходятся, концовок 2^k (двойка, умноженная сама на себя k раз):

Таблица 4.2

Развилка	Концовок
1	2
2	4
3	8
5	32
10	1024

Посмотри на последнюю строку. Десять простых выборов «да или нет» – и тысяча концовок, каждую надо написать и нарисовать. Поэтому в настоящих играх ветки почти всегда снова сходятся.

Правило сложения. Если ветки дальше разные, считаем концовки в каждой и складываем. Налево – 3 концовки, направо – 2, всего $3 + 2 = 5$. Когда умножать, а когда складывать? Выбор идёт **один за другим** («сначала спутник, потом дорога») – умножаем. Выбор **одно из двух**, и дальше ветки устроены по-разному – складываем.

Ветки сходятся. Если две дороги ведут в одну и ту же деревню, дерево превращается в карту со стрелками. Концовок на ней меньше, чем путей: разные игроки придут к одной концовке разными дорогами. Число путей считают так. У старта пишем 1. У каждого места пишем сумму чисел тех мест, откуда в него ведут стрелки. Число у финала – ответ.



Рисунок 4.5 – Ветки сходятся в деревне: концовок две, а путей четыре. Число у места – сколько путей в него ведёт

Порядок. Отдельный случай – когда выбирают не «что», а «в каком порядке». Три рычага А, Б, В можно дёрнуть по очереди шестью способами. Первым – любой из трёх, вторым – любой из двух оставшихся, последним – тот, что остался: $3 \cdot 2 \cdot 1 = 6$. Такое произведение записывают с восклицательным знаком: $3! = 6$, $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$ («четыре факториал»).

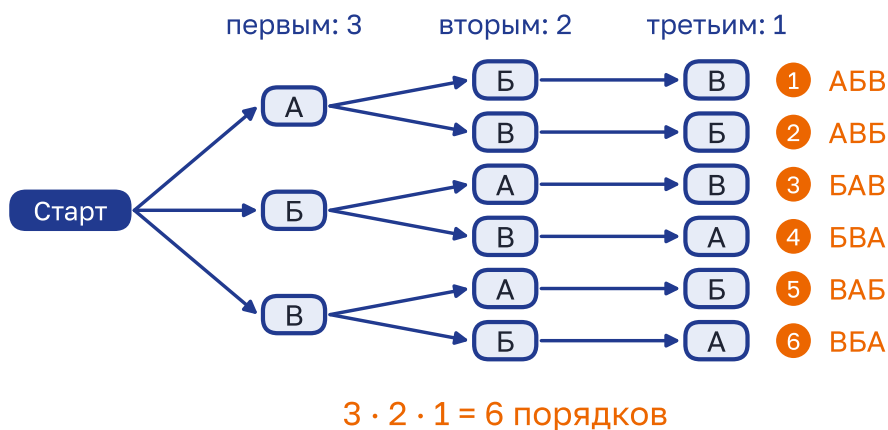


Рисунок 4.6 – Три рычага: первым – любой из трёх, вторым – из двух оставшихся, третьим – последний. Всего 6 порядков

Запомни. Выборы один за другим – умножай. Разные ветки – складывай. k развилок по 2 варианта без схождений – 2^k концовок. Ветки сходятся – рисуй карту и пиши у каждого места сумму чисел, из которых в него пришли.

Совет. Не пытайся нарисовать дерево с тридцатью листьями. Нарисуй начало – две-три развилки, – посчитай, сколько получилось, и проверь это правилом умножения. Рисунок нужен, чтобы понять, где умножать, а где складывать.

4.5. Шанс: доля, кубик и «1 из 5»

НА ПАЛЬЦАХ

$$20\% = 1/5 = 1 \text{ из } 5$$

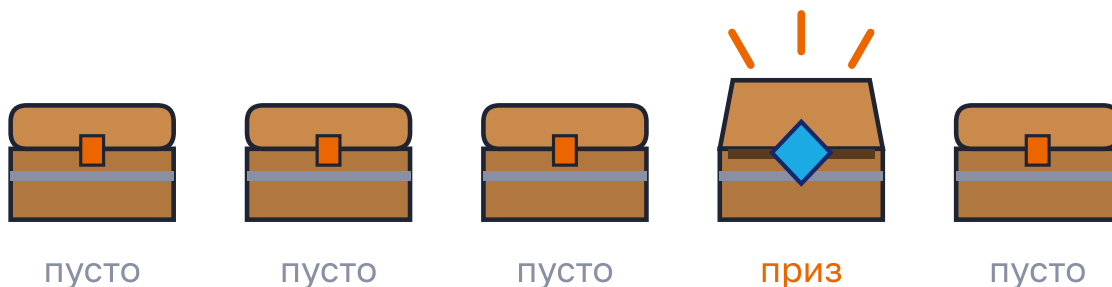


Рисунок 4.7 – Пять одинаковых сундуков, приз только в одном

Шанс 20 % – это 1 из 5: сундуков пять, приз в одном, а какой откроешь – решает случай.

Вероятность, или шанс, – это доля. Сколько исходов нам подходят, делим на сколько всего исходов:

$$P = \frac{m}{n},$$

где P – вероятность, m – число подходящих исходов, n – число всех исходов. Одно условие: все исходы должны быть **равновозможными**, то есть ни один не выпадает чаще другого. У честного кубика шесть граней, и ни одна не тяжелее остальных. Шанс выбросить шестёрку $P = 1/6$. Шанс выбросить чётное число – 2, 4 или 6 – $P = 3/6 = 1/2$.

Вероятность – всегда число от 0 до 1. Ноль – не бывает никогда (выбросить 7 на одном кубике), единица – бывает всегда (выбросить число меньше 10). В играх шанс обычно пишут в процентах, а в задачах просят то долю, то проценты, то «1 из скольких». Это одно и то же число в разных нарядах:

Таблица 4.3

Проценты	Доля	Дробь	«1 из ...»
50 %	0,5	1/2	1 из 2
25 %	0,25	1/4	1 из 4
20 %	0,2	1/5	1 из 5
10 %	0,1	1/10	1 из 10
5 %	0,05	1/20	1 из 20

Перевод: проценты делим на 100 – получаем долю ($20:100 = 0,2$). «1 из скольких» – это 100, делённое на проценты ($100:20 = 5$).

«1 из 5» – это **в среднем**, а не по расписанию. Шанс 20 % не значит, что предмет упадёт ровно с пятого монстра. Он может упасть с первого, а может не упасть ни с одного из десяти. Но если монстров много, доля удачных попыток будет близка к 20 %. Поэтому геймдизайнер прикидывает: 50 монстров при шансе 20 % дадут в среднем $50 \cdot 0,2 = 10$ предметов. У одного игрока выйдет 7, у другого 13, а в среднем по всем игрокам – около 10.

Все исходы вместе – это 100 %. Из сундука с шансом 20 % выпадает ключ, с шансом 30 % – зелье, а в остальных случаях он пустой. Значит, пустым он бывает в $100 - 20 - 30 = 50$ % случаев. Так же «не выпадет» – это $1 - P$: шанс попадания 0,7 – шанс промаха $1 - 0,7 = 0,3$.

Два кубика. В настольных играх часто бросают два кубика и складывают очки. Каждый кубик может дать 6 чисел, и они не зависят друг от друга. По правилу умножения всего $6 \cdot 6 = 36$ равновозможных пар. Суммы при этом выпадают **неодинаково часто**. Сумму 2 даёт только пара «1 и 1», а сумму 7 – целых шесть пар: 1 и 6, 2 и 5, 3 и 4, 4 и 3, 5 и 2, 6 и 1.



Рисунок 4.8 – Все 36 исходов двух кубиков. В клетке – сумма очков. Сумма 7 выпадает чаще всех: 6 клеток из 36

Шанс суммы 7 равен $6/36 = 1/6$, а шанс суммы 2 – всего $1/36$. Заметь: «1 и 6» и «6 и 1» – разные исходы. Первый кубик дал единицу, второй шестёрку – или наоборот.

Оба события сразу. Если события друг от друга не зависят, их шансы перемножают. Две шестёрки подряд на одном кубике: $\frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$. Два пустых сундука подряд при шансе 50 %:

$$0,5 \cdot 0,5 = 0,25.$$

Шанс попасть в зону. Бывает, что случайно не грань кубика, а точка на поле: враг бьёт наугад в любую клетку. Тогда все клетки равновозможны, и шанс попасть в зону – доля её клеток. Зона 5×4 клетки на поле 20×10 клеток: $\frac{5 \cdot 4}{20 \cdot 10} = \frac{20}{200} = 0,1$, то есть 10 %.

Запомни. Шанс = подходящие исходы : все исходы. $20 \% = 0,2 = 1/5 = \text{«1 из 5»}$ – в среднем, а не каждый пятый раз. Все исходы вместе – 100 %. Независимые события сразу – шансы умножаются. На двух кубиках 36 исходов, чаще всего выпадает сумма 7.

4.6. Монета не помнит: серия неудач

НА ПАЛЬЦАХ



Рисунок 4.9 – Пять решек подряд. Что будет в шестой раз?

У монеты нет памяти. После пяти решек шанс орла в шестой раз – всё те же $1/2$.

Игрок открыл пять сундуков с шансом ключа 20 %, и все пустые. «Ну теперь-то точно повезёт!» – думает он. Нет. Шестой сундук не знает про пять предыдущих, и шанс ключа в нём – те же 20 %. Генератор случайных чисел в игре похож на монету или кубик: он не помнит прошлых бросков. Удача не копится, и игра ничего тебе не «должна». Эту ошибку так часто делают, что у неё есть имя – **ошибка игрока**.

Откуда берётся ощущение, что «теперь-то выпадет»? Заранее, **до первого броска**, длинная серия и правда редкая. Пять решек подряд: $\left(\frac{1}{2}\right)^5 = \frac{1}{32}$, в среднем одна серия из 32. Но когда пять решек уже выпали, эта редкость позади. Осталась одна монета и один бросок, а у него шанс орла – $1/2$.

Совсем другой вопрос – какой шанс, что за n попыток предмет выпадет **хотя бы раз**. Его считают до начала серии и через «ни разу». Шанс, что предмет не выпадет в одной попытке, – $1 - p$. Шанс, что не выпадет ни разу за n попыток, – $(1 - p)$, умноженное само на себя n раз, то есть $(1 - p)^n$. А «хотя бы раз» – всё остальное:

$$P_{\text{хотя бы раз}} = 1 - (1 - p)^n.$$

Здесь p – шанс в одной попытке (долей, 10 % – это 0,1), n – число попыток. Проверь себя на двух попытках при шансе 10 %: «не выпало» в одной – 0,9, в обеих – $0,9 \cdot 0,9 = 0,81$, «хотя бы раз» – $1 - 0,81 = 0,19$, то есть 19 %. Вот что выходит при разном числе попыток:

Таблица 4.4

Попыток	Хотя бы раз
1	10 %
2	19 %
3	27 %
5	41 %
10	65 %
20	88 %

Даже двадцать попыток не дают 100 %. «Шанс 10 %, значит, за 10 попыток точно выпадет» – та же ошибка игрока, только вывернутая наизнанку. Посчитать такое можно на калькуляторе или в Python:

```
p = 0.1          # шанс в одной попытке
n = 10           # сколько попыток
ни_разу = (1 - p) ** n  # ** – возведение в степень
print(round(1 - ни_разу, 2))  # 0.65
```

Иногда геймдизайнеры сами дают игре память. Они добавляют **счётчик неудач**: «если редкий предмет не выпал 9 раз подряд, на десятый он выпадет точно». Такую гарантию делают, чтобы невезучие игроки не бросали игру от обиды. Но это отдельное правило, и программист записывает его переменной-счётчиком (как – в главе 6). Если в условии задачи такого правила нет, шанс у каждой попытки прежний.

Запомни. Прошлые неудачи не меняют шанс следующей попытки, если в правилах нет счётчика. «Хотя бы раз за n попыток» считают через «ни разу»: $1 - (1 - p)^n$. Даже много попыток не дают 100 %.

Совет. Слова «уже 5 раз не выпало» в условии – почти всегда ловушка. Спроси себя: есть ли в правилах игры счётчик? Нет – отвечай тем же шансом, что и для одной попытки.

4.7. Разбираем задачи

Задача 4.1 Потерянные вещи

разминка · логическая задача, таблица исключений, ответ строкой цифр

В подземном квесте три гнома – Бур, Гром и Тиль – потеряли по одной вещи: фонарь, кирку и карту. Герой должен вернуть каждому его вещь. Подсказки:

1. Бур не терял ни фонарь, ни карту.
2. Тиль не терял фонарь.

Вещи пронумерованы: 1 – фонарь, 2 – кирка, 3 – карта. Запиши номера вещей Бура, Грома и Тили – по порядку, без пробелов.

Дано: три гнома, три вещи, у каждого ровно одна; две подсказки.

Найти: номера вещей Бура, Грома, Тили.

Решение

Шаг 1. Рисуем таблицу. Строки – гномы, столбцы – вещи. По подсказке 1 ставим крестики «Бур – фонарь» и «Бур – карта». По подсказке 2 – крестик «Тиль – фонарь».

Шаг 2. Ищем строку с одной пустой клеткой. У Бура пустая только кирка – ставим точку. Кирка занята, зачёркиваем её у Грома и у Тили.

Шаг 3. Дальше по цепочке. У Тили закрыты фонарь (подсказка) и кирка (занята) – остаётся карта. Грому остаётся фонарь.

Таблица 4.5

	Фонарь	Кирка	Карта
Бур	х	•	х
Гром	•	х	х
Тиль	х	х	•

Шаг 4. Записываем ответ по порядку вопроса. Бур – кирка (2), Гром – фонарь (1), Тиль – карта (3).

Сверим с подсказками. Бур не терял ни фонарь, ни карту – и правда, у него кирка. Тиль не терял фонарь – у него карта. В каждой строке и в каждом столбце по одной точке.

 **Типичная ошибка.** Записать ответ в порядке вещей, а не гномов: «кому фонарь, кому кирка, кому карта». Получится совсем другая строка. И ещё легко забыть, что точка закрывает весь столбец. Тогда у Тили останутся две пустые клетки, кирка и карта, и кажется, что решений два.

Ответ: 213.

Своя задача по программе сферы, модуль 1

Реши сам

4.1. На школьной ярмарке игр Лёва, Миша и Соня показывали каждый свою игру: платформер, квест и гонки. Лёва не делал гонки. Соня не делала ни квест, ни гонки. Жанры пронумерованы: 1 – платформер, 2 – квест, 3 – гонки. Запиши номера жанров Лёвы, Миши и Сони по порядку.

4.2. В деревне квеста на одной улице стоят четыре дома, пронумерованные слева направо от 1 до 4. В них живут мельник, пекарь, рыбак и кузнец, в каждом доме один житель. Известно:

1. Пекарь живёт рядом с кузнецом – в соседнем доме.
2. Мельник живёт левее пекаря.
3. Мельник живёт не в крайнем доме.
4. Пекарь живёт не в доме 4.

Запиши номера домов мельника, пекаря, рыбака и кузнеца по порядку.

Задача 4.2 Три хранителя

рыцари и лжецы, предположи и проверь, противоречие

У входа в храм героя встречаются три хранителя: А, Б и В. Каждый из них либо рыцарь (всегда говорит правду), либо лжец (всегда лжёт). Хранители сказали:

- А: «Б – лжец».
- Б: «В – лжец».
- В: «А и Б – оба лжецы».

Кто из них рыцарь, а кто лжец? Запиши для А, Б и В по порядку: 1 – рыцарь, 0 – лжец. Например, 110.

Дано: три персонажа, каждый рыцарь или лжец; три фразы.

Найти: кто рыцарь, кто лжец – строка из трёх цифр.

Решение

Шаг 1. Что дают первые две фразы. А говорит, что Б – лжец. Значит, А и Б разные: один рыцарь, другой лжец (см. раздел 4.2). Б говорит, что В – лжец. Значит, Б и В тоже разные.

Шаг 2. Проверяем фразу В. В утверждает, что А и Б – оба лжецы. Но мы уже знаем, что А и Б разные: один из них рыцарь. Значит, «оба лжецы» – неправда. Правду В не сказал, поэтому В – лжец.

Шаг 3. Раскручиваем цепочку. Б и В разные, В – лжец, значит, Б – рыцарь. А и Б разные, Б – рыцарь, значит, А – лжец.

Шаг 4. Проверяем все три фразы. А (лжец) сказал «Б – лжец». Б на самом деле рыцарь, фраза ложная – для лжеца правильно. Б (рыцарь) сказал «В – лжец» – правда. В (лжец) сказал «А и Б оба лжецы» – ложь, ведь Б рыцарь. Всё сходится.

Не веришь – перебери. Вариантов всего $2 \cdot 2 \cdot 2 = 8$ (каждый из трёх – рыцарь или лжец). Подойдёт только один – тот, что мы нашли.

❗ **Частая ошибка.** Решить, что раз В обвиняет обоих, то сам он честный. Громко обвинять других умеет и лжец. И не останавливайся на первом варианте, где не нашлось противоречия: проверь слова всех троих, как в шаге 4.

Ответ: 010 (Б – рыцарь, А и В – лжецы).

Своя задача по программе сферы, модуль 1

Реши сам

4.3. У двух дверей стоят стражники Том и Рик, каждый – рыцарь или лжец. Том сказал: «Хотя бы один из нас – лжец». Рик промолчал. Запиши для Тома и Рика по порядку: 1 – рыцарь, 0 – лжец.

4.4. Три жителя острова, А, Б и В, – каждый рыцарь или лжец. А сказал: «Среди нас ровно один рыцарь». Б сказал: «Среди нас ровно два рыцаря». В сказал: «Среди нас нет ни одного рыцаря». Запиши для А, Б и В по порядку: 1 – рыцарь, 0 – лжец.

Задача 4.3 Подъёмный мост

И, ИЛИ, НЕ, порядок действий, скобки

В игре через ров перекинут подъёмный мост. Программист записал условие: мост опускается, если

(рычаг 1 включён ИЛИ рычаг 2 включён) И НЕ тролль на мосту.

Команда проверяет игру в шести ситуациях:

Таблица 4.6

Ситуация	Рычаг 1	Рычаг 2	Троль на мосту
1	включён	выключен	нет
2	выключен	выключен	нет
3	включён	включён	да
4	выключен	включён	нет
5	включён	включён	нет
6	выключен	включён	да

В каких ситуациях мост опустится? Запиши номера по возрастанию, без пробелов.

Дано: условие «(P1 ИЛИ P2) И НЕ Т»; шесть ситуаций.

Найти: номера ситуаций, в которых условие выполнено.

Решение

Шаг 1. Разбираем условие на части. Сначала скобка: «P1 ИЛИ P2» – включён хотя бы один рычаг. Потом «НЕ T» – тролля на мосту нет. Между ними И: мост опустится, только если выполнены **обе** части.

Шаг 2. Считаем скобку для каждой ситуации. Хотя бы один рычаг включён в ситуациях 1, 3, 4, 5, 6. В ситуации 2 выключены оба – скобка «нет», и эту ситуацию можно сразу вычеркнуть.

Шаг 3. Считаем «НЕ тролль». Тролля нет в ситуациях 1, 2, 4, 5. В ситуациях 3 и 6 он на мосту – «НЕ тролль» даёт «нет».

Шаг 4. Соединяем через И. Нужны ситуации, где обе части – «да»: скобка «да» (1, 3, 4, 5, 6) и тролля нет (1, 2, 4, 5). В оба списка попадают 1, 4, 5.

Таблица 4.7

Ситуация	P1 ИЛИ P2	НЕ тролль	Мост
1	да	да	опустится
2	нет	да	нет
3	да	нет	нет
4	да	да	опустится
5	да	да	опустится
6	да	нет	нет

Зачем программисту такое условие? Чтобы тролль не свалился в ров, а игроку хватило любого рычага. В ответ попали как раз все случаи «тролля нет, хоть один рычаг включён».

❗ **Чаше всего выкидывают ситуацию 5:** «ИЛИ – значит, что-то одно, а тут включены оба». Получится 14. В логике ИЛИ выполнено и тогда, когда верны обе части. А кто-то не замечает скобок и читает условие как «P1 ИЛИ (P2 И НЕ T)». Тогда в ситуации 3 рычаг 1 один «пересилит» тролля, и выйдет 1345.

Ответ: 145.

Своя задача по программе сферы, модуль 2

Реши сам

4.5. Портал в игре открывается, если **(собраны все 3 кристалла И НЕ идёт бой) ИЛИ у героя есть свиток телепорта**. Ситуации: 1) кристаллы собраны, боя нет, свитка нет; 2) кристаллы собраны, идёт бой, свитка нет; 3) кристаллы не собраны, идёт бой, свиток есть; 4) кристаллы не собраны, боя нет, свитка нет; 5) кристаллы собраны, идёт бой, свиток есть. В каких ситуациях портал откроется? Запиши номера по возрастанию.

4.6. Стражник у ворот говорит герою: «Неверно, что у тебя есть и меч, и щит». Что было у героя в его рюкзаке? Выбери все случаи, в которых стражник сказал правду: 1) есть меч и щит; 2) есть только меч; 3) есть только щит; 4) нет ни меча, ни щита. Запиши номера по возрастанию.

Задача 4.4 Сколько концовок у новеллы

дерево выборов, правило умножения, правило сложения

Катя пишет визуальную новеллу – историю, в которой игрок читает текст и иногда выбирает, что делать герою. В начале игрок выбирает спутника: кота, ворону или робота. Потом, у старого моста, решает: перейти мост или пойти в обход. В конце, у запертой двери, решает: открыть её или уйти. Ветки нигде не сходятся, и у каждого пути своя концовка.

а) Сколько концовок нужно написать Кате?

б) Катя решила сократить работу: если игрок пошёл в обход, герой сразу заблудился в лесу, и история на этом заканчивается (для каждого спутника – своя концовка «Заблудился»). Сколько концовок теперь?

Дано: развилка 1 – 3 варианта; развилка 2 – 2 варианта; развилка 3 – 2 варианта; ветки не сходятся.

Найти: а) число концовок; б) число концовок, если после «в обход» история кончается.

Решение

Шаг 1. Пункт а): выборы идут один за другим. После любого спутника есть мост, а после любого решения у моста – дверь. Значит, работает правило умножения: $3 \cdot 2 \cdot 2 = 12$ концовок.

Шаг 2. Пункт б): ветки стали разными. Разберём одного спутника, например кота. Если пойти через мост, будет ещё дверь – это 2 концовки. Если пойти в обход, история кончается сразу – это 1 концовка. Ветки разные, поэтому складываем: $2 + 1 = 3$ концовки у кота.

Шаг 3. Умножаем на число спутников. У вороны и у робота всё устроено так же, по 3 концовки. Выбор спутника – первый по очереди, и после любого из них одинаковое продолжение, поэтому умножаем: $3 \cdot 3 = 9$.

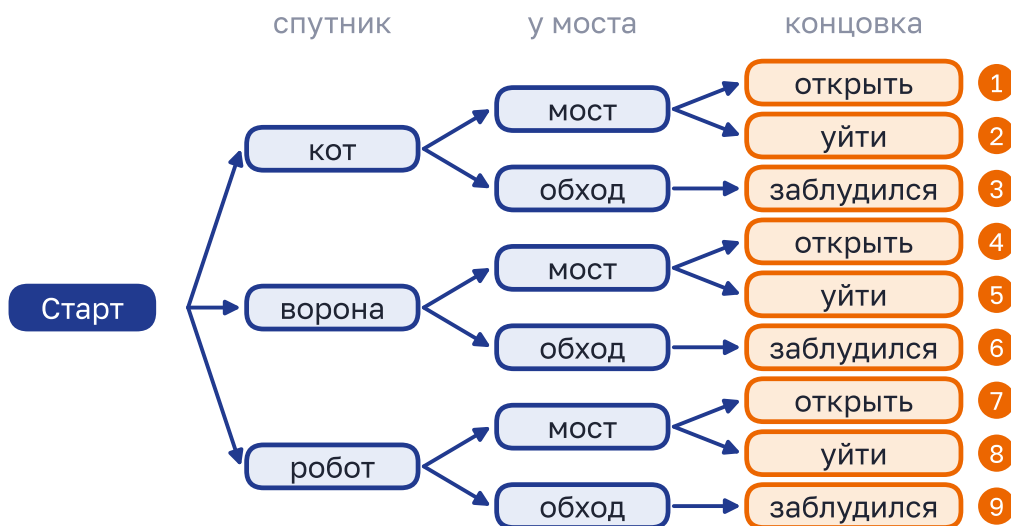


Рисунок 4.10 – Дерево выборов из пункта б). Оранжевые кружки – концовки, их 9

Сколько сэкономила Катя? $12 - 9 = 3$ концовки – по одной на спутника: вместо двух концовок «в обход» (открыть или уйти) у каждого осталась одна.

❗ **Типичная ошибка.** Сложить все варианты: $3 + 2 + 2 = 7$. Складывают, только когда выбирают одно из разных продолжений, а здесь выборы идут один за другим. А в пункте б) легко насчитать $3 \cdot 2 + 1 = 7$, будто ветка «Заблудился» одна на всех. В условии сказано: у каждого спутника своя.

Ответ: а) 12; б) 9.

Своя задача по программе сферы, модуль 4

Реши сам

4.7. В редакторе героя можно выбрать одну из 6 причёсок, один из 4 цветов костюма, одного из 3 питомцев, и ещё решить: носит герой очки или нет. Сколько разных героев можно собрать?

4.8. Сейф в квесте открывается, если дёрнуть четыре рычага по одному разу в правильном порядке. Игрок не знает порядка и пробует все подряд, не повторяясь. На одну попытку уходит 10 секунд. а) Сколько всего разных порядков? б) Сколько минут уйдёт в худшем случае, если правильный порядок окажется последним?

Задача 4.5* Карта сюжета

ветки сходятся, число путей на карте, правило умножения

Сценарист нарисовал карту сюжета. Стрелки показывают, куда можно пойти из каждого места, назад вернуться нельзя. Со старта – в Лес или к Реке. Из Леса – в Пещеру или в Деревню. От Реки – в Деревню или к Мельнице. Из Пещеры – в Замок. Из Деревни – в Замок или в Башню. От Мельницы – в Башню. Из Замка и из Башни – к Финалу.

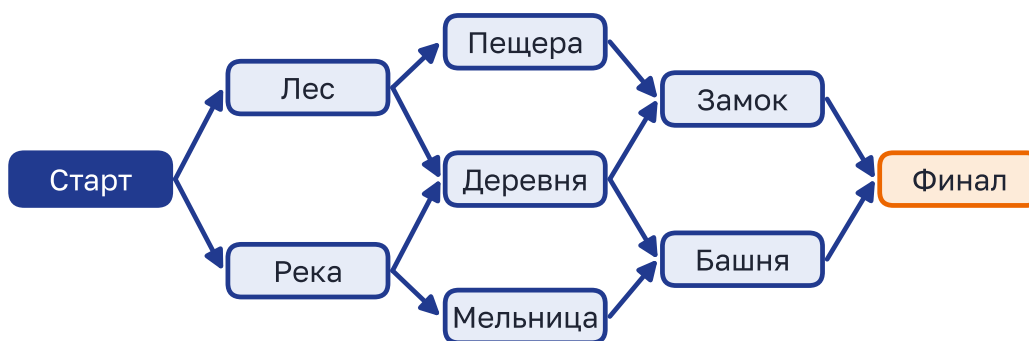


Рисунок 4.11 – Карта сюжета. Стрелки – куда можно пойти

а) Сколько разных путей ведёт от Старта к Финалу?

б) Сколько из этих путей проходят через Деревню?

Дано: карта из 9 мест, переходы – по стрелкам.

Найти: а) число путей от Старта к Финалу; б) число путей через Деревню.

Решение

Шаг 1. Почему нельзя просто умножить. Ветки сходятся: в Деревню ведут две дороги, в Замок – две, в Башню – две. Деревя здесь нет, есть карта. Поэтому считаем пути по местам, от старта к финалу.

Шаг 2. Пишем у каждого места число путей к нему. У Старта – 1. Дальше идём слева направо: у каждого места складываем числа тех мест, откуда в него ведут стрелки. Место считаем только тогда, когда посчитаны все, откуда в него приходят.

Таблица 4.8

Место	Откуда ведут стрелки	Путей
Старт	–	1
Лес, Река	из Старта	по 1
Пещера	из Леса	1
Мельница	от Реки	1
Деревня	из Леса, от Реки	$1 + 1 = 2$
Замок	из Пещеры, из Деревни	$1 + 2 = 3$
Башня	из Деревни, от Мельницы	$2 + 1 = 3$
Финал	из Замка, из Башни	$3 + 3 = 6$

Шаг 3. Читаем ответ. У Финала 6 – столько путей ведёт от Старта к Финалу.

Шаг 4. Пункт б): путь через Деревню – это два куска. Первый кусок – от Старта до Деревни, таких путей 2 (через Лес и через Реку). Второй кусок – от Деревни до Финала: через Замок или через Башню, тоже 2. Любой первый кусок можно склеить с любым вторым, и по правилу умножения выходит $2 \cdot 2 = 4$ пути.

Для верности выпишем все 6 путей: Лес – Пещера – Замок; Лес – Деревня – Замок; Лес – Деревня – Башня; Река – Деревня – Замок; Река – Деревня – Башня; Река – Мельница – Башня.

Через Деревню идут 4 из них, мимо – 2, как мы и насчитали.

! **Самая заманчивая ошибка** – решить, что на пути три развилки по два варианта (у Старта, потом у Леса или Реки, потом у Деревни), и ответить $2^3 = 8$. Правило умножения работает, только если ветки не сходятся, а здесь они сходятся трижды. В пункте б) можно сложить $2 + 2 = 4$ и попасть в ответ по чистой случайности. На другой карте это не сработает: куски пути склеивают умножением.

Ответ: а) 6; б) 4.

Своя задача по программе сферы, модуль 4

Реши сам

4.9. На карте сюжета со Старта можно пойти в Лес, в Поле или на Берег. Из Леса – в Овраг или на Мост. Из Поля – на Мост. С Берега – на Мост или к Пристани. Из Оврага – в Замок. С Моста – в Замок или к Пристани. От Пристани – в Замок. Назад возвращаться нельзя. Сколько разных путей ведёт от Старта к Замку?

4.10.* На карте из задачи 4.9 сколько путей от Старта к Замку проходят через Мост?

Задача 4.6 Луч по базе

вероятность как доля, площадь, «1 из N», среднее число попаданий

Игровое поле – прямоугольник 40×25 клеток. Игрок тайно построил на нём прямоугольную базу 8×5 клеток. Противник не видит базу и бьёт магическим лучом в случайную клетку поля – любую с одинаковым шансом.

а) Какова вероятность, что луч попадёт в базу? Ответ в процентах.

б) Это «1 из скольких»?

в) За игру противник бьёт 50 раз. Сколько попаданий в базу будет в среднем?

Дано: поле 40×25 клеток; база 8×5 клеток; все клетки равновозможны; 50 ударов.

Найти: а) P в %; б) N в «1 из N »; в) среднее число попаданий.

Решение

Шаг 1. Считаем все исходы. Луч может попасть в любую клетку поля: $40 \cdot 25 = 1000$ клеток – это n .

Шаг 2. Считаем подходящие исходы. Подходят клетки базы: $8 \cdot 5 = 40$ клеток – это m .

Шаг 3. Вероятность. $P = m/n = 40/1000 = 0,04$. В процентах: $0,04 \cdot 100 = 4$ %.

Шаг 4. «1 из скольких». $100 : 4 = 25$ – один удар из 25. Можно и так: $1000 : 40 = 25$ – поле в 25 раз больше базы.

Шаг 5. Среднее число попаданий. $50 \cdot 0,04 = 2$. Это в среднем: в одной игре противник попадёт 0 раз, в другой – 4, а если сыграть много игр, выйдет около двух на игру.

Прикинем. База – маленький кусок поля, так что 4 % похоже на правду. А 50 ударов – это дважды по 25, отсюда и 2 попадания.

❗ Не складывай стороны вместо площадей: $(8 + 5) : (40 + 25) = 13 : 65 = 0,2$ – в пять раз больше правды. Шанс зависит от того, **сколько клеток** занимает база, то есть от площади. А в пункте б) рука сама пишет 4, но это проценты, а спрашивают «1 из скольких».

Ответ: а) 4 %; б) 1 из 25; в) 2.

По мотивам задачи НТО 2025/26, профиль «Разработка компьютерных игр» (старший трек, 8–11 классы), первый отборочный этап, инженерный тур, «Геймдизайн. База»

Реши сам

4.11. В игре про рыбалку на Обском море при каждом забросе удочки вытаскиваешь рыбу с шансом 60 %, водоросли – с шансом 30 %, а в остальных случаях – сундук с сокровищем. а) Каков шанс вытащить сундук, в процентах? б) Это «1 из скольких»? в) Сколько сундуков в среднем вытащит игрок за 50 забросов?

4.12.* Поле – прямоугольник 40×25 единиц. Игрок построил на нём круглую базу радиусом 5 единиц, база не выходит за края поля. Противник бьёт лучом в случайную точку поля. Какова вероятность попасть в базу? Ответ в процентах, округли до целых.

Задача 4.7* Редкий питомец

независимые попытки, ошибка игрока, «хотя бы раз», счётчик неудач

После победы над боссом с шансом 10 % выпадает редкий питомец – дракончик. Шанс одинаковый при каждой победе. Друг Димы победил босса 9 раз, и дракончик ни разу не выпал. «Всё, в десятый раз точно выпадет», – говорит друг.

а) Какова вероятность, что дракончик выпадет при десятой победе? Ответ в процентах.

б) Какой была вероятность ещё до первого боя, что за 10 побед дракончик выпадет хотя бы раз? Ответ в процентах, округли до целых.

в) Разработчики добавили правило: «если дракончик не выпал 9 раз подряд, при десятой победе он выпадет точно». Какова теперь вероятность для друга Димы при десятой победе? Ответ в процентах.

Дано: $p = 10 \% = 0,1$ за одну победу; уже 9 неудач; $n = 10$ побед.

Найти: а) шанс на 10-й победе; б) $P_{\text{хотя бы раз}}$ за 10 побед; в) шанс на 10-й победе со счётчиком.

Решение

Шаг 1. Пункт а). По условию шанс одинаковый при каждой победе, счётчика неудач в правилах нет. Игра не помнит девять прошлых боёв. Значит, при десятой победе шанс тот же – 10 %.

Шаг 2. Пункт б): считаем через «ни разу». Шанс, что дракончик не выпадет в одной победе: $1 - 0,1 = 0,9$. Что не выпадет ни разу за 10 побед – 0,9, умноженное само на себя 10 раз. Считаем по шагам: $0,9^2 = 0,81$; $0,9^5 = 0,81 \cdot 0,81 \cdot 0,9 = 0,59049$; $0,9^{10} = 0,59049 \cdot 0,59049 \approx 0,3487$.

Шаг 3. «Хотя бы раз» – всё остальное. $1 - 0,3487 = 0,6513$, то есть $65,13\% \approx 65\%$.

Шаг 4. Пункт в). Теперь в правилах есть счётчик. Друг уже проиграл 9 раз подряд, значит, по новому правилу при десятой победе дракончик выпадет обязательно. Вероятность – 100 %.

Как 10 % и 65 % уживаются вместе? До начала серии шанс «хотя бы раз за 10» был 65 % – неплохой, но далеко не 100 %. У друга же серия уже почти закончилась: остался один бой, а у одного боя шанс 10 %. Примерно каждый третий игрок (около 35 %) так и проходит все 10 боёв без дракончика. Для них разработчики и вводят гарантию.

 **Главная ловушка** – ответить в пункте а) 100 %: «шанс 1 из 10, а 9 раз уже было». Это ошибка игрока: шанс «1 из 10» – в среднем, а не по расписанию. В пункте б) тянет сложить: $10 \cdot 10 = 100\%$. Если бы шансы складывались, за 11 побед вышло бы 110 %, а больше 100 % не бывает. И не забудь вычесть $0,9^{10}$ из единицы: 35 % – это шанс «ни разу», а не «хотя бы раз».

Ответ: а) 10 %; б) 65 %; в) 100 %.

Своя задача по программе сферы, модуль 1

Реши сам

4.13.* Из сундука с шансом 20 % выпадает золотой ключ. Шанс одинаковый для каждого сундука, счётчика неудач в игре нет. Игрок открыл 4 сундука, и все пустые. а) Каков шанс найти ключ в пятом сундуке? Ответ в процентах. б) Какой была вероятность ещё до первого сундука, что хотя бы в одном из пяти найдётся ключ? Ответ в процентах, округли до целых.

4.14. В настольной игре бросают два кубика и складывают очки. Лиза бросила три раза, и сумма 7 ни разу не выпала. Какова вероятность, что при четвёртом броске сумма будет 7? Ответ десятичной дробью, округли до сотых.

Машины состояний: как думает NPC



Стражник в игре замечает тебя, бежит следом, бьёт, а потеряв из виду, возвращается к обходу. Кажется, что он думает. А у него всего лишь несколько режимов и правила, по которым он прыгает из одного в другой. Такая схема называется машиной состояний. В «Урсуле» персонажей программируют так же, на языке ПРИМС. Задачи по таким схемам похожи одна на другую: дают схему и список событий, а спрашивают, где окажется персонаж, сколько раз что-то сработает или какая стрелка лишняя. Ответ – номер состояния, число или строка цифр.

5.1. Состояние и переход

НА ПАЛЬЦАХ



Рисунок 5.1 – Ночник с одной кнопкой

У ночника два режима, и кнопка переводит его из одного в другой. Быть сразу и включённым, и выключенным он не может.

Возьми ночник с одной кнопкой. Он бывает в двух режимах: выключен или включён. Нажал кнопку – режим сменился. Нажал ещё раз – вернулся обратно. Больше ночник ничего не умеет, и всё его поведение помещается в картинку сверху.

Такая картинка называется **машиной состояний**. Ещё её называют **конечным автоматом** (по-английски finite state machine). «Конечный» значит, что режимов не бесконечно много, их можно пересчитать. Деталей в машине всего четыре:

- **Состояние** – режим, в котором объект находится сейчас и делает только определённые вещи: «выключен», «спит», «патрулирует». Рисуем скруглённым прямоугольником, внутри – имя.
- **Переход** – смена одного состояния другим. Рисуем стрелкой.
- **Событие** – то, что случилось и заставило сменить состояние: «нажали кнопку», «игрок обнаружен», «таймер выполнен». Пишем на стрелке.
- **Начальное состояние** – то, в котором объект оказывается при запуске игры. К нему ведёт стрелка от чёрной точки.

Главное правило: **в каждый момент объект находится ровно в одном состоянии**. Человек или спит, или не спит. Ноутбук или в спящем режиме, или работает. Если по твоей схеме выходит, что стражник сейчас и «гонится», и «патрулирует», в схеме ошибка.

Имя состояния отвечает на вопрос «что объект делает прямо сейчас?». «Ждёт игрока», «бежит к двери», «открыт» – хорошие имена. «Состояние 1» или «Новый узел» – плохие: через неделю ты сам не вспомнишь, что там происходит. Если состояния названы точно, схему поймёт даже тот, кто языка программирования не знает.

Пример посложнее – светофор на переходе у школы. Кнопки здесь нет, события светофор создаёт сам. Строчка «вход: таймер 30 с» значит: как только загорелся красный, светофор за-

водит **таймер** – будильник на 30 секунд. Время вышло – приходит событие «таймер выполнен», и светофор переходит к следующему сигналу. Подробнее про «вход» – в разделе 5.2.

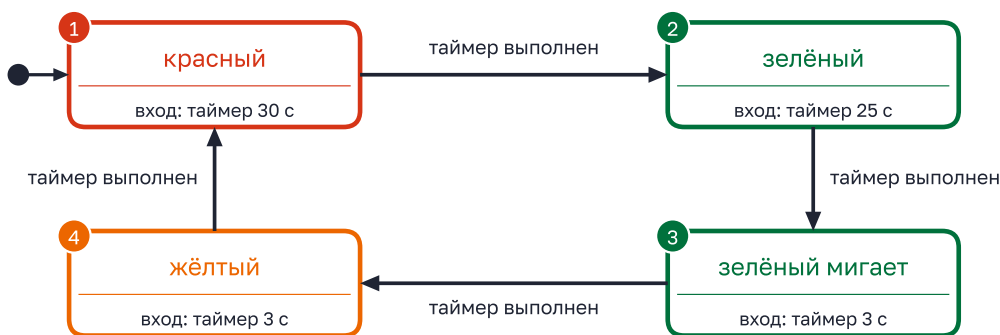


Рисунок 5.2 – Светофор на переходе. Каждое состояние на входе заводит свой таймер, а по событию «таймер выполнен» светофор переходит дальше по кругу

Машины состояний прячутся почти в каждой игре. Привидения в старой аркаде Рас-Ман то гонятся за героем, то разбегаются по углам поля, а когда Пакман съедает большую точку, синеют и ударают. Крипер в Minecraft бродит, замечает игрока, подходит, начинает шипеть – и взрывается, если ты не успел отбежать. Каждое «то, то, то» – отдельное состояние.

Запомни. Машина состояний – это прямоугольники-состояния, стрелки-переходы и события на стрелках. Объект всегда находится только в одном состоянии. Стрелка от чёрной точки показывает, с чего всё начинается.

5.2. Подпись на стрелке: событие [условие] / действие

НА ПАЛЬЦАХ

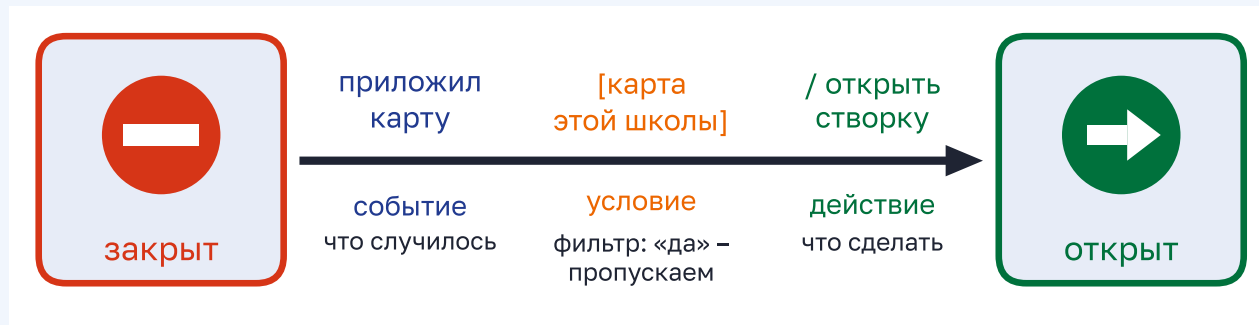


Рисунок 5.3 – Турникет у входа в школу

Приложил карту – это событие. Турникет проверяет, своя ли карта, – это условие. Если своя, открывает створку – это действие.

Турникет в школе реагирует не на любую карту. Приложишь проездной на автобус – створка останется закрытой. Одного события для такого правила мало. Поэтому полная подпись на стрелке в нашей книге всегда выглядит так:

событие [условие] / действие

- **Событие** – что случилось. Его пишут как уже произошедшее: «нажата кнопка», «игрок обнаружен», «перемещение выполнено».
- **Условие** – в квадратных скобках. Это фильтр: переход сработает, только если в этот момент условие верно. На вебинаре сферы его называли **ограждающим условием**. Условие неверно – объект остаётся, где был.
- **Действие** – после косой черты. Что объект делает в момент перехода: открыть створку, показать реплику, выдать монеты.

Условия и действия может не быть. Подпись «таймер выполнен» – одно событие, без фильтра и без действия. А вот событие есть почти всегда: без него непонятно, когда переходить.

Где ещё живут действия. Действие можно поставить не только на стрелку, но и внутрь состояния. Внутри прямоугольника под чертой пишут строки трёх видов:

- **вход:** ... – выполняется один раз, в момент, когда объект попал в это состояние. Светофор на входе в «красный» заводит таймер.
- **выход:** ... – один раз, в момент, когда объект уходит из состояния. Удобно, чтобы убрать за собой: погасить надпись, выключить индикатор.
- **событие / действие** без стрелки – это **внутреннее событие**. Действие выполняется каждый раз, когда событие случилось, а состояние не меняется. «Вход» и «выход» при этом не срабатывают.

Или умная лампочка. Когда ей меняют цвет с тёплого на холодный, новое состояние не нужно: лампочка как была включена, так и осталась. Поэтому смену цвета пишут внутренним событием в состоянии «включена»: «цвет изменён / задать цвет».

Вот дверь в подвал из квеста. Состояния пронумерованы – так потом проще записывать ответ.

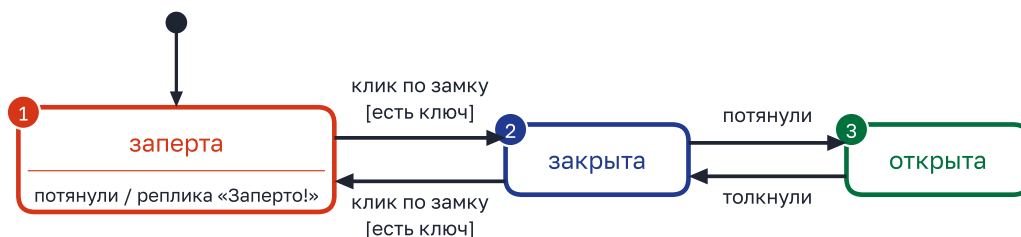


Рисунок 5.4 – Дверь в подвал. В состоянии «заперта» есть внутреннее событие: потянул дверь – увидел реплику, а дверь осталась запертой

Читаем схему. Сначала дверь заперта. Клик по замку срабатывает, только если у героя есть ключ, – об этом говорит условие в скобках. Тогда дверь становится просто закрытой, и её можно открыть, потянув за ручку. Если потянуть запертую дверь, сработает внутреннее событие: появится реплика «Заперто!», а дверь так и останется запертой.

Теперь по шагам: что машина делает, когда приходит одно событие. На этих пяти шагах держатся все задачи главы, так что их стоит выучить.

1. Смотрим на текущее состояние и ищем стрелку, которая из него **выходит** и подписана этим событием.
2. Если на стрелке есть условие, проверяем его. Верно – стрелка подходит, неверно – не подходит.

3. Нашлась подходящая стрелка – по порядку выполняем «выход» старого состояния, действие на стрелке и «вход» нового.
4. Стрелки нет, но в состоянии есть внутреннее событие с таким названием – выполняем его действие и остаёмся на месте.
5. Ничего не нашлось – событие пропадает. Машина не запоминает его «на потом».

Запомни. Подпись на стрелке: событие [условие] / действие. «Вход» и «выход» срабатывают по одному разу – когда попали в состояние и когда ушли из него. Внутреннее событие действует, но состояние не меняет. Нет подходящей стрелки – событие пропало.

Совет. Прежде чем трассировать чужую схему, выпиши, какие события в ней вообще встречаются. Событие, которого нет ни на одной стрелке и ни в одном состоянии, можно сразу вычеркнуть из списка: оно ни на что не влияет.

5.3. Как думает NPC: стражник и торговец

НА ПАЛЬЦАХ

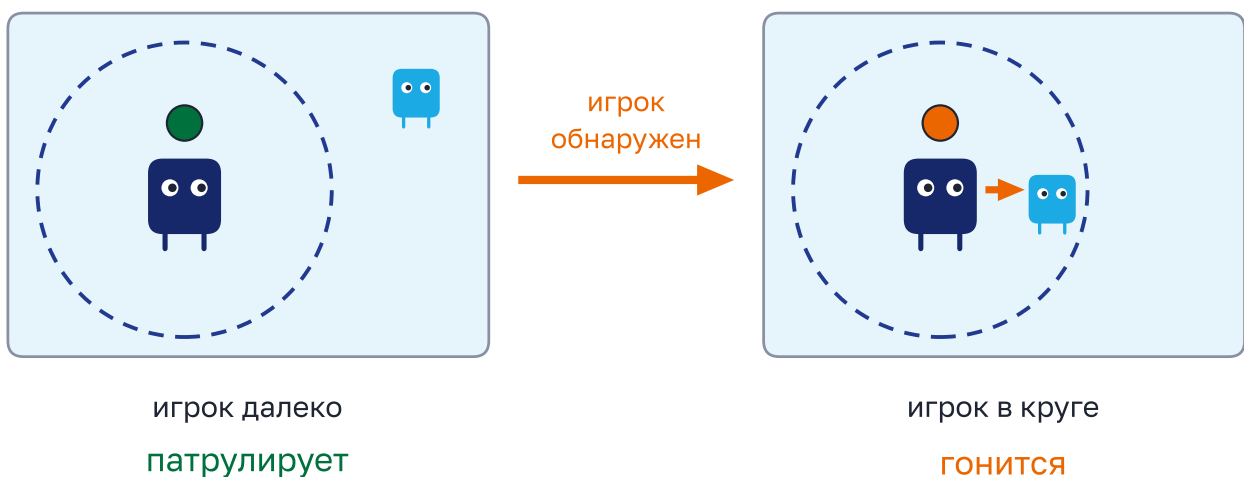


Рисунок 5.5 – Стражник и круг, в котором он замечает игрока

У стражника два режима: обход и погоня. Переключает их одно событие – игрок вошёл в круг. Шар над головой меняет цвет, и по нему видно, какой режим сейчас включён.

NPC – персонаж, которым управляет не игрок, а сама игра (подробнее – в главе 1). Поведение любого NPC раскладывается на состояния. Нарисуем стражника ворот. Почти такого же робота-охранника собирали в «Урсуле» на вебинаре сферы.

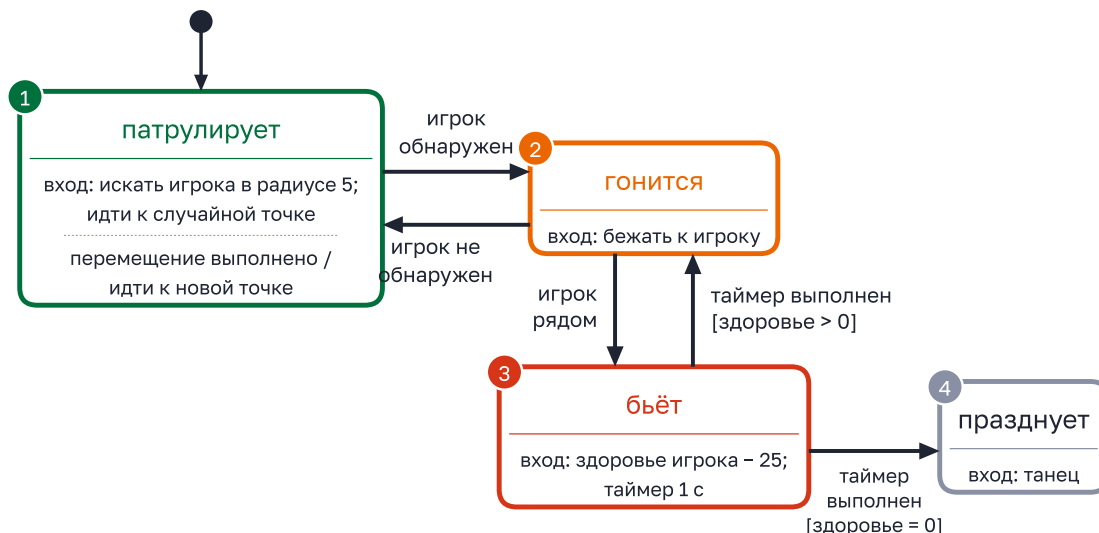


Рисунок 5.6 – Стражник ворот. В состоянии 1 под пунктиром – внутреннее событие: дошёл до точки – выбрал новую

Идём по схеме.

- **1 «патрулирует».** На входе стражник включает поиск игрока в радиусе 5 и идёт к случайной точке. Дошёл – приходит событие «перемещение выполнено», и он выбирает новую точку. Это внутреннее событие: состояние не меняется, стражник просто ходит и ходит.
- **2 «гонится».** Игрок вошёл в круг – событие «игрок обнаружен». Стражник бежит к нему. Если игрок выбежал из круга (событие «игрок не обнаружен»), стражник возвращается к обходу.
- **3 «бьёт».** Догнал – событие «игрок рядом». На входе один удар: у игрока минус 25 здоровья. Сразу заводится таймер на 1 секунду – пауза между ударами.
- Таймер вышел – развилка по условию. Здоровье ещё есть – обратно в «гонится», а от туда при следующем «игрок рядом» снова в «бьёт». Здоровья не осталось – в **4 «празднует».**

Из «празднует» не выходит ни одной стрелки: для героя игра закончилась. Про такие состояния – в разделе 5.5.

Торговец устроен ещё проще. Он стоит за прилавком в состоянии «ждёт». Игрок подошёл – торговец переходит в «торгует» и говорит: «Зелья по 30 монет!» (это действие на стрелке). Внутри «торгует» два внутренних события: «купить [монет не меньше 30] / минус 30 монет, выдать зелье» и «купить [монет меньше 30] / реплика «Не хватает монет»». Игрок ушёл – торговец снова «ждёт». Покупка режим торговца не меняет, поэтому она записана внутренним событием, а не стрелкой.

Как это делают в «Урсуле». Так было в сентябре 2026 года, на вебинаре сферы по ПРИМС; интерфейс может обновиться.

- Поведение рисуют в отдельном редакторе «Кибериада Graph Editor». «Создать новый граф» – начать новую программу, «создать новый нод» – добавить состояние. Готовую схему сохраняют файлом и загружают в карточку модели в «Урсуле», в поле «файл алгоритма».

- Команды и события собраны в **модули**: обнаружение, движение, индикация (цветной шар над объектом), вывод текста (реплики), взаимодействие (проходимость, «уничтожить объект»), атака, здоровье, триггеры, переменные, победа и поражение и другие.
- Одна программа работает на всех копиях модели на карте. Если нужно, чтобы один ключ был настоящим, а остальные ложными, делают две разные модели с разными именами.

Состояния стражника в ПРИМС записываются примерно так. Названия команд в редакторе могут звучать чуть иначе, смысл тот же.

«патрулирует»

вход: Обнаружение – искать игрока в радиусе 5
 Индикация – цвет зелёный, включить индикатор
 Движение – случайная точка в радиусе 5
 Движение – двигаться по координатам
 событие «перемещение выполнено»:
 Движение – случайная точка в радиусе 5
 Движение – двигаться по координатам
 переход «игрок обнаружен» → «гонится»

«гонится»

вход: Индикация – цвет оранжевый, включить индикатор
 Движение – двигаться к объекту: игрок
 переход «игрок не обнаружен» → «патрулирует»
 переход «игрок рядом» → «бьёт»

Новички в этой записи чаще всего спотыкаются о две вещи. **Модуль сам не включается**: на вебинаре робот сначала вообще не замечал игрока – в «патрулирует» забыли команду поиска, и событию «игрок обнаружен» просто неоткуда было взяться. И наоборот, **включённая команда работает и после ухода из состояния**, пока её не выключат. Поиск, включённый в «патрулирует», продолжает работать в «гонится» – поэтому там и может прийти «игрок не обнаружен». Если в каком-то состоянии поиск нужно остановить, это пишут отдельной командой на входе.

Запомни. Поведение NPC – его состояния и события между ними. То, что не меняет режим (дошёл до точки, купил зелье), записывают внутренним событием. Событие приходит только от включённого модуля: не включил поиск – «игрок обнаружен» не придёт никогда.

Совет. Дай каждому состоянию свой цвет индикатора. Тогда в игре сразу видно, в каком состоянии NPC, и ошибку в схеме находишь в разы быстрее. Потренироваться в ПРИМС можно на курсе «Программирование автономного поведения персонажей в видеоиграх» в «Урсуле».

5.4. Трассировка: куда приведут события

НА ПАЛЬЦАХ



Рисунок 5.7 – Кот-NPC, фишка и карточки-события

Трассировка похожа на настольную игру. Тянешь карточку и двигаешь фишку по стрелке с тем же словом. Если из клетки с фишкой такой стрелки нет, фишка стоит.

Трассировка – проследить работу программы шаг за шагом, как будто ты сам компьютер. Для машины состояний дают схему и список событий по порядку, а спрашивают, где окажется персонаж, сколько раз сработает действие, в каких состояниях он побывает.

Главная опасность – держать всё в голове. На третьем событии забываешь, где был после первого. Надёжнее завести таблицу и заполнять строку за строкой: какое состояние было, какое пришло событие, что произошло, какое состояние стало. Каждая новая строка начинается с того состояния, которым закончилась предыдущая.

Пример по схеме стражника из раздела 5.3. У игрока 100 здоровья. События приходят так: перемещение выполнено, игрок обнаружен, перемещение выполнено, игрок рядом, таймер выполнен.

Таблица 5.1

Шаг	Было	Событие	Что произошло	Стало
1	1	перемещение выполнено	внутреннее: новая точка	1
2	1	игрок обнаружен	переход	2
3	2	перемещение выполнено	в «гонится» его никто не ждёт – пропало	2
4	2	игрок рядом	переход, удар: $100 - 25 = 75$	3

Шаг	Было	Событие	Что произошло	Стало
5	3	таймер выполнен	$75 > 0$ – обратно в «гонится»	2

Итог: стражник гонится за игроком, у которого 75 здоровья. Посмотри на шаг 3. Событие пришло, но в состоянии «гонится» для него нет ни стрелки, ни внутреннего события, и оно пропадает. Как раз такие строки чаще всего пропускают – и сбиваются в номерах.

Когда событий очень много. Бывает, что машина сама ходит по кругу, по таймерам, и спрашивают, где она будет через тысячу секунд. Трассировать тысячу шагов не нужно – хватит деления с остатком.

НА ПАЛЬЦАХ



Рисунок 5.8 – Урок физкультуры: дорога до дорожки и круги по стадиону

До дорожки ты идёшь один раз – это хвост. Дальше бежишь круг за кругом. Целые круги можно выбросить: после каждого ты снова в той же точке.

Начнём с машины без хвоста. Светофор из раздела 5.1 проходит круг «красный → зелёный → мигает → жёлтый» за $30 + 25 + 3 + 3 = 61$ с, дальше всё повторяется. Где он будет через 130 с? Делим с остатком: $130 = 2 \cdot 61 + 8$. Два целых круга ничего не меняют, поэтому через 130 с светофор в той же точке круга, что и через 8 с. Красный горит с 0-й по 30-ю секунду круга – значит, горит красный.

Часто перед кругом есть **хвост** – состояния, которые проходят один раз, в самом начале: «спит», «просыпается». Как дорога от раздевалки: по ней не бегают кругами. Тогда сначала вычитают время хвоста и только потом отбрасывают круги. Так устроена задача 5.6.

Запомни. Трассируй по таблице: было – событие – что произошло – стало. Каждое событие – отдельная строка, даже если оно пропало. Если машина ходит по кругу, отбрось целые круги делением с остатком.

Совет. В задачах с таймерами проверь, не попадает ли спрошенный момент ровно на смену состояний. На 30-й секунде красный как раз сменяется зелёным. Если попадает – перечитай условие: там должно быть сказано, как считать такой момент.

5.5. Тупики, лишние и недостижимые переходы

НА ПАЛЬЦАХ

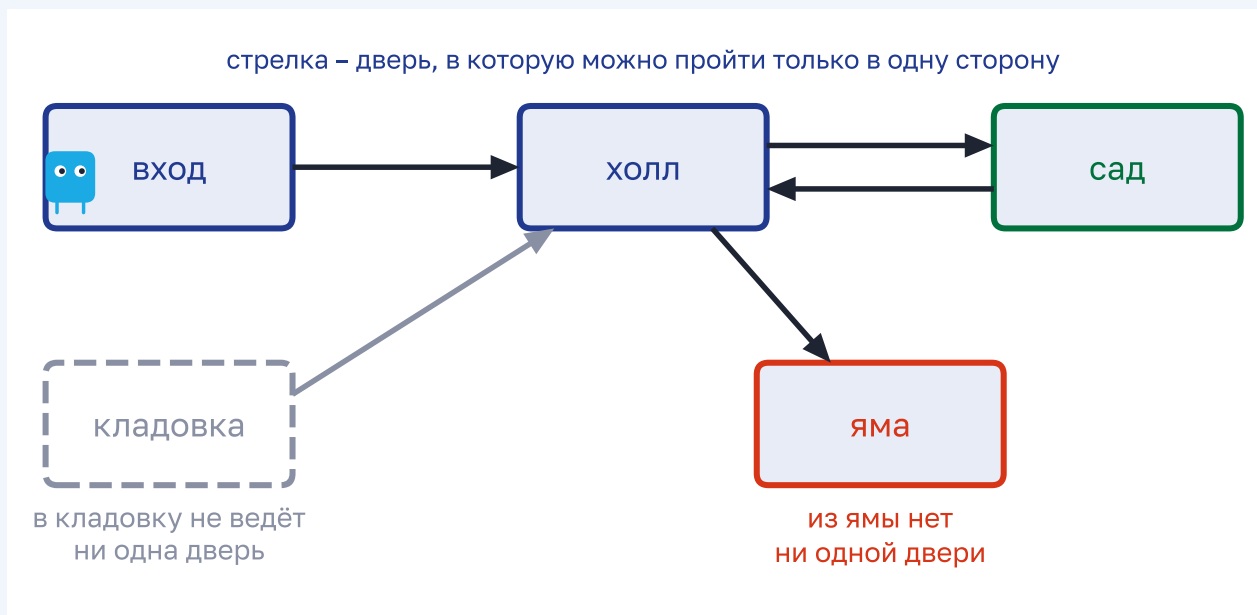


Рисунок 5.9 – Дом с дверями, которые открываются только в одну сторону

От входа можно дойти до холла, сада и ямы. В кладовку не попасть никак. А из ямы не выбраться.

Схема может быть нарисована аккуратно и всё равно работать неправильно. Ошибки в таких схемах попадают одни и те же, их ловят и в «кейсе по отладке чужого кода» из программы сферы.

- **Недостижимое состояние** – в него нельзя попасть из начального ни по какой цепочке стрелок, как в кладовку на картинке. Персонаж никогда не окажется в этом состоянии, и всё, что там написано, не сработает ни разу.
- **Тупик** – состояние, из которого не выходит ни одной стрелки. Попал – остался навсегда. Иногда так и задумано: «победа», «поражение», «празднует». А иногда это ошибка: если стражнику забыли нарисовать стрелку «игрок не обнаружен», он будет гнаться вечно.
- **Лишний переход** – стрелка, которая никогда не сработает. Так бывает, если её условие в этом состоянии не может быть верным, или если её событие в этом состоянии не может прийти: поиск игрока выключен, а на стрелке «игрок обнаружен». Стрелка из недостижимого состояния – тоже лишняя: выходить из него некому.

Ещё одна неприятность – **две стрелки из одного состояния по одному событию**, у которых условия могут оказаться верными одновременно (или условий нет вовсе). Какая сработает?

По схеме не понять. Условия на таких стрелках должны исключать друг друга, как у стражника: [здоровье > 0] и [здоровье = 0].

Как найти недостижимые состояния. Работает та же волна, что для комнат в главе 3.

1. Обведи начальное состояние.
2. Посмотри на стрелки, которые выходят из обведённых состояний, и обведи, куда они ведут. Лишние стрелки, которые никогда не срабатывают, пропускай.
3. Повторяй, пока обводится что-то новое.
4. Всё, что осталось необведённым, – недостижимо.

Тупики ищутся ещё проще: у каждого состояния посчитай стрелки, которые из него **выходят**. Внутренние события не в счёт – они никуда не ведут. Ноль стрелок – тупик. Потом реши по смыслу, задуман он или нет.

Запомни. Недостижимое – не попасть. Тупик – не выйти. Лишний переход – никогда не сработает. Тупик бывает задуманным (конец игры), недостижимое состояние – почти всегда ошибка.

Совет. Проверяй стрелки «туда» и «обратно» парами. Есть «игрок обнаружен» – ищи рядом «игрок не обнаружен». Есть «открыть» – ищи «закрыть». Стрелка без пары – первый кандидат в ошибку.

5.6. Вложенные состояния: одна стрелка вместо многих

НА ПАЛЬЦАХ

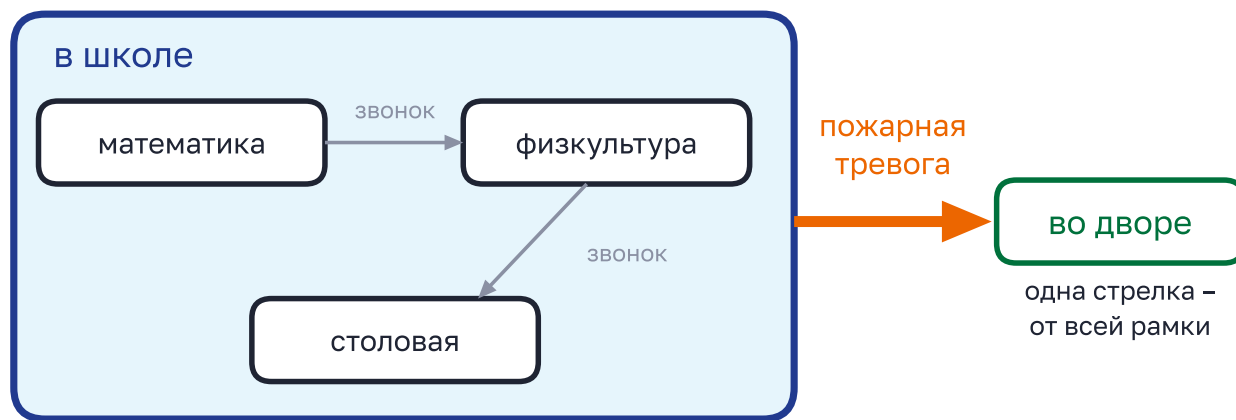


Рисунок 5.10 – Школа и пожарная тревога

На каком бы уроке ты ни был, по тревоге все выходят во двор. Рисовать стрелку от каждого кабинета не нужно – хватит одной, от всей рамки «в школе».

Вернёмся к роботу-охраннику с вебинара сферы. Кусты травы в той игре – островок безопасности: игрок прячется в них, и трава выпускает **триггер** – сигнал, который получают объекты

вокруг. Робот по триггеру должен замереть. Но в каком состоянии триггер его застанет? Может, робот патрулирует, а может, уже гонится. Пришлось бы тянуть стрелку «триггер активирован» из каждого состояния: два состояния – две стрелки, восемь состояний – восемь. Схема превращается в клубок.

Выход – **вложенные**, или **иерархические**, состояния. Отсюда буква «И» в названии ПРИМС: программирование расширенных иерархических машин состояний. Несколько состояний кладут внутрь одного большого – **родительского** (его ещё называют составным). Те, что внутри, – **дочерние**.

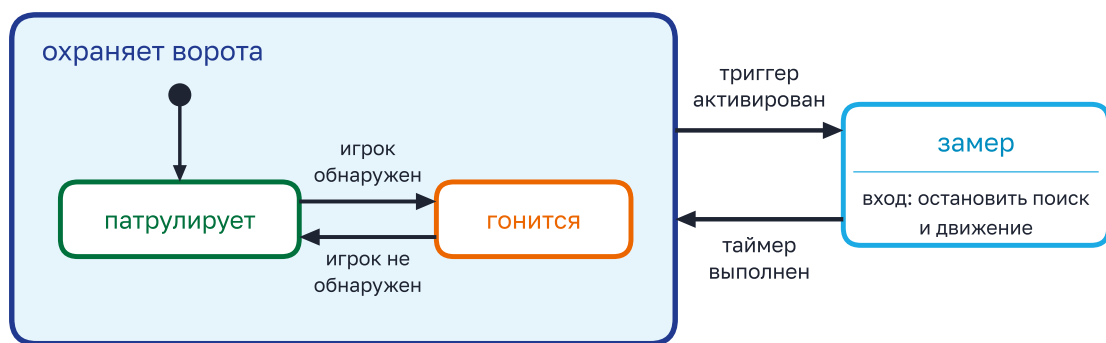


Рисунок 5.11 – Робот-охранник. «Патрулирует» и «гонится» лежат внутри родительского состояния «охраняет ворота». Стрелка «триггер активирован» выходит из рамки, поэтому работает из любого дочернего состояния

Правила чтения такие.

- Объект в дочернем состоянии находится заодно и в родительском. Робот, который гонится, в то же время «охраняет ворота».
- Стрелка, которая выходит из рамки родителя, срабатывает, в каком бы дочернем состоянии ни был объект. Робот замрёт по триггеру и во время обхода, и во время погони.
- Стрелка, которая входит в рамку родителя, ведёт в **начальное дочернее** состояние – к нему внутри рамки идёт стрелка от своей чёрной точки. После «замер» робот вернётся не туда, где его застал триггер, а в «патрулирует». В схемах этой книги родитель не помнит, где объект был раньше: вход в рамку всегда ведёт в начальное дочернее.

Сколько стрелок экономим (стрелки от чёрных точек не считаем)? Без родителя роботу нужны две стрелки «триггер активирован» – из «патрулирует» и из «гонится». Вместе с остальными выходит $2 + 2 + 1 = 5$ стрелок. С родителем $2 + 1 + 1 = 4$. Для двух дочерних состояний выигрыш маленький. А у босса с пятью приёмами и тремя общими событиями («ранен», «игрок сбежал», «повержен») выходит уже 15 стрелок против 3.

Так же на вебинаре объясняли ночной режим светофора. Четыре дневных сигнала кладут в родителя «дневной режим», и одна стрелка «нажата кнопка «ночь»» из его рамки переводит светофор в «мигает жёлтым», какой бы сигнал ни горел. В «Кибериаде» состояние делали дочерним кнопкой «Сделать дочерним» на нижней панели: нажимаешь её и выбираешь, в какое состояние вложить.

Запомни. Стрелка из рамки родителя действует из любого дочернего состояния. Стрелка в рамку ведёт в начальное дочернее. Общее для нескольких состояний событие – повод сложить их в одного родителя.

5.7. Разбираем задачи

Задача 5.1 Дверь в подвал

разминка · трассировка, условие на переходе, внутреннее событие

Дверь в подвал работает по схеме из раздела 5.2. У героя есть ключ. События приходят по порядку: 1) потянули; 2) клик по замку; 3) толкнули; 4) потянули; 5) толкнули; 6) клик по замку; 7) потянули.

В каком состоянии дверь окажется в конце и сколько раз игрок увидит реплику «Заперто!»? Запиши номер состояния и число подряд, без пробелов.

Дано: схема двери (раздел 5.2), начальное состояние 1 «заперта», ключ есть; семь событий.

Найти: номер состояния в конце; сколько раз была реплика «Заперто!».

Решение

Шаг 1. Что происходит. У двери три состояния. Реплика «Заперто!» – внутреннее событие в состоянии 1: она появляется, когда дверь потянули, пока та заперта. Условие [есть ключ] у нас всегда верно. Трассируем по таблице.

Шаг 2. Таблица.

Таблица 5.2

Шаг	Было	Событие	Что произошло	Стало
1	1	потянули	«Заперто!» (1-й раз)	1
2	1	клик по замку	ключ есть – переход	2
3	2	толкнули	нет стрелки – пропало	2
4	2	потянули	переход	3
5	3	толкнули	переход	2
6	2	клик по замку	ключ есть – переход	1
7	1	потянули	«Заперто!» (2-й раз)	1

Шаг 3. Ответ. Дверь в состоянии 1, реплика была 2 раза. Записываем подряд: 12.

Представь это в игре. Герой отпер дверь, открыл её, закрыл, снова запер своим ключом – и потянул. Конечно, заперто. Реплика появлялась, только пока дверь была заперта: в самом начале и в самом конце.

❗ **Типичная ошибка.** На шаге 3 решить, что закрытая дверь от толчка открылась или заперлась. Из «закрыта» стрелка «толкнули» не выходит, событие пропадает. А ещё бывает, что на шаге 6 клик по замку считают бесполезным, хотя из «закрыта» он ведёт обратно в «заперта». С этой ошибкой дверь в конце окажется открытой, и выйдет ответ 31.

Ответ: 12 (дверь заперта, реплика «Заперто!» – 2 раза).

Своя задача по программе сферы, модуль 2

Реши сам

5.1. Та же дверь в подвал (раздел 5.2), но ключа у героя **нет**. События: клик по замку, потянули, потянули, толкнули, клик по замку, потянули. В каком состоянии дверь в конце и сколько раз появилась реплика «Заперто!»? Запиши номер состояния и число подряд, без пробелов.

5.2. Умная лампа: 1 «выключена» (начальное), 2 «белый свет», 3 «ночник». Переходы: 1 → 2 «кнопка»; 2 → 3 «кнопка»; 3 → 1 «кнопка»; 2 → 1 «хлопок»; 3 → 1 «хлопок». События: кнопка, хлопок, хлопок, кнопка, кнопка, кнопка, хлопок, кнопка. В каком состоянии лампа в конце? Запиши номер.

Задача 5.2 Бесконечный сундук

действия на входе, на выходе и на переходе, трассировка

Начинающий разработчик запрограммировал сундук с монетами (рисунок ниже). Игрок, не отходя, кликнул по сундуку 9 раз подряд. а) Сколько монет он получил? б) Сколько раз прозвучал звук «хлоп»?

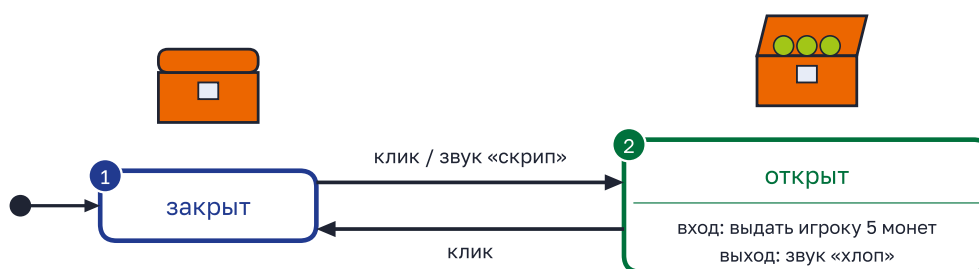


Рисунок 5.12 – Сундук с монетами. В состоянии «открыт» есть действия на входе и на выходе, на стрелке 1 → 2 – действие «звук «скрип»»

Дано: 1 «закрит» (начальное), 2 «открыт»; вход в 2 – выдать 5 монет; выход из 2 – звук «хлоп»; 9 кликов.

Найти: сколько монет; сколько раз «хлоп».

Решение

Шаг 1. За что дают монеты. Каждый клик переключает сундук: закрыт → открыт → закрыт → ... Монеты выдаются не за клик, а **на входе** в «открыт». Значит, нужно посчитать, сколько раз

сундук входил в «открыт». «Хлоп» звучит **на выходе** из «открыт» – посчитаем и выходы.

Шаг 2. Трассировка. Сундук начинает закрытым. Нечётные клики – 1-й, 3-й, 5-й, 7-й, 9-й – переводят его 1 → 2, это входы в «открыт». Чётные – 2-й, 4-й, 6-й, 8-й – переводят 2 → 1, это выходы из «открыт».

Шаг 3. Монеты. Входов в «открыт» 5, за каждый 5 монет: $5 \cdot 5 = 25$ монет.

Шаг 4. «Хлоп». Выходов из «открыт» 4. После 9-го клика сундук остался открытым, и в пятый раз из него ещё не выходили.

Проверим себя. Входов 5 и выходов 4 – вместе 9, как и кликов. Звук «скрип» на стрелке 1 → 2 тоже прозвучал 5 раз.

А теперь глазами геймдизайнера. Такой сундук можно «доить» бесконечно: закрыл, открыл – ещё 5 монет. Это настоящий баг геймдизайна: за минуту игрок накликает сотни монет. Лечится одной стрелкой: по клику из «открыт» вести не обратно в «закрыт», а в новое состояние «пуст», из которого выхода нет. Такой тупик задуман – сундук своё уже отдал.

 **Частая ошибка** – умножить все клики на 5 и получить 45: монеты даёт не клик, а вход в состояние. Или решить, что «вход» срабатывает один раз за всю игру, и ответить 5. «Вход» срабатывает при **каждом** попадании в состояние. А в пункте б) многие отвечают 5: забыли, что в конце сундук открыт.

Ответ: а) 25 монет; б) 4 раза.

Своя задача по программе сферы, модуль 2

Реши сам

5.3. Другой сундук: 1 «закрыт» (начальное) и 2 «открыт». Стрелка 1 → 2: «клик / выдать 2 монеты», стрелка 2 → 1: «клик / выдать 1 монету». Внутри состояний действий нет. Игрок кликнул 7 раз. Сколько монет он получил?

5.4. Исправленный сундук: 1 «закрыт» (начальное), 2 «открыт», 3 «пуст». Переходы: 1 → 2 «клик» и 2 → 3 «клик». В «открыт» на входе: выдать 5 монет. В «пуст» на входе: реплика «Пусто»; там же внутреннее событие «клик / реплика «Пусто»». Игрок кликнул 8 раз. а) Сколько монет он получил? б) Сколько раз увидел реплику «Пусто»?

Задача 5.3 Стражник ворот

NPC, трассировка, условие на переходе, пропавшее событие

Стражник работает по схеме из раздела 5.3. У игрока 100 здоровья, ниже нуля оно не опускается. События приходят по порядку: 1) игрок обнаружен; 2) игрок рядом; 3) игрок не обнаружен; 4) таймер выполнен; 5) игрок рядом; 6) таймер выполнен; 7) игрок не обнаружен; 8) перемещение выполнено; 9) игрок обнаружен; 10) игрок рядом; 11) таймер выполнен.

а) В каком состоянии стражник после всех событий? Запиши номер. б) Сколько здоровья осталось у игрока?

Дано: схема стражника (раздел 5.3), начальное состояние 1; удар – минус 25 здоровья; 100 здоровья; 11 событий.

Найти: номер состояния в конце; здоровье игрока.

Решение

Шаг 1. Где меняется здоровье. Здоровье меняется только в одном месте – на входе в «бьёт». Значит, считать удары нужно, когда стражник **входит** в состояние 3. Условия есть только на стрелках из «бьёт»: куда идти по таймеру, решает здоровье.

Шаг 2. Таблица.

Таблица 5.3

Шаг	Было	Событие	Что произошло	Стало
1	1	игрок обнаружен	переход	2
2	2	игрок рядом	переход, удар: $100 - 25 = 75$	3
3	3	игрок не обнаружен	из «бьёт» такой стрелки нет – пропало	3
4	3	таймер выполнен	$75 > 0$ – в «гонится»	2
5	2	игрок рядом	переход, удар: $75 - 25 = 50$	3
6	3	таймер выполнен	$50 > 0$ – в «гонится»	2
7	2	игрок не обнаружен	переход	1
8	1	перемещение выполнено	внутреннее: новая точка	1
9	1	игрок обнаружен	переход	2
10	2	игрок рядом	переход, удар: $50 - 25 = 25$	3
11	3	таймер выполнен	$25 > 0$ – в «гонится»	2

Шаг 3. Ответ. Стражник в состоянии 2 «гонится», у игрока 25 здоровья.

Проверка. Ударов три: «игрок рядом» пришло три раза (шаги 2, 5 и 10), и каждый раз стражник был в «гонится». $100 - 3 \cdot 25 = 25$.

Заметка для разработчика. Шаг 3 – подозрительное место. Игрок убежал, пока стражник замахивался, а событие пропало. Если бы игрок больше не вернулся, стражник после таймера оказался бы в «гонится» и ждал события, которое уже было. В главе 6 такие места ищут специально.

❗ Соблазн на шаге 3 – перевести стражника в «патрулирует»: «игрок же убежал». Но из «бьёт» стрелки «игрок не обнаружен» нет. Кто так ошибся, получает другую цепочку: в «патрулирует» таймер и «игрок рядом» пропадают, ударов выйдет всего два, и в пункте б) получится 50.

Ответ: а) 2 («гонится»); б) 25.

Своя задача по программе сферы, модуль 2

Реши сам

5.5. Торговец: 1 «ждёт» (начальное), 2 «торгует». Стрелки: 1 → 2 «игрок подошёл / реплика «Зелья по 30 монет!»» и 2 → 1 «игрок ушёл». Внутри «торгует» два внутренних события: «купить [монет не меньше 30] / минус 30 монет, выдать зелье» и «купить [монет меньше 30] / реплика «Не хватает монет»». У игрока 100 монет. События: игрок подошёл, купить, купить, игрок ушёл, купить, игрок подошёл, купить, купить. а) Сколько зелий купил игрок? б) Сколько монет у него осталось?

5.6. Стражник из раздела 5.3, но у игрока всего 70 здоровья. Удар по-прежнему отнимает 25, ниже нуля здоровье не опускается. События: игрок обнаружен, игрок рядом, таймер выполнен, игрок не обнаружен, игрок обнаружен, игрок рядом, таймер выполнен, игрок рядом, таймер выполнен, игрок не обнаружен. Запиши номера состояний после каждого события – подряд, 10 цифр.

Задача 5.4 Сторожевой пёс

недостижимое состояние, тупик, лишний переход

Команда нарисовала поведение сторожевого пса на даче (рисунок ниже) и просит найти ошибки. Состояния пронумерованы, переходы помечены буквами. Что пёс делает внутри состояний, на рисунке не показано: считай, что таймеры и поиск игрока включаются там, где нужно. а) Какое состояние недостижимо? б) Какое состояние – тупик? в) Какие переходы не сработают никогда? Запиши ответ строкой: номер, номер, потом все такие буквы по алфавиту – например, «12кл».

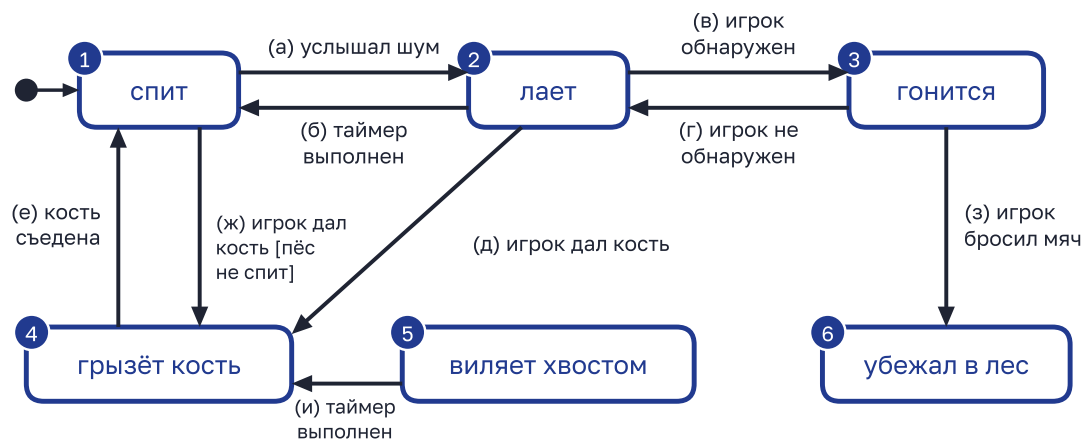


Рисунок 5.13 – Сторожевой пёс. Начальное состояние – 1 «спит»

Дано: шесть состояний, девять переходов (а)–(и), начальное состояние 1.

Найти: недостижимое состояние; тупик; все переходы, которые никогда не сработают.

Решение

Шаг 1. Сначала переход с невозможным условием – он влияет на волну. Стрелка (ж) выходит из «спит» с условием [пёс не спит]. Но пока пёс в состоянии «спит», он спит – условие здесь ложно всегда. Переход (ж) не сработает никогда.

Шаг 2. Волна от начального состояния. Обводим 1. Из 1 выходят (а) → 2 и (ж), но (ж) пропускаем. Обводим 2. Из 2 выходят (б) → 1 (уже обведено), (в) → 3, (д) → 4. Обводим 3 и 4. Из 3 выходят (г) → 2 и (з) → 6 – обводим 6. Из 4 выходит (е) → 1. Из 6 не выходит ничего. Новых состояний больше нет, стоп.

Обведены 1, 2, 3, 4, 6. Необведённым осталось 5 «виляет хвостом»: в него не входит ни одна стрелка. Состояние 5 недостижимо.

Шаг 3. Ещё раз по стрелкам. Вспомни правило из раздела 5.5: стрелка из недостижимого состояния тоже лишняя. Из 5 выходит (и) «таймер выполнен». Подпись нормальная, но пса в состоянии 5 не бывает, и выходить по этой стрелке некому. Значит, никогда не срабатывают два перехода: (ж) и (и).

Шаг 4. Тупики. Считаем стрелки, которые выходят из каждого состояния: из 1 – (а), (ж); из 2 – (б), (в), (д); из 3 – (г), (з); из 4 – (е); из 5 – (и); из 6 – ни одной. Тупик – 6 «убежал в лес». По смыслу он не задуман: пёс умчался за мячом, и дача осталась без сторожа до конца игры. Нужна стрелка обратно, например «таймер выполнен» в «спит».

Шаг 5. Ответ. Недостижимое – 5, тупик – 6, переходы, которые не сработают, – (ж) и (и): 5бжи.

Что делать команде. Пёс, виляющий хвостом, нарисован, но в игре его никто не увидит. Чтобы он появился, нужна стрелка в «виляет хвостом», например «игрок погладил» из «лает». Тогда оживёт и переход (и).

⚠ **Чаще всего** называют только (ж) и забывают (и): стрелка (и) нарисована правильно, и кажется, что с ней всё в порядке. Бывает, что недостижимым называют 4 «грызёт кость»: «туда ведёт лишняя стрелка (ж)». Но в 4 ведёт ещё и рабочая стрелка (д) из «лает». А кто-то записывает тупиком 5. В «виляет хвостом» не входит ни одной стрелки, но выходит одна – (и). Тупик – это «нельзя выйти», а не «нельзя войти».

Ответ: 56жи.

Своя задача по программе сферы, модуль 5

Реши сам

5.7. Кузнец в деревне: 1 «ждёт» (начальное), 2 «разговаривает», 3 «куёт», 4 «спит», 5 «храпит», 6 «чинит». Переходы: $1 \rightarrow 2$, $2 \rightarrow 1$, $2 \rightarrow 3$, $3 \rightarrow 1$, $3 \rightarrow 6$, $6 \rightarrow 1$, $4 \rightarrow 5$, $5 \rightarrow 4$, $4 \rightarrow 1$. Все переходы рабочие, события не важны. Какие состояния недостижимы? Запиши номера по возрастанию, без пробелов.

5.8. Летучая мышь в пещере. 1 «висит» (начальное; вход: Обнаружение – остановить поиск игрока; таймер 30 с). 2 «летает» (вход: Обнаружение – искать игрока в радиусе 6; Движение – двигаться к случайной точке; таймер 20 с). 3 «атакует» (вход: Атака – атаковать игрока). Переходы: а) $1 \rightarrow 2$ «таймер выполнен»; б) $1 \rightarrow 3$ «игрок обнаружен»; в) $2 \rightarrow 3$ «игрок обнаружен»; г) $3 \rightarrow 2$ «игрок не обнаружен»; д) $2 \rightarrow 1$ «таймер выполнен». Какой переход не сработает никогда? Запиши букву.

Задача 5.5 Каменный голем

вложенные состояния, число переходов, трассировка

Босс «Каменный голем» ждёт героя у входа на арену. Когда герой входит, начинается бой: голем по таймеру по кругу бьёт кулаком, кидает камни и прикрывается щитом. Здоровье голема кончилось – он рассыпается. Герой убежал с арены – голем возвращается ко входу. Схема – на рисунке.

а) Сколько стрелок пришлось бы нарисовать, если бы родительского состояния «бой» не было? Стрелки от чёрных точек не считай. б) События пришли так: игрок вошёл, таймер, таймер, игрок покинул арену, игрок вошёл, таймер. В каком состоянии голем? Запиши номер.

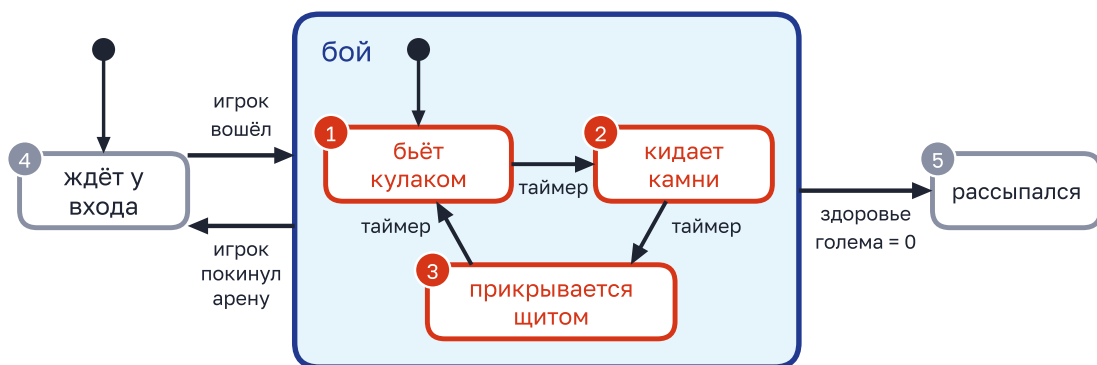


Рисунок 5.14 – Каменный голем. Три приёма лежат внутри родительского состояния «бой». «Таймер» – коротко от «таймер выполнен»

Дано: родитель «бой» с дочерними 1, 2, 3 (начальное дочернее – 1); снаружи 4 (начальное всей схемы) и 5.

Найти: а) число стрелок без родителя; б) состояние после шести событий.

Решение

Шаг 1. Считаем стрелки на схеме. Внутри «бой» три стрелки «таймер» по кругу: $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$. Снаружи тоже три: «игрок вошёл» (из 4 в рамку), «игрок покинул арену» (из рамки в 4), «здоровье голема = 0» (из рамки в 5). Всего $3 + 3 = 6$.

Шаг 2. Убираем родителя. Внутренний круг остаётся – 3 стрелки. «Игрок вошёл» ведёт в начальное дочернее, то есть прямо в 1, – 1 стрелка. А две стрелки из рамки придётся размножить: каждая должна выходить из каждого из трёх приёмов. «Игрок покинул арену» – 3 стрелки, «здоровье голема = 0» – ещё 3. Итого $3 + 1 + 3 + 3 = 10$.

Шаг 3. Трассировка для пункта б).

Таблица 5.4

Шаг	Было	Событие	Стало
1	4	игрок вошёл	1 – начальное дочернее
2	1	таймер	2
3	2	таймер	3
4	3	игрок покинул арену	4 – по стрелке из рамки
5	4	игрок вошёл	1 – снова начальное дочернее
6	1	таймер	2

Голем в состоянии 2 «кидает камни».

А если приёмов больше? 10 стрелок против 6 – с родителем схема уже заметно проще. Будь у голема пять приёмов, без родителя вышло бы $5 + 1 + 5 + 5 = 16$ стрелок, а с родителем $5 + 3 = 8$. Чем больше дочерних состояний, тем больше выигрыш.

⚠ Типичная ошибка. На шаге 5 продолжить с того места, где голем прервался, – с «прикрывается щитом». Тогда после таймера выйдет 1, а не 2. Вход в рамку всегда ведёт в начальное дочернее. В пункте а) частый неверный ответ – 8: одну из двух стрелок рамки размножили, а про вторую забыли.

Ответ: а) 10; б) 2.

Своя задача по программе сферы, модуль 5

Реши сам

5.9. Светофор с ночным режимом. Четыре сигнала из раздела 5.1 (красный, зелёный, зелёный мигает, жёлтый – по кругу по таймеру) сложили в родителя «дневной режим». Из его рамки стрелка «нажата кнопка «ночь»» ведёт в состояние «мигает жёлтым», а из «мигает жёлтым» стрелка «нажата кнопка «день»» ведёт обратно в рамку (начальное дочернее – «красный»). Сколько стрелок понадобилось бы без родительского состояния? Стрелки от чёрных точек не считай.

5.10. Робот-охранник из раздела 5.6. Пронумеруем состояния: 1 «патрулирует», 2 «гонится», 3 «замер». Робот начинает в 1. События: игрок обнаружен, триггер активирован, игрок не обнаружен, игрок обнаружен, таймер выполнен, игрок обнаружен, триггер активирован, таймер выполнен. Запиши номера состояний после каждого события – подряд, 8 цифр.

Задача 5.6* Дракон по расписанию

таймеры, хвост и круг, деление с остатком

Дракон живёт по таймерам (рисунок ниже). В начале игры он спит 7 с, потом 3 с просыпается, а дальше ходит по кругу: ищет добычу, летит, дышит огнём, отдыхает – и снова ищет добычу. Все переходы – по событию «таймер выполнен».

а) В каком состоянии дракон через 200 с после начала игры? Запиши номер. б) Сколько раз к этому моменту он начинал дышать огнём?

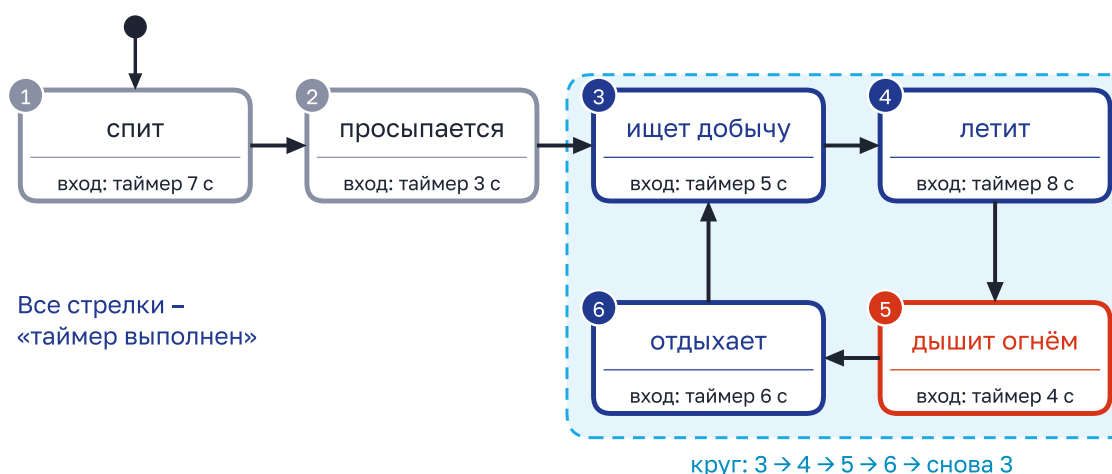


Рисунок 5.15 – Дракон по таймерам. Состояния 1 и 2 – хвост, их дракон проходит один раз. Состояния 3–6 – круг

Дано: хвост: 1 – 7 с, 2 – 3 с; круг: 3 – 5 с, 4 – 8 с, 5 – 4 с, 6 – 6 с; $T = 200$ с.

Найти: а) состояние в момент 200 с; б) сколько раз дракон входил в состояние 5.

Решение

Шаг 1. Куда ведут стрелки. Из каждого состояния выходит одна-единственная стрелка, поэтому путь дракона известен заранее: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow 4 \rightarrow \dots$. Состояния 1 и 2 – хвост, они бывают один раз. Дальше круг $3 \rightarrow 4 \rightarrow 5 \rightarrow 6$ повторяется. Трассировать 200 секунд по шагам долго, поэтому вычтем хвост и отбросим целые круги.

Шаг 2. Хвост. $7 + 3 = 10$ с. На 10-й секунде дракон впервые входит в «ищет добычу». До момента 200 с остаётся $200 - 10 = 190$ с кругов.

Шаг 3. Длина круга. $5 + 8 + 4 + 6 = 23$ с.

Шаг 4. Отбрасываем целые круги. $8 \cdot 23 = 184$, а $9 \cdot 23 = 207$ – уже больше 190. Значит, полных кругов 8, и ещё остаётся $190 - 184 = 6$ с девятого круга.

Шаг 5. Где дракон на 6-й секунде круга. Разложим круг по времени, считая от его начала:

Таблица 5.5

№	Состояние	С секунды круга	До секунды
3	ищет добычу	0	5
4	летит	5	13
5	дышит огнём	13	17
6	отдыхает	17	23

Число 6 лежит между 5 и 13 – дракон летит (состояние 4), и летит уже 1 с. На границу между состояниями момент не попадает, спорить не о чем.

Шаг 6. Огонь. В каждом полном круге дракон один раз входит в «дышит огнём» – за 8 кругов это 8 раз. В девятом круге прошло только 6 с, а огонь начинается с 13-й секунды круга – до него ещё не дошли. Всего 8.

Проверим в секундах от начала игры. Хвост кончается на 10-й, восемь кругов – на $10 + 184 = 194$ -й. С 194-й по 199-ю дракон ищет добычу, с 199-й летит. На 200-й секунде он летит уже 1 с – сходится.

 **Главная ловушка** – забыть про хвост и делить 200 на 23: $200 = 8 \cdot 23 + 16$. На 16-й секунде круга дракон дышит огнём, и выходит неверный ответ 5. Бывает и наоборот – хвост считают частью круга и получают круг $1 \rightarrow \dots \rightarrow 6$ длиной 33 с. Но из 6 стрелка ведёт в 3, а не в 1, – хвост проходится только один раз.

Ответ: а) 4 («летит»); б) 8 раз.

По мотивам задачи НТО 2025/26, профиль «Разработка компьютерных игр» (старший трек, 8–11 классы), первый отборочный этап, предметный тур по информатике, «Функциональный граф «буква А»»

Реши сам

5.11. Светофор из раздела 5.1 включили ровно в 8:00:00, и в этот момент загорелся красный. Какой сигнал горит в 8:01:40? Запиши номер состояния.

5.12.* Призрак в замке по каждому событию «тик» переходит в следующее состояние. Стрелки: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$, а из 6 – обратно в 4. Призрак начинает в 1. а) В каком состоянии он будет после 1000 тиков? б) Сколько раз за эти 1000 тиков он входил в состояние 4?

Переменные, условия и поиск ошибок



Игра помнит, сколько у героя монет, нашёл ли он ключ и сколько у него здоровья. Для этого у программиста есть переменные. А условия решают, что будет дальше: откроются ворота или нет. Здесь ты научишься читать чужую программу строчку за строчкой, как её читает компьютер, и находить в ней ошибку. Ещё посчитаем баланс в электронной таблице и увидим, зачем программу режут на части. На отборе такие задачи выглядят так: «что напечатает программа», «в какой строке ошибка», «что окажется в ячейке». Ответ – число, строка цифр или короткая формула.

6.1. Переменная – коробка с именем

НА ПАЛЬЦАХ



Рисунок 6.1 – Три коробки с именами. В каждой лежит одно значение

Положил в коробку новое значение – старое пропало. Коробка «монеты» помнит только последнее число.

Переменная – это коробка с именем, в которой лежит одно значение. Имя придумывает программист: `монеты`, `здоровье`, `ключ_найден`. Значение меняется по ходу игры: герой подобрал монету – в коробке стало на одну больше. Когда игра хочет узнать, хватит ли денег на меч, она заглядывает в коробку и читает число.

В играх чаще всего встречаются три вида переменных.

- **Счётчик** – число, которое растёт на единицу при каждом событии: сколько монет собрано, сколько врагов побеждено, сколько раз игрок проиграл.
- **Флаг** (флажок) – переменная «да или нет». `ключ_найден` сначала «нет», герой подобрал ключ – стало «да». Флаги помнят, что уже случилось: поговорил ли герой со стражником, выполнен ли квест.
- **Число с границами** – здоровье, энергия, сытость. Оно растёт и падает, но не выходит за рамки: здоровье не бывает больше максимума и меньше нуля.

В книге программы записаны двумя способами. **Псевдокод** – это программа по-русски, как её пишут на доске. Его не запустишь, зато легко читать. Второй способ – язык **Python**. Его программы можно запустить на компьютере и проверить.

Главная команда с переменной – **присваивание**: положить значение в коробку. В псевдокоде её пишем стрелкой:

```
монеты ← 12  
монеты ← монеты + 5
```

Первая строка кладёт в коробку `монеты` число 12. Вторая читается так: «возьми, что лежит в коробке `монеты`, прибавь 5 и положи обратно». Было 12, стало 17. Это не уравнение из мате-

матики. Там « $x = x + 5$ » не бывает никогда, а в программе это самая обычная команда. Стрелка показывает, куда кладём: справа считаем, слева – коробка, куда попадёт ответ.

В Python вместо стрелки пишут знак `=`. Тот же пример, а заодно флаг и здоровье:

```
монеты = 12          # в коробке 12
монеты = монеты + 5   # достали 12, прибавили 5, положили 17
ключ_найден = False   # флаг: ключа пока нет
ключ_найден = True    # герой подобрал ключ
здоровье = 100
здоровье = здоровье - 30
print(монеты, ключ_найден, здоровье)  # 17 True 70
```

Всё, что после знака `#`, – **комментарий**. Компьютер его пропускает, это записка для человека. `True` значит «да», `False` – «нет» (как в главе 4). Команда `print` печатает на экран то, что стоит в скобках. Программа выведет `17 True 70`.

У каждой переменной должно быть **начальное значение** – то, что лежит в коробке до начала игры. Забыл его задать – и счётчик монет начнёт не с нуля, а с чего придётся. В Python программа в таком случае просто остановится с ошибкой.

На вебинаре сферы показывали, что в «Урсуле» поведение персонажей собирают из модулей. Среди них есть «Счётчики» и «Переменные». Там же упоминали глобальные переменные – общие для всей игры, а не для одного персонажа. Устроены они так же: у переменной есть имя и значение, и её можно проверять в условиях.

Запомни. В коробке всегда одно значение, новое стирает старое. `монеты ← монеты + 5` читай справа налево: достань, прибавь, положи обратно. Без начального значения переменную не используют.

Совет. Читаешь чужую программу – заведи на черновике по клеточке на каждую переменную. Выполнилась строка – зачеркни старое число и напиши новое. Держать пять чисел в голове трудно, а на бумаге – легко.

6.2. Условие: если – иначе

НА ПАЛЬЦАХ

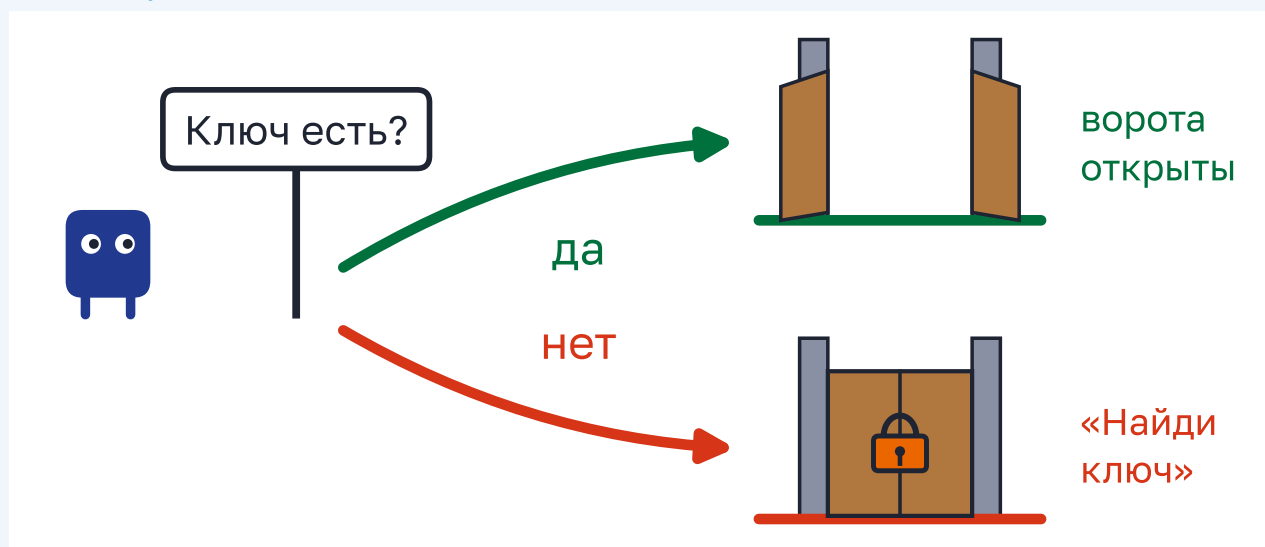


Рисунок 6.2 – Герой у развилки. Табличка спрашивает: «Ключ есть?»

«Да» – дорога к открытым воротам. «Нет» – к запертым. Каждый раз игра идёт ровно по одной дороге.

Условие – это проверка «да или нет»: «монет не меньше 50?», «ключ найден?». По ответу программа выбирает, что делать дальше. В псевдокоде:

```
если монеты ≥ 50 то
    сказать «Покупай меч»
иначе
    сказать «Накопи ещё»
```

Если условие верно, выполняются строки под **если**. Если неверно – строки под **иначе**. Одна из двух веток выполнится обязательно, а обе сразу – никогда. Часть **иначе** можно не писать. Тогда при неверном условии программа ничего не делает и идёт дальше.

Числа сравнивают такими знаками:

Таблица 6.1

Словами	Псевдокод	Python
больше	>	>
меньше	<	<
не меньше	≥	≥
не больше	≤	≤
равно	=	=
не равно	≠	≠

«Не меньше» – это «больше или равно»: $50 \geq 50$ верно. «Не больше» – «меньше или равно». Заметь: в Python «равно» – два знака `==`. Один знак `=` уже занят, он кладёт значение в коробку. `здоровье = 0` обнуляет здоровье, а `здоровье == 0` спрашивает, равно ли оно нулю.

Та же программа на Python:

```
монеты = 35
if монеты ≥ 50:
    print("Покупай меч")
else:
    print("Накопи ещё", 50 - монеты)
```

`if` значит «если», `else` – «иначе». После условия ставят двоеточие. Строки, которые относятся к ветке, сдвигают вправо на 4 пробела – это называется **отступ**. По отступу Python понимает, где ветка кончается. У героя 35 монет, $35 \geq 50$ – неверно. Выполнится ветка `else`, и программа напечатает `Накопи ещё 15`.

Когда вариантов больше двух, пишут цепочку `иначе если` (в Python – `elif`):

```
если здоровье > 60 то
    цвет ← зелёный
иначе если здоровье > 20 то
    цвет ← жёлтый
иначе
    цвет ← красный
```

Самое опасное место в условиях – **граница**. «Хотя бы 50 монет» – это ≥ 50 , и с 50 монетами меч купить можно. «Больше 50 монет» – это > 50 , и с 50 монетами нельзя. Всегда проверяй условие на самом граничном числе: 50 подходит или нет?

В машинах состояний из главы 5 условие стоит на стрелке в квадратных скобках: «разговор [монеты ≥ 50] / выдать меч». Это то же самое «если»: переход сработает, только когда условие верно.

Запомни. Условие верно – выполняется ветка `если`, неверно – ветка `иначе`. Одна из двух, не обе. «Хотя бы», «не меньше» – это $\geq$. В Python сравнивают знаком `==`, а один `=` кладёт в коробку.

6.3. Порядок проверок

НА ПАЛЬЦАХ



Рисунок 6.3 – Одна и та же цепочка проверок в двух порядках. У игрока 120 очков

Проверки идут сверху вниз, и первое «да» уводит игрока из цепочки. Если первой стоит мягкая проверка « ≥ 10 », золото не достанется никому.

Цепочку `если – иначе если – иначе` программа проверяет **сверху вниз** и останавливается на **первом** верном условии. Всё, что ниже, она даже не читает. Поэтому порядок строк в цепочке меняет результат.

Пусть за уровень дают медаль: от 100 очков – золото, от 50 – серебро, от 10 – бронзу. Правильно так:

```
если очки  $\geq 100$  то медаль ← золото
иначе если очки  $\geq 50$  то медаль ← серебро
иначе если очки  $\geq 10$  то медаль ← бронза
иначе медаль ← нет
```

120 очков: первое условие верно – золото, дальше не смотрим. 70 очков: первое неверно, второе верно – серебро. Теперь переставим строки и начнём с «очки ≥ 10 ». Для 120 очков это условие тоже верно, программа выдаст бронзу и остановится. Такая программа дарит бронзу всем, у кого хотя бы 10 очков, и золото не выдаёт никогда.

Отсюда два правила.

- **Сначала строгие проверки, потом мягкие.** « ≥ 100 » раньше « ≥ 50 », « ≥ 50 » раньше « ≥ 10 ».
- **Особый случай – первым.** Если клетка, номер которой делится и на 3, и на 5, особенная (в ней сундук), проверяй её до проверок «делится на 3» и «делится на 5». Иначе её перехватит проверка «делится на 3», и до сундука программа не дойдёт.

Цепочка и отдельные проверки – не одно и то же. Если написать три отдельных `если` без `иначе`, программа проверит все три. Каждое верное условие перезапишет коробку, и в ней останется значение из **последнего** сработавшего `если`. Для медалей с отдельными `если` правильный порядок обратный: сначала бронза, потом серебро, потом золото. Тогда последним

сработает самое строгое верное условие. Не веришь – возьми 120 очков и пройди обе программы по строчкам.

Запомни. Цепочка иначе если останавливается на первом «да». Строгие проверки ставь выше мягких, особый случай – выше всех. В отдельных если без иначе побеждает последнее сработавшее.

Совет. Проверь цепочку на самом «сильном» игроке: у кого больше всех очков или чей номер клетки подходит под все условия сразу. Если он получает не то, что задумано, – порядок строк неверный.

6.4. Трассировочная таблица: следим за переменными

НА ПАЛЬЦАХ



1 < 3 – на следующий шаг энергии не хватает, цикл закончен

Рисунок 6.4 – Программа как кино по кадрам. На каждом кадре – значения коробок после очередного шага

Энергии 10, каждый шаг стоит 3. После трёх шагов осталось 1 – на четвёртый не хватает, и герой останавливается.

Многие программы повторяют одно и то же много раз. Такое повторение называется **цикл**. В этой главе их два вида.

- **Цикл «для»** повторяет строки заданное число раз: для ход от 1 до 5: выполнит тело цикла пять раз, а в коробке ход по очереди окажутся 1, 2, 3, 4, 5. **Тело цикла** – это строки с отступом под ним.
- **Цикл «пока»** повторяет, пока условие верно: пока энергия ≥ 3 : . Перед каждым повтором условие проверяется заново. Стало неверным – цикл закончился.

Чтобы узнать, что сделает программа с циклом, её **трассируют** – выполняют вручную и записывают значения в **трассировочную таблицу**: столбцы – переменные и проверки, строки – шаги. Вот программа с картинки:

```

энергия ← 10
шаги ← 0
пока энергия ≥ 3:
    энергия ← энергия - 3
    шаги ← шаги + 1

```

Таблица 6.2

Шаг	энергия ≥ 3?	энергия	шаги
начало	–	10	0
1	да	7	1
2	да	4	2
3	да	1	3
4	нет – конец	1	3

Правила заполнения:

- первая строка – начальные значения, до цикла;
- одна строка – один повтор цикла; в неё пишем значения **после** всех команд тела;
- проверку условия записываем отдельным столбцом, словами «да» или «нет»;
- последняя строка – проверка, на которой цикл закончился. Её часто забывают, и тогда кажется, что был ещё один шаг.

В Python цикл «для» пишут через `range`: `for ход in range(1, 6):`. Запомни ловушку: `range(1, 6)` даёт числа 1, 2, 3, 4, 5, а **6 не входит**. Второе число – это «до, но не включая». Цикл «пока» по-английски `while`:

```

энергия = 10
шаги = 0
while энергия ≥ 3:
    энергия = энергия - 3
    шаги = шаги + 1
    print("шаг", шаги, "энергия", энергия)
print("всего шагов:", шаги)  # всего шагов: 3

```

Строка с `print` внутри цикла на каждом повторе печатает шаг 1 энергия 7, шаг 2 энергия 4, шаг 3 энергия 1. Это та же трассировочная таблица, только её пишет сама программа. Такие строки называют **логом**, а окно, куда программа их печатает, – **консолью**. В программе сферы отладку учат вести как раз через консоль и логи: когда игра ведёт себя странно, программист печатает значения переменных и смотрит, где они пошли не так.

НА ПАЛЬЦАХ



$$12 : 3 = 4, \text{ остаток } 0$$

$$12 \% 3 = 0 - \text{ делится}$$



$$14 : 3 = 4, \text{ остаток } 2$$

$$14 \% 3 = 2 - \text{ не делится}$$

Рисунок 6.5 – Раскладываем монеты кучками по 3. Что не влезло в кучку – остаток

12 монет легли ровно в 4 кучки – остаток 0, 12 делится на 3. Из 14 монет две лишние – остаток 2.

Ещё одна вещь для задач с клетками и ходами – **остаток от деления**. «Число делится на 3» значит «остаток от деления на 3 равен нулю». 12 делится на 3 (остаток 0), 14 не делится (остаток 2): $4 \cdot 3 = 12$, и до 14 не хватает ещё двух. В Python остаток даёт знак `%`: `14 % 3` равно 2, поэтому проверка «клетка делится на 3» пишется `клетка % 3 == 0`.

Запомни. Трассировка – выполняем программу вручную, строка таблицы на каждый шаг. Первая строка – начальные значения, последняя – проверка, на которой цикл кончился. `range(1, 6)` – это 1, 2, 3, 4, 5, без шестёрки.

Совет. Не торопись «догадаться», чем кончится цикл. На проверке «здоровье < 30» при здоровье ровно 30 ответ – «нет», и такие строки решают всю задачу. Иди по таблице честно, строка за строкой.

6.5. Как найти ошибку в чужой программе

НА ПАЛЬЦАХ

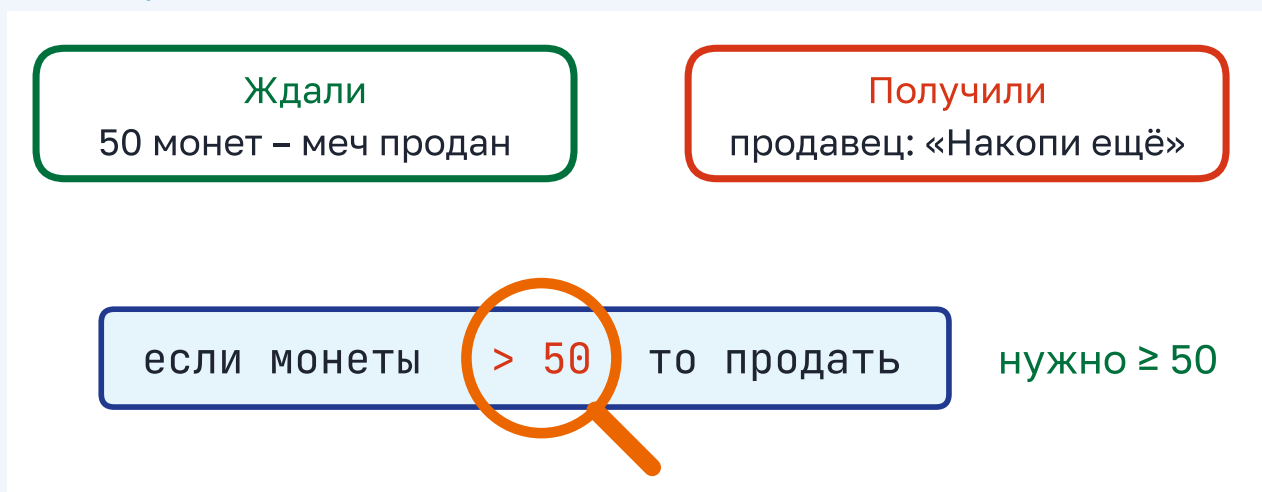


Рисунок 6.6 – Отладка начинается с двух фраз: что ждали и что получили

У героя ровно 50 монет, а продавец не отдаёт меч. Лупа наведена на строку-виновника: там «больше», а нужно «не меньше».

Баг – ошибка в программе: игра делает не то, что задумал автор. **Отладка** – поиск и исправление таких ошибок. В программе сферы есть задание «кейс по отладке чужого кода». Дают описание, как игра должна работать, программу и жалобу игрока. Нужно найти строку, которая всё портит. Действуй так.

1. **Сформулируй баг одной фразой:** «делал это – ждал этого – получил вот это». Например: «принёс 3 руды – ждал меч – кузнец просит ещё 2».
2. **Трассируй программу на том, что делал игрок.** Не «вообще», а с его числами и в его порядке.
3. **Ищи первое расхождение.** Пройди таблицу и найди первую строку, после которой значение в коробке стало не таким, как по описанию. Ошибка – в команде, которая это значение поменяла.
4. **Исправь одно место и проверь снова.** Ошибок бывает несколько: починил одну – проснулась вторая.
5. **Проверь границы:** ровно на пороге, на единицу меньше, на единицу больше.

Большинство ошибок в логике квестов – из короткого списка. Проверяй его пункт за пунктом.

- **Граница:** $>$ вместо $\geq$, $<$ вместо $\leq$. С 50 монетами при условии «монеты > 50 » меч не купить.
- **Положил вместо прибавил:** `руда  $\leftarrow$  1` вместо `руда  $\leftarrow$  руда + 1`. Счётчик застрял на единице.
- **Равно вместо «меньше или равно»:** `если здоровье = 0`. Здоровье прыгает $20 \rightarrow -5$ и мимо нуля проскакивает.
- **Порядок проверок:** мягкое условие стоит выше строгого (раздел 6.3).
- **И вместо ИЛИ** (или наоборот): дверь открывается с одним ключом, хотя нужны ключ и факелы.

- **Сброс не там:** счётчик $\leftarrow 0$ попал внутрь цикла или в обработку события, и значение каждый раз обнуляется.
- **Нет рамок:** после лечения здоровье 130 при максимуме 100; после удара – минус 10.
- **Флаг никто не меняет:** программа проверяет `квест_выполнен`, но ни одна строка не делает его «да».

Шпаргалка по отладке есть и в справочнике в конце книги.

Запомни. Баг – «ждали одно, получили другое». Трассируй программу с числами игрока, ищи первое расхождение с описанием. Меняй одно место за раз и проверяй границу.

Совет. В задаче «найди все ошибки» после первой находки не останавливайся. Мысленно исправь её и прогони программу ещё раз. Если игра всё ещё работает не по описанию, где-то прячется вторая ошибка.

6.6. Электронные таблицы: формулы и ссылки

НА ПАЛЬЦАХ

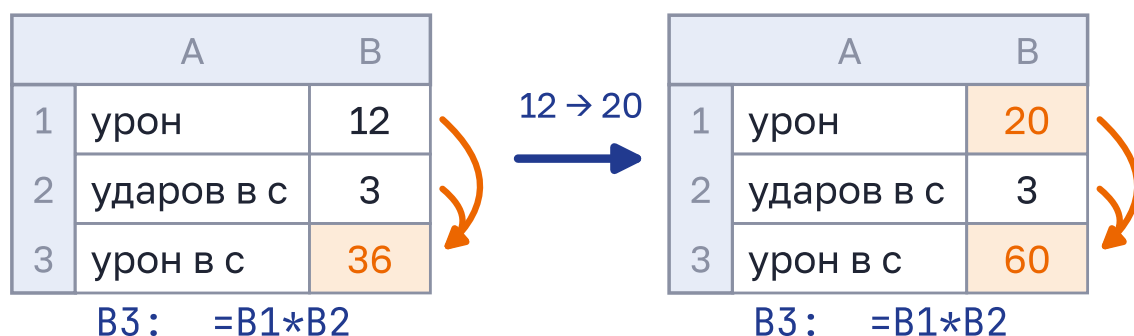


Рисунок 6.7 – В ячейке B3 не число, а формула: «умножь B1 на B2»

Поменяли урон с 12 на 20 – ячейка B3 сама пересчиталась: было 36, стало 60.

Баланс (глава 2) – это когда в игре нет ни всесильного оружия, ни бесполезного. Геймдизайнеры считают его в **электронных таблицах**: Excel, LibreOffice Calc, Яндекс Таблицы и других. Таблица состоит из **ячеек**. У каждой есть **адрес** – буква столбца и номер строки: B3 – столбец В, строка 3. Можно думать о ячейке как о переменной: адрес – её имя, внутри – значение.

В ячейку можно написать число, текст или **формулу**. Формула начинается со знака `=`: `=B1*B2` значит «умножь то, что в B1, на то, что в B2». Звёздочка – умножение, косая черта – деление. На экране ты увидишь не формулу, а результат – 36. Главное удобство: поменял урон в B1, и все формулы, которые на него ссылаются, пересчитались сами. В главе 2 так строили ряд цен.



Рисунок 6.8 – Формулу из E2 скопировали в E3 и E4. Слева ссылка уехала в пустые ячейки, справа доллар удержал её на B7

Обычная ссылка при копировании едет вниз вместе с формулой. Доллар прибивает её гвоздём к одной ячейке.

Копирование формул. Формулу пишут один раз и тянут за уголок вниз или вправо. При этом **ссылки сдвигаются**: формула `=B2*C2` из строки 2, скопированная в строку 5, превратится в `=B5*C5`. Скопировал на 3 строки ниже – все номера строк выросли на 3. Скопировал на один столбец вправо – все буквы сдвинулись на одну: B → C. Такие ссылки называются **относительными**. Обычно так и надо: каждая строка считает своё оружие.

Но бывает, что все строки должны смотреть в **одну и ту же** ячейку – например, в здоровье босса. Тогда ставят знак доллара: `$B$7`. Такая ссылка **абсолютная**: при копировании она не сдвигается. Доллар перед буквой держит столбец, перед цифрой – строку. Забыл доллар – формула уплзёт вниз вместе с остальными и будет читать пустую ячейку. Пустая ячейка в формуле считается нулём.

Функции – готовые команды таблицы. Вот самые нужные. В русской версии их названия пишут по-русски, а части внутри скобок разделяют точкой с запятой.

- `=СУММ(B2:B6)` – сумма ячеек от B2 до B6. Двоеточие значит «диапазон»: B2, B3, B4, B5, B6.
- `=МАКС(B2:B6)` и `=МИН(B2:B6)` – самое большое и самое маленькое число.
- `=СРЗНАЧ(B2:B6)` – среднее арифметическое.
- `=ЕСЛИ(B2 ≥ C2; "открыта"; "закрыта")` – условие прямо в ячейке. Если B2 не меньше C2, в ячейке будет «открыта», иначе «закрыта». Это то же `если – иначе` из раздела 6.2.

В английской версии те же функции называются SUM, MAX, MIN, AVERAGE, IF, а части разделяют запятой.

	A	B	C	D	E	
1	Оружие	Урон	Ударов/с	Урон/с	Время, с	
2	Кинжал	12	3	=B2*C2	=\$B\$7/D2	формулы скопированы вниз
3	Меч	30	1,5	↓	↓	
4	Молот	80	0,5	↓	↓	
5	Лук	20	2	↓	↓	
6						
7	HP босса	720				

Рисунок 6.9 – Таблица баланса: урон в секунду и время боя с боссом. Видны формулы в строке 2, остальные строки получены копированием

На рисунке – таблица баланса к задаче 6.6. В D2 урон в секунду: урон удара, умноженный на число ударов в секунду. В E2 – за сколько секунд оружие победит босса: здоровье босса из B7 делим на урон в секунду. У B7 стоят доллары, потому что босс один на все строки.

Запомни. Адрес ячейки – буква столбца и номер строки. Формула начинается с = . При копировании обычные ссылки сдвигаются на столько же строк и столбцов, а ссылки с долларом (\$B\$7) стоят на месте. Пустая ячейка в формуле – ноль.

6.7. Функции и модули: делим программу на части

НА ПАЛЬЦАХ

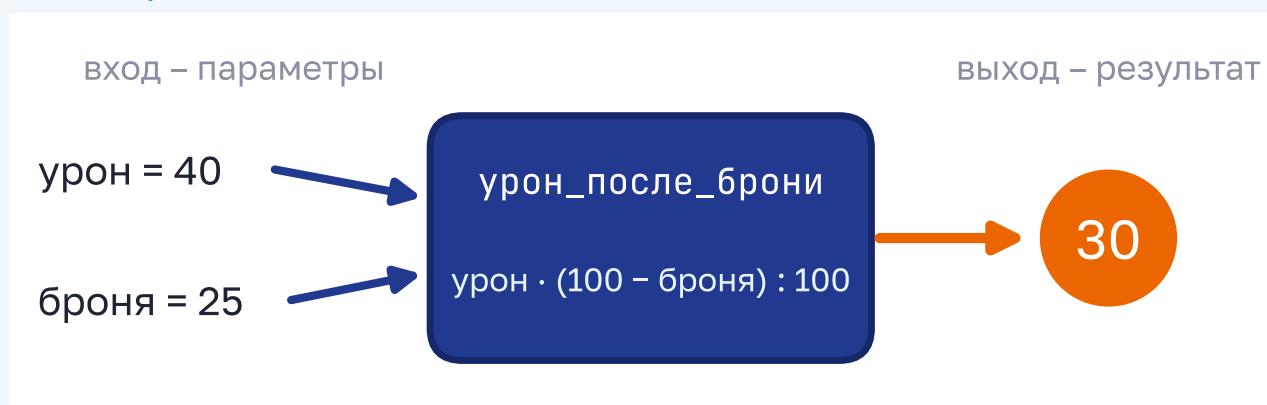


Рисунок 6.10 – Функция – машинка с именем. На входе урон и броня, на выходе урон после брони

Как машинка устроена внутри, знать не обязательно. Дал ей 40 и 25 – получил 30. Дал 60 и 50 – получил 30.

Программа настоящей игры – тысячи строк. Если писать их одной простыней, в ней утонешь. Поэтому программу режут на части, и у каждой части своя работа: одна считает урон, другая двигает персонажа, третья показывает диалог. Такие части называют **функциями**, а круп-

ные наборы функций – **модулями**. Когда программу собирают из таких частей, это называют **модульным программированием**.

Зачем это нужно:

- **Не повторяться.** Урон считается в десяти местах игры – формулу пишут один раз и вызывают по имени.
- **Чинить в одном месте.** Нашёл ошибку в подсчёте урона – исправил в функции, и она исправилась везде.
- **Читать легче.** Строка `урон_после_брони(40, 25)` говорит сама за себя, даже если не смотреть внутрь.
- **Делить работу в команде.** Один пишет модуль боя, другой – модуль диалогов, и друг другу не мешают.

У функции есть **имя**, **параметры** – коробки для входных данных – и **результат**, который она возвращает. На Python:

```
def урон_после_брони(урон, броня):  
    return урон * (100 - броня) / 100  
  
print(урон_после_брони(40, 25))    # 30.0  
print(урон_после_брони(60, 50))    # 30.0
```

Разберём построчно. `def` – «определить функцию»: дальше идут её имя и параметры в скобках. Строка с отступом – тело функции. `return` – «вернуть»: результат вычисления уходит туда, откуда функцию позвали. Сама по себе функция ничего не делает, пока её не **вызовут**. Вызов – это имя и числа в скобках: `урон_после_брони(40, 25)`. При вызове числа раскладываются по параметрам **по порядку**: первое – в `урон`, второе – в `броня`. Дальше считается $40 \cdot (100 - 25) / 100 = 30$. Python печатает `30.0`: после деления он показывает число с дробной частью, пусть и нулевой.

Порядок чисел при вызове – частый источник багов. `урон_после_брони(25, 40)` посчитает удар силой 25 по броне 40 % и вернёт 15. Программа не упадёт и ошибку не покажет: она честно выполнит то, что написано.

Функция может вызывать другую функцию, а внутри у неё могут быть условия и циклы. Трассируют её так же: заводят коробки для параметров и идут по строчкам. Когда функция вернула результат, её коробки выбрасывают, а число подставляют на место вызова.

В «Урсуле» модульный подход виден сразу. У персонажа там целый набор модулей, его разбирали на вебинаре сферы: обнаружение, движение, анимация, счётчики, атаки, повествование, триггеры, переменные. Программист не пишет движение с нуля, а подключает готовый модуль и настраивает его.

Запомни. Функция – кусок программы с именем, параметрами и результатом. Она работает, только когда её вызвали. Числа при вызове ложатся в параметры по порядку. Модульная программа – из частей, у каждой своя работа.

Совет. Трассируя программу с функциями, рисуй для каждого вызова отдельную рамочку со своими коробками. Коробка `здоровье` внутри функции и коробка `здоровье` снаружи – разные, даже если имя одно.

6.8. Разбираем задачи

Задача 6.1 Кошелёк торговца

разминка · переменные, присваивание, порядок команд

В игре у торговца волшебный кошелёк. Программа, которая следит за его монетами, такая:

```
монеты ← 12
монеты ← монеты + 8
монеты ← монеты - 5
цена ← 10
монеты ← монеты - цена
монеты ← монеты * 2
```

Сколько монет окажется в переменной `монеты` после выполнения программы? Ответ – целое число.

Дано: шесть команд присваивания; переменные `монеты` и `цена`.

Найти: значение `монеты` в конце.

Решение

Шаг 1. Что происходит. Команды выполняются строго по порядку, сверху вниз. Каждая берёт текущее значение из коробки, считает и кладёт результат обратно. Заведём трассировочную таблицу. Пока в коробку `цена` ничего не положили, ставим прочерк.

Шаг 2. Идём по строкам.

Таблица 6.3

Шаг	Что делает	монеты	цена
1	кладёт 12	12	–
2	+8	20	–
3	–5	15	–
4	цена ← 10	15	10
5	–цена	5	10
6	·2	10	10

На шаге 2 берём 12, прибавляем 8 – получаем 20. На шаге 3: $20 - 5 = 15$. Шаг 4 кладёт 10 в другую коробку, `монеты` не меняются. Шаг 5: $15 - 10 = 5$. Шаг 6: $5 \cdot 2 = 10$.

Шаг 3. Проверяем на Python.

```

монеты = 12
монеты = монеты + 8
монеты = монеты - 5
цена = 10
монеты = монеты - цена
монеты = монеты * 2
print(монеты)    # 10

```

Проверим другим путём. До удвоения торговец получил 8 и отдал $5 + 10 = 15$, в кошельке $12 + 8 - 15 = 5$ монет. Удвоили – 10.

❗ **Типичная ошибка.** Удвоить не то число: сосчитать $12 \cdot 2$ в начале или удвоить всё, что торговец получил ($20 \cdot 2 = 40$), а траты вычесть потом. Удвоение – последняя команда. Она действует на то, что лежит в коробке к этому моменту, то есть на 5. Ещё путаются со строкой `цена ← 10`: кажется, что она что-то отнимает у монет. Нет, она кладёт число в другую коробку.

Ответ: 10.

Своя задача по программе сферы, модуль 2

Реши сам

6.1. Программа считает опыт героя:

```

опыт ← 25
опыт ← опыт + 15
бонус ← 3
опыт ← опыт * бонус
опыт ← опыт - 50

```

Чему равен `опыт` после выполнения программы?

6.2. Игрок хочет поменять местами предметы в двух ячейках рюкзака. Предметы обозначены номерами. Программист написал:

```

левая ← 4
правая ← 9
левая ← правая
правая ← левая

```

а) Какие числа окажутся в `левая` и `правая` после программы? Запиши два числа через пробел.

б) Какой вариант на самом деле меняет предметы местами? Запиши его номер.

1. `правая ← левая`, потом `левая ← правая`
2. `запас ← левая`, потом `левая ← правая`, потом `правая ← запас`
3. `левая ← правая`, потом `запас ← левая`, потом `правая ← запас`

Задача 6.2 Что посоветует торговец

если – иначе если, границы, $\geq$ и $>$

Торговец советует покупку по тому, сколько у героя монет:

если монеты ≥ 100 то сказать «Бери меч»	(ответ 1)
иначе если монеты ≥ 40 то сказать «Бери щит»	(ответ 2)
иначе если монеты > 0 то сказать «Бери яблоко»	(ответ 3)
иначе сказать «Приходи с монетами»	(ответ 4)

К торговцу по очереди подходят пять героев. У Ани 100 монет, у Бори 39, у Вики 40, у Гоши 0, у Димы 250. Какой ответ получит каждый? Запиши номера ответов по порядку героев, без пробелов.

Дано: цепочка из трёх проверок и ветки «иначе»; монеты героев: 100, 39, 40, 0, 250.

Найти: номера ответов для пяти героев подряд.

Решение

Шаг 1. Как работает цепочка. Проверки идут сверху вниз. Первое верное условие – и программа говорит свою фразу и дальше не смотрит. Если все три неверны, срабатывает последнее «иначе».

Шаг 2. Аня, 100 монет. $100 \geq 100$ – верно: знак « $\geq$ » пропускает и само число 100. Ответ 1.

Шаг 3. Боря, 39. $39 \geq 100$ – нет. $39 \geq 40$ – нет, не хватило одной монеты. $39 > 0$ – да. Ответ 3.

Шаг 4. Вика, 40. $40 \geq 100$ – нет. $40 \geq 40$ – да. Ответ 2.

Шаг 5. Гоша, 0. $0 \geq 100$ – нет, $0 \geq 40$ – нет, $0 > 0$ – нет: ноль не больше нуля. Срабатывает «иначе». Ответ 4.

Шаг 6. Дима, 250. $250 \geq 100$ – да, сразу ответ 1. То, что 250 больше и 40, и 0, уже неважно: до этих строк программа не дойдёт.

Таблица 6.4

Герой	Монеты	Первое «да»	Ответ
Аня	100	≥ 100	1
Боря	39	> 0	3
Вика	40	≥ 40	2
Гоша	0	иначе	4
Дима	250	≥ 100	1

Самые богатые, Аня и Дима, идут за мечом, а Гоша без гроша слышит «приходи с монетами». Торговец ведёт себя, как задумано.

⚠ Типичная ошибка. Прочитать « $\geq$ » как « $>$ ». Тогда Аня со своими 100 монетами получит щит, а Вика с 40 – яблоко, и выйдет 23341. Граничные числа 100 и 40 в условии задачи стоят не случайно. А Диме кто-то выдаёт «яблоко»: последнее верное условие для него – « > 0 ». Но в цепочке побеждает первое «да», а не последнее.

Ответ: 13241.

Своя задача по программе сферы, модуль 2

Реши сам

6.3. За прохождение уровня дают звёзды:

```
если время ≤ 60 то звёзды ← 3
иначе если время ≤ 90 то звёзды ← 2
иначе если время ≤ 120 то звёзды ← 1
иначе звёзды ← 0
```

Время – в секундах. Пять игроков прошли уровень за 90, 61, 121, 60 и 15 секунд. Запиши число звёзд каждого по порядку, без пробелов.

6.4. Другой программист записал звёзды отдельными проверками, без «иначе»:

```
звёзды ← 0
если время ≤ 60 то звёзды ← 3
если время ≤ 90 то звёзды ← 2
если время ≤ 120 то звёзды ← 1
```

Сколько звёзд получают игроки со временем 50, 85, 110 и 130 секунд? Запиши числа по порядку, без пробелов.

Задача 6.3 Зелья в походе

цикл «для», трассировочная таблица, условие с И

Герой идёт через болото. Каждый ход болото отнимает 20 здоровья. Если после этого здоровья меньше 30 и в сумке есть зелье, герой сразу пьёт одно зелье: +30 здоровья. Программа:

```
здоровье ← 70
зелья ← 3
для ход от 1 до 5:
    здоровье ← здоровье - 20
    если здоровье < 30 и зелья > 0 то
        здоровье ← здоровье + 30
        зелья ← зелья - 1
```

Сколько здоровья будет у героя после пятого хода и сколько зелий останется? Запиши два числа через пробел.

Дано: начальное здоровье 70, зелий 3; 5 ходов по –20; зелье +30, если здоровье < 30 и зелья есть.

Найти: здоровье и число зелий после 5 ходов.

Решение

Шаг 1. Порядок внутри хода. Цикл повторится 5 раз. В каждом повторе сначала минус 20, потом проверка. Условие с И: нужны обе части сразу (глава 4). Здоровье меньше 30 **и** зелья есть.

Шаг 2. Трассировочная таблица. Во втором столбце – здоровье сразу после удара болота, до проверки. В последних двух – значения в конце хода.

Таблица 6.5

Ход	После -20	Меньше 30 и зелья есть?	Здоровье	Зелья
начало	–	–	70	3
1	50	нет	50	3
2	30	нет: 30 не меньше 30	30	3
3	10	да	40	2
4	20	да	50	1
5	30	нет: 30 не меньше 30	30	1

На ходу 2: $50 - 20 = 30$. Условие «меньше 30» неверно: 30 не меньше самого себя. Зелье не пьем. На ходу 3: $30 - 20 = 10$, пьем зелье: $10 + 30 = 40$, зелий $3 - 1 = 2$. На ходу 4: $40 - 20 = 20$, пьем – 50, зелий 1. На ходу 5: $50 - 20 = 30$ – опять ровно на пороге, зелье не пьем.

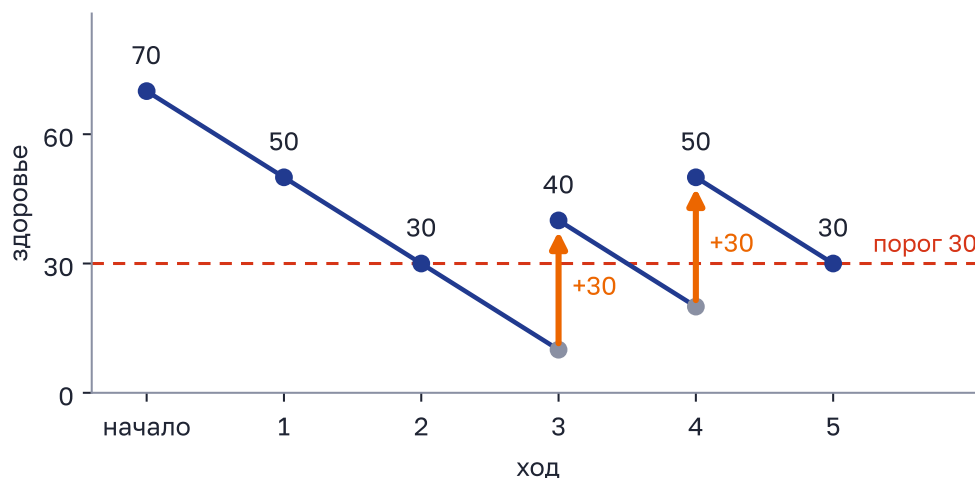


Рисунок 6.11 – Здоровье героя по ходам. Серая точка – здоровье сразу после удара болота, оранжевая стрелка – выпитое зелье, пунктир – порог 30

Шаг 3. Проверяем на Python.

```
здоровье = 70
зелья = 3
for ход in range(1, 6):          # ход = 1, 2, 3, 4, 5
    здоровье = здоровье - 20
    if здоровье < 30 and зелья > 0:
        здоровье = здоровье + 30
        зелья = зелья - 1
    print(ход, здоровье, зелья)  # лог: строка таблицы
print(здоровье, зелья)          # 30 1
```

Проверка без таблицы. Болото отняло $5 \cdot 20 = 100$, выпито два зелья: $2 \cdot 30 = 60$. Итого $70 - 100 + 60 = 30$. Зелий было 3, выпито 2 – осталось 1. Сходится с таблицей.

! Типичная ошибка. На ходу 2 или 5 решить, что при здоровье 30 герой пьёт зелье. Условие «меньше 30», а не «не больше 30». С такой ошибкой зелья уйдут на ходах 2, 4 и 5, и выйдет 60 0. И не сбейся со счёта повторов: «от 1 до 5» – это пять ходов, не четыре и не шесть.

Ответ: 30 1.

Своя задача по программе сферы, модуль 2

Реши сам

6.5. В игре на выживание есть сытость. Программа:

```
сытость ← 40
еда ← 3
для день от 1 до 6:
    сытость ← сытость - 12
    если сытость ≤ 20 и еда > 0 то
        сытость ← сытость + 30
        еда ← еда - 1
```

Сколько сытости и сколько еды будет после шестого дня? Запиши два числа через пробел.

6.6*. У фонтана в квесте две фляги: А на 3 литра и В на 5 литров, сначала обе пустые. Переливать можно только «до конца»: пока не опустеет одна фляга или не наполнится другая. Поэтому переливание из В в А записано так:

```
сколько ← наименьшее из (В, 3 - А)
В ← В - сколько
А ← А + сколько
```

Герой делает по порядку: наполнить В ($В < 5$); перелить из В в А; вылить А ($А < 0$); перелить из В в А; наполнить В; перелить из В в А. Сколько литров будет во флягах А и В в конце? Запиши два числа через пробел.

Задача 6.4 Сундуки на тропе

порядок проверок, цикл, остаток от деления

Тропа в игре – это клетки с номерами от 1 до 30. Геймдизайнер задумал так. Если номер клетки делится и на 3, и на 5, в ней сундук: +10 монет, и больше ничего. Если номер делится только на 3 – одна монета. Если только на 5 – две монеты. В остальных клетках ничего нет. Программист написал:

```
монеты ← 0
для клетка от 1 до 30:
    если клетка делится на 3 то монеты ← монеты + 1
    иначе если клетка делится на 5 то монеты ← монеты + 2
    иначе если клетка делится на 15 то монеты ← монеты + 10
```

а) Сколько монет соберёт герой, пройдя всю тропу, по этой программе?

б) Сколько монет он должен был собрать по задумке геймдизайнера?

Дано: клетки 1–30; задумка: делится на 15 → +10; только на 3 → +1; только на 5 → +2; порядок проверок в программе: 3, 5, 15.

Найти: а) монеты по программе; б) монеты по задумке.

Решение

Шаг 1. Что не так с программой. Число, которое делится и на 3, и на 5, делится на 15. Таких клеток две: 15 и 30. Для клетки 15 программа сначала спросит «делится на 3?» – да. Выдаст одну монету и остальные проверки пропустит. Строка про сундук не сработает **ни разу**: любое число, которое делится на 15, раньше поймает проверка «на 3».

Шаг 2. Считаем по программе. Проверка «на 3» ловит все клетки, номер которых делится на 3: 3, 6, 9, 12, 15, 18, 21, 24, 27, 30. Это $30 : 3 = 10$ клеток, по одной монете – 10 монет. Проверка «на 5» достаётся только тем клеткам, которые не поймала первая: из 5, 10, 15, 20, 25, 30 уходят 15 и 30. Остаются 5, 10, 20, 25 – четыре клетки по две монеты, это 8. Итого $10 + 8 = 18$.

Шаг 3. Считаем по задумке. Сундуки – клетки 15 и 30: $2 \cdot 10 = 20$ монет. Только на 3 – десять клеток без 15 и 30, это 8 клеток и 8 монет. Только на 5 – те же 5, 10, 20, 25: $4 \cdot 2 = 8$. Итого $20 + 8 + 8 = 36$.

Шаг 4. Как починить. Поставить особый случай первым: сначала «делится на 15», потом «на 3», потом «на 5».

Шаг 5. Проверяем на Python. «Делится на 3» – это «остаток от деления на 3 равен нулю», `клетка % 3 == 0`. Сначала программа как у программиста, потом – с сундуком на первом месте.

```

монеты = 0
for клетка in range(1, 31):           # 1, 2, ..., 30
    if клетка % 3 == 0:
        монеты = монеты + 1
    elif клетка % 5 == 0:
        монеты = монеты + 2
    elif клетка % 15 == 0:
        монеты = монеты + 10
print(монеты)                         # 18

монеты = 0
for клетка in range(1, 31):
    if клетка % 15 == 0:               # особый случай – первым
        монеты = монеты + 10
    elif клетка % 3 == 0:
        монеты = монеты + 1
    elif клетка % 5 == 0:
        монеты = монеты + 2
print(монеты)                         # 36

```

Откуда разница в 18 монет? Сундуки пропали. Вместо $2 \cdot 10 = 20$ монет в клетках 15 и 30 герой получил по одной, то есть потерял $20 - 2 = 18$ монет. И правда: $36 - 18 = 18$.

❗ **Типичная ошибка.** В пункте а) посчитать сундуки: «строка про 15 в программе есть – значит, работает». Она есть, но до неё никогда не доходит очередь. В пункте б) бывает перебор в другую сторону: клетке 15 дают и сундук, и монету за «на 3», и монеты за «на 5». А по задумке в клетке с сундуком – «и больше ничего».

Ответ: а) 18; б) 36.

По мотивам задачи НТО 2025/26, профиль «Разработка компьютерных игр» (старший трек, 8–11 классы), заключительный этап, предметный тур по информатике, «Квест»

Реши сам

6.7. Клетки уровня пронумерованы от 1 до 40. Задумка: если номер делится и на 2, и на 7, в клетке портал, +10 кристаллов и больше ничего. Только на 2 – один кристалл, только на 7 – три кристалла. Программист проверяет в порядке «на 2», «на 7», «на 14» цепочкой `иначе если`. а) Сколько кристаллов даёт его программа за весь уровень? б) Сколько должно быть по задумке?

6.8. Чтобы невезучие не бросали игру, в ней есть счётчик неудач (о нём говорилось в главе 4):

```
неудачи ← 0
предметы ← 0
для победа от 1 до 12:
    если повезло или неудачи = 4 то
        предметы ← предметы + 1
        неудачи ← 0
    иначе
        неудачи ← неудачи + 1
```

Удача сама, без счётчика, выпала только на победах номер 2 и 11. Сколько предметов получит игрок за 12 побед? На какой победе предмет выдаст счётчик, а не удача? Запиши два числа через пробел: число предметов и номер победы.

Задача 6.5 Кузнец просит ещё руды

отладка чужого кода, счётчик, граница условия

Описание в документе игры: «Кузнец выковывает меч, когда игрок принесёт 3 куска руды. Каждый подобранный кусок прибавляет к счётчику 1. Если руды 3 или больше, кузнец говорит «Держи меч!», забирает 3 куска и отмечает, что меч выдан. Иначе говорит, сколько руды ещё нужно». Программа:

```
1  руда ← 0
2  меч_выдан ← нет
3  когда игрок подобрал руду:
4      руда ← 1
5  когда игрок заговорил с кузнецом:
6      если руда > 3 то
7          сказать «Держи меч!»
8          руда ← руда - 3
9          меч_выдан ← да
10     иначе
11         сказать «Принеси ещё», 3 - руда
```

Строки с **когда** – это события, как в машинах состояний из главы 5: строки под ними выполняются каждый раз, когда событие случилось. Жалоба с плейтеста – пробной игры (глава 2): «Подобрал 3 куска руды, подошёл к кузнецу – а он говорит: «Принеси ещё 2»».

Найди все строки с ошибками – из-за которых программа работает не по описанию. Запиши их номера по возрастанию, без пробелов.

Дано: описание механики; программа из 11 строк; жалоба «3 руды – просит ещё 2».

Найти: номера всех строк с ошибками.

Решение

Шаг 1. Баг одной фразой. Принёс 3 руды – ждал меч – кузнец просит ещё 2.

Шаг 2. Трассируем действия игрока. В начале `руда = 0`. Игрок подобрал первый кусок – строка 4: `руда ← 1`. Второй кусок – снова строка 4: `руда ← 1`. Третий – опять 1. Строка кладёт в коробку единицу, а не прибавляет её.

Таблица 6.6

Событие (строки)	руда
начало (1)	0
подобрал 1-й кусок (4)	1
подобрал 2-й кусок (4)	1
подобрал 3-й кусок (4)	1
заговорил с кузнецом (6, 11)	1

Разговор: $1 > 3$ – нет, строка 11 говорит «Принеси ещё» и $3 - 1 = 2$. Жалоба объяснена: **строка 4** должна быть `руда ← руда + 1`.

Шаг 3. Мысленно чиним строку 4 и прогоняем ещё раз. Теперь после трёх кусков `руда = 3`. Разговор: $3 > 3$ – нет! Три не больше трёх. Кузнец скажет «Принеси ещё 0» – бессмыслица. По описанию «3 или больше», а это $\geq$. **Строка 6** должна быть `если руда  $\geq$  3 то`.

Шаг 4. Проверяем остальные строки по описанию. Строка 8 забирает 3 куска – так и сказано. Строка 9 ставит флаг «меч выдан» – тоже по описанию. Строки 1 и 2 задают начальные значения: руды 0, меч не выдан. Больше ошибок нет.

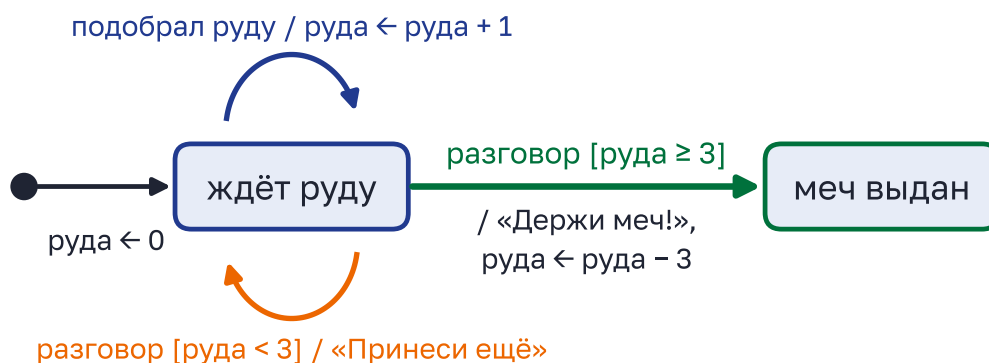


Рисунок 6.12 – Кузнец после исправления – машина состояний. Подпись на стрелке – «событие [условие] / действие»

Шаг 5. Проверяем на Python. Событие «подобрал руду» три раза – это цикл из трёх повторов.

```

# как было
руда = 0
for кусок in range(3):
    руда = 1                # строка 4
if руда > 3:                # строка 6
    print("Держи меч!")
else:
    print("Принеси ещё", 3 - руда)    # Принеси ещё 2

# после исправления строк 4 и 6
руда = 0
for кусок in range(3):
    руда = руда + 1
if руда ≥ 3:
    print("Держи меч!")        # Держи меч!
else:
    print("Принеси ещё", 3 - руда)

```

❗ **Типичная ошибка.** Найти строку 4 и остановиться: жалоба объяснена, ответ 4. А после её исправления кузнец всё равно не отдаст меч за 3 руды – мешает строка 6. Поэтому после каждой находки программу прогоняют заново. Бывает и лишняя находка – строка 8: «зачем забирать руду?». Так написано в описании – это задумка, а не баг.

Ответ: 46.

Своя задача по программе сферы, модуль 5

Реши сам

6.9. Описание: «Дверь в сокровищницу открывается, только если у героя есть ключ и зажжены все 4 факела. Зажёг факел – счётчик факелов растёт на 1». Программа:

```

1  факелы ← 0
2  ключ ← нет
3  когда герой зажёг факел:
4      факелы ← факелы + 1
5  когда герой подобрал ключ:
6      ключ ← да
7  когда герой толкнул дверь:
8      факелы ← 0
9      если ключ = да или факелы = 4 то
10         открыть дверь

```

Жалоба: «Нашёл ключ и сразу толкнул дверь, ни одного факела не зажёл, – а она открылась». Найди все строки с ошибками и запиши их номера по возрастанию, без пробелов.

6.10. Описание: «Здоровье не бывает больше 100. Зелье +40, но не выше 100. Шип отнимает 25. Когда здоровье закончилось, игрок проиграл». Программа:

```

1  здоровье ← 100
2  проигрыш ← нет
3  когда герой выпил зелье:
4      здоровье ← здоровье + 40
5      если здоровье > 100 то здоровье ← 100
6  когда герой наступил на шип:
7      здоровье ← здоровье - 25
8      если здоровье = 0 то проигрыш ← да

```

Герой по очереди: наступил на шип 3 раза, выпил зелье, наступил на шип ещё 3 раза. а) Чему будет равно `здоровье` и что будет в `проигрыш` ? б) В какой строке ошибка?

Задача 6.6 Таблица баланса

электронные таблицы, копирование формул, абсолютная ссылка, баланс

Геймдизайнер сравнивает четыре оружия в бою с боссом (таблица – на рисунке в разделе 6.6). В столбце В – урон одного удара, в С – ударов в секунду, в В7 – здоровье босса, 720. В D2 написана формула `=B2*C2`, в E2 – `=$B$7/D2`. Обе формулы скопировали вниз до строки 5.

Таблица 6.7

	А	В	С
2	Кинжал	12	3
3	Меч	30	1,5
4	Молот	80	0,5
5	Лук	20	2

- а) Какая формула окажется в ячейке E4?
 б) Каким оружием босса можно победить быстрее всего и за сколько секунд?
 в) Помощник по ошибке написал в E2 формулу `=B7/D2`, без долларов, и тоже скопировал её вниз. Что покажет ячейка E3?

Дано: B2:C5 – урон и скорость; B7 = 720; D2 `=B2*C2`, E2 `=$B$7/D2`, скопированы до строки 5.
Найти: а) формулу в E4; б) самое быстрое оружие и время; в) значение E3 при формуле без долларов.

Решение

Шаг 1. Пункт а): как сдвигаются ссылки. Из E2 в E4 – на 2 строки вниз. Обычная ссылка D2 становится D4. Ссылка `$B$7` с долларами стоит на месте. В E4 будет `=$B$7/D4`.

Шаг 2. Считаем урон в секунду (столбец D). В каждой строке своё: $D2 = 12 \cdot 3 = 36$, $D3 = 30 \cdot 1,5 = 45$, $D4 = 80 \cdot 0,5 = 40$, $D5 = 20 \cdot 2 = 40$.

Шаг 3. Время боя (столбец E). Здоровье босса делим на урон в секунду: кинжал $720:36 = 20$ с, меч $720:45 = 16$ с, молот и лук $720:40 = 18$ с.

Таблица 6.8

Оружие	D, урон в секунду	E, секунд
Кинжал	36	20
Меч	45	16
Молот	40	18
Лук	40	18

Меньше всего времени – у меча: 16 секунд. Функция `=МИН(E2:E5)` нашла бы это число сама.

Шаг 4. Пункт в): формула без долларов. `=B7/D2`, скопированная на одну строку вниз, превращается в `=B8/D3`. Обе ссылки сдвинулись. В B8 ничего нет, пустая ячейка считается нулём. Выходит $0:45 = 0$. Таблица покажет, что меч побеждает босса за 0 секунд, – явная чепуха, по ней баг и видно.

А можно было не делить. Кто наносит больше урона за секунду, тот и побеждает быстрее. Меч бьёт реже кинжала, зато сильнее, и за секунду выдаёт больше всех – 45. Молот и лук наносят по 40 в секунду – у них и время одинаковое.

 **Типичная ошибка.** Сравнивать оружие по урону одного удара: «у молота 80 – он самый сильный». Молот бьёт раз в две секунды, в секунду у него 40, меньше, чем у меча. Вторая ошибка – в пункте а) сдвинуть и `$B$7:=$B$9/D4`. Доллары для того и ставят, чтобы ссылка не двигалась.

Ответ: а) `=$B$7/D4`; б) меч, 16 с; в) 0.

Своя задача по программе сферы, модуль 1

Реши сам

6.11. В ячейке C2 формула `=A2+B2`, её протянули вниз до C6. В ячейке D2 формула `=A2*$F$1`, её скопировали в D4. Какие формулы окажутся в C5 и в D4? Запиши обе через точку с запятой.

6.12. В таблице уровней в столбце В – сколько монет собрал игрок, в С – сколько монет нужно, чтобы открыть дверь:

Таблица 6.9

Строка	В	С
2	35	30
3	29	30
4	30	30
5	50	60
6	61	60

В D2 записана формула `=ЕСЛИ(В2 ≥ С2; "открыта"; "закрыта")`, её протянули до D6. а) В скольких строках в столбце D будет «открыта»? б) Что покажет ячейка `=СУММ(В2:В6)`? в) Что покажет `=МАКС(В2:В6) - МИН(В2:В6)`?

Задача 6.7 *. Модуль урона

функции, вызов функции из функции, цикл «пока», порядок параметров

Юля делает боевую систему и разбила её на две функции. Первая считает урон одного удара: сила минус броня, но не меньше 1. Вторая считает, сколько ударов нужно, чтобы победить врага.

```
def урон(сила, броня):  
    d = сила - броня  
    if d < 1:  
        d = 1  
    return d  
  
def удары_до_победы(здоровье, сила, броня):  
    n = 0  
    while здоровье > 0:  
        здоровье = здоровье - урон(сила, броня)  
        n = n + 1  
    return n  
  
print(удары_до_победы(50, 12, 4))  
print(удары_до_победы(30, 10, 10))  
print(удары_до_победы(50, 4, 12))
```

Что напечатает программа? Запиши три числа через пробел.

Дано: функция `урон` (сила – броня, не меньше 1); функция `удары_до_победы` (бьёт, пока здоровье > 0); три вызова.

Найти: три напечатанных числа.

Решение

Шаг 1. Что делает каждая функция. урон вычитает броню из силы. Если вышло меньше 1 (ноль или минус), урон становится 1: совсем без урона враг был бы бессмертным. удары_до_победы заводит счётчик ударов n и бьёт, пока здоровье больше нуля. Каждый удар отнимает урон(сила, броня) и прибавляет к счётчику 1. Когда здоровье стало 0 или меньше, цикл кончается, и функция возвращает n .

Шаг 2. Первый вызов: удары_до_победы(50, 12, 4). Числа ложатся в параметры по порядку: здоровье = 50, сила = 12, броня = 4. Урон: $12 - 4 = 8$, это не меньше 1 – остаётся 8. Трассируем цикл:

Таблица 6.10

Удар n	Здоровье после удара
1	42
2	34
3	26
4	18
5	10
6	2
7	-6

После шестого удара у врага осталось 2 – он ещё жив, нужен седьмой. После седьмого здоровья -6, условие « > 0 » неверно, цикл закончен. Напечатается 7. Проверка по формуле из главы 2: $50 : 8 = 6,25$, округляем вверх – 7.

Шаг 3. Второй вызов: удары_до_победы(30, 10, 10). Урон: $10 - 10 = 0$. Ноль меньше 1, значит, урон 1. Каждый удар отнимает по одному здоровью, из 30 нужно 30 ударов. Напечатается 30.

Шаг 4. Третий вызов: удары_до_победы(50, 4, 12). По порядку: здоровье = 50, сила = 4, броня = 12. Урон: $4 - 12 = -8$, меньше 1 – значит, 1. Нужно 50 ударов. Напечатается 50. Похоже, Юлия хотела посчитать то же, что в первом вызове, но перепутала силу и броню местами. Программа не ругается: для неё 4 и 12 – просто числа, и она считает слабый удар по толстой броне.

Шаг 5. Проверяем запуском. Программа печатает 7, 30 и 50 – каждое число на своей строке. Сильный удар по слабой броне – всего 7 ударов. Когда броня гасит удар целиком, враг теряет по единичке, и ударов нужно столько, сколько у него здоровья: 30 и 50.

⚠ Типичная ошибка. В первом вызове ответить 6: « $50 : 8 = 6,25$, значит, 6». После шести ударов у врага 2 здоровья, и цикл «пока здоровье > 0 » делает седьмой удар. Вторая ошибка – в третьем вызове подставить числа «по смыслу», а не по порядку: сила 12, броня 4 – и ответить 7. Компьютер кладёт числа в параметры строго по порядку. А во втором вызове иногда получают урон 0 и решают, что цикл бесконечный. Как раз от этого в функции урон и стоит проверка $d < 1$.

Ответ: 7 30 50.

Реши сам

6.13. Что напечатает программа?

```
def награда(уровень):  
    if уровень ≥ 10:  
        return 50  
    return 5 * уровень  
  
всего = 0  
for y in range(8, 12):  
    всего = всего + награда(y)  
print(всего)
```

Напоминание: `return` сразу завершает функцию, строки ниже уже не выполняются.

6.14*. * Функция лечения не даёт здоровью подняться выше максимума:

```
def лечение(здоровье, зелье, максимум):  
    здоровье = здоровье + зелье  
    if здоровье > максимум:  
        здоровье = максимум  
    return здоровье  
  
hp = 70  
hp = лечение(hp, 50, 100)  
hp = hp - 45  
hp = лечение(hp, 100, 30)  
print(hp)
```

В последнем вызове программист хотел выпить зелье на 30 при максимуме 100, но перепутал числа. а) Что напечатает программа? б) Что напечатала бы она, если бы последний вызов был записан правильно?

Сюжет и диалоги: язык Ink



Помочь лису или прогнать? Пойти в лес или к реке? В игре с выборами это решаешь ты, а сценарист заранее рисует схему: куски текста и стрелки между ними. В «Урсуле» такие истории пишут на языке Ink. Здесь ты научишься читать короткую историю на Ink, считать по схеме прохождения и концовки и находить ошибки в чужом коде. Задачи в главе устроены как на отборе: дан кусок истории, а спрашивают, что увидит игрок, сколько всего концовок или в какой строке ошибка. Ответ – число, строка цифр или номера строк.

7.1. Ветвящийся сюжет: узлы, выборы, последствия

НА ПАЛЬЦАХ

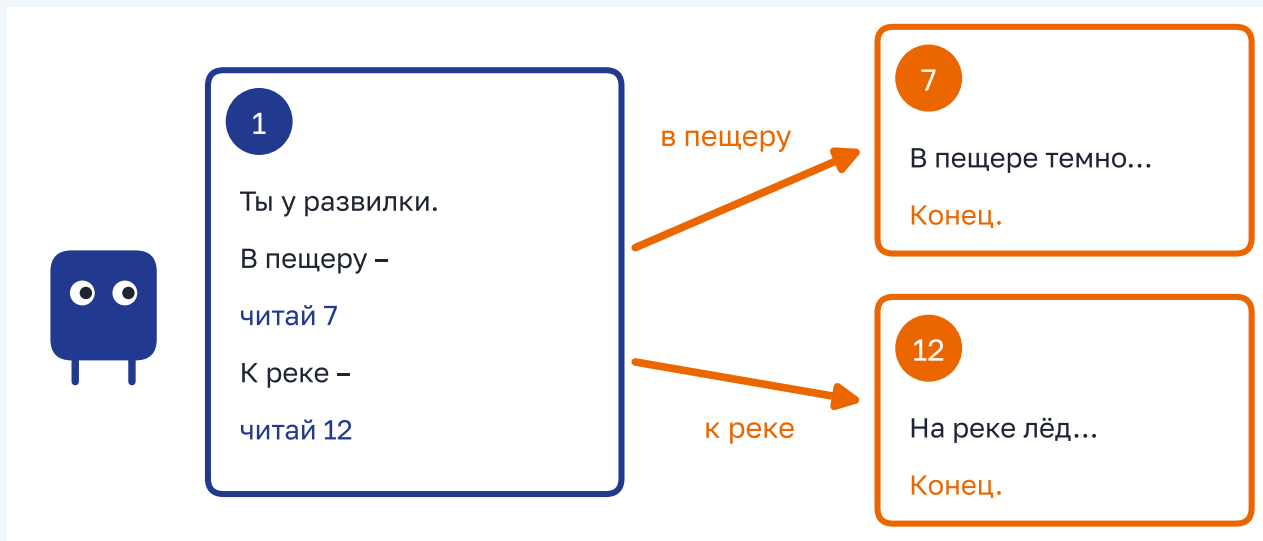


Рисунок 7.1 – Книга-игра: каждый абзац говорит, какой читать следующим

Абзац – это узел. «Читай абзац 7» – переход. Куда идти, решает читатель.

Обычная книга читается одной дорогой: страница за страницей. А в книге-игре абзац кончается так: «Если идёшь в пещеру – читай абзац 7, если к реке – абзац 12». Два читателя одной книги могут прочитать две разные истории. Сюжет в компьютерной игре с выборами устроен точно так же, только листать страницы за тебя будет программа.

Такой сюжет называют **ветвящимся**, или нелинейным: у него есть развилки. Собран он из таких деталей:

- **Узел** – кусок истории, который читается целиком: сцена, место, разговор. В книге-игре это абзац.
- **Выбор** – кнопка, которую нажимает игрок: «Пойти в пещеру», «Промолчать», «Отдать яблоко».
- **Переход** – куда ведёт выбор, то есть в какой узел история прыгнет дальше. «Читай абзац 7» – переход.
- **Концовка** – узел, на котором история заканчивается.
- **Последствие** – то, что изменилось из-за выбора: другая сцена, другая реплика, открылась или закрылась дорога.

Выбор без последствий игрок быстро раскусывает. На вебинаре сферы об этом говорили так: интересен **значимый выбор** – тот, после которого игрок видит, что история пошла по-другому. В Minecraft: Story Mode после важного решения на экране появляется надпись: такой-то персонаж это запомнит. И правда запоминает: с кем ты поссорился, тот дальше с тобой не дружит. А три одинаковые двери без единой подсказки – это не выбор, а лотерея.

Сюжеты бывают разной формы. Их удобно узнавать по картинке.

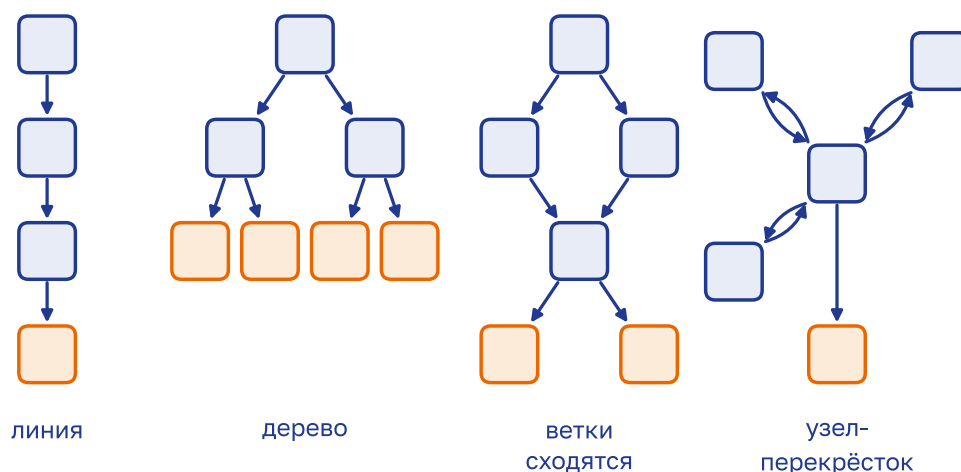


Рисунок 7.2 – Четыре формы сюжета. Прямоугольник – узел, стрелка – переход, оранжевые прямоугольники – концовки

- **Линия.** Выборов нет или они ничего не меняют. Так устроены почти все платформеры: история одна, а игрок выбирает только, как прыгать.
- **Дерево.** Каждая развилка разводит историю навсегда. Концовок очень много: 5 развилок по 2 варианта – это уже $2^5 = 32$ концовки (как считать, мы разбирали в главе 4). Каждую надо написать, и сценарист быстро выдыхается.
- **Ветки сходятся.** После развилки дороги снова встречаются в общем узле. Так делают почти во всех сюжетных играх: выбор меняет одну-две сцены, а дальше история снова общая.
- **Узел-перекрёсток.** Из одного узла можно уйти в несколько мест и вернуться обратно. Прихожая, из которой ведут двери в комнаты. Ларёк, где можно спросить о разном. Игрок сам решает, в каком порядке всё смотреть.

Запомни. Узел – кусок истории, выбор – кнопка, переход – куда она ведёт, концовка – узел, где всё кончается. Выбор значимый, если у него есть последствия, которые игрок замечает.

7.2. Граф сюжета: считаем прохождения и концовки

НА ПАЛЬЦАХ

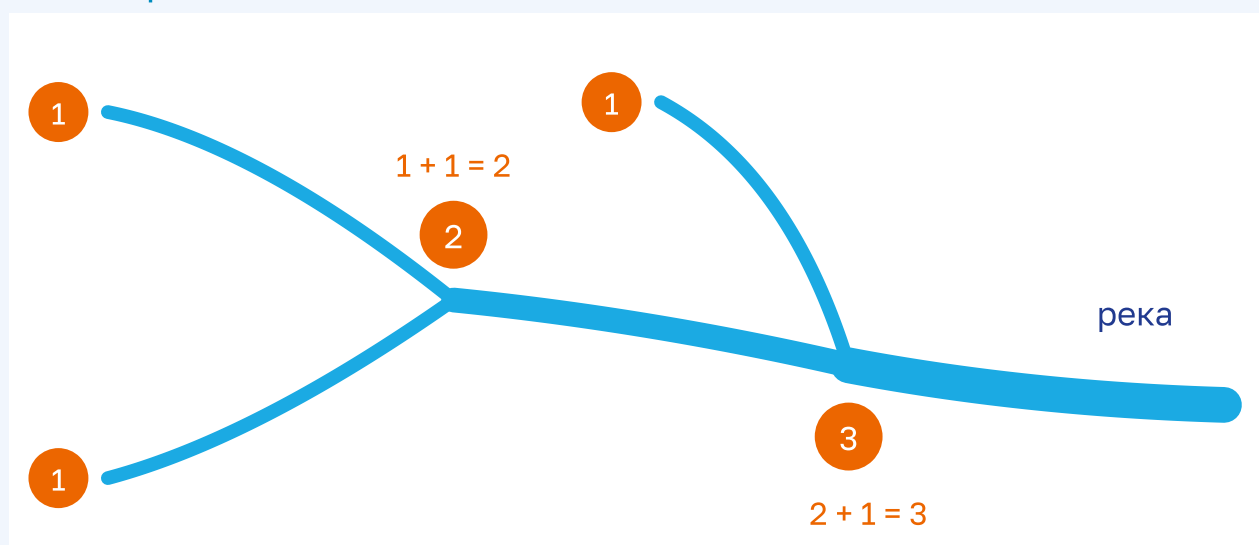


Рисунок 7.3 – Ручьи сливаются в реку

Дороги к узлу складываются, как ручьи: сколько путей пришло по каждой стрелке, столько и слилось в узле.

Нарисуй каждый узел прямоугольником, а каждый переход – стрелкой. Получится **граф сюжета** – та же схема, что карта уровня в главе 3 и карта сюжета в главе 4. На нём видно всё сразу: где развилки, где ветки сходятся, где концовки. Граф удобно нарисовать раньше, чем писать реплики: так сразу понятно, сколько работы впереди.

По графу легко сосчитать **концовки** – узлы, где история заканчивается и куда можно попасть. Чуть сложнее с **прохождениями**. Прохождение – это вся цепочка нажатых кнопок от начала до концовки, и их обычно больше, чем концовок: к одной концовке ведут разные дороги. А по **самому короткому и самому длинному** прохождению (сколько в них кнопок) прикидывают, сколько минут займёт игра.

Прохождения считают так же, как пути в главе 4. У начального узла пишем 1. У каждого следующего узла – сумму чисел тех узлов, откуда в него ведут стрелки. Узел считаем, когда посчитаны все, кто в него ведёт. Одно новое правило: если из узла в один и тот же узел ведут **две разные кнопки**, это две разные стрелки. Прохождения тоже разные – игрок нажимал разное. Число такого узла прибавляем дважды.

Пример – история про снежную крепость во дворе.

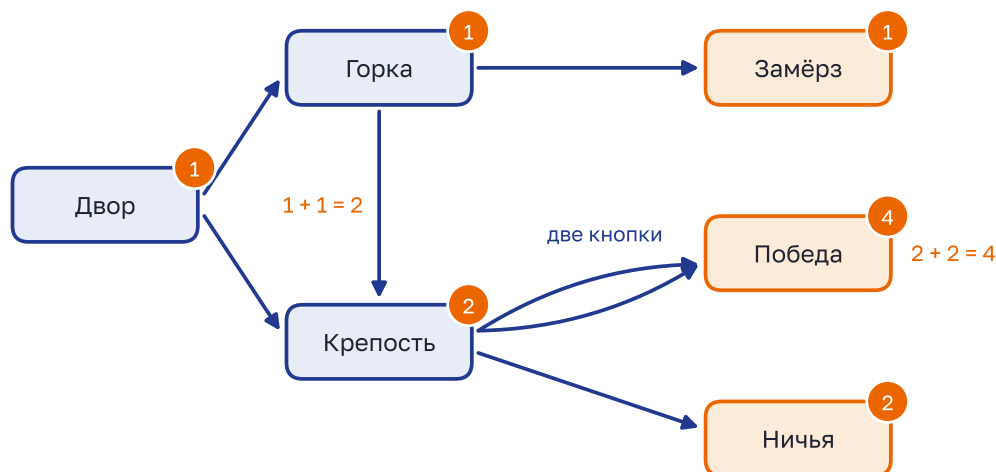


Рисунок 7.4 – Граф истории про снежную крепость. Оранжевое число у узла – сколько прохождений в него ведёт. Из «Крепости» в «Победу» ведут две кнопки

Таблица 7.1

Узел	Откуда стрелки	Прохождений
Двор	начало	1
Горка	Двор	1
Крепость	Двор, Горка	$1 + 1 = 2$
Замёрз	Горка	1
Победа	Крепость, 2 кнопки	$2 + 2 = 4$
Ничья	Крепость	2

Концовок три: «Замёрз», «Победа», «Ничья». Прохождений $1 + 4 + 2 = 7$. Самое короткое – 2 кнопки (Двор → Крепость → концовка), самое длинное – 3 (Двор → Горка → Крепость → концовка). А чтобы увидеть все концовки, хватит трёх прохождений – по одному на каждую.

Этот способ работает, только когда стрелки ведут вперёд. Если стрелка возвращает игрока в узел, где он уже был (приходя с дверями в комнаты), по кругу можно ходить сколько угодно. Там нужен другой подсчёт – он разобран в задаче 7.7.

Запомни. У начала – 1, у узла – сумма чисел тех, кто в него ведёт. Две кнопки в один узел – две стрелки. Всего прохождений – сумма чисел у всех концовок.

Совет. Прежде чем считать, выпиши таблицу «узел – откуда в него ведут стрелки». Так не пропустишь стрелку, которая идёт издалека, через несколько узлов.

7.3. Первая история на Ink

НА ПАЛЬЦАХ

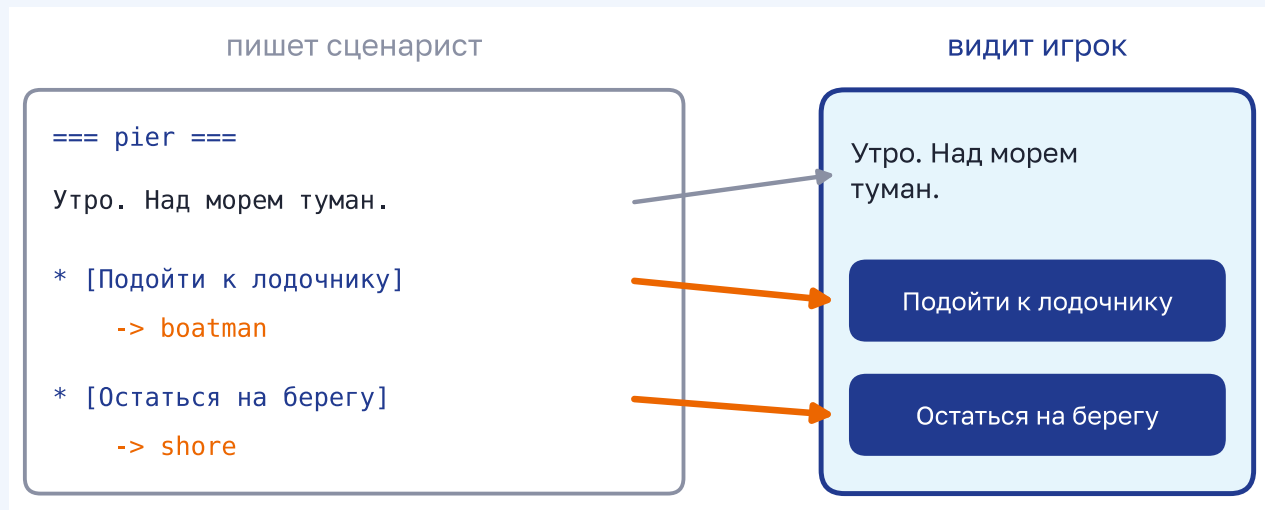


Рисунок 7.5 – Слева – что пишет сценарист, справа – что видит игрок

Обычная строка – это текст на экране. Строка со звёздочкой – кнопка. Стрелка говорит, куда кнопка ведёт.

Ink – язык для историй с выборами. Его придумала британская студия inkle, а писать на нём удобно в бесплатном редакторе Inky: историю пишешь слева, а справа сразу в неё играешь. Ink ничего не рисует и не двигает героев – он только выводит текст и кнопки. В «Урсуле» есть модуль повествования: он связывает историю на Ink с 3D-игрой, и выбор в разговоре может менять то, что происходит на сцене. Так рассказывали на установочном вебинаре сферы. Вот целая история. Её можно вставить в Inky и сыграть.

```
// История «Туман над Обским морем»
→ pier

=== pier ===
Утро. Над Обским морем туман.
Лодочник Семёныч машет тебе с причала.
* [Подойти к лодочнику] → boatman
* [Остаться на берегу] → shore

=== boatman ===
– Туман к обеду уйдёт, – говорит Семёныч. – Поплывём к острову?
* [Поплыть] → island
* [Отказаться] → shore

=== island ===
На острове ты находишь старый маяк.
→ END

=== shore ===
Ты кидаешь камешки в воду и ждёшь, пока рассеется туман.
→ END
```

Теперь построчно.

- `// История...` – **комментарий**. Всё, что после двух косых черт, Ink не показывает. Это записка для себя.
- `→ pier` – **переход** в узел `pier`. Стрелку набирают двумя знаками: минус `-` и «больше» `>`. Ink читает файл с самой первой строки, поэтому первой строкой мы сразу отправляем историю в начальный узел.
- `≡ pier ≡` – начало **узла** с именем `pier` («причал»): по три знака «равно» с каждой стороны. Имя пишут латинскими буквами без пробелов, вместо пробела ставят подчёркивание: `found_river`. Узел тянется до следующей строки с `≡`.
- Обычные строки (`Утро. Над Обским морем туман.`) – текст, который игрок прочитает. Каждая строка – новый абзац.
- `* [Подойти к лодочнику] → boatman` – **выбор**. Звёздочка в начале говорит: это кнопка. В квадратных скобках – что написано на кнопке. После стрелки – куда игрок попадёт, если её нажмёт.
- `→ END` – история закончилась. Узлы `island` и `shore` – концовки.

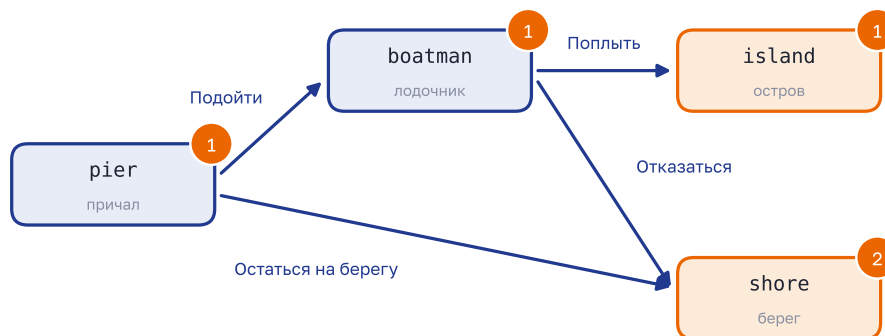


Рисунок 7.6 – Граф истории «Туман над Обским морем». Два прохождения ведут на берег, одно – на остров

Граф нарисовать легко: каждый узел – прямоугольник, каждая стрелка `→` внутри узла – стрелка на графе. Концовки две, прохождений три: «Остаться на берегу»; «Подойти к лодочнику» `→` «Поплыть»; «Подойти к лодочнику» `→` «Отказаться».

Квадратные скобки. Если текст выбора написать без скобок, он появится и на кнопке, и потом в самой истории, как будто герой произнёс его вслух. Строка `* «Привет!»` покажет кнопку «Привет!», а после нажатия – реплику «Привет!» в тексте. Строка `* [Промолчать]` покажет кнопку, но в историю слово «Промолчать» не попадёт. Реплику героя удобно писать без скобок, а действие – в скобках.

НА ПАЛЬЦАХ

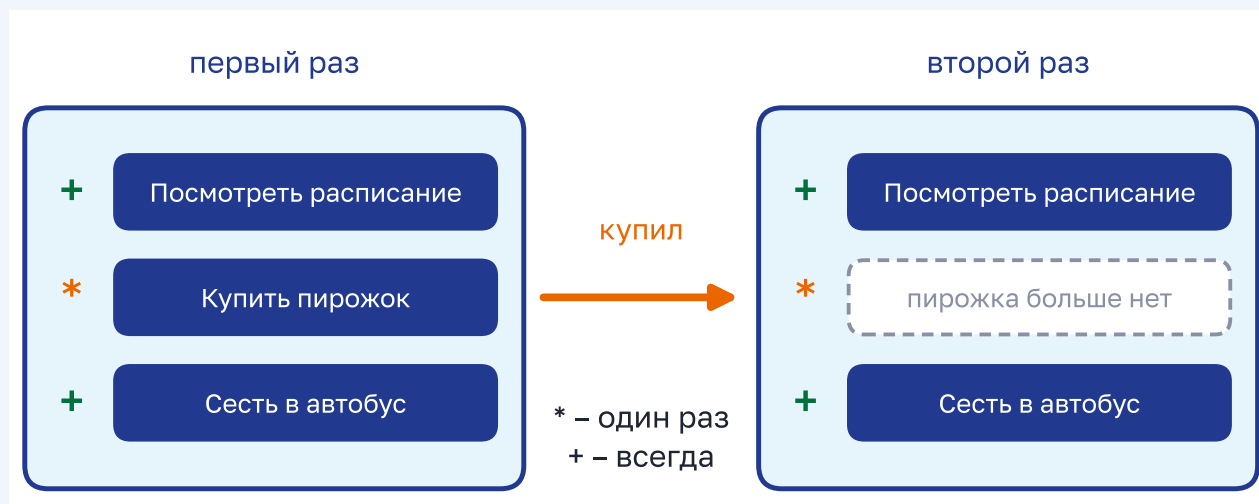


Рисунок 7.7 – Остановка у школы: пирожок купил – кнопки больше нет, а расписание можно смотреть снова

Звёздочка – одноразовая кнопка, как последний пирожок в ларьке. **Плюс** – кнопка навсегда.

Звёздочка и плюс. Выбор со звёздочкой * исчезает, как только его нажали. Выбор с плюсом + остаётся навсегда. Разница видна в узле-перекрёстке, куда игрок возвращается снова и снова:

```
→ stop

≡ stop ≡
Остановка у школы. Автобус придёт через пять минут.
+ [Посмотреть расписание]
    Автобус идёт до Академгородка без пересадок.
    → stop
* [Купить пирожок]
    Пирожок с капустой, ещё тёплый.
    → stop
+ [Сесть в автобус] → END
```

Сначала на остановке три кнопки. Купил пирожок – в следующий раз кнопок уже две: второй пирожок не продадут. Расписание можно смотреть сколько угодно, у этой кнопки плюс. Строки под выбором – что происходит после нажатия, а `→ stop` в конце возвращает игрока на остановку. Отступы слева Ink не замечает, их ставят, чтобы человеку было легче читать.

Место сбора. Бывает, что выборы только слегка меняют текст, а дальше история общая. Тогда после выборов ставят строку, которая начинается с чёрточки . Это **место сбора**: сюда история приходит после любого выбора, у которого нет своего перехода.

→ meet

≡≡≡ meet ≡≡≡

Лис смотрит на тебя из-за ёлки.

* «Привет!»

* [Промолчать]

Ты молчишь.

- Лис фыркает и убегает.

→ END

Нажмёшь «Привет!» – увидишь реплику «Привет!», а потом «Лис фыркает и убегает». Нажмёшь «Промолчать» – «Ты молчишь» и то же «Лис фыркает и убегает». На графе место сбора – это ветки, которые сходятся.

Запомни. ≡≡≡ имя ≡≡≡ – начало узла, → имя – переход, → END – конец истории. * – кнопка, которая исчезает, + – кнопка, которая остаётся. Текст в [] виден только на кнопке. - – место сбора.

Совет. Проверяй историю в Inky после каждого нового узла: пройди его сам, пока помнишь, что задумал. Опечатку в одном узле найти легко, а в двадцати сразу – тяжело.

7.4. Переменные: история запоминает выбор

НА ПАЛЬЦАХ

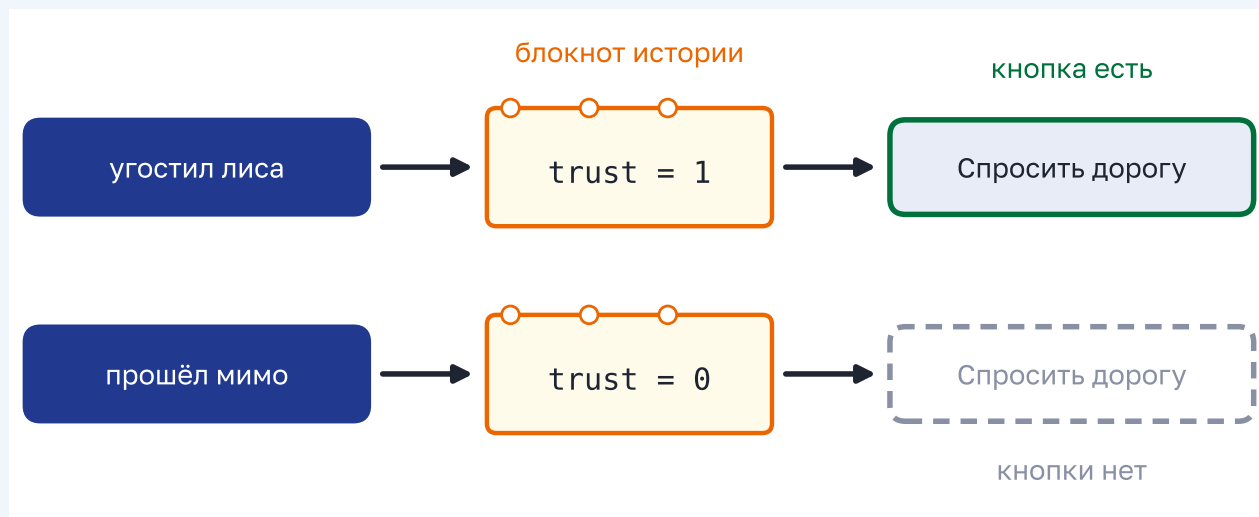


Рисунок 7.8 – Блокнот истории: что было раньше, решает, какие кнопки покажут потом

Угостил лиса – в блокнот легла запись. У развилки история заглядывает в блокнот: запись есть – кнопка «Спросить дорогу» появится.

Чтобы выбор имел последствия через несколько сцен, история должна его запомнить. Для этого есть **переменные** – коробки с именем, в которых лежит число или «да/нет» (подробно о

них – в главе 6). В Ink с переменными работают так:

- `VAR trust = 0` – завести переменную `trust` («доверие») и положить в неё 0. Пишут в самом начале файла.
- `VAR has_map = false` – переменная «да/нет». `true` – «да», `false` – «нет».
- `~ trust = trust + 1` – изменить значение. **Тильда** `~` в начале строки говорит Ink: это команда, её не надо показывать на экране.
- `{trust}` в тексте – подставить значение: «У тебя {coins} монет» превратится в «У тебя 5 монет».
- `* {trust > 0} [Спросить дорогу]` – **выбор с условием**. Кнопка появится, только если условие в фигурных скобках верно. Если неверно, игрок её просто не увидит.
- `{has_map: Текст, если да.|Текст, если нет.}` – **текст по условию**: до вертикальной черты – если условие верно, после черты – если нет.
- `{trust ≥ 3: → glade}` – **переход по условию**: если условие верно, история уходит в узел `glade`.

Сравнивают знаками `>`, `<`, `≥` (больше или равно), `≤`, `=` (равно, два знака!) и `≠` (не равно). Знаки из двух символов набирают подряд, без пробела: `>` и `=` – это `≥`. Условия соединяют словами `and` («и»), `or` («или»), `not` («не»). Таблица знаков есть в справочнике в конце книги.

Ещё Ink сам считает, сколько раз игрок побывал в каждом узле. Имя узла в условии – это число посещений: `{kitchen}` верно, если игрок уже был на кухне, а `{not kitchen}` – если ещё не был. Заводить для этого переменную не нужно.

Вот история, где выбор в начале открывает кнопку в середине и меняет концовку.

```
VAR trust = 0           // доверие лиса
VAR has_map = false     // есть ли у героя карта

→ forest

=== forest ===
На опушке сидит лис. Он смотрит на твой рюкзак.
* [Угостить лиса печеньем]
  ~ trust = trust + 1
  Лис хрустит печеньем и виляет хвостом.
* [Пройти мимо]
  Лис провожает тебя взглядом.
- Впереди развилка тропинок.
* {trust > 0} [Спросить у лиса дорогу]
  ~ has_map = true
  Лис рисует лапой на песке карту.
* [Выбрать тропинку наугад]
- → lake

=== lake ===
{has_map: Ты сразу выходишь к озеру.|Ты долго плутаешь и выходишь к озеру к вечеру.}
→ END
```

Посчитаем прохождения. На опушке 2 кнопки, у развилки тоже 2 – кажется, что прохождений $2 \cdot 2 = 4$. Но кнопка «Спросить у лиса дорогу» появляется, только если `trust > 0`, то есть

только после печенья. Кто прошёл мимо, у развилки увидит одну кнопку. Прохождений **три**: печенье → спросить дорогу; печенье → наугад; мимо → наугад. Условие вырезает из графа стрелку: нарисовать её можно, а пройти по ней получится не всегда.

Порядок переходов по условию. Если подряд стоят несколько строк вида {условие: → узел}, Ink проверяет их сверху вниз и уходит по **первой** верной. Поэтому сначала пишут самое строгое условие:

```
{trust ≥ 3: → berry_glade}  
{trust ≥ 1: → edge}  
→ bushes
```

При `trust = 3` верны оба условия, но сработает первое – игрок попадёт на поляну. При `trust = 1` – на опушку. При 0 и меньше оба условия ложны, и последняя строка отправит игрока в кусты. Поменяешь первые две строки местами – до поляны не доберётся никто: любой с доверием 3 раньше уйдёт на опушку.

Запомни. `VAR` заводит переменную, `~` меняет её, {имя} показывает. `* {условие} [...]` – кнопка появится, только если условие верно. Условия перехода проверяются сверху вниз, срабатывает первое верное. «Равно» – это `=`.

Совет. Называй переменные так, чтобы имя говорило само за себя: `has_key`, `trust`, `coins`. Через неделю `x1` и `x2` не вспомнит никто. Рядом пиши комментарий по-русски: `// доверие лиса`.

7.5. Недостижимые концовки и другие ошибки

НА ПАЛЬЦАХ

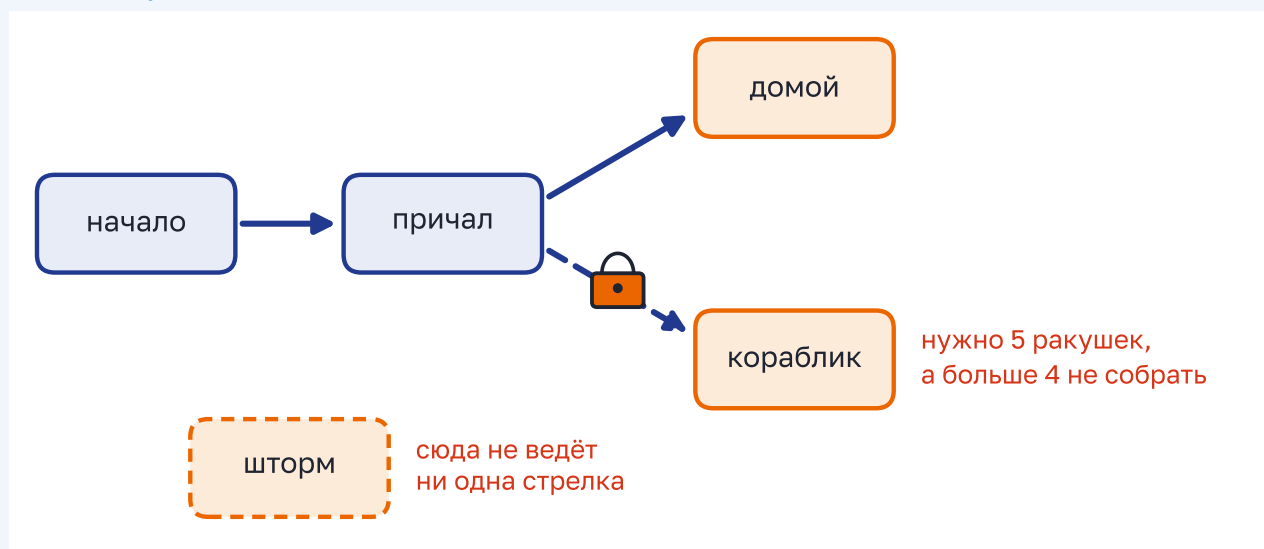


Рисунок 7.9 – Комната, в которую не ведёт ни одна дверь

Концовка написана, но ни одна дорога к ней не ведёт или дверь заперта навсегда. Игрок её никогда не увидит.

Недостижимая концовка – концовка, которую игрок не увидит, как бы ни играл. Сценарист её написал, а дороги к ней нет. Причины всего две. Первая: **к узлу не ведёт ни одна стрелка**. Узел есть, а `→ имя_узла` нет нигде в файле. Так бывает, когда переход переделали и забыли про старый узел. Ink об этом не предупреждает – узел просто лежит без дела. Вторая: **стрелка есть, но её условие не выполнится никогда**. Скажем, кнопка «Купить кораблик» требует `shells ≥ 10`, а собрать за всю игру можно самое большее 4 ракушки.

Как искать. Для каждой концовки найди в коде все `→ имя` и `{...: → имя}`, которые в неё ведут. Если их нет – причина первая. Если есть, но с условием, посчитай, какое **самое большое** и **самое маленькое** значение может получить переменная за всю игру. Для этого пройди по графу и на каждой развилке бери сначала вариант, где прибавляют больше всего, а потом – где меньше всего. Если условие требует значения за этими границами, концовка недостижима. Когда вариантов мало, надёжнее выписать все прохождения в таблицу.

Другие частые ошибки в историях на Ink:

Таблица 7.2

Ошибка в коде	Что будет
опечатка в имени: <code>→ forrest</code>	Inky не запустит историю: узел не найден
узел кончился без <code>→</code>	история обрывается на полуслове
нет <code>~</code> перед <code>coins = coins + 5</code>	строка появится на экране как текст
<code>=</code> вместо <code>==</code> в условии	Inky покажет ошибку или странный текст
<code>*</code> в узле, куда возвращаются	со второго раза кнопки уже нет

Ошибка в коде	Что будет
все кнопки перекрёстка со *	кнопки кончатся, история оборвётся

Про последнюю ошибку подробнее. Если в узле-перекрёстке все кнопки со звёздочками, рано или поздно игрок нажмёт их все. Кнопок не останется, и Inky сообщит, что история «кончилась» – а концовки не было. Лечится это **запасным выбором** – строкой `* → finale` без текста. Такой выбор игрок не видит: Ink выбирает его сам, когда других кнопок не осталось.

Все эти ошибки ловят **плейтестом** – пробной игрой. В программе сферы ему отдельное занятие. Пройди каждую концовку хотя бы один раз. Потом дай сыграть другу и молча смотри, где он застрял, что пропустил и где заскучал.

Запомни. Концовка недостижима, если в неё не ведёт ни одна стрелка или условие на стрелке не выполнится никогда. Для проверки считай самое большое и самое маленькое значение переменной. Без `~` команда становится текстом.

Совет. Выпиши на листок все концовки, а рядом отмечай, какую уже видел в плейтесте. Не отмеченная после десяти прохождений – первый кандидат в недостижимые.

7.6. Хороший диалог

НА ПАЛЬЦАХ

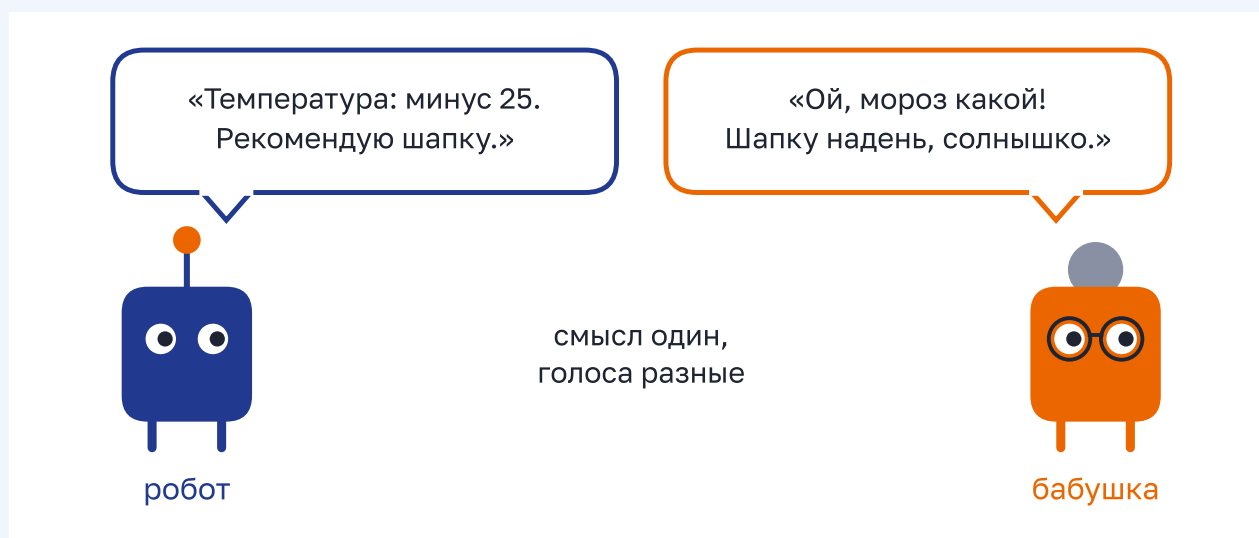


Рисунок 7.10 – Одно и то же – двумя голосами

Смысл один: на улице холодно, надень шапку. А кто говорит, понятно без подписи.

Диалог в игре читают на бегу: игрок хочет дальше, к действию. Поэтому у хорошего диалога свои правила.

У каждого персонажа свой голос. Робот говорит точно и с числами. Бабушка – ласково, про еду и тепло. Капитан катера – коротко и командами. Проверка простая: закрой имя говорящего. Если по реплике всё равно понятно, кто это, – голос получился.

Одна реплика – одна мысль. Длинный абзац на экране пролистывают не читая. Держи реплику в пределах одной-двух строк, примерно до 15 слов. Не влезает – разбей на две реплики или дай игроку кнопку «Расскажи подробнее». У старших школьников на втором отборочном этапе НТО 2025/26 по разработке игр даже для тизера – короткого ролика про игру – было правило: меньше 15 слов на экран.

Реплика что-то делает. Сообщает то, что нужно игроку, показывает характер или двигает сюжет. «Привет. – Привет. – Как дела? – Нормально» – четыре пустые реплики подряд.

Показывай, а не объясняй. Вместо «Я очень злой сторож» лучше: «Ещё раз наступишь на мою грядку – пожалеешь». Игрок сам поймёт, что сторож злой, и поверит больше.

Выборы разные, а последствия заметные. Кнопки «Да», «Ага» и «Конечно» – это один выбор в трёх разных костюмах. Дай игроку разные пути: вежливо, дерзко, промолчать. И пусть персонаж потом ответит по-разному. Кнопка короткая и подсказывает, что скажет герой. Полную реплику можно спрятать за скобками: на кнопке «[Соврать]», а в тексте – то, что герой сказал.

Сравни два начала одного разговора со стражником на мосту. Это куски узлов, остальные узлы истории не показаны.

≡ guard ≡

Стражник: Здравствуйте. Я стражник этого моста. Я охраняю его уже много лет и не пропускаю никого, у кого нет пропуска, потому что таков приказ.

* [Да] → bridge

* [Хорошо] → bridge

≡ guard ≡

– Стоять! – Стражник опускает копьё. – Пропуск есть?

* [Показать пропуск] → bridge

* «Я гонец короля!» → lie

* [Развернуться и уйти] → road

В первом варианте стражник рассказывает о себе, как в анкете, а обе кнопки ведут в одно место. Выбора тут нет, есть только видимость. Во втором одна реплика сразу показывает характер и задаёт вопрос, а три кнопки ведут в три разных узла. Честно, соврать, уйти – у каждого пути свои последствия.

Запомни. Узнаёшь героя без подписи – голос есть. Одна реплика – одна мысль, примерно до 15 слов. Выбор, который ведёт туда же, куда и другой, – не выбор.

7.7. Разбираем задачи

Задача 7.1 Три вопроса о сюжете

разминка · нелинейный сюжет, знаки Ink, значимый выбор

Ответь на три вопроса. В ответ запиши цифры выбранных вариантов по порядку, без пробелов.

а) В какой из игр сюжет ветвится: от выборов игрока зависит, как пойдёт история? 1 – «Тетрис»; 2 – Super Mario Bros.; 3 – Minecraft: Story Mode; 4 – Angry Birds.

б) Сценарист хочет, чтобы кнопка «Спросить про остров» была видна каждый раз, когда игрок возвращается к лодочнику. Каким знаком начать строку с этим выбором в Ink? 1 – *; 2 – +; 3 – →; 4 – ~.

в) Как называют выбор, после которого игрок видит последствия: персонаж это запомнил, история пошла по-другому? 1 – случайный; 2 – значимый; 3 – линейный; 4 – запасной.

Дано: три вопроса, в каждом четыре варианта и ровно один верный.

Найти: три цифры по порядку.

Решение

Шаг 1. Вопрос а). Проверим каждую игру вопросом: «Могут ли два игрока увидеть разные истории?» В «Тетрисе» истории нет совсем – падают фигуры. В Super Mario Bros. история одна: спасти принцессу. Игрок выбирает, как прыгать, но сюжет от этого не меняется. В Angry Birds свиньи украли яйца, птицы мстят – и тоже без развилки. В Minecraft: Story Mode игрок принимает решения в разговорах, и от них зависит, как поведут себя персонажи и что случится дальше. Ответ – 3.

Шаг 2. Вопрос б). Выбор со звёздочкой * исчезает после первого нажатия. Вернётся игрок к лодочнику – кнопки уже нет. Чтобы кнопка оставалась, нужен плюс +. Стрелка → – переход, кнопку она не рисует. Тильда ~ – команда для переменной. Ответ – 2.

Шаг 3. Вопрос в). Выбор с заметными последствиями называют значимым – так его называли и на вебинаре сферы. «Запасной» – это выбор * → узел без текста, который Ink делает сам, когда кнопок не осталось. «Линейный» бывает сюжет, а не выбор. Ответ – 2.

Шаг 4. Собираем ответ. Цифры по порядку: 3, 2, 2.

Заметь: в каждом вопросе мы не просто выбрали ответ, а для каждого из трёх лишних вариантов назвали причину. Не нашлась причина для какого-то варианта – вернись к нему, ошибка сидит там.

 **Типичная ошибка.** В вопросе а) выбрать Super Mario Bros.: там есть тайные трубы, которые перебрасывают через несколько миров сразу. Но трубы только сокращают дорогу по уровням: принцесса и финал остаются теми же. В вопросе б) спутать * и +: звёздочку ставят чаще, и рука тянется к ней по привычке.

Ответ: 322.

Реши сам

7.1. Соедини знак Ink с тем, что он делает. Знаки: 1 – ; 2 – ; 3 – ; 4 – ; 5 – ; 6 –  в начале строки. Что они делают: А – выбор, который остаётся; Б – начало узла; В – изменить значение переменной; Г – перейти в другой узел; Д – место сбора, где ветки снова сходятся; Е – выбор, который исчезает после нажатия. Запиши номера знаков для А, Б, В, Г, Д, Е по порядку, без пробелов.

7.2. История про школьный буфет:

```
→ canteen

=== canteen ===
Школьный буфет. До звонка пять минут.
* Взять компот.
    Компот из сухофруктов, ещё тёплый.
    → canteen
* [Взять булочку]
    Булочка с маком.
    → canteen
+ [Посмотреть меню]
    Сегодня сырники.
    → canteen
+ [Пойти на урок] → END
```

Игрок нажал по очереди: «Взять компот.», «Посмотреть меню», «Взять булочку», «Пойти на урок». а) Сколько кнопок было на экране при каждом из четырёх посещений буфета? Запиши четыре числа подряд. б) Сколько строк текста (не считая кнопок) прочитает игрок за всю игру?

Задача 7.2 Где искать Бублика

граф сюжета, подсчёт прохождений и концовок

Щенок убежал с дачи, и игрок отправляется его искать. Вот история на Ink:

```

→ gate

=== gate ===
Щенок Бублик убежал с дачи.
* [Искать в лесу] → forest
* [Искать у реки] → river
* [Спросить соседку] → neighbor

=== neighbor ===
- Видела, к реке побежал! - говорит тётя Валя.
* [Бежать к реке] → river
* [Не поверить и пойти в лес] → forest

=== forest ===
В лесу темнеет. Где-то впереди лай.
* [Идти на лай] → found_forest
* [Вернуться к реке] → river

=== river ===
На мокром песке следы лап.
* [Идти по следам] → found_river
* [Позвать Бублика] → found_river
* [Сдаться и вернуться домой] → home

=== found_forest ===
Бублик застрял в кустах малины. Ты его вытаскиваешь.
→ END

=== found_river ===
Бублик гоняет уток у берега. Вы вместе идёте домой.
→ END

=== home ===
Бублик уже сидит у крыльца и виляет хвостом.
→ END

```

Прохождение – вся цепочка нажатых кнопок от начала до конца истории. а) Сколько у истории концовок? б) Сколько у неё разных прохождений? Ответ – два числа.

Дано: 7 узлов; начало – gate ; переходы – как в коде.

Найти: а) число концовок; б) число прохождений.

Решение

Шаг 1. Выписываем стрелки. Идём по коду и для каждого узла записываем, куда ведут его кнопки:

Таблица 7.3

Узел	Куда ведут кнопки
gate	forest, river, neighbor
neighbor	river, forest
forest	found_forest, river

Узел	Куда ведут кнопки
river	found_river (две кнопки), home

Шаг 2. Концовки. Узлы с $\rightarrow$ END – это found_forest, found_river и home. В каждую ведёт хотя бы одна стрелка, значит, все три достижимы. Концовок 3.

Шаг 3. Рисуем граф и считаем. У начала пишем 1. Дальше – по правилу из раздела 7.2: у узла сумма чисел тех, кто в него ведёт. Узел считаем, только когда посчитаны все, кто в него ведёт. Поэтому river – после forest: из леса тоже можно вернуться к реке.

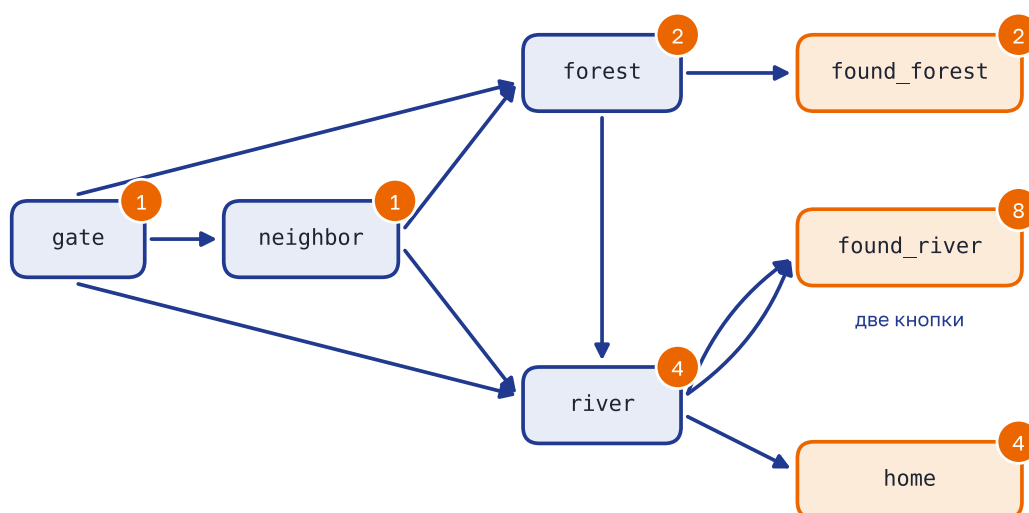


Рисунок 7.11 – Граф истории про Бублика. У узла – сколько прохождений в него ведёт. Из «river» в «found_river» идут две кнопки

Таблица 7.4

Узел	Откуда ведут стрелки	Прохождений
gate	– (начало)	1
neighbor	из gate	1
forest	из gate, из neighbor	$1 + 1 = 2$
river	из gate, neighbor, forest	$1 + 1 + 2 = 4$
found_forest	из forest	2
found_river	из river, две кнопки	$4 + 4 = 8$
home	из river	4

Шаг 4. Складываем концовки. Прохождений всего $2 + 8 + 4 = 14$.

Проверка. Посчитаем по-другому – по первой кнопке. Через «Искать в лесу»: сразу на лай (1) или к реке и там 3 кнопки (3) – всего 4. Через «Искать у реки»: 3. Через «Спросить соседку»: к реке – 3, в лес – ещё 4, всего 7. Итого $4 + 3 + 7 = 14$. Сходится.

Концовок 3, а прохождений 14 – почти в пять раз больше. Откуда столько? Ветки всё время сходятся у реки: дорог много, а мест, где всё кончается, мало.

❗ **Типичная ошибка.** Посчитать две кнопки «Идти по следам» и «Позвать Бублика» за одну стрелку – они ведь ведут в один узел. Тогда у `found_river` будет 4 вместо 8, а всего выйдет 10. Кнопки разные – значит, и прохождения разные. А если посчитать `river` раньше `forest`, потеряется стрелка «Вернуться к реке», и у реки выйдет 2 вместо 4.

Ответ: а) 3 концовки; б) 14 прохождений.

Своя задача по программе сферы, модуль 4

Реши сам

7.3. Класс на экскурсии в зоопарке:

→ entrance

=== entrance ===

Вход в зоопарк. Куда пойдёт класс?

* [К медведям] → bears

* [К жирафам] → giraffes

=== bears ===

Бурый медведь спит на солнце.

* [К тиграм] → tigers

* [В кафе] → cafe

=== giraffes ===

Жираф тянется к ветке.

* [К тиграм] → tigers

* [К пингвинам] → penguins

=== tigers ===

Амурский тигр ходит вдоль решётки.

* [В кафе] → cafe

* [К выходу] → exit

=== penguins ===

Пингвины ныряют по очереди.

* [В кафе] → cafe

* [Идти за пингвинами до конца дорожки] → lost

=== cafe ===

Весь класс ест мороженое.

→ END

=== exit ===

Автобус уже ждёт у ворот.

→ END

=== lost ===

Ты отстал от класса, и учительница звонит тебе на телефон.

→ END

Сколько у истории концовок и сколько разных прохождений? Ответ – два числа.

7.4. Вернись к истории про Бублика из задачи 7.2. а) В скольких прохождениях игрок поговорил с соседкой? б) Сколько кнопок нажимает игрок в самом коротком прохождении и сколько – в самом длинном?

Задача 7.3 Лис у ручья

переменные, выбор с условием, переход по условию

```
VAR trust = 0      // доверие лиса
VAR berries = 2    // ягоды в корзинке

→ meet

=== meet ===
Под ёлкой сидит лис и смотрит на твою корзинку.
* [Дать ягоду]
  ~ trust = trust + 2
  ~ berries = berries - 1
* [Топнуть ногой]
  ~ trust = trust - 1
* [Сделать вид, что не заметил]
- Лис идёт за тобой по тропинке.
→ stream

=== stream ===
Тропинку перегородил ручей.
* {berries = 2} [Бросить лису ягоду]
  ~ trust = trust + 1
  ~ berries = berries - 1
* [Перейти ручей по камням]
* [Позвать лиса с собой]
  ~ trust = trust + 1
- → glade

=== glade ===
{trust ≥ 3: → berry_glade}
{trust ≥ 1: → edge}
→ bushes

=== berry_glade ===
Лис приводит тебя к поляне, красной от брусники.
→ END

=== edge ===
Лис провожает тебя до опушки и убегает.
→ END

=== bushes ===
Лис фыркает и прячется в кустах.
→ END
```

Концовки: 1 – брусничная поляна (berry_glade), 2 – опушка (edge), 3 – кусты (bushes).

- а) Игрок топнул ногой, а у ручья бросил лису ягоду. Какой концовкой закончится история и чему будет равно доверие `trust` ?
- б) Сколько всего разных прохождений у истории и в скольких из них лис приведёт игрока на брусничную поляну?

Дано: в начале `trust = 0`, `berries = 2`; две развилки; три перехода по условию в узле `glade`.

Найти: а) концовку и `trust`; б) число прохождений и число прохождений к поляне.

Решение

Шаг 1. Что происходит. Каждая кнопка меняет доверие: ягода под ёлкой +2, топнуть -1, позвать с собой или бросить ягоду у ручья +1. Кнопка «Бросить лису ягоду» появляется, только если ягод ровно 2, то есть если ягоду ещё не отдавали. В узле `glade` концовку выбирают три строки, сверху вниз.

Шаг 2. Пункт а): следим за переменными.

Таблица 7.5

После	<code>trust</code>	<code>berries</code>
начало	0	2
«Топнуть ногой»	$0 - 1 = -1$	2
«Бросить лису ягоду»	$-1 + 1 = 0$	$2 - 1 = 1$

Кнопка «Бросить» была на экране: под ёлкой ягоду не отдавали, и `berries = 2` было верно. В узле `glade`: $0 \geq 3$ – нет, $0 \geq 1$ – нет. Остаётся последняя строка – кусты. Концовка 3, доверие 0.

Шаг 3. Пункт б): сколько кнопок у ручья. Под ёлкой 3 кнопки. После «Дать ягоду» ягод остаётся 1, и у ручья кнопки «Бросить» не будет – только 2 кнопки. После двух других кнопок ягод по-прежнему 2, у ручья 3 кнопки. Прохождений $2 + 3 + 3 = 8$.

Шаг 4. Доверие в конце каждого прохождения.

Таблица 7.6

Под ёлкой (<code>trust</code>)	Бросить ягоду	По камням	Позвать с собой
дать ягоду (2)	кнопки нет	2 – опушка	3 – поляна
топнуть (-1)	0 – кусты	-1 – кусты	0 – кусты
не заметить (0)	1 – опушка	0 – кусты	1 – опушка

Доверие 3 и больше только в одной клетке: дать ягоду, а потом позвать лиса с собой. На поляну ведёт 1 прохождение из 8.

Проверка. По таблице: поляна – 1, опушка – 3, кусты – 4. Вместе $1 + 3 + 4 = 8$ – столько же, сколько мы насчитали в шаге 3.

Выходит, поляна – редкая награда, и это честно: чтобы до неё дойти, лиса надо и угостить, и позвать с собой. Один раз топнул – доверия уже не хватит, даже если потом извиниться

ягодой.

❗ **Типичная ошибка.** Забыть, что кнопка с условием исчезает, и насчитать $3 \cdot 3 = 9$ проходов. Тогда появится лишнее «дать ягоду – бросить ягоду» с доверием 3, и поляна выйдет в 2 прохождениях. А в пункте а) тянет решить, что брошенная ягода сделала лиса добрым, и выбрать опушку. Но доверие складывается из всех нажатых кнопок, поэтому считать надо с самого начала.

Ответ: а) концовка 3 (кусты), `trust = 0`; б) 8 прохождений, на поляну – 1.

Своя задача по программе сферы, модуль 4

Реши сам

7.5. Робот в кружке робототехники:

```
VAR energy = 3      // заряд робота

→ lab

=== lab ===
Робот Винтик стоит в кружке робототехники. Заряд: {energy}.
* [Отправить его за инструментами]
  ~ energy = energy - 2
* [Попросить протереть окна]
  ~ energy = energy - 1
* [Ничего не просить]
- → corridor

=== corridor ===
В коридоре есть розетка.
* [Подзарядить Винтика]
  ~ energy = energy + 2
* [Не заряжать]
- → hall

=== hall ===
{energy ≥ 3: → dance}
{energy > 0: → slow}
→ sleep

=== dance ===
Винтик танцует на празднике кружка.
→ END

=== slow ===
Винтик медленно едет по залу и мигает лампочкой.
→ END

=== sleep ===
Винтик выключается посреди зала.
→ END
```

Концовки: 1 – танцует, 2 – едет медленно, 3 – выключается. а) Игрок отправил Винтика за инструментами и не стал его заряжать. Какой будет концовка и какой заряд? б) Сколько из всех прохождений заканчиваются танцем?

7.6. В истории про лиса из задачи 7.3 сценарист заменил условие `{berries = 2}` на `{berries > 0}`. Сколько теперь разных прохождений и в скольких из них лис приведёт на брусничную поляну?

Задача 7.4 Ракушки на берегу

недостижимая концовка, наибольшее значение переменной

```

VAR shells = 0      // сколько ракушек у героя

→ beach

=== beach ===
Берег Обского моря. В песке блестят ракушки.
* [Собрать ракушки у воды]
    ~ shells = shells + 3
* [Поискать под камнями]
    ~ shells = shells + 2
- → dune

=== dune ===
На песчаной горке мальчишка строит замок.
* [Помочь ему с башней]
    Он дарит тебе ракушку.
    ~ shells = shells + 1
* [Пройти мимо]
- → pier

=== pier ===
У причала торговец сувенирами.
* {shells ≥ 4} [Обменять ракушки на бусы] → beads
* {shells ≥ 5} [Обменять ракушки на кораблик в бутылке] → ship
* [Пойти домой] → home

=== beads ===
Ты везёшь маме бусы из ракушек.
→ END

=== ship ===
Кораблик в бутылке встанет на твою полку.
→ END

=== home ===
Ракушки гремят в кармане всю дорогу домой.
→ END

=== storm ===
Налетает шторм, и катер возвращается к причалу.
→ END

```

Концовки: 1 – бусы (beads), 2 – кораблик (ship), 3 – домой (home), 4 – шторм (storm). Какие концовки игрок не увидит никогда, как бы ни играл? Запиши их номера по возрастанию, без пробелов.

Дано: 4 концовки; в начале shells = 0 ; ракушки прибавляются на двух развилках; у двух кнопок есть условие.

Найти: номера недостижимых концовок.

Решение

Шаг 1. Ищем стрелки в каждую концовку. Пройдём по коду и найдём все → , которые ведут в КОНЦОВКИ:

Таблица 7.7

Концовка	Откуда ведёт стрелка	Условие
1 beads	из pier	$shells \geq 4$
2 ship	из pier	$shells \geq 5$
3 home	из pier	нет
4 storm	ниоткуда	–

В узел `storm` не ведёт ни одна стрелка. Он написан, но попасть в него нельзя. Концовка 4 недостижима.

Шаг 2. Самое большое число ракушек. Условия зависят от `shells`. На берегу за один раз нажимают одну кнопку: больше всего даёт «Собрать у воды» – 3 ракушки. У горки больше даёт «Помочь с башней» – ещё 1. Больше ракушек взять негде: $3 + 1 = 4$.

Шаг 3. Проверяем условия. Для бус нужно $4 \geq 4$ – верно, если собрать у воды и помочь мальчишке. Концовка 1 достижима. Для кораблика нужно 5 или больше, а больше 4 не бывает никогда. Концовка 2 недостижима. Домой можно уйти всегда – концовка 3 достижима.

Проверим перебором. До причала всего $2 \cdot 2 = 4$ варианта:

Таблица 7.8

На берегу	У горки	<code>shells</code> у причала
у воды (+3)	помочь (+1)	4
у воды (+3)	мимо	3
под камнями (+2)	помочь (+1)	3
под камнями (+2)	мимо	2

Пять не встречается ни разу. Кораблик и правда не достать.

Шаг 4. Ответ. Недостижимы концовки 2 и 4.

Откуда берутся такие концовки? Скорее всего, так: сначала за ракушки у воды давали 4, и кораблик был достижим. Потом число уменьшили до 3 – а про кораблик забыли.

⚠ Типичная ошибка. Сложить обе кнопки на берегу: $3 + 2 + 1 = 6$ – и решить, что кораблик можно получить. Но на развилке нажимают одну кнопку, обе сразу не выбрать. А `storm` проглядеть легко: сам узел написан без единой ошибки, и беду видно, только если поискать, откуда в него идут.

Ответ: 24.

Своя задача по программе сферы, модуль 4

Реши сам

7.7. Ночь на крыше школы:

```
VAR stars = 0      // сколько созвездий и планет нашёл
VAR tea = false    // пил ли чай

→ roof

=== roof ===
Ночь на крыше школы. Учитель настраивает телескоп.
* [Посмотреть на Луну]
  ~ stars = stars + 1
* [Налить чаю из термоса]
  ~ tea = true
- → telescope

=== telescope ===
Учитель уступает тебе телескоп.
* [Найти Большую Медведицу]
  ~ stars = stars + 2
* [Найти Юпитер]
  ~ stars = stars + 1
- → finale

=== finale ===
{stars ≥ 3 and tea: → astronomer}
{stars ≥ 3: → star_map}
{tea: → warm}
→ sleepy

=== astronomer ===
Учитель записывает тебя в кружок астрономии.
→ END

=== star_map ===
Ты рисуешь карту звёздного неба.
→ END

=== warm ===
Ты греешь руки о кружку и смотришь на небо.
→ END

=== sleepy ===
Ты засыпаешь в спальнике прямо на крыше.
→ END

=== clouds ===
Небо затянули тучи, и все спускаются вниз.
→ END
```

Концовки: 1 – `astronomer`, 2 – `star_map`, 3 – `warm`, 4 – `sleepy`, 5 – `clouds`. Какие концовки недостижимы? Запиши номера по возрастанию, без пробелов.

7.8. Ярмарка в Академгородке:

```
VAR coins = 0           // монеты от продажи

→ fair

=== fair ===
Ярмарка в Академгородке. У тебя {coins} монет.
* [Продать банку варенья]
  ~ coins = coins + 3
  → fair
* [Продать корзину огурцов]
  ~ coins = coins + 2
  → fair
+ [Пойти к прилавкам] → shop

=== shop ===
Прилавки с игрушками и сладостями.
* {coins ≥ 6} [Купить самокат] → scooter
* {coins ≥ 5} [Купить мяч] → ball
* {coins = 4} [Купить воздушного змея] → kite
* [Купить мороженое] → icescream

=== scooter ===
Ты едешь домой на новом самокате.
→ END

=== ball ===
Ты играешь в мяч с друзьями.
→ END

=== kite ===
Змей поднимается над Обским морем.
→ END

=== icescream ===
Мороженое тает быстрее, чем ты успеваешь его съесть.
→ END
```

Концовки: 1 – самокат, 2 – мяч, 3 – змей, 4 – мороженое. Какие концовки недостижимы? Запиши номера по возрастанию, без пробелов.

Задача 7.5 Билет в музей

отладка чужого кода, тильда, == и =, звёздочка и плюс

Задумка сценариста: «У героя 5 монет. Смотрителя можно расспрашивать про динозавров сколько угодно раз. Билет стоит 2 монеты, купить его можно один раз. С билетом открывается кнопка «Пройти в зал». В зале герой видит скелет и сколько монет у него осталось». Начинающий сценарист написал (номера строк – не часть кода):

```

1  VAR coins = 5
2  VAR ticket = false
3  → hall
4  ≡≡ hall ≡≡
5  Вход в музей. Смотритель сидит у кассы.
6  * [Спросить про динозавров]
7    - Скелет в зале настоящий!
8    → hall
9  * {coins ≥ 2} [Купить билет]
10    coins = coins - 2
11    ~ ticket = true
12    → hall
13  * {ticket = true} [Пройти в зал] → dino_hall
14  ≡≡ dino_hall ≡≡
15  Над тобой висит скелет динозавра. Монет осталось: {coins}.
16  → END

```

а) Найди все строки, из-за которых история работает не по задумке. Запиши их номера по возрастанию через запятую.

б) Строки исправили. Игрок купил билет и прошёл в зал. Какое число он увидит в последней фразе?

Дано: задумка; код из 16 строк.

Найти: а) номера строк с ошибками; б) число монет в конце после исправления.

Решение

Шаг 1. Сверяем код с задумкой по пунктам. Задумка – это техническое задание. Каждое её предложение ищем в коде.

- «Расспрашивать сколько угодно раз» – строка 6. У кнопки звёздочка `*`, а звёздочка исчезает после первого нажатия. Спросил один раз – кнопки больше нет. Нужен плюс: `+ [Спросить про динозавров]`. **Ошибка в строке 6.**
- «Билет стоит 2 монеты» – строка 10. В начале нет тильды `~`. Inky не поймёт, что это команда, и покажет на экране строку «coins = coins - 2», а монет останется 5. Нужно `~ coins = coins - 2`. **Ошибка в строке 10.**
- «Купить можно один раз» – строка 9: звёздочка, условие `coins ≥ 2`. Всё верно.
- «С билетом открывается кнопка» – строка 13. В условии один знак `=`, а сравнивают двумя. Inky не запустит историю: в строке 13 он найдёт ошибку. Нужно `{ticket = true}` или короче `{ticket}`. **Ошибка в строке 13.**
- Строки 11, 12, 15, 16 делают то, что сказано в задумке.

Шаг 2. Пункт б). После исправления: было 5 монет, билет стоит 2, $5 - 2 = 3$. В зале игрок прочитает «Монет осталось: 3».

Проверили в Inky. Мы запустили исправленную историю: расспросили смотрителя дважды, купили билет, прошли в зал – на экране «Монет осталось: 3». А с ошибкой в строке 10 было бы «Монет осталось: 5».

Заметь, как по-разному прячутся эти три ошибки. Строку 13 Inky покажет сам. Строку 10 заметишь по странному тексту на экране. А строку 6 найдёт только плейтест, где игрок захочет

спросить второй раз. Так что мало дожидаться, пока Inky перестанет ругаться: код сверяют с задумкой по каждому предложению.

! Типичная ошибка. Найти только ошибку в строке 13 – про неё Inky «кричит» – и остановиться. Остальные две Inky не считает ошибками: для него это законный текст и законная кнопка со звёздочкой. Ещё одна ловушка – «исправить» строку 9 на плюс. Тогда билет можно покупать снова и снова, а по задумке – один раз.

Ответ: а) 6, 10, 13; б) 3.

Своя задача по программе сферы, модуль 4

Реши сам

7.9. Задумка: «Фонарик в избушке можно взять один раз. В окно можно смотреть сколько угодно. С фонариком открывается дорога в лес, и там история заканчивается». Код (номера строк – не часть кода):

```
1  VAR torch = false
2  → hut
3  ≡ hut ≡
4  Избушка лесника. На столе лежит фонарик.
5  + [Взять фонарик]
6    ~ torch = true
7    → hut
8  + [Посмотреть в окно]
9    За окном метёт метель.
10   → hut
11  * {torch} [Пойти в лес] → forrest
12  ≡ forest ≡
13  Фонарик освещает тропинку к дому.
```

В коде две неправильные строки, а в одном узле не хватает строки. Запиши номера неправильных строк по возрастанию и имя узла, где не хватает строки.

7.10. Строчка с ошибкой:

```
VAR apples = 0
→ garden
=== garden ===
Сад на даче. Яблоки уже спели.
* [Сорвать яблоки с ветки]
  ~ apples = apples + 5
* [Собрать яблоки с земли]
  apples = apples + 3
- → basket
=== basket ===
Сосед заглядывает через забор.
* [Отдать два яблока соседу]
  ~ apples = apples - 2
* [Оставить всё себе]
- В корзинке {apples} яблок.
→ END
```

Игрок собрал яблоки с земли и отдал два соседу. Какое число он увидит в последней фразе?

Задача 7.6 Три голоса

голос персонажа, длина реплики

В игре три персонажа: 1 – робот-экскурсовод в музее, 2 – бабушка на даче, 3 – капитан прогулочного катера. Сценарист перемешал их реплики:

1. «Внимание, уважаемый посетитель: до закрытия нашего музея осталось ровно двенадцать минут, поэтому я настоятельно рекомендую вам пройти в зал номер три, где выставлены первые роботы.»
2. «Ой, руки-то ледяные! Садись к печке, я блинов напекла.»
3. «Полный вперёд! Держись за поручень – волна идёт.»
4. «Ваш вопрос не распознан. Повторите его, пожалуйста, медленнее.»
5. «Туман – не беда. Я по этой воде тридцать лет хожу.»
6. «Кабачков в этом году – хоть соседям раздавай. Возьми пару для мамы.»

а) Кому принадлежит каждая реплика? Запиши номера персонажей для реплик 1–6 по порядку, без пробелов.

б) Одна реплика слишком длинная для экрана – в ней больше 15 слов. Какая, и сколько в ней слов? Тире словом не считаем.

Дано: 3 персонажа, 6 реплик; правило «не больше 15 слов на экран».

Найти: а) персонаж для каждой реплики; б) номер длинной реплики и число слов в ней.

Решение

Шаг 1. Приметы голосов. Прежде чем раскладывать реплики, выпишем, как говорит каждый:

- **робот** – вежливо и сухо, с точными числами, говорит «вы», слова из инструкций («рекомендую», «не распознан»);

- **бабушка** – ласково, на «ты», про еду, тепло, огород;
- **капитан** – коротко, командами, про воду и волны, гордится опытом.

Шаг 2. Раскладываем. Реплика 1: «уважаемый посетитель», «ровно двенадцать минут», «рекомендую» – робот (1). Реплика 2: «руки-то ледяные», печка, блины – бабушка (2). Реплика 3: «Полный вперед!», поручень, волна – капитан (3). Реплика 4: «вопрос не распознан» – так говорит машина, робот (1). Реплика 5: вода, «тридцать лет хожу» – капитан (3). Реплика 6: кабачки, «возьми для мамы» – бабушка (2).

Шаг 3. Считаем слова. Короткие реплики видно сразу, длинная – первая. Считаем по одному: «Внимание (1), уважаемый (2), посетитель (3): до (4) закрытия (5) нашего (6) музея (7) осталось (8) ровно (9) двенадцать (10) минут (11), поэтому (12) я (13) настоятельно (14) рекомендую (15) вам (16) пройти (17) в (18) зал (19) номер (20) три (21), где (22) выставлены (23) первые (24) роботы (25)». 25 слов – больше 15. В остальных репликах от 7 до 11 слов.

Шаг 4. Как исправить. Разбить на две короткие реплики: «До закрытия музея – 12 минут». «Рекомендую зал номер 3: там первые роботы». Голос робота остался, а читать стало легко.

Каждому герою досталось по две реплики, и ни одну нельзя отдать двоим сразу. Значит, голоса у персонажей получились.

 **Типичная ошибка.** Отдать реплику 4 бабушке: «Повторите, пожалуйста» звучит вежливо. Но бабушка говорит внуку «ты», а «вопрос не распознан» скажет только машина. При подсчёте слов часто теряют короткие «я», «в», «до» – а они тоже слова.

Ответ: а) 123132; б) реплика 1, 25 слов.

Своя задача по программе сферы, модуль 4

Реши сам

7.11. Персонажи: 1 – белка-болтушка из Академгородка, 2 – строгая библиотекарь, 3 – первоклассник Петя. Реплики: 1) «Тише, пожалуйста. Книги возвращают до пятницы, будьте добры.» 2) «А можно я тоже с вами? Я уже умею читать!» 3) «Орешки! Орешки! У тебя в кармане точно есть орешки, я чувствую!» 4) «Шапку забыл! Мама будет ругаться...» 5) «Читальный зал закрывается через десять минут.» 6) «Бегу-бегу, на сосне шишки поспели!» Запиши номера персонажей для реплик 1–6 по порядку, без пробелов.

7.12. История про зимний двор. Развилки пронумерованы: 1 – горка (hill), 2 – шапка (hat), 3 – снеговик (snowman).

```
VAR helped = false      // помог ли малышу
VAR red_hat = false      // надел ли красную шапку
VAR carrot = false       // отдал ли морковку

→ hill

=== hill ===
Снежная горка во дворе. Малыш никак не может залезть наверх.
* [Помочь малышу]
  ~ helped = true
* [Скатиться самому]
  - → hat

=== hat ===
Мама зовёт: «Шапку поменяй, эта промокла!»
* [Надеть красную]
  ~ red_hat = true
* [Надеть зелёную]
  - → snowman

=== snowman ===
Ребята лепят снеговика.
* [Отдать ему свою морковку]
  ~ carrot = true
* [Выложить ему улыбку из камешков]
  - → evening

=== evening ===
{helped: Мама малыша угощает тебя пирогом. | Малыш так и сидит внизу горки.}
{carrot: У снеговика морковка – он лучший во дворе. | Снеговик улыбается каменной улыбкой.}
→ END
```

а) На каких развилках выбор значимый – меняет то, что игрок увидит потом? Запиши номера развилок по возрастанию, без пробелов. б) Прохождений у истории 8. Сколько среди них разных вечерних сцен (две последние строки)?

Задача 7.7* Половинки ключа

узел-перекрёсток, выборы со звёздочкой, число посещений, перебор

Игрок ищет на даче две половинки ключа от сарая.

→ hall

=== hall ===

Прихожая старой дачи. Где-то спрятан ключ от сарая.

* [Осмотреть кухню] → kitchen

* [Подняться на чердак] → attic

* [Заглянуть в чулан] → closet

* {kitchen and attic} [Открыть сарай] → shed

+ [Уехать в город] → city

=== kitchen ===

На кухне в банке с гречкой лежит половинка ключа.

→ hall

=== attic ===

На чердаке, в коробке с ёлочными игрушками, – вторая половинка.

→ hall

=== closet ===

В чулане только старые лыжи.

→ hall

=== shed ===

Ты складываешь половинки ключа и открываешь сарай. Там старый велосипед деда.

→ END

=== city ===

Ты садишься в электричку и едешь в город.

→ END

Прохождение – вся цепочка нажатых кнопок. а) Сколько у истории разных прохождений? б) В скольких из них игрок откроет сарай?

Дано: узел-перекрёсток hall : три комнаты со *, сарай со * и условием, город с + .

Найти: а) число прохождений; б) число прохождений, которые кончаются сараем.

Решение

Шаг 1. Что происходит. Из прихожей игрок уходит в комнату и сразу возвращается. У комнат звёздочки: в каждую можно зайти только один раз. Кнопка «Открыть сарай» появляется, когда игрок побывал и на кухне, и на чердаке – об этом условие {kitchen and attic}. «Уехать в город» с плюсом видна всегда. История кончается в сарае или в городе.

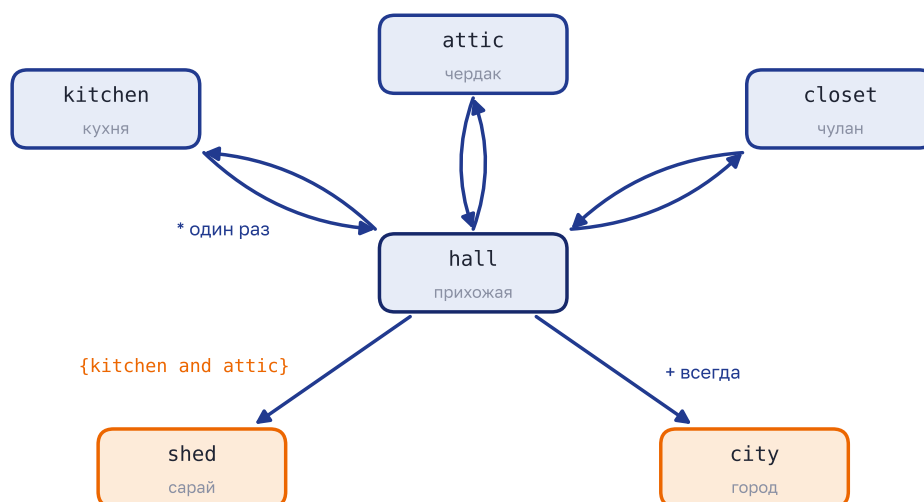


Рисунок 7.12 – Прихожая – узел-перекрёсток. Из комнат игрок возвращается обратно, закончить можно только в сарае или в городе

Шаг 2. Почему сумма по стрелкам не работает. Здесь стрелки ведут обратно в прихожую, и правило «у узла – сумма входящих» заикнется. Посмотрим иначе. Любое прохождение – это **порядок осматриваемых комнат**, а в конце – сарай или город. Например: кухня, чулан, город. Или: чердак, кухня, сарай.

Шаг 3. Считаем порядки комнат. Комнат три, каждая – не больше одного раза.

- Ни одной комнаты: 1 способ (сразу уехал).
- Одна комната: 3 способа.
- Две комнаты по порядку: первую выбираем из 3, вторую – из 2 оставшихся, $3 \cdot 2 = 6$.
- Все три по порядку: $3 \cdot 2 \cdot 1 = 6$.

Всего порядков $1 + 3 + 6 + 6 = 16$.

Шаг 4. Концовка «город». Уехать можно после любого порядка – кнопка с плюсом есть всегда. Прохождений с городом 16.

Шаг 5. Концовка «сарай». Сарай открывается, если среди осматриваемых есть и кухня, и чердак. Две комнаты: только «кухня, чердак» и «чердак, кухня» – 2 порядка. Три комнаты: во всех 6 порядках есть обе – 6. Одна комната или ни одной – сарай закрыт. Прохождений с сараем $2 + 6 = 8$.

Шаг 6. Всего. $16 + 8 = 24$ прохождения.

Проверка. Прохождения с сараем можно пересчитать и иначе. В них обязательно есть кухня и чердак, а чулан – есть или нет. Без чулана порядков 2: «кухня, чердак» и «чердак, кухня». С чуланом – его можно поставить в начало, в середину или в конец каждого из этих двух порядков: $2 \cdot 3 = 6$. Вместе $2 + 6 = 8$ – сходится.

Для любопытных – проверка на Python. Если ты уже пробовал программировать, пусть компьютер сам переберёт все порядки:

```

from itertools import permutations
rooms = ["кухня", "чердак", "чулан"]
city = shed = 0
for k in range(4):                                # сколько комнат осмотрел: 0, 1, 2 или 3
    for order in permutations(rooms, k):
        city += 1                                  # из прихожей всегда можно уехать
        if "кухня" in order and "чердак" in order:
            shed += 1                               # обе половинки найдены – сарай открыт
print(city, shed, city + shed)                     # 16 8 24

```

`permutations(rooms, k)` перебирает все порядки из `k` комнат. Программа печатает `16 8 24`. Саму историю мы тоже прошли всеми способами программой на движке Ink – вышло 24 прохождение, из них 8 кончатся сараем.

Сарай – награда для того, кто обыскал дачу, поэтому прохождений с ним меньше: 8 из 24, то есть треть. А уехать можно в любой момент, даже не разувшись.

❗ Типичная ошибка. Считать только прохождения, где осмотрены все три комнаты: $3 \cdot 2 \cdot 1 = 6$ порядков. Но уехать или открыть сарай можно и раньше. Бывает, кажется, что кнопка «Открыть сарай» появляется после любых двух комнат. Нет: условию нужны кухня и чердак, чулан не помогает. И не забудь: после сарая история кончается, в город уже не уехать.

Ответ: а) 24; б) 8.

Своя задача по программе сферы, модуль 4

Реши сам

7.13.* В истории из задачи 7.7 сценарист решил, что хватит одной половинки ключа, и заменил условие `{kitchen and attic}` на `{kitchen or attic}`. Сколько теперь разных прохождений и в скольких из них игрок откроет сарай?

7.14.* Другая дача:

```
→ hall

=== hall ===
Прихожая старой дачи.
* [Осмотреть кухню] → kitchen
* [Подняться на чердак] → attic
* [Заглянуть в чулан] → closet
* {closet} [Открыть сундук из чулана] → chest
* → night

=== kitchen ===
На кухне пахнет яблоками.
→ hall

=== attic ===
На чердаке пыльно и темно.
→ hall

=== closet ===
В чулане стоит тяжёлый сундук.
→ hall

=== chest ===
В сундуке - дедушкины письма и компас.
→ END

=== night ===
Стемнело, и ты ложишься спать.
→ END
```

Строка `* → night` – запасной выбор: Ink делает его сам, когда других кнопок не осталось. а) Сколько у истории разных прохождений? б) В скольких из них игрок увидит концовку «Стемнело, и ты ложишься спать»?

3D и визуал: Blender



Сундук, ёлка, робот-помощник в игре – всё это 3D-модели, а собраны они из самых простых кусочков: точек, отрезков и плоских граней. Здесь мы научимся их считать, посмотрим, как Blender сам достраивает вторую половину модели и размножает забор из одной доски, как не сделать стул высотой с девятиэтажку и чем «1920 × 1080» отличается от «PNG». На отборе такие задачи короткие: «сколько вершин получится», «какой масштаб поставить», «сколько процентов экрана занято», «найди лишнее в ряду». Ответ – число или строка цифр.

8.1. Из чего сделана 3D-модель: вершины, рёбра, грани

НА ПАЛЬЦАХ

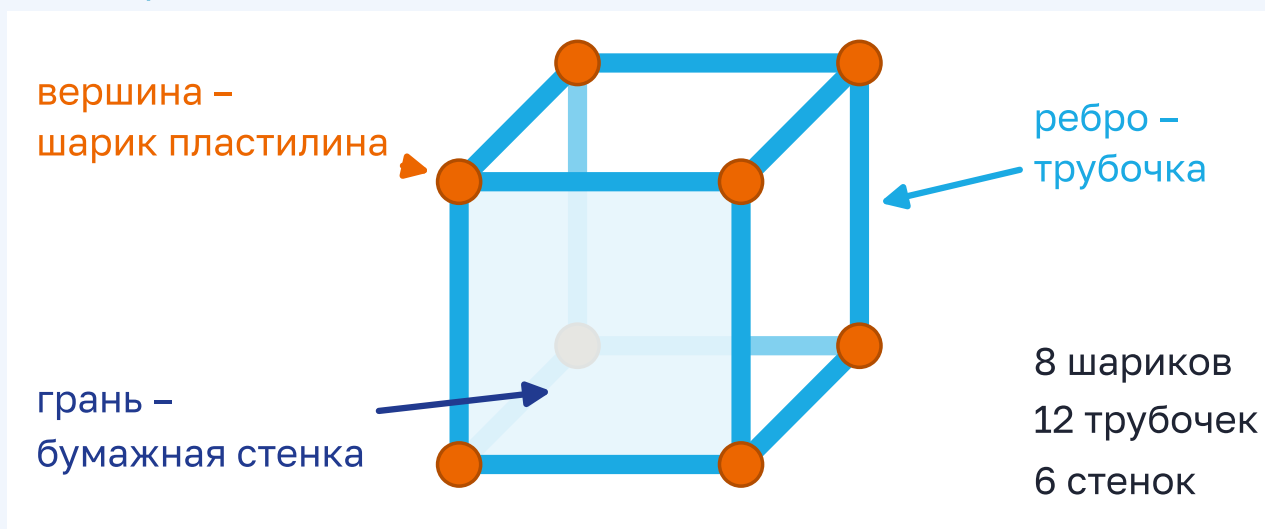


Рисунок 8.1 – Каркас куба из трубочек для коктейля и пластилиновых шариков

Шарики – вершины, трубочки – рёбра, бумажные стенки – грани. Из таких же кусочков сделана любая модель в игре.

Возьми восемь шариков пластилина и двенадцать трубочек для коктейля и собери из них каркас куба. Заклей каждую сторону бумагой – получится коробка. Компьютер строит 3D-модели точно так же, только вместо пластилина и трубочек у него числа.

- **Вершина** – точка в пространстве. Компьютер хранит для неё три числа – координаты x , y , z .
- **Ребро** – отрезок, который соединяет две вершины.
- **Грань** – плоский кусок поверхности, натянутый на рёбра, как бумага на каркас. Грань ещё называют **полигоном**, поэтому такая модель называется **полигональной**.

Все вершины, рёбра и грани модели вместе называют **сеткой** (по-английски mesh – «меш»). Если граней в сетке мало и модель выглядит угловатой, её называют **low-poly** (читается «лоу-поли», «мало полигонов»). На вебинаре сферы обещали: в Blender на курсе будут делать как раз такие модели, и они под силу любому школьнику. К тому же они лёгкие – игра в браузере с ними не тормозит.

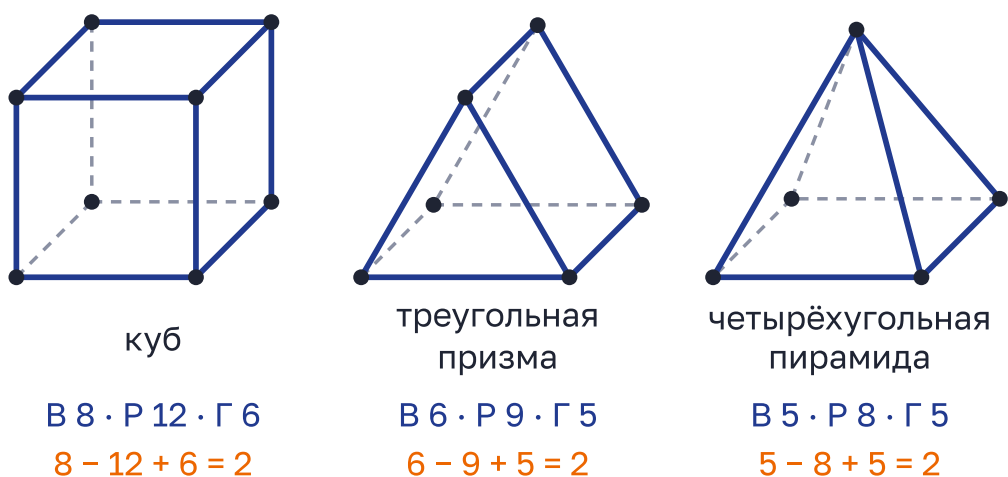


Рисунок 8.2 – Три простых тела. В – вершины, Р – рёбра, Г – грани. Пунктир – рёбра, которые спрятаны сзади

Как считать, чтобы не сбиться. Считай не всё подряд, а по частям: сначала верх, потом низ, потом бока. Возьмём **призму** – тело, у которого сверху и снизу одинаковые многоугольники, а бока – прямоугольники. Пусть в основании n -угольник (у шестигранного карандаша $n = 6$).

- Вершины: n сверху и n снизу, всего $2n$.
- Рёбра: n по краю верха, n по краю низа и n вертикальных, всего $3n$.
- Грани: верх, низ и n боковых, всего $n + 2$.

У **пирамиды** внизу n -угольник, а все бока сходятся в одной верхней точке. Вершин $n + 1$ (все углы основания и верхушка), рёбер $2n$ (n в основании и n к верхушке), граней $n + 1$ (дно и n треугольников).

Проверка Эйлера. Больше двухсот пятидесяти лет назад математик Леонард Эйлер заметил, что для любого тела без дырок работает одно и то же равенство. Словами: вершины минус рёбра плюс грани – всегда два.

$$В - Р + Г = 2.$$

В таблице ниже это проверено для всех наших тел, а на рисунке выше – с числами. Вышло 1 или 3 – значит, где-то ребро пропущено или посчитано дважды. Считай заново. А если известны два числа из трёх, третье можно найти без подсчёта: $Р = В + Г - 2$.

Таблица 8.1

Тело	Вершины	Рёбра	Грани	Проверка Эйлера
Куб	8	12	6	$8 - 12 + 6 = 2$
Треугольная призма	6	9	5	$6 - 9 + 5 = 2$
Призма, n углов в основании	$2n$	$3n$	$n + 2$	$2n - 3n + n + 2 = 2$
Четырёхугольная пирамида	5	8	5	$5 - 8 + 5 = 2$
Пирамида, n углов в основании	$n + 1$	$2n$	$n + 1$	$n + 1 - 2n + n + 1 = 2$

Треугольники. Видеокарта рисует модели только из треугольников. Четырёхугольную грань она режет диагональю на 2 треугольника, пятиугольную – на 3, а грань с n углами – на $n - 2$ треугольника. Поэтому куб для видеокарты – это $6 \cdot 2 = 12$ треугольников. Когда говорят, что модель «лёгкая» или «тяжёлая», обычно считают треугольники.

разрезаем диагоналями из одного угла



Рисунок 8.3 – Грань режут на треугольники диагоналями из одного угла. Треугольников всегда на 2 меньше, чем углов

Запомни. Вершина – точка, ребро – отрезок, грань (полигон) – плоский кусок. Призма: $2n, 3n, n + 2$. Пирамида: $n + 1, 2n, n + 1$. Проверка: $V - P + \Gamma = 2$. Грань с n углами – это $n - 2$ треугольника.

Совет. Blender считает сам. Открой меню наложений над окном 3D-вида (Overlays) и поставь галочку Statistics – в углу появятся числа вершин, рёбер, граней и треугольников. Так удобно проверять свои подсчёты на настоящей модели.

Для любопытных. У бублика (в Blender есть такой примитив – Torus) выйдет $V - P + \Gamma = 0$: у него есть дырка, и правило Эйлера для него звучит по-другому.

8.2. Прimitives: с чего начинается модель

НА ПАЛЬЦАХ

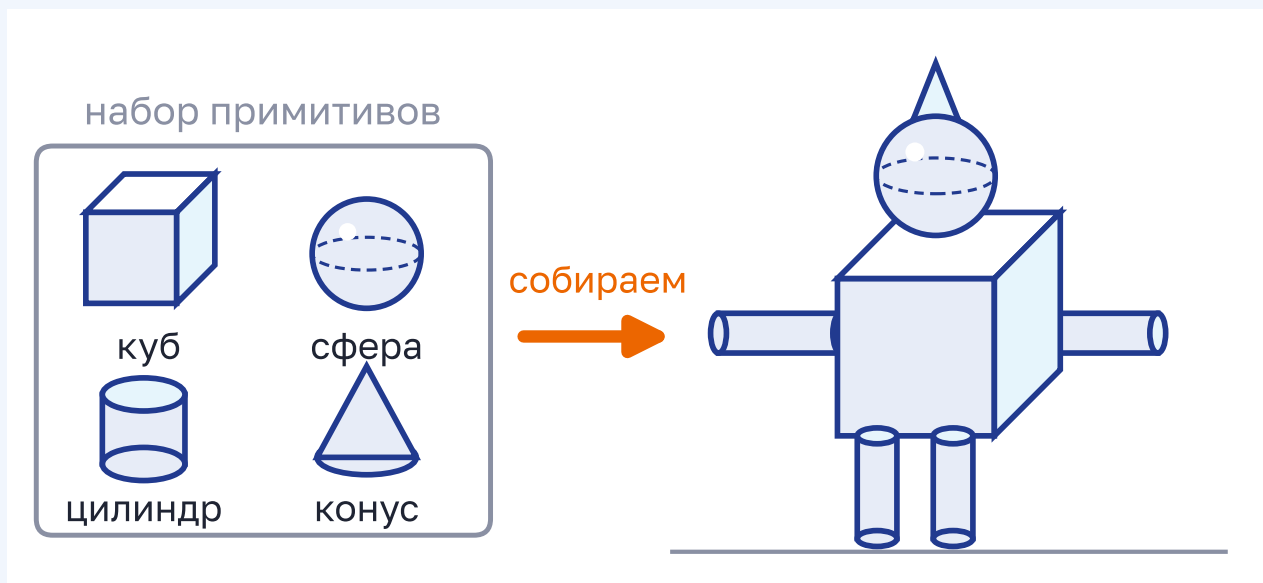


Рисунок 8.4 – Слева – готовые детали Blender, справа – робот, собранный из них

Сначала собираем фигуру из готовых деталей, как из конструктора. Мелочи – потом.

Примитив – готовая простая фигура, которую Blender создаёт одной командой: куб, сфера, цилиндр, конус, плоскость, бублик. Почти любая модель в игре начинается с примитивов. Робот на картинке – куб-корпус, шар-голова, цилиндры-руки и цилиндры-ноги, конус-антенна.

На вебинаре сферы этот первый шаг называли **блокингом** – черновой моделью из простых форм без мелких деталей. На нём проверяют пропорции (голова не слишком ли большая?) и силуэт. Фаски, дверцы и болтики добавляют потом.

Несколько команд, с которых начинается работа (они же – в шпаргалке в конце книги). Клавиши срабатывают, когда курсор мыши стоит над окном 3D-вида.

- **Shift + A** – добавить объект. В открывшемся меню выбирают Mesh, а в нём Cube (куб), Cylinder (цилиндр) или UV Sphere (сфера).
- **G, R, S** – переместить, повернуть, масштабировать. После буквы можно нажать X, Y или Z – тогда объект движется только вдоль этой оси. Например, R, Y, 90 и Enter поворачивают объект на 90° вокруг оси Y.
- **Tab** – переключиться между **режимом объекта** (двигаем модель целиком) и **режимом редактирования** (двигаем отдельные вершины, рёбра и грани).
- **1, 3, 7 на цифровом блоке** – вид спереди, сбоку и сверху.

У каждого примитива есть настройки, которые видно сразу после его создания – в маленьком окошке внизу слева. Главная для нас – число вершин по кругу у цилиндра, конуса и сферы. У стандартного цилиндра их 32, значит, всего у него $2 \cdot 32 = 64$ вершины. Для low-poly это много, и на вебинаре число вершин цилиндра уменьшали до 12. Стандартная UV-сфера ещё «тяжелее»: у неё 482 вершины.

Размеры примитивов по умолчанию лучше выучить. **Куб – 2 м** в каждую сторону (вершины стоят на расстоянии 1 м от центра). **Цилиндр – радиус 1 м и высота 2 м**. Это пригодится в разделе 8.5.

Запомни. Модель начинают с примитивов и собирают черновик (блокинг), детали – потом. Shift + A – добавить, G, R, S – двигать, вращать, масштабировать, Tab – режим редактирования. Куб по умолчанию – 2 м, у цилиндра 32 вершины по кругу.

Совет. Сразу давай объектам понятные имена (на вебинаре куб-корпус назвали RobotBase) и раскладывай их по коллекциям – это как папки в списке объектов (окно Outliner справа вверху). Через час работы в сцене будет тридцать цилиндров, и без имён ты их не различишь.

8.3. Зеркало: модификатор Mirror

НА ПАЛЬЦАХ

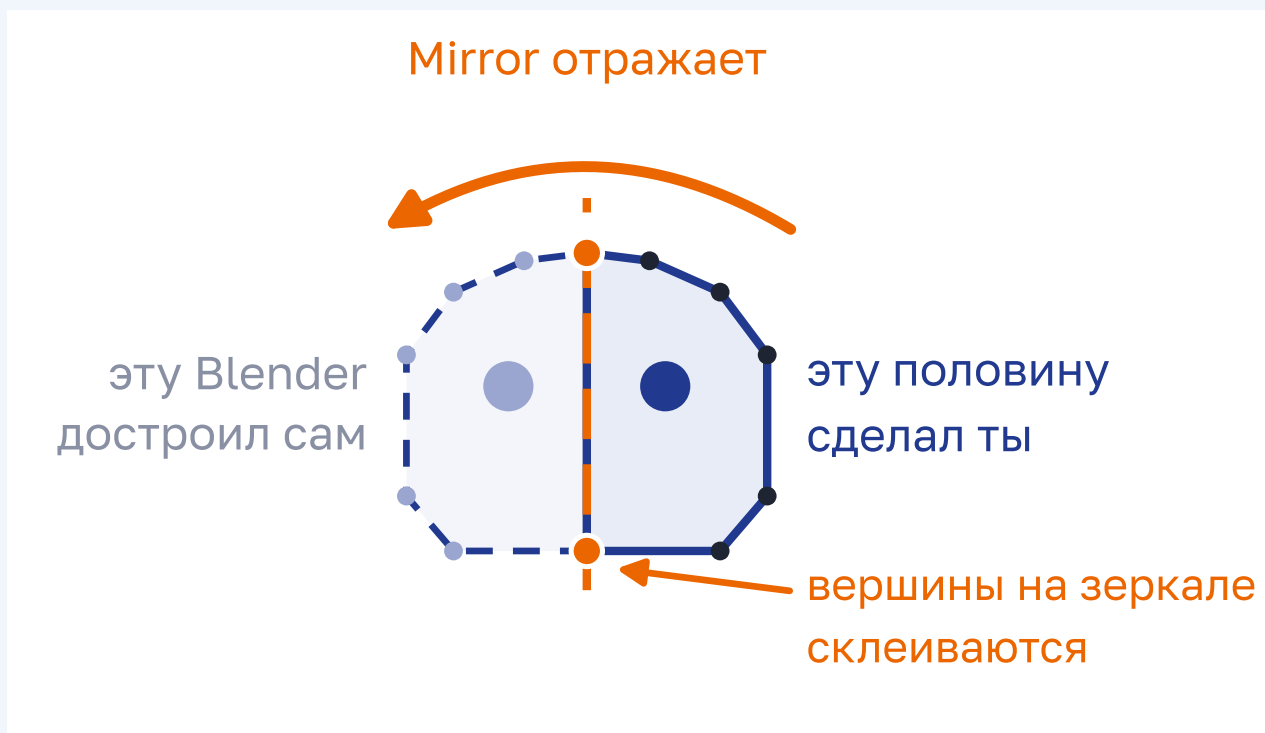


Рисунок 8.5 – Mirror отражает сделанную половину и склеивает её со второй по средней линии

Как клякса на сложенном листе: рисуешь половину бабочки, складываешь бумагу – бабочка целая.

У большинства персонажей, машин и сундуков левая половина – зеркальная копия правой. Лепить обе половины вручную долго, и они всё равно выйдут чуть разными. Поэтому делают одну половину, а вторую достраивает **модификатор**.

Модификатор – это правило, которое Blender применяет к модели «на лету». Исходная сетка остаётся нетронутой, а на экране ты видишь результат правила. Модификатор можно выключить.

чить, поменять его настройки или удалить – и модель вернётся к исходной. Модификаторы добавляют на вкладке с гаечным ключом в окне свойств справа: кнопка Add Modifier, в группе Generate – нужный модификатор.

Mirror (зеркало) отражает модель по оси – по умолчанию по X, то есть слева направо. Ты двигаешь вершину правого уха – левое ухо двигается само. В настройках Mirror есть галочка **Merge** (объединить), по умолчанию она включена. Она склеивает вершины, которые лежат ровно на зеркале, со своими отражениями. Без неё посередине модели остаётся шов из вершин, стоящих друг на друге.

Отсюда правило подсчёта. Пусть в половине V вершин, и k из них лежат на зеркале. Каждая вершина не на зеркале получает двойника, а вершины на зеркале склеиваются сами с собой:

$$\text{вершин в целой модели} = 2V - k.$$

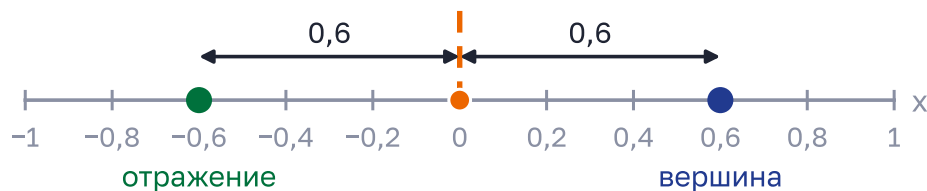
Пример. У половины маски на картинке 7 вершин, 2 из них на зеркале (оранжевые). Целая маска: $2 \cdot 7 - 2 = 12$ вершин. Без Merge было бы $2 \cdot 7 = 14$ – по две вершины в одной точке сверху и снизу.

Где стоит зеркало. Mirror отражает не относительно середины мира, а относительно **центра объекта** – маленькой оранжевой точки, которую Blender рисует у выделенного объекта (по-английски origin). Если центр стоит в $x = 0$, вершина из $x = 0,6$ уходит в $x = -0,6$: координата просто меняет знак. Если центр сдвинут, считаем расстояние до центра и откладываем его в другую сторону. Формулой: отражение точки x относительно центра x_0 –

$$x' = 2x_0 - x.$$

Координаты y и z при отражении по X не меняются.

центр объекта в 0: $0,6 \rightarrow -0,6$



центр объекта в 0,2: $0,6 \rightarrow -0,2$

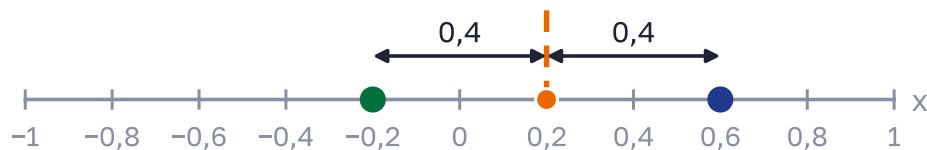


Рисунок 8.6 – Одна и та же вершина в 0,6 при разных центрах объекта. Центр в 0,2 – и отражение попадает в -0,2: второе ухо уехало внутрь головы

Пример. Центр объекта случайно сдвинули в $x = 0,2$. Вершина уха стоит в $x = 0,6$. До центра $0,6 - 0,2 = 0,4$, значит, отражение – на 0,4 левее центра: $0,2 - 0,4 = -0,2$. По формуле: $2 \cdot 0,2 - 0,6 = -0,2$. Два уха стоят несимметрично – это частый баг новичков.

Запомни. Mirror – делаешь половину, вторую строит Blender. Вершины: $2V - k$, где k – вершины на зеркале (при включённом Merge). Зеркало проходит через центр объекта: $x' = 2x_0 - x$, при центре в нуле просто $x' = -x$.

Совет. Включи в Mirror ещё и галочку Clipping – тогда вершины посередине не смогут случайно заехать за зеркало, и в модели не появится дырка или складка по средней линии.

8.4. Массив: модификатор Array

НА ПАЛЬЦАХ

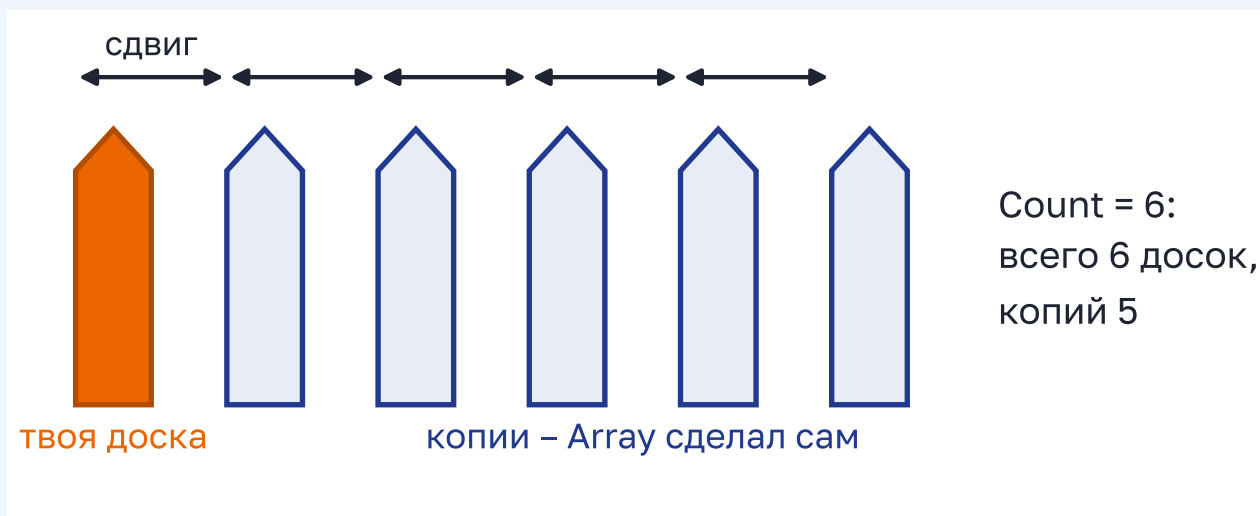


Рисунок 8.7 – Одна доска и модификатор Array – получается забор

Как печать-штамп: вырезал одну доску, а дальше Array ставит её оттиски в ряд с одинаковым шагом.

Array (массив) повторяет модель несколько раз с одинаковым сдвигом: доски забора, ступеньки, шпалы, колонны, звенья цепи. Из его настроек нам хватит двух.

- **Count** – сколько штук всего, **вместе с оригиналом**. Count = 6 – это твоя доска и 5 копий.
- **Relative Offset** (относительный сдвиг) – на сколько ширин модели сдвинута каждая следующая копия. Сдвиг 1 – копии стоят вплотную, 1,5 – между ними зазор в половину ширины, 2 – зазор в целую ширину. По умолчанию сдвиг 1 по оси X. Есть и Constant Offset – сдвиг, заданный прямо в метрах.

Если ширина детали a , а относительный сдвиг k , то **шаг** – расстояние от начала одной копии до начала следующей – равен $s = k \cdot a$. Длина всего ряда:

$$L = a + (n - 1) \cdot s.$$

Словами: первая деталь целиком и ещё $n - 1$ шагов. Почему не $n \cdot s$? Шагов между n досками на один меньше, чем досок: между двумя досками один шаг, между тремя – два. А последняя доска добавляет только свою ширину, а не целый шаг с зазором.

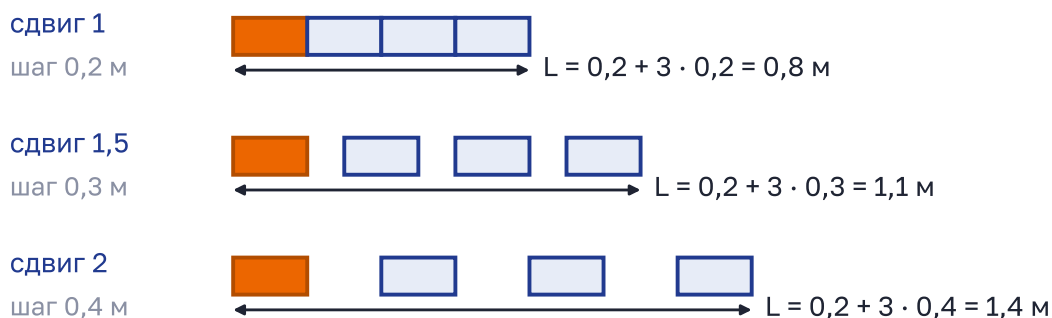


Рисунок 8.8 – Четыре детали шириной 0,2 м при разных сдвигах. Длину ряда меряем от левого края первой до правого края последней

Вершины и треугольники множатся на Count: у доски-бруска 8 вершин, у массива из 30 досок – $30 \cdot 8 = 240$. Если в Array включить Merge, вершины соседних копий, которые оказались в одной точке, склеятся – как в Mirror. Так делают сплошную трубу или рельс из одинаковых кусков.

Сколько получилось объектов? Один. Все копии Array и отражения Mirror – части того же самого объекта, и в списке объектов (Outliner) по-прежнему одна строчка. Двигаешь объект – двигается весь забор.

Несколько модификаторов подряд. Модификаторы стоят в списке – в стопке – и работают сверху вниз: каждый следующий берёт то, что получилось у предыдущего. Array по X на 6 штук, потом Array по Y на 4 ряда, потом Mirror – это $6 \cdot 4 \cdot 2 = 48$ штук вместе с оригиналом. Числа перемножаются.

Запомни. Count – всего штук, с оригиналом. Шаг $s = k \cdot a$, длина ряда

$L = a + (n - 1) \cdot s$. Вершины умножаются на Count. Модификаторы в стопке работают сверху вниз, копии перемножаются. Объект после модификаторов – всё равно один.

8.5. Масштаб, единицы и экспорт в игру

НА ПАЛЬЦАХ

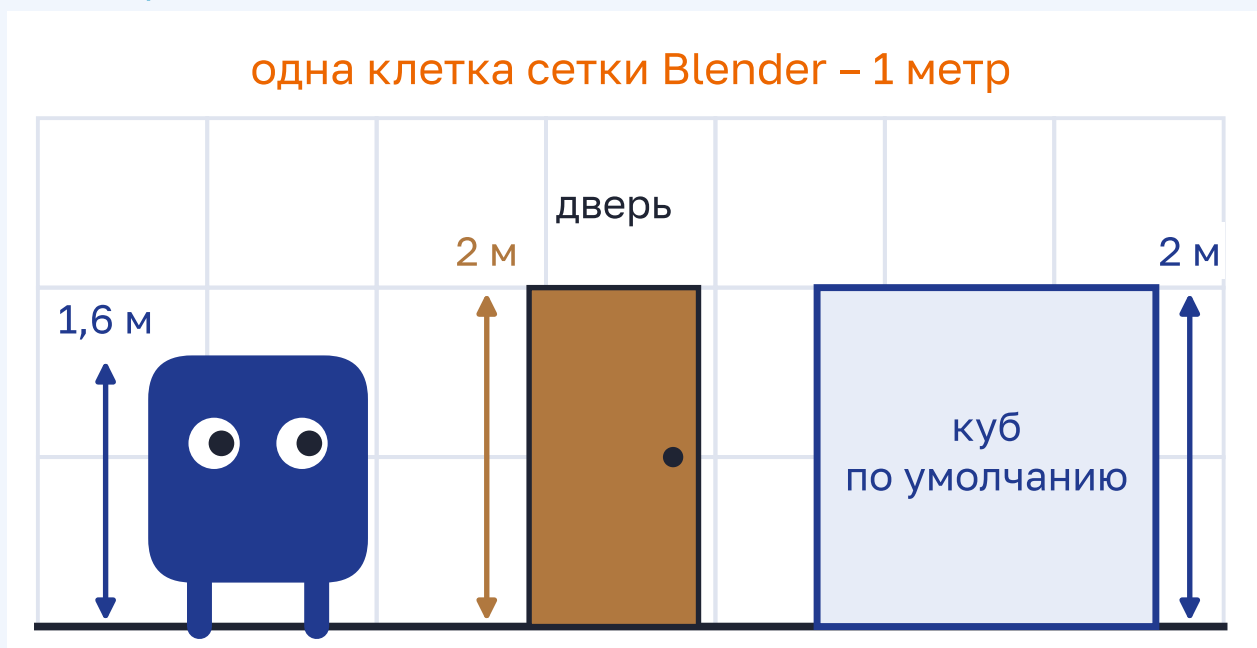


Рисунок 8.9 – Сетка Blender, вид сбоку: герой, дверь и куб по умолчанию

Одна клетка сетки – метр. Герою ростом 1,6 м нужна дверь хотя бы 2 м, а куб по умолчанию ровно с эту дверь по высоте.

В Blender одна **единица длины** по умолчанию – **1 метр**. Это видно в настройках сцены (раздел Units): там стоит метрическая система и метры. Ось **Z смотрит вверх**, X – вправо, Y – вглубь. На вебинаре сферы робота делали ростом около 1,6 м, сверяясь с клетками сетки.

Движок – программа, которая запускает игру и рисует её на экране, – тоже считает в метрах. Поэтому размеры в Blender должны быть настоящими: стул – 0,45 м до сиденья, дверь – около 2 м, герой – 1,6–1,8 м. Модель, сделанная «на глаз», в игре окажется игрушечной или огромной.

Scale и Dimensions. Нажми N в окне 3D-вида – справа откроется панель с числами объекта. В ней две похожие строчки.

- **Dimensions** (размеры) – настоящие размеры объекта в метрах.
- **Scale** (масштаб) – во сколько раз объект растянут относительно своей сетки.

Они связаны умножением:

$$\text{размер} = \text{размер сетки} \cdot \text{Scale}.$$

У стандартного куба сетка 2 м. Если нужен ящик высотой 0,5 м, Scale по Z должен быть $0,5 : 2 = 0,25$. Главная ловушка: написать Scale 0,5 и получить ящик высотой $2 \cdot 0,5 = 1$ м.

После масштабирования его обычно **применяют**: Ctrl + A, пункт Scale (или All Transforms – сразу всё, так делали на вебинаре). Сетка пересчитывается под новый размер, Scale снова становится 1, а Dimensions остаются прежними. Зачем это нужно? Если объект сильно растянули в одну сторону, **фаска** (срезанный уголок на ребре) выходит на нём кривой: с одной

стороны широкая, с другой узкая. После Ctrl + A фаски получаются одинаковыми со всех сторон.

Экспорт в игру. Готовую модель переносят в движок файлом. На вебинаре сферы делали так: выделить все объекты модели, выбрать Файл → Экспорт и формат **GLB**. В настройках экспорта включить «только выделенные объекты» и «применить модификаторы» – тогда Mirror и Array попадут в файл как настоящая сетка. Потом файл загружали в редактор «Урсулы» и добавляли как окружение. Если бы у модели были анимации, они подгрузились бы вместе с ней.

Запомни. 1 единица Blender = 1 м, Z – вверх. Размер = размер сетки · Scale; куб по умолчанию – 2 м, поэтому нужный Scale – это нужный размер, делённый на 2. Масштаб применяют (Ctrl + A → Scale). В игру – файл GLB с применёнными модификаторами.

Совет. Поставь рядом с моделью «эталон роста» – куб, растянутый до 1,6 × 0,4 × 0,3 м. Если сундук получился выше эталона, что-то не так с масштабом. Перед экспортом эталон удали или не выделяй.

8.6. Растровая картинка: пиксели и разрешение

НА ПАЛЬЦАХ

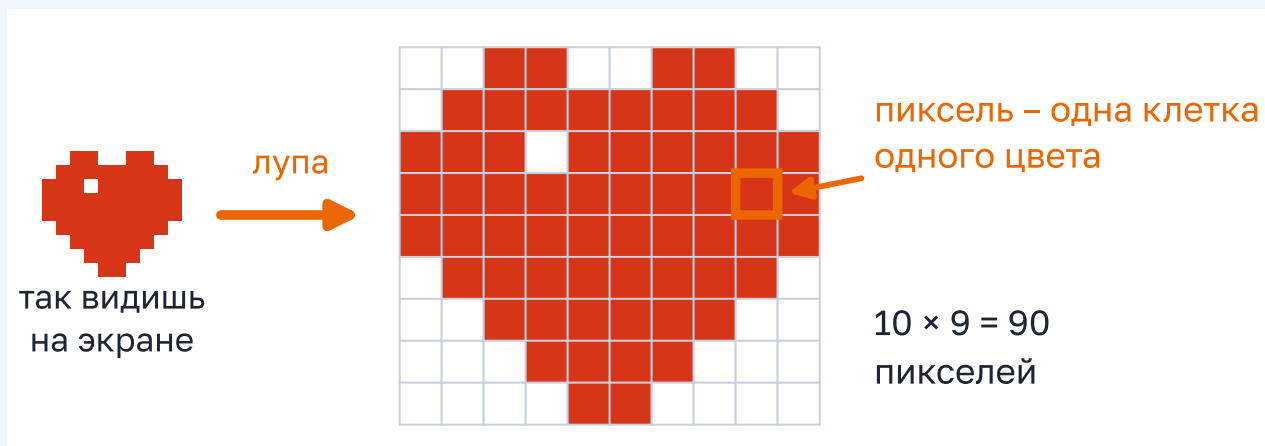


Рисунок 8.10 – Сердечко из игры – вблизи это клетки одного цвета

Экран – как тетрадь в клетку, где каждую клетку закрашивают одним цветом. Клетка называется пикселем.

Растровая картинка – мозаика из маленьких квадратов. Каждый квадратик – **пиксель** – закрашен одним цветом. Фотография, скриншот, текстура на 3D-модели – всё это растр. Каждый пиксель экрана смешивает свой цвет из трёх лампочек: красной, зелёной и синей (по-английски Red, Green, Blue – отсюда **RGB**). У каждой лампочки 256 уровней яркости, от 0 до 255, поэтому всего оттенков $256 \cdot 256 \cdot 256 = 16\,777\,216$.

Разрешение – сколько пикселей в картинке по ширине и по высоте. Записывают «ширина × высота»: 1920 × 1080 значит 1920 пикселей в строке и 1080 строк. Всего пикселей – произведение:

$$1920 \cdot 1080 = 2\,073\,600.$$

У популярных разрешений есть имена.

Таблица 8.2

Имя	Разрешение	Всего пикселей
HD (720p)	1280 × 720	921 600
Full HD (1080p)	1920 × 1080	2 073 600
4K	3840 × 2160	8 294 400



Рисунок 8.11 – Разрешения в одном масштабе. 4K вдвое шире и вдвое выше Full HD, поэтому пикселей в нём в 4 раза больше

Здесь сидит ловушка. 4K больше Full HD **в 2 раза** по ширине и **в 2 раза** по высоте, а пикселей – в $2 \cdot 2 = 4$ раза. С текстурами то же самое: текстура 2048 × 2048 вместо 1024 × 1024 – это в 4 раза больше пикселей и памяти, хотя «цифра выросла вдвое». Размеры текстур художники по привычке делают степенями двойки: 256, 512, 1024, 2048 – каждое следующее число вдвое больше предыдущего.

Доля экрана. Когда проверяют интерфейс, считают, какую часть экрана он занимает. Площадь панели в пикселях делим на площадь экрана и умножаем на 100 %. Шкала здоровья 400 × 50 на экране Full HD: $400 \cdot 50 = 20\,000$ пикселей, $20\,000 : 2\,073\,600 \cdot 100 \approx 0,96\%$. В требованиях к играм финала НТО 2025/26 для старшего трека было, например, такое правило: интерфейс по краям экрана в обычном состоянии занимает не больше 15 % экрана.

Формат файла – это не разрешение. Одну и ту же картинку 1920 × 1080 можно сохранить в разных **форматах** – это способ упаковать пиксели в файл.

- **PNG** – сжимает без потерь и умеет хранить прозрачность. Удобно для значков и спрайтов.
- **JPG** – сжимает сильнее, но немного портит картинку, прозрачности нет. Годится для фотографий.

- **BMP** – почти не сжимает, файл большой.
- **SVG** – вообще не растр, а **вектор**: в файле записаны не пиксели, а линии и фигуры, как формулы. Такую картинку можно увеличивать сколько угодно без «ступенек». Рисунки в этой книге, кстати, тоже SVG.

А ещё бывают форматы не картинок: MP4 – видео, GLB и FBX – 3D-модели. «1080p», «4K», «1280 × 720» – это разрешения; «PNG», «MP4», «GLB» – форматы. В задачах «найди лишнее» их любят смешивать.

Запомни. Пиксель – одна клетка одного цвета. Разрешение – ширина × высота в пикселях; всего пикселей – произведение. Вдвое больше по каждой стороне – вчетверо больше пикселей. Формат (PNG, JPG) – как упакован файл, разрешение (1920 × 1080) – сколько в нём пикселей. SVG – вектор.

8.7. Эскиз и силуэт персонажа

НА ПАЛЬЦАХ

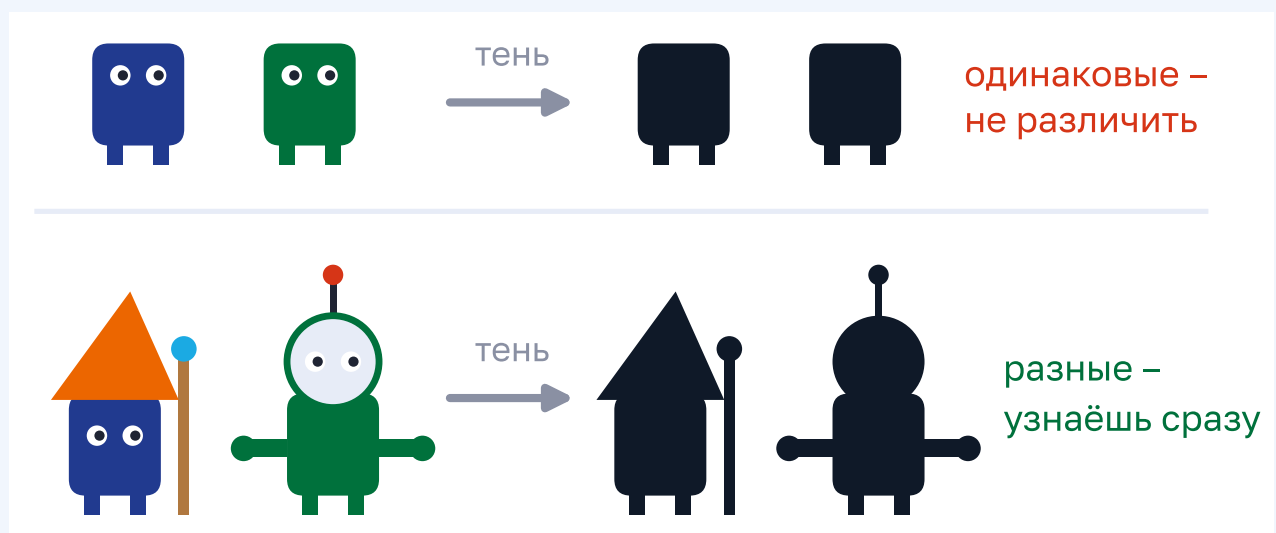


Рисунок 8.12 – Сверху – герои отличаются только цветом, снизу – формой. Справа – их тени

В тени цвет пропадает, а форма остаётся. Хорошего героя узнают даже по тени.

До Blender модель рождается на бумаге. **Эскиз** – быстрый рисунок карандашом, по которому видно идею: кто это, какого роста, что у него в руках. Эскиз не обязан быть красивым, он должен быть понятным. В «Урсуле» есть конкурс «Идея и визуал», куда присылают эскизы персонажей. На вебинаре сферы говорили, что по самым интересным эскизам сделают настоящие 3D-модели.

Силуэт – контур фигуры, залитый одним тёмным цветом, как тень на стене. Проверка простая: закрась героя целиком чёрным. Если по кляксе всё ещё понятно, кто это, – силуэт хороший. Два героя, которые отличаются только цветом куртки, в тени сливаются. Колпак, посох, антенна, огромные уши – делают силуэт узнаваемым. Игроку это нужно, чтобы за долю секунды отличить друга от врага в тёмной пещере или издали.

Виды спереди, сбоку и сверху. Чтобы по эскизу было удобно моделировать, рисуют **виды**: как фигура выглядит ровно спереди, ровно сбоку и ровно сверху. На вебинаре такие картинки называли **референсами** – их переносят в Blender, ставят на фон и строят модель по-верх (вид спереди – клавиша 1, сбоку – 3, сверху – 7). Нужны хотя бы два вида: спереди и сбоку.

Один вид обманывает. Цилиндр спереди – прямоугольник, а сверху – круг. Конус и пирамида спереди – одинаковые треугольники, а различить их можно только сверху: у конуса там круг, у четырёхугольной пирамиды – квадрат с диагоналями-рёбрами, сходящимися к верхушке.

Персонажей для 3D рисуют и моделируют в **Т-позе**: стоит прямо, ноги на ширине плеч, руки вытянуты в стороны. Так на вебинаре моделировали и робота: в этой позе удобно пользоваться симметрией – одну руку делаешь, вторую отражает Mirror. Потом внутрь модели ставят скелет, чтобы её оживить.

Запомни. Эскиз – идея на бумаге, референсы – виды спереди и сбоку для моделирования. Силуэт проверяют заливкой чёрным: героя должно быть видно по форме, а не только по цвету. Персонажа моделируют в Т-позе.

Совет. Нарисуй три силуэта своих героев, покажи другу и спроси, кто есть кто. Если он путается – преувеличь одну деталь каждого: у мага колпак выше, у робота голова круглее, у рыцаря щит больше.

8.8. Разбираем задачи

Задача 8.1 Карандаш для инвентаря

разминка · вершины, рёбра, грани, призма, формула Эйлера

Для игры-квеста нужен карандаш, который лежит в инвентаре героя. Художник сделал его low-poly: шестигранный незаточенный карандаш – это шестиугольная призма, закрытая с обоих концов. Сколько у этой модели вершин, рёбер и граней? Ответ – три числа через пробел в этом порядке.

Дано: шестиугольная призма, $n = 6$.

Найти: В, Р, Г – ?

Решение

Шаг 1. Что за тело. Призма: сверху и снизу одинаковые шестиугольники, по бокам – прямоугольники. Считаем по частям: верх, низ, бока.

Шаг 2. Вершины. У шестиугольника 6 углов. Сверху 6 вершин, снизу 6: $6 + 6 = 12$.

Шаг 3. Рёбра. По краю верхнего шестиугольника 6 рёбер, по краю нижнего – ещё 6, и 6 вертикальных рёбер соединяют верх с низом: $6 + 6 + 6 = 18$.

Шаг 4. Грани. Верх, низ и 6 боковых граней – по одной на каждую сторону шестиугольника:
 $1 + 1 + 6 = 8$.

Шаг 5. Проверка Эйлера. $12 - 18 + 8 = 2$ – сошлось.

А по формулам? У призмы $2n = 12$, $3n = 18$, $n + 2 = 8$ – те же числа. Возьми настоящий карандаш: у него 6 граней-боков и два торца, всего 8.

! **Чаще всего забывают торцы** и отвечают «6 граней» – это только бока. Другая ошибка – считать рёбра по граням: у 6 боковых граней по 4 ребра, у двух торцов по 6, всего $24 + 12 = 36$. Но каждое ребро – общая сторона двух граней, и так оно посчитано дважды: $36 : 2 = 18$. Если Эйлер не дал 2, первым делом ищи такой двойной счёт.

Ответ: 12 18 8.

Своя задача по программе сферы, модуль 3

Реши сам

8.1. Крыша сторожевой башни в игре – пятиугольная пирамида вместе с дном. Сколько у неё вершин, рёбер и граней? Ответ – три числа через пробел.

8.2. Для настольной игры внутри видеоигры смоделировали двадцатигранный кубик: у него 20 треугольных граней и 12 вершин. Сколько у модели рёбер?

Задача 8.2 Ящик из куба

разминка · масштаб, размеры куба по умолчанию, Scale и Dimensions

Для склада в игре нужен ящик шириной 0,8 м (по оси X), глубиной 0,6 м (по Y) и высотой 0,5 м (по Z). Ты добавил в Blender стандартный куб и собираешься растянуть его, вписав числа в строку Scale. Какие значения Scale нужно поставить по X, Y и Z? Ответ – три числа через пробел.

Дано: куб по умолчанию – 2 м по каждой оси; нужно $0,8 \times 0,6 \times 0,5$ м.

Найти: Scale по X, Y, Z – ?

Решение

Шаг 1. Что происходит. Размер = размер сетки · Scale. Сетка куба – 2 м по каждой оси. Значит, Scale = нужный размер : 2.

Шаг 2. По X. $0,8 : 2 = 0,4$.

Шаг 3. По Y. $0,6 : 2 = 0,3$.

Шаг 4. По Z. $0,5 : 2 = 0,25$.

Проверка. $2 \cdot 0,4 = 0,8$ м, $2 \cdot 0,3 = 0,6$ м, $2 \cdot 0,25 = 0,5$ м. В панели N в строке Dimensions будут 0,8, 0,6 и 0,5 м.

Все три Scale меньше 1, и это понятно: куб по умолчанию высотой с дверь, а ящик нам нужен примерно по колено взрослому.

❗ Типичная ошибка. Вписать в Scale сами размеры: 0,8, 0,6, 0,5. Blender послушно умножит их на 2, и ящик выйдет $1,6 \times 1,2 \times 1$ м – в человеческий рост. Scale – это «во сколько раз», а не «сколько метров».

Ответ: 0,4 0,3 0,25.

Своя задача по программе сферы, модуль 3

Реши сам

8.3. Колонну для храма делают из стандартного цилиндра (радиус 1 м, высота 2 м). Колонна должна быть диаметром 0,4 м и высотой 3 м. Какие значения Scale поставить по X, Y и Z? Ответ – три числа через пробел.

8.4. Новичок моделировал стол и думал, что одна клетка сетки – это сантиметр. Высота стола у него получилась 75 единиц. а) Какой высоты в метрах окажется стол в игре? б) На какое число нужно умножить Scale стола, чтобы он стал высотой 0,75 м? Ответ – два числа через пробел.

Задача 8.3 Шлем рыцаря

модификатор Mirror, объединение вершин, отражение координат

Художник смоделировал правую половину шлема рыцаря и добавил модификатор Mirror по оси X. Галочка Merge включена, центр объекта стоит в $x = 0$. В половине шлема 26 вершин, и 6 из них лежат ровно на средней линии, в $x = 0$. а) Сколько вершин будет у целого шлема? б) Вершина на краю забрала стоит в точке (0,35; -0,1; 1,6). В какой точке окажется её отражение? Ответ: число вершин и координата x отражения через пробел.

Дано: $V = 26$ вершин в половине, $k = 6$ на зеркале; Merge включён; центр в $x_0 = 0$; вершина (0,35; -0,1; 1,6).

Найти: вершин в целом шлеме; отражение вершины.

Решение

Шаг 1. Что делает Mirror. Каждая вершина отражается на другую сторону зеркала. Вершины, которые лежат на самом зеркале, отражаются сами в себя, и Merge склеивает их с копией.

Шаг 2. Вершины не на зеркале. $26 - 6 = 20$. У каждой появляется двойник: $20 \cdot 2 = 40$.

Шаг 3. Вершины на зеркале. 6 штук, двойников у них нет – они склеились: остаётся 6.

Шаг 4. Всего. $40 + 6 = 46$. По формуле: $2 \cdot 26 - 6 = 46$.

Шаг 5. Отражение вершины. Зеркало по X, центр в нуле – у координаты x меняется знак, y и z остаются прежними: (-0,35; -0,1; 1,6). По формуле: $x' = 2 \cdot 0 - 0,35 = -0,35$.

46 – чуть меньше, чем $2 \cdot 26 = 52$: склеилась только средняя линия. Край забрала справа в 35 см от середины, и слева тоже в 35 см – шлем симметричный.

❗ **Ответ 52** получается, если забыть, что вершины на зеркале склеиваются. Он верен, только если Merge выключен. А ещё иногда меняют знак у всех трёх координат, хотя Mirror по X трогает только x.

Ответ: 46 вершин; отражение в точке $(-0,35; -0,1; 1,6)$. Строка ответа: «46 -0,35».

Своя задача по программе сферы, модуль 6

Реши сам

8.5. Половина лодки состоит из 40 вершин, 8 из них лежат на плоскости зеркала. К лодке добавили Mirror. Сколько вершин будет у целой лодки, если галочка Merge включена? А если выключена? Ответ – два числа через пробел.

8.6. У головы персонажа сделана правая половина, к ней добавлен Mirror по оси X. Центр объекта случайно оказался в точке $x = 0,3$, а вершина правого уха стоит в $x = 0,9$. В какой координате x окажется левое ухо?

Задача 8.4 Забор у фермы

модификатор Array, шаг и длина ряда, вершины копий

Для фермы в игре делают забор. Одна штакетина – брусок шириной 0,1 м (по оси X) с 8 вершинами. К ней добавили Array: Count = 40, Relative Offset по X = 1,5. а) Какой длины получится забор в метрах? б) Сколько вершин у всего забора? в) Сколько строк-объектов покажет список Outliner для этого забора? Ответ – три числа через пробел.

Дано: $a = 0,1$ м; $n = 40$; $k = 1,5$; у штакетины 8 вершин.

Найти: L, число вершин, число объектов.

Решение

Шаг 1. Шаг массива. Относительный сдвиг 1,5 – это полторы ширины штакетины: $s = 1,5 \cdot 0,1 = 0,15$ м. Между досками остаётся зазор $0,15 - 0,1 = 0,05$ м – 5 см, в такую щель пролезет разве что мышь.

Шаг 2. Сколько шагов. Досок 40, шагов между ними на один меньше: $40 - 1 = 39$.

Шаг 3. Длина забора. Первая доска целиком плюс 39 шагов:

$L = 0,1 + 39 \cdot 0,15 = 0,1 + 5,85 = 5,95$ м.

Шаг 4. Вершины. Каждая копия – та же штакетина: $40 \cdot 8 = 320$ вершин.

Шаг 5. Объекты. Array не создаёт новых объектов, все копии – часть одной штакетины. В Outliner одна строчка.

Прикинем на глаз. Без зазора 40 досок по 0,1 м дали бы 4 м. С зазором выходит больше, но меньше, чем $40 \cdot 0,15 = 6$ м, – наши 5,95 м туда и попали. Забор в шесть метров – одна сто-

рона небольшого огорода на даче.

❗ **Рука сама пишет** $40 \cdot 0,15 = 6$ м. Но так мы добавляем лишний зазор после последней доски, а его нет. А кто-то решит, что сдвиг 1,5 – это 1,5 метра, и получит забор длиной почти 60 м. *Relative Offset* меряется в ширинах самой детали.

Ответ: 5,95 320 1.

Своя задача по программе сферы, модуль 6

Реши сам

8.7. Железную дорогу в игре собирают из шпал. Шпала – брусок шириной 0,25 м вдоль пути с 8 вершинами. Модификатор Array: Count = 21, Relative Offset вдоль пути = 2. Какой длины получится путь в метрах и сколько у него вершин? Ответ – два числа через пробел.

8.8. Мост через ручей собирают из досок шириной 0,3 м. Array ставит их с Relative Offset = 1,2. Какое наименьшее значение Count нужно поставить, чтобы мост был длиной не меньше 6 м?

Задача 8.5 Не закрой игру интерфейсом

пиксели, разрешение, проценты от площади экрана

Команда делает игру для экрана Full HD (1920 × 1080). Договорились: интерфейс по краям экрана (HUD) должен занимать не больше 15 % экрана. Сейчас на экране шкала здоровья 480 × 60 пикселей, мини-карта 300 × 300 и панель предметов 720 × 90. Панели не перекрываются. а) Какую долю экрана в процентах занимает интерфейс? Округли до десятых. б) Дизайнер хочет добавить постоянное окно заданий 400 × 500 пикселей. Уложится ли тогда интерфейс в 15 %? Ответ: число и «да» или «нет».

Дано: экран 1920 × 1080; панели 480 × 60, 300 × 300, 720 × 90; новое окно 400 × 500; предел 15 %.

Найти: доля интерфейса в процентах; уложится ли с новым окном.

Решение

Шаг 1. Площадь экрана в пикселях. $1920 \cdot 1080 = 2\,073\,600$.

Шаг 2. Площадь каждой панели.

- шкала здоровья: $480 \cdot 60 = 28\,800$;
- мини-карта: $300 \cdot 300 = 90\,000$;
- панель предметов: $720 \cdot 90 = 64\,800$.

Шаг 3. Весь интерфейс. $28\,800 + 90\,000 + 64\,800 = 183\,600$ пикселей.

Шаг 4. Доля. $183\,600 : 2\,073\,600 \cdot 100 = 8,85... \approx 8,9$ %.

Шаг 5. С окном заданий. Окно: $400 \cdot 500 = 200\,000$ пикселей. Вместе: $183\,600 + 200\,000 = 383\,600$. Доля: $383\,600 : 2\,073\,600 \cdot 100 \approx 18,5\%$. Это больше 15 % – не уложится.

Проверка. 15 % экрана – это $2\,073\,600 \cdot 0,15 = 311\,040$ пикселей. Сейчас занято 183 600 – меньше, с окном 383 600 – больше. Сходится.

И на глаз так же: мини-карта шириной 300 пикселей – примерно шестая часть ширины экрана ($1920 : 300 = 6,4$), небольшой квадрат в углу, а две другие панели – узкие полосы. Вместе они не тянут и на десятую часть экрана.

⚠ Не сравнивай стороны вместо площадей: «мини-карта 300 из 1920 по ширине – это почти 16 %, уже больше предела». Доля экрана – это доля пикселей, то есть площадей. Ещё хитрее ошибаются так: складывают ширины и высоты панелей по отдельности и перемножают суммы. Получается площадь одного большого прямоугольника, которого на экране нет.

Ответ: а) 8,9 %; б) нет (с окном выходит около 18,5 %).

По мотивам задачи НТО 2025/26, профиль «Разработка компьютерных игр» (старший трек, 8–11 классы), заключительный этап, инженерный тур, блок «Арт», «Полишинг проекта»

Реши сам

8.9. Игра рассчитана на экран HD (1280×720). В интерфейсе есть шкала энергии 320×40 пикселей и мини-карта 200×200 , они не перекрываются. Какую долю экрана в процентах они занимают вместе? Округли до десятых.

8.10. Игру с экрана Full HD перенесли на экран 4K (3840×2160). а) Во сколько раз больше пикселей на экране 4K? б) Кнопка на Full HD была высотой 25 пикселей. Интерфейс растянули на весь новый экран, сохранив пропорции. Какой высоты в пикселях стала кнопка? Ответ – два числа через пробел.

Задача 8.6 Лишнее значение

термины: разрешение и формат, растр и вектор, сетка, RGB, симметрия

Из каждого ряда выбери одно лишнее значение. В ответ запиши номера выбранных вариантов по порядку рядов, без пробелов.

Ряд 1: 1) 1280×720 ; 2) Full HD; 3) 4K; 4) MP4.

Ряд 2: 1) PNG; 2) JPG; 3) SVG; 4) BMP.

Ряд 3: 1) ребро; 2) пиксель; 3) грань; 4) вершина.

Ряд 4 (печатные заглавные буквы): 1) Ж; 2) Я; 3) Ф; 4) Ш.

Ряд 5: 1) красный; 2) зелёный; 3) жёлтый; 4) синий.

Дано: 5 рядов по 4 значения.

Найти: в каждом ряду – значение, которое не подходит к трём остальным.

Решение

Шаг 1. В каждом ряду сначала ищем, что объединяет троих, и только потом – кто выпадает.

- **Ряд 1.** 1280 × 720, Full HD, 4K – разрешения экрана или видео. MP4 – формат видео-файла, а не разрешение. Лишнее – **4**.
- **Ряд 2.** PNG, JPG, BMP – форматы растровых картинок из пикселей. SVG – векторный формат: в нём записаны линии и фигуры. Лишнее – **3**.
- **Ряд 3.** Ребро, грань, вершина – части полигональной сетки 3D-модели. Пиксель – клетка растровой картинке. Лишнее – **2**.
- **Ряд 4.** Ж, Ф, Ш симметричны слева направо: левую половину отражаешь модификатором Mirror – получаешь целую букву. У Я левая и правая половины разные. Лишнее – **2**.
- **Ряд 5.** Красный, зелёный, синий – три цвета RGB, из которых экран смешивает цвет каждого пикселя. Жёлтый экран получает смесь красного и зелёного. Лишнее – **3**.

Шаг 2. Собираем строку: 4, 3, 2, 2, 3 – «43223».

Шаг 3. Перед тем как сдать. Рядов пять – и цифр в ответе пять. У каждой тройки общий признак назван одним-двумя словами: «разрешения», «растровые форматы», «части сетки», «симметричные буквы», «цвета RGB». Спорить не о чем.

 **Типичная ошибка.** В ряду 1 выбрать 4K, «потому что оно самое большое», или 1280 × 720, «потому что записано числами». Лишнее ищут по смыслу, а не по виду записи: Full HD и 1920 × 1080 – одно и то же разрешение, просто по-разному названное. В ряду 5 многие выбирают синий, вспоминая краски на уроке рисования. Но экран смешивает не краски, а свет, и у света свои три основных цвета.

Ответ: 43223.

По мотивам задачи НТО 2025/26, профиль «Разработка компьютерных игр» (старший трек, 8–11 классы), первый отборочный этап, инженерный тур, «Арт. Лишнее значение»

Реши сам

8.11. Команда делает квест в «Урсуле». Для каждой работы выбери программу: а) 3D-модель сундука; б) диалог с торговцем, где игрок выбирает ответ; в) поведение стражника: он патрулирует и замечает игрока; г) карандашный эскиз героя на компьютере. Программы: 1) ПРИМС; 2) графический редактор, например Krita; 3) Blender; 4) Inky (язык Ink). Ответ – четыре цифры по порядку работ, без пробелов.

8.12. На рисунке – вид спереди и вид сверху четырёх фигур А, Б, В, Г. Для примера слева показан цилиндр. Узнай каждую фигуру: 1) куб; 2) конус; 3) четырёхугольная пирамида; 4) шар. Ответ – номера для А, Б, В, Г по порядку, без пробелов.

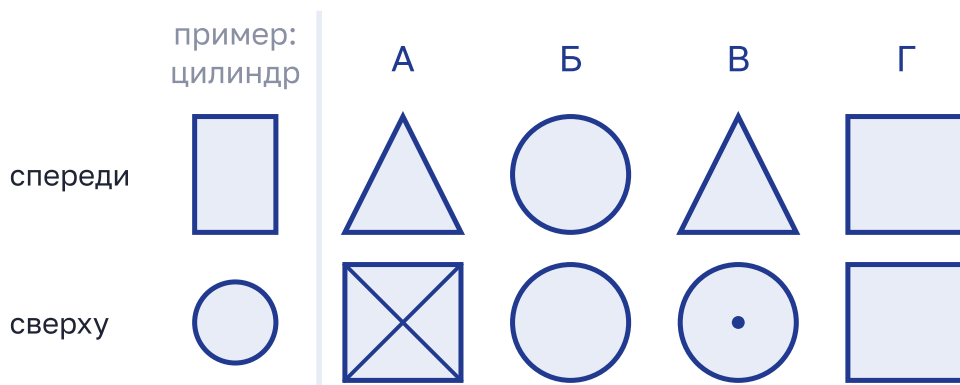


Рисунок 8.13 – Виды спереди и сверху. Для примера слева – цилиндр

Задача 8.7* Лес из одной ёлки

треугольники, стопка модификаторов Array и Mirror, бюджет сцены

В игре на опушке нужен лес. Low-poly ёлка собрана из двух частей: ствол – цилиндр с 6 вершинами по кругу (с верхней и нижней крышкой), крона – конус с 6 вершинами по кругу (с дном). К ёлке добавили три модификатора, сверху вниз: Array по X с Count = 6, Array по Y с Count = ?, Mirror по X. Mirror отражает лес на другую сторону тропинки, ёлки не касаются зеркала. Команда договорилась: во всей сцене не больше 6000 треугольников, а дом и герой уже занимают 3500. а) Сколько треугольников в одной ёлке? б) Какое наибольшее значение Count можно поставить во втором Array, чтобы сцена уложилась в договорённость? Ответ – два числа через пробел.

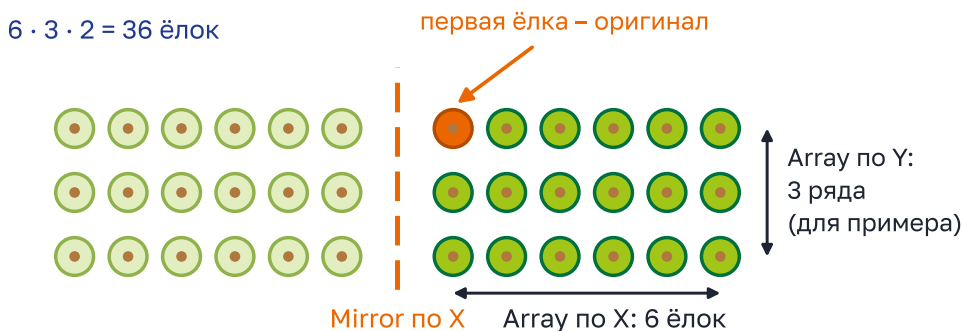


Рисунок 8.14 – Лес сверху: одна ёлка, два массива и зеркало. Здесь для примера во втором массиве 3 ряда

Дано: ствол – шестиугольная призма, крона – шестиугольная пирамида; Array X: 6; Array Y: m – ?; Mirror по X; лимит 6000 треугольников, занято 3500.

Найти: треугольников в ёлке; наибольшее m .

Решение

Шаг 1. Что за тела. Цилиндр с 6 вершинами по кругу – это шестиугольная призма. Конус с 6 вершинами по кругу – шестиугольная пирамида.

Шаг 2. Треугольники в стволе. 6 боковых граней – четырёхугольники, по 2 треугольника: $6 \cdot 2 = 12$. Две крышки – шестиугольники, по $6 - 2 = 4$ треугольника: $2 \cdot 4 = 8$. Ствол: $12 + 8 = 20$.

Шаг 3. Треугольники в кроне. 6 боковых граней уже треугольники: 6. Дно – шестиугольник: 4. Крона: $6 + 4 = 10$.

Шаг 4. Одна ёлка. $20 + 10 = 30$ треугольников.

Для любопытных – быстрая проверка. Если тело без дырок и все его грани разрезаны на треугольники, треугольников всегда $2 \cdot V - 4$ (это следует из формулы Эйлера). У ствола 12 вершин: $2 \cdot 12 - 4 = 20$. У кроны 7 вершин: $2 \cdot 7 - 4 = 10$. То же, что и по граням.

Шаг 5. Сколько ёлок даёт один ряд второго массива. Первый Array даёт ряд из 6 ёлок. Каждая единица Count во втором Array добавляет такой ряд, а Mirror удваивает всё: $6 \cdot 2 = 12$ ёлок на каждый ряд, то есть $12 \cdot 30 = 360$ треугольников.

Шаг 6. Сколько треугольников свободно. $6000 - 3500 = 2500$.

Шаг 7. Сколько рядов войдёт. $2500 : 360 = 6,94\dots$ Семь рядов не влезут, считаем целые – округляем **вниз**: $m = 6$. Проверка: при $m = 6$ ёлок $6 \cdot 6 \cdot 2 = 72$, треугольников $72 \cdot 30 = 2160 \leq 2500$. При $m = 7 - 2520 > 2500$, перебор.

72 ёлки – приличная опушка, а берут они меньше половины бюджета всей сцены. Low-poly для того и придумали: одна «тяжёлая» стандартная сфера (482 вершины, 960 треугольников) стоит как 32 наши ёлки.

⚠ Опаснее всего забыть, что Mirror удваивает лес: $2500 : 180 = 13,9$ и ответ 13 – сцена вылезет за бюджет почти вдвое. Вторая ошибка – считать грани, а не треугольники: у ёлки $8 + 7 = 15$ граней, и ответ выйдет завышенным. Третья – округлить 6,94 до 7: «сколько поместится» округляют вниз.

Ответ: 30 6.

Своя задача по программе сферы, модуль 6

Реши сам

8.13.* Вдоль дороги ставят фонари. Фонарь – это столб (цилиндр с 8 вершинами по кругу, с обеими крышками) и плафон-куб. Модификаторы сверху вниз: Array вдоль дороги с Count = ?, потом Mirror на другую сторону дороги (фонари зеркала не касаются). Для фонарей осталось 1000 треугольников. Какое наибольшее значение Count можно поставить?

8.14.* Из куба делают сплошную балку: Array с Count = 10 и Relative Offset = 1 (кубы вплотную), галочка Merge в Array включена. Сколько вершин у балки?

Ответы и решения

Сначала реши задачу сам и только потом сверяйся. Не сошлось – найди в решении шаг, на котором твой путь разошёлся с нашим.

К главе 1. Игры и жанры: язык геймдизайнера

1.1. Ответ: 4123.

Подсказка: цифры пишем для букв А–Г по порядку. Если вышло «2341», ты записал буквы для частей 1–4.

1.2. Ответ: 35142.

1.3. Ответ: 6231.

1.4. Ответ: 56231.

Подсказка: у игры может быть два жанра сразу; внутри каждой игры цифры идут по возрастанию, поэтому «56», а не «65».

1.5. Ответ: 3223.

1.6. Ответ: 432.

Подсказка: в третьем вопросе ищем не лишнюю игру, а ту, что подходит, – квест.

1.7. Ответ: 2467.

1.8. Ответ: 134.

Подсказка: утверждение 5 – ловушка: задания от NPC бывают и в RPG, и в стратегиях, это «квест» во втором смысле.

1.9. Ответ: 41325.

1.10. Ответ: 2413.

1.11. Ответ: 61.

Подсказка: если вышло 48 – ты забыл про начатую третью стопку яблок, в неё войдёт ещё 13.

1.12. Ответ: 5.

Подсказка: сначала найди, сколько нужно верных ответов: 70 % от 18 – это 12,6, округляем **вверх**, до 13. Ошибок – всё остальное.

1.13. Ответ: 10.

Подсказка: Б, В, Г, Д выстраиваются в ряд двумя способами (В и Г меняются местами), а свободное дело Е встаёт в любой из пяти промежутков этого ряда.

1.14. Ответ: 20.

Подсказка: ком за комом – Б, В, Г – идут строго по порядку. Свободных дел два, вставляй их по одному: Д – в один из четырёх промежутков ряда из трёх, потом Е – в один из пяти промежутков получившегося ряда из четырёх.

К главе 2. Геймдизайн: механики, баланс, документ игры

2.1. Ответ: 8.

2.2. Ответ: 75; на 6,25 %.

Подсказка: второй процент берут от 100 – от урона после зелья, а не от исходных 80. Поэтому +25 % и –25 % не уничтожают друг друга.

2.3. Ответ: 4512.

2.4. Ответ: 1321.

Подсказка: про каждый набросок спроси по очереди: что игрок получает за действие? куда это можно потратить?

2.5. Ответ: 88 мин.

2.6. Ответ: 33 бревна.

Подсказка: сначала число порций досок (округли вверх), потом брёвна на них – и не забудь 6 брёвен на опоры.

2.7. Ответ: 5.

Подсказка: выпиши цены и строку «всего потрачено», как в задаче 2.4.

2.8. Ответ: а) 500; б) 1000.

Подсказка: с 9-го уровня на 10-й – это 9-й по счёту переход, до него 8 шагов по +50. С 1-го уровня до 6-го – 5 переходов, а не 6.

2.9. Ответ: 46.

2.10. Ответ: 34.

Подсказка: сравни числа в «Цели» и в «Уровнях»: сколько кирпичей нужно и сколько их можно скатать. И поищи, как игрок может проиграть.

2.11. Ответ: а) 32 %; б) 75 %.

Подсказка: на плейтестах было разное число людей, поэтому сравнивают проценты, а не число нашедших.

2.12. Ответ: а) 4; б) 33 %.

2.13. Ответ: а) на 100 монет больше; б) на 100 монет меньше.

Подсказка: теплица окупается за $300 : 20 = 15$ мин. Сравни это с тем, сколько осталось до конца ярмарки.

2.14. Ответ: с луком, на 1 с раньше.

Подсказка: урон в секунду здесь не годится. Найди число выстрелов (вверх), а между первым и последним выстрелом промежутков на один меньше, чем выстрелов.

К главе 3. Левел-дизайн: карты и пути

3.1. Ответ: 21 м.

3.2. Ответ: 1,5 с.

Подсказка: по сторонам шагов $7 + 3 = 10$, а с диагоналями – столько, сколько большая разница, то есть 7.

3.3. Ответ: (11; 5).

3.4. Ответ: 9.

Подсказка: раскрой все циклы в столбик – получится 12 команд; иди по ним пальцем по карте, а не складывай сдвиги.

3.5. Ответ: 11.

3.6. Ответ: 16.

Подсказка: запусти две волны: от С до К, потом новую – от К до Ф – и сложи. Ключ стоит в тупике, от него придётся вернуться назад.

3.7. Ответ: 11 м.

3.8. Ответ: 19 м.

Подсказка: туда и обратно – две разные задачи. Туда можно прыгнуть с обрыва, а обратно этой дороги нет.

3.9. Ответ: 11.

3.10. Ответ: 3.

Подсказка: отмечай по кругам, куда можно дойти и какие ключи там лежат; посмотри, где лежит жёлтый ключ.

3.11. Ответ: 7.

3.12. Ответ: 30.

Подсказка: зарядка делит путь на две части. Посчитай пути до неё, потом пути от неё до финиша (как от нового старта) и перемножь – правило умножения из раздела 3.5.

3.13. Ответ: 3124.

3.14. Ответ: 135.

3.15. Ответ: 32.

3.16. Ответ: 24.

Подсказка: «хотя бы 5» – это «ровно 5» плюс «ровно 6»; посчитай каждое так же, как в задаче 3.8, и сложи.

К главе 4. Логика, выборы и случайность

4.1. Ответ: 231.

4.2. Ответ: 2314.

Подсказка: начни с мельника. Он не в крайнем доме, значит, в доме 2 или 3, а пекарь правее его и не в доме 4.

4.3. Ответ: 10.

Подсказка: попробуй «пусть Том – лжец». Тогда его фраза оказывается правдой.

4.4. Ответ: 100.

Подсказка: начни с В – его фраза похожа на «Я лжец». Потом заметь, что А и Б не могут оба сказать правду.

4.5. Ответ: 135.

Подсказка: здесь ИЛИ стоит снаружи скобок. Если свиток есть, портал откроется при любых кристаллах и в любом бою.

4.6. Ответ: 234.

4.7. Ответ: 144.

Подсказка: «очки или нет» – тоже выбор, из двух вариантов.

4.8. Ответ: а) 24; б) 4 мин.

Подсказка: порядков $4! = 24$; секунды переведи в минуты – раздели на 60.

4.9. Ответ: 8.

Подсказка: к Пристани ведут стрелки и с Берега, и с Моста. Сначала найди число путей к Мосту, потом к Пристани.

4.10. Ответ: 6.

Подсказка: склей два куска пути, как в задаче 4.5: от Старта до Моста и от Моста до Замка.

4.11. Ответ: а) 10 %; б) 1 из 10; в) 5.

4.12. Ответ: 8 %.

Подсказка: площадь круга – πr^2 . Её делят на площадь поля.

4.13. Ответ: а) 20 %; б) 67 %.

4.14. Ответ: 0,17.

Подсказка: прошлые броски не в счёт. Остаётся шанс суммы 7 на двух кубиках.

К главе 5. Машины состояний: как думает NPC

5.1. Ответ: 13.

Подсказка: без ключа условие [есть ключ] ложно, и клик по замку ничего не меняет.

5.2. Ответ: 2.

5.3. Ответ: 11 монет.

Подсказка: монеты теперь выдаются на стрелках, а переходов «туда» и «обратно» при 7 кликах получается разное число.

5.4. Ответ: а) 5 монет; б) 7 раз.

Подсказка: «вход» в «пуст» срабатывает один раз, а каждый следующий клик – внутреннее событие.

5.5. Ответ: а) 3 зелья; б) 10 монет.

Подсказка: «купить», пока торговец в состоянии «ждёт», пропадает.

5.6. Ответ: 2321232344.

Подсказка: после третьего удара здоровья 0, и таймер ведёт уже в «празднует». Последнее событие там пропадает.

5.7. Ответ: 45.

Подсказка: в 5 стрелка входит, но только из 4, а в 4 попасть нельзя.

5.8. Ответ: б.

Подсказка: на входе в «висит» поиск игрока выключается, поэтому в этом состоянии событию «игрок обнаружен» неоткуда взяться.

5.9. Ответ: 9.

5.10. Ответ: 23331231.

5.11. Ответ: 2.

Подсказка: сначала переведи время в секунды: 1 мин 40 с = 100 с.

5.12. Ответ: а) 5; б) 333 раза.

Подсказка: хвост 1 → 2 → 3 занимает 3 тика, круг 4 → 5 → 6 – тоже 3 тика.

К главе 6. Переменные, условия и поиск ошибок

6.1. Ответ: 70.

6.2. Ответ: а) 9 9; б) 2.

Подсказка: после строки левая ← правая четвёрка пропала навсегда. Чтобы её не потерять, сначала отложи её в третью коробку.

6.3. Ответ: 22033.

6.4. Ответ: 1110.

Подсказка: здесь нет «иначе», поэтому проверяются все три строки. Кто прошёл быстрее 120 секунд, у того последняя строка перезапишет звёзды единицей.

6.5. Ответ: 28 1.

Подсказка: здесь « ≤ 20 », а не « < 20 ». На четвёртый день сытость 22 – еду не тратим.

6.6. Ответ: 3 4.

Подсказка: веди таблицу с двумя столбцами А и В. В последнем переливании в А помещается только 1 литр: $3 - 2 = 1$.

6.7. Ответ: а) 29; б) 47.

6.8. Ответ: 3 7.

Подсказка: после каждой выдачи предмета счётчик обнуляется. После победы 2 неудачи идут на победах 3, 4, 5, 6 – на седьмой их уже 4.

6.9. Ответ: 89.

Подсказка: жалобу объясняет «или» в строке 9. Но если заменить его на «и», дверь всё равно не откроется никогда: строка 8 обнуляет факелы перед проверкой.

6.10. Ответ: а) -10, «нет»; б) 8.

Подсказка: здоровье идёт 75, 50, 25, 65, 40, 15, -10 и через ноль перепрыгивает. Нужно « ≤ 0 ».

6.11. Ответ: `=A5+B5` ; `=A4*$F$1` .

6.12. Ответ: а) 3; б) 205; в) 32.

6.13. Ответ: 185.

Подсказка: `range(8, 12)` – это 8, 9, 10, 11. С десятого уровня награда перестаёт расти.

6.14. Ответ: а) 30; б) 85.

Подсказка: числа ложатся в параметры по порядку. В последнем вызове зелье стало 100, а максимум – 30.

К главе 7. Сюжет и диалоги: язык Ink

7.1. Ответ: 415263.

7.2. Ответ: а) 4332; б) 8 строк.

Подсказка: текст «Взять компот.» написан без квадратных скобок, поэтому после нажатия он появляется и в истории.

7.3. Ответ: 3 концовки, 7 прохождений.

7.4. Ответ: а) 7; б) 2 и 4 кнопки.

Подсказка: посчитай прохождения, которые начинаются из узла `neighbor`, так же, как от начала: у `neighbor` пиши 1.

7.5. Ответ: а) концовка 2, заряд 1; б) 4.

7.6. Ответ: 9 прохождений, на поляну – 2.

Подсказка: теперь и после «Дать ягоду» у ручья три кнопки: ягод остаётся 1, а $1 > 0$.

7.7. Ответ: 15.

Подсказка: для концовки 1 нужно `stars` ≥ 3 и чай. Три набираются только через Луну ($1 + 2$), а на крыше выбирают что-то одно: Луну или чай.

7.8. Ответ: 13.

Подсказка: каждую кнопку со звёздочкой в узле `fair` можно нажать один раз. Выпиши все суммы, которые могут получиться: 0, 2, 3, 5.

7.9. Ответ: строки 5 и 11; не хватает $\rightarrow$ END в узле `forest`.

7.10. Ответ: -2.

Подсказка: строка без `~` не прибавила яблок, а два соседу отдали из нуля.

7.11. Ответ: 231321.

7.12. Ответ: а) 13; б) 4.

Подсказка: переменную `red_hat` записывают, но нигде потом не проверяют – значит, шапка ни на что не влияет.

7.13. Ответ: 30 прохождений, с сараем – 14.

Подсказка: сарай закрыт, только если не осмотрено ни кухни, ни чердака: сразу уехал или осмотрел один чулан.

7.14. Ответ: а) 11; б) 0.

Подсказка: после чулана кнопка «Открыть сундук» не исчезает, пока её не нажали, поэтому запасной выбор не сработает никогда.

К главе 8. 3D и визуал: Blender

8.1. Ответ: 6 10 6.

8.2. Ответ: 30.

Подсказка: по Эйлеру $P = B + G - 2$. Или иначе: у 20 треугольников $20 \cdot 3 = 60$ сторон, и каждое ребро – общая сторона двух граней.

8.3. Ответ: 0,2 0,2 1,5.

Подсказка: по X и Y у цилиндра размер задаёт диаметр – 2 м, а не радиус 1 м.

8.4. Ответ: 75 0,01.

8.5. Ответ: 72 80.

8.6. Ответ: -0,3.

Подсказка: зеркало проходит через центр объекта, а не через ноль: $x' = 2 \cdot 0,3 - 0,9$.

8.7. Ответ: 10,25 168.

8.8. Ответ: 17.

Подсказка: здесь известна длина, а искать надо Count. Шаг $0,3 \cdot 1,2 = 0,36$ м; при 16 досках мост 5,7 м – мало, при 17 – 6,06 м. «Чтобы хватило» – округляем вверх.

8.9. Ответ: 5,7 %.

8.10. Ответ: 4 50.

Подсказка: пикселей стало в 4 раза больше, но высота экрана выросла только в 2 раза – а высота кнопки растёт вместе с высотой экрана.

8.11. Ответ: 3412.

8.12. Ответ: 3421.

Подсказка: А и В спереди одинаковые – смотри на вид сверху.

8.13. Ответ: 12.

Подсказка: столб – восьмиугольная призма, $8 \cdot 2 + 2 \cdot 6 = 28$ треугольников, с плафоном 40. Mirror удваивает ряд, поэтому каждая единица Count стоит $2 \cdot 40 = 80$ треугольников; $1000 : 80 = 12,5$ – округляем вниз.

8.14. Ответ: 44.

Подсказка: как в Mirror с Merge. Без склейки $10 \cdot 8 = 80$ вершин, а на каждом из 9 стыков 4 вершины одного куба совпадают с 4 вершинами соседнего.

Справочник

Всё, что может понадобиться под рукой во время решения: формулы и правила из глав, мини-шпаргалки по ПРИМС, «Урсуле», Ink и Blender, словарь слов из мира разработки игр и все «Запомни» из глав на нескольких страницах.

Формулы и правила

Если в условии задачи на отборе правило записано по-своему, решай по условию. Игру придумал её автор, и законы в ней такие, какие он написал.

Здоровье, урон и удары

Сколько нужно ударов. Здоровье врага делим на урон одного удара и округляем **вверх**:

$$n = \frac{HP}{d} - \text{округлить вверх до целого,}$$

где HP – здоровье (очки жизни), d – урон одного удара, n – число ударов. У врага 100 очков здоровья, удар снимает 30: $100 : 30 = 3,33...$ Трёх ударов мало (останется 10 очков), значит, нужно **4** удара.

Урон в секунду:

$$\text{урон в секунду} = d \cdot k,$$

где k – сколько ударов в секунду. Меч бьёт на 12 два раза в секунду: $12 \cdot 2 = 24$ урона в секунду. Враг со здоровьем 120 продержится $120 : 24 = 5$ секунд.

Время боя по ударам. Если между ударами проходит t секунд, а первый удар случается сразу, то последний, n -й, будет через $(n - 1) \cdot t$. Как со столбиками забора: между пятью столбиками четыре промежутка.

Таблица С.1

Правило	Формула	Пример
Броня снижает урон на p %	$d \cdot \left(1 - \frac{p}{100}\right)$	удар 40, броня 25 %: $40 \cdot 0,75 = 30$
Лечение не поднимает здоровье выше максимума	$\min(HP + \text{лечение}, HP_{\max})$	$85 + 30$ при максимуме 100 – будет 100, а не 115
Средний урон с критическим ударом (шанс p , урон в m раз больше)	$d \cdot (1 - p) + d \cdot m \cdot p$	удар 20, шанс 25 %, крит $\times 2$: $20 \cdot 0,75 + 40 \cdot 0,25 = 25$

Проценты, ресурсы, прогрессии

p процентов от числа A : $A \cdot \frac{p}{100}$. Скидка 15 % на меч за 600 монет – это $600 \cdot 0,15 = 90$ монет, меч стоит 510.

Какую часть A составляет от B : $\frac{A}{B} \cdot 100\%$. Собрал 12 кристаллов из 48 – это 25 %.

Проценты друг за другом не складываются. +20 %, потом –20 %: $100 \cdot 1,2 \cdot 0,8 = 96$, а не 100. Второй процент берётся уже от нового числа.

Сколько ждать, чтобы накопить:

$$\text{время} = \frac{\text{цена} - \text{уже есть}}{\text{доход за единицу времени}} - \text{округлить вверх.}$$

Улучшение стоит 250 монет, у тебя 40, шахта приносит 45 монет в минуту:

$$(250 - 40) : 45 = 4,67 - \text{ждать 5 минут.}$$

Сложность растёт на одно и то же число (арифметическая прогрессия):

$a_n = a_1 + (n - 1) \cdot d$. Враг первого уровня – 50 здоровья, каждый уровень +15. На 8-м уровне: $50 + 7 \cdot 15 = 155$.

Сложность растёт в одно и то же число раз (геометрическая прогрессия): $b_n = b_1 \cdot q^{n-1}$. Первое улучшение стоит 10 монет, каждое следующее вдвое дороже. Пятое: $10 \cdot 2^4 = 160$.

Логика: И, ИЛИ, НЕ

«Истина» – 1, «ложь» – 0.

Таблица С.2

A	B	$A \text{ и } B$	$A \text{ или } B$	НЕ A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

«И» истинно, только когда верны оба. «ИЛИ» ложно, только когда неверны оба. Ворота «откроются, если есть ключ И здоровье больше 50»: с ключом и здоровьем 40 – не откроются.

Таблица С.3

Знак в тексте	В Ink	Читается	Где ловушка
$>$	$>$	больше	$50 > 50$ – ложь
$\geq$	$\geq$	больше или равно	$50 \geq 50$ – истина
$<$	$<$	меньше	
$\leq$	$\leq$	меньше или равно	
$=$	$=$	равно	в Ink один знак $=$ кладёт значение, а сравнивают двумя
$\neq$	$\neq$	не равно	

Комбинаторика: варианты, ветки, пути

Таблица С.4

Правило	Формула	Пример
Правило умножения: выбираем по шагам, на каждом шаге свой выбор	$n_1 \cdot n_2 \cdot n_3 \cdot \dots$	4 причёски, 5 цветов, 3 шапки: $4 \cdot 5 \cdot 3 = 60$ разных героев
Правило сложения: выбираем что-то одно из разных групп	$n_1 + n_2$	3 меча или 2 лука: $3 + 2 = 5$ видов оружия
Перестановки: сколькими способами выстроить n предметов в ряд	$n! = 1 \cdot 2 \cdot \dots \cdot n$	4 рычага дёрнуть по очереди: $4! = 24$ порядка
Концовки: k развилок по 2 варианта, ветки нигде не сходятся	2^k	3 развилки: $2^3 = 8$ концовок

Если ветки сюжета снова сходятся в один узел, формула 2^k не работает – рисуй схему и считай пути по ней.

Пути по клеткам (ходим только вправо и вниз). В каждой клетке пишем число путей до неё: в клетках верхнего ряда и левого столбца – по 1, в остальных – сумма числа сверху и числа слева. Число в клетке финиша – ответ.

Таблица С.5

1	1	1
1	2	3
1	3	6

Из левого верхнего угла в правый нижний (2 шага вправо и 2 вниз) ведут 6 путей. Если клетка закрыта стеной, пиши в ней 0 и считай дальше так же.

Вероятность и случайность

$$P(A) = \frac{m}{n},$$

где m – число подходящих исходов, n – число всех равновозможных исходов.

Таблица С.6

Правило	Формула	Пример
Вероятность всегда от 0 до 1	$0 \leq P \leq 1$	0 – не бывает никогда, 1 – бывает всегда
Событие не случится	$P(\text{не } A) = 1 - P(A)$	шанс промаха при попадании 0,7: $1 - 0,7 = 0,3$
Оба независимых события сразу	$P(A) \cdot P(B)$	две монеты, обе орлом: $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$

Правило	Формула	Пример
Хотя бы один раз за n попыток (шанс в каждой p)	$1 - (1 - p)^n$	предмет выпадает с шансом 10 %, три монстра: $1 - 0,9^3 = 0,271$
Два кубика	всего $6 \cdot 6 = 36$ исходов	сумма 7 выпадает 6 способами: $\frac{6}{36} = \frac{1}{6}$
Попасть в зону на поле	$\frac{\text{площадь зоны}}{\text{площадь поля}}$	зона 5×4 на поле 20×10 : $\frac{20}{200} = 0,1$

Среднее, чего ждать от случайности (математическое ожидание): каждое значение умножаем на его вероятность и складываем. В сундуке с шансом 50 % – 10 монет, 30 % – 20 монет, 20 % – 50 монет. В среднем: $10 \cdot 0,5 + 20 \cdot 0,3 + 50 \cdot 0,2 = 5 + 6 + 10 = 21$ монета. Сумма всех вероятностей должна быть ровно 1 (100 %) – проверяй это первым делом.

Карта и пути

Таблица С.7

Что	Как найти	Пример
Шагов по клеткам, если ходить только по сторонам (без диагоналей)	$ x_2 - x_1 + y_2 - y_1 $	из (1; 2) в (6; 5): $5 + 3 = 8$ шагов, если на пути нет стен
Шагов, если можно и по диагонали (каждый шаг одинаковый)	большее из $ x_2 - x_1 $ и $ y_2 - y_1 $	из (1; 2) в (6; 5): 5 шагов
Время пути	$\text{время} = \frac{\text{путь}}{\text{скорость}}$	8 клеток со скоростью 2 клетки в секунду: 4 с
Масштаб карты	настоящее расстояние = число клеток $\times$ размер клетки	8 клеток по 2 м: 16 м

Вертикальные чёрточки $|...|$ значат «без знака минус»: $|3 - 7| = 4$.

Кратчайший путь в лабиринте (волновой алгоритм). В клетке старта пиши 0. Во все свободные соседние клетки (по сторонам) – 1. В их ещё не помеченных соседей – 2, и так далее, пока волна не дойдёт до финиша. Число в клетке финиша – длина кратчайшего пути. Чтобы найти сам путь, иди от финиша назад, каждый раз в соседнюю клетку с числом на 1 меньше.

3D: размеры, координаты, полигоны

Таблица С.8

Что	Правило	Пример
Масштаб в k раз	длины $\times k$, площади $\times k^2$, объём $\times k^3$	куб увеличили в 2 раза: грань в 4 раза больше, объём в 8 раз

Что	Правило	Пример
Отражение по оси X (центр отражения в 0)	$(x; y; z) \rightarrow (-x; y; z)$	$(1,5; 2; 0,3) \rightarrow (-1,5; 2; 0,3)$
Отражение по оси X, центр в точке x_0	$x' = 2x_0 - x$	центр в 1, точка в 4: $x' = 2 - 4 = -2$
Ряд копий со сдвигом на s (длина объекта a , n штук)	$L = a + (n - 1) \cdot s$	доска 0,2 м, 10 копий, сдвиг 0,3 м: $0,2 + 9 \cdot 0,3 = 2,9$ м
Копии вплотную (сдвиг равен длине)	$L = n \cdot a$	10 досок по 0,5 м: 5 м
Куб	8 вершин, 12 рёбер, 6 граней	
Цилиндр, n вершин на окружности	$2n$ вершин, $3n$ рёбер, $n + 2$ грани	$n = 8$: 16 вершин, 24 ребра, 10 граней
Проверка для замкнутой фигуры без дыр (формула Эйлера)	вершины – рёбра + грани = 2	куб: $8 - 12 + 6 = 2$
Сколько треугольников в грани с n углами	$n - 2$	квадрат – 2, восьмиугольник – 6; куб – 12 треугольников

В Blender одна единица длины по умолчанию – 1 метр, а ось Z смотрит **вверх**.

Экран и кадры

Таблица С.9

Что	Как найти	Пример
Пикселей на экране	ширина × высота	$1920 \cdot 1080 = 2\,073\,600$
Время одного кадра	$\frac{1000}{\text{кадров в секунду}}$ мс	60 кадров в секунду: около 16,7 мс; 30 – около 33,3 мс
Длительность анимации	$\frac{\text{число кадров}}{\text{кадров в секунду}}$	120 кадров при 24 кадрах в секунду: 5 с

Мини-шпаргалки по инструментам

Машина состояний и ПРИМС

ПРИМС – «программирование расширенных иерархических машин состояний». Это визуальный язык: программу не пишут строчками, а рисуют схемой. В «Урсуле» так задают поведение персонажей и интерактивных предметов.

Таблица С.10

Элемент	Что это	Как выглядит
---------	---------	--------------

Элемент	Что это	Как выглядит
Состояние	режим работы: «ждёт игрока», «патрулирует», «атакует». У каждого состояния есть имя	скруглённый прямоугольник
Событие	то, что случилось и заставило реагировать: «игрок подошёл», «перемещение выполнено»	подпись на стрелке или строка внутри состояния
Действие	то, что объект делает: «включить красный индикатор», «двигаться к игроку»	на входе в состояние, на выходе, на переходе
Условие	фильтр: переход сработает, только если условие верно («здоровье = 100»). На вебинаре сферы его называли ограждающим	в квадратных скобках на стрелке
Переход	смена состояния	стрелка

Подпись на стрелке в нашей книге всегда одна и та же: **событие [условие] / действие**. Например: «игрок кликнул [есть ключ] / открыть ворота».

Правила, которые спасают от ошибок:

- В каждый момент объект находится **ровно в одном** состоянии. Если состояние составное (внутри него другие), то ещё и в одном из его дочерних.
- Переход из **родительского** состояния срабатывает, в каком бы дочернем состоянии ни был объект. Кнопка «ночной режим» переключит светофор и из «зелёный мигает», и из «жёлтый».
- Где выполняется действие: **на входе** – один раз, когда попали в состояние; **на выходе** – один раз, когда уходим; **на переходе** – в момент перехода; **по внутреннему событию** – каждый раз, когда событие случилось, а состояние не меняется.
- Событие без условия срабатывает всегда. Событие с условием – только если условие верно в этот момент. Если условие ложно, объект остаётся в том же состоянии.
- Имя состояния отвечает на вопрос «что сейчас делает объект?». «Состояние 1» – плохое имя, «ждёт ключ» – хорошее.

Как проследить работу машины: заведи таблицу «шаг – текущее состояние – событие – условие верно? – новое состояние – действие» и заполняй строку за строкой. Не прыгай через шаги.

«Урсула»: что где

Так было в сентябре 2026 года, на вебинарах сферы. Интерфейс может обновиться – смысл останется.

- **Две части работы.** В редакторе «Урсулы» строят сцену: расставляют объекты, двигают, поворачивают, меняют размер. Поведение рисуют в отдельном редакторе диаграмм

«Кибериада Graph Editor» и сохраняют файлом с расширением `.graphml`. Этот файл загружают в карточку модели – в поле «файл алгоритма».

- **Какие бывают объекты:** проходимое окружение, непроходимое окружение, интерактивные объекты, mobs (персонажи и монстры), области с эффектом. Программу можно подключить только к **интерактивным объектам и мобам**.
- **Модули** – наборы команд и событий для поведения. Среди них: обнаружение, движение, взаимодействие (в нём проходимость), индикация (цветной шар над объектом), вывод текста (реплики), атака, здоровье, триггеры, переменные, счётчики, анимация, победа и поражение.
- **Имя объекта в программе должно совпадать с именем на сцене.** Написал «ворота», а объект называется «R1 gates» – программа не найдёт свои ворота.
- **Модуль не работает сам.** Чтобы робот заметил игрока, в его состоянии нужно включить поиск игрока (модуль обнаружения). Забыл – робот пройдёт мимо тебя.
- **Радиус** в командах обнаружения и в других – в единицах сцены. Слишком большой радиус – ворота откроются кликом через всю карту, слишком маленький – до них не дотянуться.
- **Проверяй после каждого изменения:** загрузил программу – перешёл к игре – посмотрел. И сохраняй проект: история «работал весь день, забыл сохранить, всё пропало» случалась на самом деле.

Язык Ink

Ink – язык для интерактивных историй: текст с выборами, развилками и переменными. Писать удобно в редакторе Inky: историю пишешь слева, играешь справа.

Таблица С.11

Запись	Что делает
≡ gate ≡	начало узла с именем <code>gate</code> – кусок истории, куда можно перейти
→ gate	перейти в узел <code>gate</code>
→ END	история закончилась
* Текст	выбор, который исчезает после того, как его выбрали
+ Текст	выбор, который остаётся (к нему можно вернуться снова)
* [Текст]	текст в квадратных скобках виден на кнопке выбора, но не повторяется в истории
-	место сбора: здесь ветки выборов снова сходятся
VAR coins = 0	объявить переменную и дать ей начальное значение
~ coins = coins + 5	изменить переменную
{coins}	вставить в текст значение переменной

Запись	Что делает
* {has_key} [Открыть]	выбор появится, только если условие в фигурных скобках верно
{gate}	сколько раз игрок побывал в узле gate ; 0 – ещё не был
// текст	комментарий: Ink его не показывает

Ещё две записи с вертикальной чертой:

- {coins ≥ 5: Богат!|Беден.} – текст по условию: до черты – если условие верно, после черты – если нет;
- {~0рёл|Решка} – Ink сам выберет случайный вариант из списка.

Сравнение – = , «не равно» – ≠ , «и» – and , «или» – or , «не» – not .

Пример целиком. Эту историю мы проверили: она запускается и работает.

```

VAR coins = 0           // число монет
VAR has_key = false     // флажок: есть ключ или нет

→ gate                 // начать с узла gate

=== gate ===
Перед тобой запёртые ворота. Рядом сидит ёж.
* [Поговорить с ежом] → hedgehog
* {has_key} [Открыть ворота ключом] → finish
+ [Посчитать монеты]
    У тебя {coins} монет.
    → gate

=== hedgehog ===
Ёж фыркает: «Принеси яблоко!»
* [Отдать яблоко]
    ~ coins = coins + 5
    ~ has_key = true
    Ёж отдаёт тебе ключ и пять монет.
- → gate

=== finish ===
{coins ≥ 5: Ворота открыты, и карман звенит!|Ворота открыты.}
→ END

```

Сначала игрок видит два выбора: «Поговорить с ежом» и «Посчитать монеты». Кнопка «Открыть ворота ключом» появится только после разговора с ежом, когда has_key станет true . «Поговорить с ежом» после первого раза исчезнет (звёздочка), а «Посчитать монеты» останется (плюс).

Совет. Ошибки, на которых спотыкаются чаще всего: узел, в который ведёт → , назван по-другому (одна буква – и Ink его не найдёт); ветка ни к чему не ведёт, и история обрывается; в строке `coins = coins + 5` забыли знак `~` в начале – и Ink просто напечатает её как текст; выбор со звёздочкой в узле, куда игрок возвращается, – после первого раза его уже нет.

Blender

Клавиши работают, когда курсор мыши стоит над окном 3D-вида. Внизу окна Blender подсказывает, что сейчас нажато.

Таблица С.12

Клавиши	Что делают
колёсико мыши, зажатое	вращать вид
Shift + колёсико, зажатое	сдвигать вид
колёсико, прокрутка	приблизить или отдалить
цифры на цифровом блоке: 1, 3, 7	вид спереди, сбоку (справа), сверху
Shift + A	добавить объект: куб, цилиндр, сферу и другие примитивы
G, R, S	переместить, повернуть, масштабировать; после буквы можно нажать X, Y или Z – только вдоль этой оси
R, Y, 90	повернуть на 90° вокруг оси Y
Tab	переключиться между режимом объекта и режимом редактирования
1, 2, 3 на верхнем ряду (в режиме редактирования)	выделять вершины, рёбра или грани
A	выделить всё
Shift + D	копия объекта
Alt + D	связанная копия: правишь форму одной – меняются все
E	выдавить (экструдировать) выделенное
I	вставить грань внутрь грани (Inset)
Ctrl + B	фаска на рёбрах (Bevel)
Ctrl + R	разрезать кольцом рёбер (Loop Cut)
M	объединить вершины, например «по расстоянию» – убрать сдвоенные
Alt + Z	режим прозрачности: видно и выделяется то, что сзади

Клавиши	Что делают
/ (на цифровом блоке)	показать только выделенный объект и вернуться обратно
Ctrl + Z	отменить

Модификаторы – это правила, которые меняют форму, не трогая исходную сетку, пока модификатор не применён.

Таблица С.13

Модификатор	Что делает	На что смотреть
Mirror (зеркало)	отражает объект по оси: делаешь половину персонажа – вторая появляется сама	отражает относительно центра объекта (оранжевая точка, origin)
Array (массив)	повторяет объект рядом несколько раз: забор, лестница, рельсы	число копий и сдвиг; сдвиг 1 по оси означает «вплотную»

Материал. Три главных ползунка: **базовый цвет**, **металличность** (есть ли отражения, как у металла) и **шероховатость** (насколько отражения размыты: больше – матовый пластик, меньше – глянец).

Экспорт в игру. «Файл» → «Экспорт» → формат **GLB** (glTF 2.0). Если нужны не все объекты – включи «только выделенные». Если на модели остались модификаторы – включи «применить модификаторы», иначе в игре может приехать половина персонажа без зеркала.

Как искать ошибку

Чужая программа не работает – не переписывай всё сразу. Иди по списку.

- 1. Что должно быть и что есть?** Запиши одной фразой: «робот должен остановиться, когда видит игрока, а он едет дальше».
- 2. В каком состоянии объект сейчас?** Для этого и нужны индикаторы и реплики: цвет шара над роботом говорит, какое состояние сработало.
- 3. Проследи путь по таблице** – шаг за шагом, как в разделе про машины состояний.
- 4. Проверь частые причины:**
 - о имя объекта или узла написано по-разному в двух местах;
 - о нужный модуль не включён на входе в состояние (событие некому поймать);
 - о нет перехода обратно («погнался за игроком» есть, «потерял игрока» – нет, и робот гонится вечно);
 - о условие никогда не бывает верным: «здоровье = 100», хотя здоровье всегда меньше; перепутаны $<$ и $\leq$;
 - о переменную проверяют, но никто её не меняет;
 - о действие стоит на входе, а должно повторяться – значит, нужно внутреннее событие;
 - о проект не сохранили или загрузили старый файл.
- 5. Меняй одно за раз** и сразу проверяй. Поменял три вещи – не узнаешь, какая помогла.

Документ игры (ГДД)

ГДД – геймдизайн-документ: описание игры, по которому её делает команда. В программе сферы ГДД проверяют по тому, есть ли в нём цели, механики, сюжет и персонажи и связаны ли разделы между собой.

Таблица С.14

Раздел	Что написать
Концепция	название, жанр, для кого игра, в одной-двух фразах – о чём она и что должен почувствовать игрок
Цель игры	что нужно сделать, чтобы победить; как можно проиграть
Механики	что делает игрок и как игра отвечает: событие, условие, изменение мира
Игровой цикл	действия, которые игрок повторяет снова и снова
Сюжет	завязка, главные события, развилки и концовки
Персонажи	кто они, чего хотят, как себя ведут (у NPC – состояния)
Уровни и локации	карта, путь игрока, где препятствия, подсказки и точка победы
Управление и интерфейс	какие клавиши, что видно на экране: здоровье, инвентарь, подсказки
Визуальный стиль	эскизы, референсы, цвета

Жанры: коротко

Одна игра часто сочетает несколько жанров. Примеры – известные игры для твоего возраста.

Таблица С.15

Жанр	Главное в игре	Например
Платформер	бег и прыжки по площадкам, препятствия	Super Mario Bros.
Головоломка	решить задачу: сложить, передвинуть, догадаться	Tetris, Angry Birds, Portal 2
Квест (приключение)	исследовать мир, собирать предметы, решать загадки, чтобы дойти до цели	Machinarium
Песочница	свободно строить и менять мир, цели ставишь себе сам	Minecraft

Жанр	Главное в игре	Например
Ролевая игра (RPG)	герой растёт: опыт, уровни, умения, выборы в сюжете	Pokémon
Стратегия	планировать, распоряжаться ресурсами и войсками	Plants vs. Zombies (защита башнями)
Симулятор	похоже повторить занятие из жизни: ферма, полёт, управление	Stardew Valley (ферма)
Гонки	приехать первым	Mario Kart 8
Визуальная новелла	история в картинках, игрок выбирает реплики и решения	
Ритм-игра	попадать действиями в такт музыке	Just Dance

Словарик

Таблица С.16

Слово	Что значит
Ассет	готовая часть игры: модель, текстура, звук, анимация
Баг	ошибка в игре или программе: что-то работает не так, как задумано
Баланс	настройка чисел в игре (здоровье, урон, цены), чтобы было не слишком легко и не слишком трудно
Вершина, ребро, грань	точка, отрезок между двумя точками и плоский кусок поверхности модели. Грань ещё называют полигоном
Геймдизайнер	тот, кто придумывает правила игры и то, что в ней переживает игрок
Геймплей	то, что игрок делает в игре, сам процесс игры
ГДД	геймдизайн-документ: описание игры для команды
Действие (в ПРИМС)	то, что объект делает: включить индикатор, двигаться, атаковать
Динамика	то, что на самом деле происходит, когда игрок играет по правилам (механикам)
Движок	программа, в которой собирают и запускают игру. «Урсула» – редактор и движок сферы

Слово	Что значит
Игровой цикл	повторяющаяся цепочка действий: бой → награда → улучшение → новый бой
Индикатор	знак, по которому видно состояние объекта: цветной шар над ежом, шкала здоровья
Инвентарь	«карман» игрока: предметы, которые он собрал
Квест	задание: игрок узнаёт цель, встречается препятствие, выполняет условие и получает исход. Квест бывает не взят, взят, выполнен или провален
Концовка	один из вариантов окончания истории
Левел-дизайн	проектирование уровней: где стены, пути, враги, подсказки и точка победы
Локация	место в игре: сад, пещера, корабль
Low-poly (низкополигональная модель)	модель из небольшого числа граней, угловатая на вид; такие быстро делать и легко показывать в браузере
Машина состояний (конечный автомат)	описание поведения как набора состояний и правил перехода между ними
Механика	правило, по которому игра отвечает на действие игрока: событие + условие → изменение мира
Моб	движущийся персонаж, которым управляет игра: монстр, животное, охранник
Модификатор	в Blender: правило, которое меняет форму модели (Mirror, Array), не трогая исходную сетку
Нарратив	то, как игра рассказывает историю: сюжет, диалоги, записки, сама окружающая обстановка
NPC	неигровой персонаж: им управляет не игрок, а программа
Отладка	поиск и исправление ошибок в программе
Петля обратной связи	действие игрока → отклик игры → понимание → новое действие
Плейтест	проверка игры на живых игроках: смотрим, где они путаются и скучают
Полезная игра	игра с полезным содержанием: помогает учиться или говорить о важном
ПРИМС	программирование расширенных иерархических машин состояний – визуальный язык, на котором в «Урсуле» задают поведение

Слово	Что значит
Прототип	быстрая черновая версия игры, чтобы проверить, работает ли идея
Проходимость	можно ли пройти сквозь объект. Закрытые ворота непроходимы, трава – проходима
Путь игрока	порядок, в котором игрок встречает механики, испытания, врагов и персонажей
Развилка	место в сюжете или на карте, где игрок выбирает, куда идти дальше
Референс	картинка-образец, по которой рисуют или моделируют
Реиграбельность	желание пройти игру ещё раз – например, чтобы увидеть другую концовку
Сеттинг	мир игры: время, место, законы (средневековое фэнтези, космос, современный город)
Событие (в ПРИМС)	то, что произошло и может изменить поведение: игрок подошёл, кнопка нажата
Состояние	режим, в котором объект находится сейчас и делает только определённые действия
Текстура	картинка, которую «натягивают» на поверхность модели
Триггер	срабатывание по событию: игрок вошёл в область – что-то случилось
Узел (в Ink)	именованный кусок истории, в который можно перейти
Условие	проверка «да или нет», от которой зависит, что случится дальше
Урон	сколько здоровья снимает удар или опасность
Флажок (флаг)	переменная «да или нет»: ключ взят, ёж накормлен, ворота открыты
Экспорт	сохранить модель в формате, который понимает другая программа, например GLB для игры
Эскиз	быстрый рисунок от руки, чтобы показать идею
HP	очки здоровья (от английского hit points)
GLB	формат файла 3D-модели вместе с материалами; его понимает «Урсула»
GraphML	формат файла, в котором «Кибериада» сохраняет диаграмму поведения

Шпаргалка: всё «Запомни» из глав

Глава 1. Игры и жанры: язык геймдизайнера

Запомни. Цель – чего хочешь. Правила – что можно и что за это будет. Препятствие – что мешает. Обратная связь – как игра показывает результат. Если в задаче просят «сопоставить часть игры с описанием», спроси себя про каждую фразу: «Она про то, чего добиться, как устроен мир, что мешает или как игра ответила?»

Запомни. Жанр определяют по главному действию игрока. Тема и картинка не в счёт: игра про ферму бывает и симулятором, и стратегией. Если признаков двух жанров поровну, перед тобой смесь – и тогда в задаче обычно просят отметить все подходящие.

Запомни. Проверка на механику: разложи фразу на «событие → условие → изменение мира». Разложилась – механика. Не разложилась – перед тобой сеттинг, сюжет или атмосфера. А в инвентаре неполная стопка занимает целую ячейку, поэтому число ячеек – это деление с округлением вверх.

Запомни. Квест – и жанр, и задание. Шаги задания: выдача → цель → препятствие → условие → исход. Состояний у задания четыре: не взят, взят, выполнен, провален, – и в каждый момент только одно. Свободное дело можно вставить в ряд из трёх дел четырьмя способами, в ряд из четырёх – пятью.

Запомни. Прочитай, в каком виде нужен ответ, **до** того, как начнёшь решать. «Все верные» – по возрастанию. «Сопоставь» – по порядку букв. Лишний пробел или запятая могут сделать верный ответ неверным.

Глава 2. Геймдизайн: механики, баланс, документ игры

Запомни. Цикл – кольцо: действие → награда → новая цель → снова действие. Выпало любое звено – кольцо рвётся, и игрок уходит. Короткие циклы (секунды) вложены в длинные (уровень, вся игра).

Запомни. Удары, ходы и порции обмена бывают только целыми – округляй вверх, чтобы хватило. Минус $p\%$ – умножить на $(1 - p/100)$, плюс $p\%$ – на $(1 + p/100)$. Проценты подряд перемножают, а не складывают.

Запомни. От первого числа ряда до n -го – $n - 1$ шаг, как и от 1-го уровня до n -го – $n - 1$ переход. Шаг «плюс»: первое $+(n - 1) \cdot$ прибавка. Шаг «умножить»: первое $\cdot$ множитель в степени $n - 1$. Сумму надёжнее считать таблицей со строкой «всего».

Запомни. Время = недостающее : доход, округлить вверх. Если доход дан «за действие», сначала находим число действий (вверх), потом время. Улучшение окупается за «цена : прибавка» минут.

Запомни. Плейтест проводят на тех, кто игру ещё не видел. Смотри и записывай, а не объясняй. Долю ушедших считают от дошедших до этапа, а не от всех.

Глава 3. Левел-дизайн: карты и пути

Запомни. Координаты $(x; y)$: сначала столбец, потом строка. Без диагоналей шагов столько, сколько разница по x плюс разница по y . С диагоналями – сколько большая из двух разниц. Клетки в метры – умножением на размер клетки.

Запомни. Каждая команда движения меняет одну координату на единицу. Чтобы узнать, где окажется робот, проще всего сложить сдвиги: сколько всего раз вправо минус сколько раз влево – это изменение x ; вверх минус вниз – изменение y . А вот врежется ли он в стену, по сумме не узнать: тут придётся пройти путь клетка за клеткой.

Запомни. Волна: старт – 0, соседи – 1, их соседи – 2... Число в финише – длина кратчайшего пути. Путь – обратным ходом, по числам на 1 меньше. Клетку, где число уже стоит, не переписываем.

Запомни. Граф: комната – вершина, проход – ребро, над ребром длина. Кратчайший путь – метками: закрепляй комнату с самым маленьким числом и обновляй соседей. Ключи – по кругам: куда можно дойти → какие ключи там → какие двери они открывают.

Запомни. Только вправо и вверх: клетка = слева + снизу. На нижнем и левом краю – по 1, но только до первой запрещённой клетки. Запрещённая клетка – 0. Если выбор идёт по шагам, варианты шагов перемножают: так считают путь через обязательную клетку.

Запомни. Игрок идёт туда, где свет, где виден ориентир и куда ведут «хлебные крошки». Одинаковые коридоры, темнота везде и подсказки текстом каждые десять секунд – признаки слабого уровня.

Глава 4. Логика, выборы и случайность

Запомни. Строки – кто, столбцы – где. Подсказка с «не» – крестик. Одна свободная клетка в строке – точка. Точка закрывает свой столбец для всех остальных.

Запомни. Предположи – проверь – ищи противоречие. Противоречие значит, что предположение неверно. «А говорит, что Б лжец» – они разные. «А говорит, что Б рыцарь» – они одинаковые.

Запомни. И – нужны обе части. ИЛИ – хватит одной, обе тоже годятся. НЕ – наоборот. Порядок: скобки, НЕ, И, ИЛИ.

Запомни. Выборы один за другим – умножай. Разные ветки – складывай. k развилок по 2 варианта без схождений – 2^k концовок. Ветки сходятся – рисуй карту и пиши у каждого места сумму чисел, из которых в него пришли.

Запомни. Шанс = подходящие исходы : все исходы. $20\% = 0,2 = 1/5 = \text{«1 из 5»}$ – в среднем, а не каждый пятый раз. Все исходы вместе – 100% . Независимые события сразу – шансы умножаются. На двух кубиках 36 исходов, чаще всего выпадает сумма 7.

Запомни. Прошлые неудачи не меняют шанс следующей попытки, если в правилах нет счётчика. «Хотя бы раз за n попыток» считают через «ни разу»: $1 - (1 - p)^n$. Даже много попыток не дают 100% .

Глава 5. Машины состояний: как думает NPC

Запомни. Машина состояний – это прямоугольники-состояния, стрелки-переходы и события на стрелках. Объект всегда находится только в одном состоянии. Стрелка от чёрной точки показывает, с чего всё начинается.

Запомни. Подпись на стрелке: событие [условие] / действие. «Вход» и «выход» срабатывают по одному разу – когда попали в состояние и когда ушли из него. Внутреннее событие действует, но состояние не меняет. Нет подходящей стрелки – событие пропало.

Запомни. Поведение NPC – его состояния и события между ними. То, что не меняет режим (дошёл до точки, купил зелье), записывают внутренним событием. Событие приходит только от включённого модуля: не включил поиск – «игрок обнаружен» не придёт никогда.

Запомни. Трассируй по таблице: было – событие – что произошло – стало. Каждое событие – отдельная строка, даже если оно пропало. Если машина ходит по кругу, отбрось целые круги делением с остатком.

Запомни. Недостижимое – не попасть. Тупик – не выйти. Лишний переход – никогда не сработает. Тупик бывает задуманным (конец игры), недостижимое состояние – почти всегда ошибка.

Запомни. Стрелка из рамки родителя действует из любого дочернего состояния. Стрелка в рамку ведёт в начальное дочернее. Общее для нескольких состояний событие – повод сложить их в одного родителя.

Глава 6. Переменные, условия и поиск ошибок

Запомни. В коробке всегда одно значение, новое стирает старое. $\text{монеты} \leftarrow \text{монеты} + 5$ читай справа налево: достань, прибавь, положи обратно. Без начального значения переменную не используют.

Запомни. Условие верно – выполняется ветка `если`, неверно – ветка `иначе`. Одна из двух, не обе. «Хотя бы», «не меньше» – это $\geq$. В Python сравнивают знаком `=`, а один `=` кладёт в коробку.

Запомни. Цепочка `иначе если` останавливается на первом «да». Строгие проверки ставь выше мягких, особый случай – выше всех. В отдельных `если` без `иначе` побеждает последнее сработавшее.

Запомни. Трассировка – выполняем программу вручную, строка таблицы на каждый шаг. Первая строка – начальные значения, последняя – проверка, на которой цикл кончился. `range(1, 6)` – это 1, 2, 3, 4, 5, без шестёрки.

Запомни. Баг – «ждали одно, получили другое». Трассируй программу с числами игрока, ищи первое расхождение с описанием. Меняй одно место за раз и проверяй границу.

Запомни. Адрес ячейки – буква столбца и номер строки. Формула начинается с `=`. При копировании обычные ссылки сдвигаются на столько же строк и столбцов, а ссылки с долларом (`$B$7`) стоят на месте. Пустая ячейка в формуле – ноль.

Запомни. Функция – кусок программы с именем, параметрами и результатом. Она работает, только когда её вызвали. Числа при вызове ложатся в параметры по порядку. Модульная программа – из частей, у каждой своя работа.

Глава 7. Сюжет и диалоги: язык Ink

Запомни. Узел – кусок истории, выбор – кнопка, переход – куда она ведёт, концовка – узел, где всё кончается. Выбор значимый, если у него есть последствия, которые игрок замечает.

Запомни. У начала – 1, у узла – сумма чисел тех, кто в него ведёт. Две кнопки в один узел – две стрелки. Всего прохождений – сумма чисел у всех концовок.

Запомни. `≡ имя ≡` – начало узла, `→ имя` – переход, `→ END` – конец истории. `*` – кнопка, которая исчезает, `+` – кнопка, которая остаётся. Текст в `[ ]` виден только на кнопке. `-` – место сбора.

Запомни. `VAR` заводит переменную, `~` меняет её, `{имя}` показывает. `* {условие} [...]` – кнопка появится, только если условие верно. Условия перехода проверяются сверху вниз, срабатывает первое верное. «Равно» – это `=`.

Запомни. Концовка недостижима, если в неё не ведёт ни одна стрелка или условие на стрелке не выполнится никогда. Для проверки считай самое большое и самое маленькое значение переменной. Без ~ команда становится текстом.

Запомни. Узнаёшь героя без подписи – голос есть. Одна реплика – одна мысль, примерно до 15 слов. Выбор, который ведёт туда же, куда и другой, – не выбор.

Глава 8. 3D и визуал: Blender

Запомни. Вершина – точка, ребро – отрезок, грань (полигон) – плоский кусок. Призма: $2n, 3n, n + 2$. Пирамида: $n + 1, 2n, n + 1$. Проверка: $V - P + G = 2$. Грань с n углами – это $n - 2$ треугольника.

Запомни. Модель начинают с примитивов и собирают черновик (блокинг), детали – потом. Shift + A – добавить, G, R, S – двигать, вращать, масштабировать, Tab – режим редактирования. Куб по умолчанию – 2 м, у цилиндра 32 вершины по кругу.

Запомни. Mirror – делаешь половину, вторую строит Blender. Вершины: $2V - k$, где k – вершины на зеркале (при включённом Merge). Зеркало проходит через центр объекта: $x' = 2x_0 - x$, при центре в нуле просто $x' = -x$.

Запомни. Count – всего штук, с оригиналом. Шаг $s = k \cdot a$, длина ряда $L = a + (n - 1) \cdot s$. Вершины умножаются на Count. Модификаторы в стопке работают сверху вниз, копии перемножаются. Объект после модификаторов – всё равно один.

Запомни. 1 единица Blender = 1 м, Z – вверх. Размер = размер сетки · Scale; куб по умолчанию – 2 м, поэтому нужный Scale – это нужный размер, делённый на 2. Масштаб применяют (Ctrl + A → Scale). В игру – файл GLB с применёнными модификаторами.

Запомни. Пиксель – одна клетка одного цвета. Разрешение – ширина × высота в пикселях; всего пикселей – произведение. Вдвое больше по каждой стороне – вчетверо больше пикселей. Формат (PNG, JPG) – как упакован файл, разрешение (1920 × 1080) – сколько в нём пикселей. SVG – вектор.

Запомни. Эскиз – идея на бумаге, референсы – виды спереди и сбоку для моделирования. Силуэт проверяют заливкой чёрным: героя должно быть видно по форме, а не только по цвету. Персонажа моделируют в Т-позе.

Где что найти

Таблица С.17

Что	Где
Правила олимпиады, сферы, сроки	junior.ntcontest.ru
Личный кабинет участника	my.ntcontest.ru
Платформа отбора	orbita.education (заходить кнопкой из личного кабинета)
Курс сферы «Игротех» (за тест курса +5 баллов к отбору)	games.kruzhok.org («Мастерская игр «Урсула»»), вход через «Талант»
Все курсы «Урсулы»: программирование на ПРИМС, Ink, Blender, «Создание 3D-игры в Урсуле»	games.kruzhok.org/courses/all
Редактор игр «Урсула», свои проекты	games.kruzhok.org , личный кабинет (вход через «Талант»)
Лаборатория полезных игр: конкурсы, вебинары, материалы для наставников	games.kruzhok.org/laboratory
Программа сферы и список литературы (там же брошюра «Защита пасеки» про ПРИМС)	junior.ntcontest.ru → «Подготовка» → «Для наставников» → «Образовательная программа»
Памятки для Новосибирской области	nto.kvantorium54.ru
Blender	blender.org
Ink и редактор Inky (на английском)	inklestudios.com/ink
Журнал «Юный киберфизик»	телеграм-канал t.me/journal_cyberphys

Об авторе-составителе



Артём Осиновский

Методист и разработчик детского технопарка «Кванториум» (Новосибирск). Придумывает образовательные программы и сам делает для них цифровые инструменты: платформу смены «Миссия на Луну» для фестиваля «Технопром-2026», сайт с памятками НТО Юниоры для Новосибирской области. Эту книгу собрал для школьников области, чтобы к отбору

можно было подготовиться самостоятельно – даже без кружка рядом с домом.

Нашли ошибку – пишите на osinovsky.aa@kvantorium54.ru (рабочая почта): сфера, версия книги и страница. По другим вопросам и проектам – личная: artem081033@gmail.com.