

УДК 794:004.946  
ББК 77.563  
Б89

Б89 **Крис Брэдфилд**

Godot 4. Разработка игр. Создайте пять кросс-платформенных 2D- и 3D-игр с помощью одного из самых мощных игровых движков с открытым исходным кодом / пер. с англ. Е. В. Шевчук – М.: ДМК Пресс, 2026. – 256 с.: ил.

**ISBN 978-5-93700-437-6**

Новейшая версия открытого игрового движка Godot 4 предлагает огромное количество функций и возможностей и является достойной альтернативой дорогим коммерческим решениям. Для новичков это удобный и эффективный способ освоить основы разработки игр. Опытным дизайнерам Godot предлагает мощный, гибкий и настраиваемый инструмент для воплощения творческих идей. В основе книги лежит проектный подход. Вы последовательно создадите пять игровых проектов, а дополнительные ресурсы помогут глубже разобраться в принципах создания игр в Godot.

Книга предназначена всем, кто хочет научиться создавать игры на современном игровом движке. Приветствуется наличие навыков программирования на любом языке.

УДК 794:004.946  
ББК 77.563

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-80461-040-4 (англ.)  
ISBN 978-5-93700-437-6 (рус.)

Copyright © 2023 Packt Publishing  
© Оформление, перевод, издание,  
ДМК Пресс, 2026

# Оглавление

<b>Об авторе</b> .....	<b>11</b>
<b>О рецензенте</b> .....	<b>12</b>
<b>Предисловие</b> .....	<b>13</b>
<b>Глава 1. Введение в Godot 4.0</b> .....	<b>16</b>
Общие рекомендации .....	16
Как учиться эффективно.....	17
Что такое игровой движок?.....	18
Что такое Godot? .....	19
Загрузка Godot .....	20
Альтернативные способы загрузки .....	21
Обзор интерфейса Godot.....	22
Диспетчер проектов.....	22
Окно редактора .....	24
Ноды и сцены.....	27
Создание скриптов в Godot.....	28
О языке GDScript .....	28
Итоги главы.....	29
<b>Глава 2. Coin Dash – создание первой 2D-игры</b> .....	<b>30</b>
Технические требования.....	31
Настройка проекта .....	31
Векторы и 2D-координатные системы .....	33
Часть 1 – Сцена игрока .....	35
Создание сцены.....	35
Анимация спрайтов .....	36
Форма столкновений .....	39
Скрипт поведения игрока.....	40
Управление движением игрока.....	41
Переключение анимации .....	43
Управление началом и завершением движения игрока .....	44
Подготовка к обработке столкновений .....	44
Часть 2 – Сцена монеты.....	47
Настройка нод .....	47
Скрипт монеты.....	48
Часть 3 – Главная сцена .....	49
Настройка нод .....	49
Скрипт главной сцены .....	50
Подготовка к новой игре .....	50
Часть 4 – Пользовательский интерфейс.....	52
Настройка нод .....	52
Настройка сообщений.....	53
Отображение счета и времени .....	54
Динамическое обновление интерфейса с помощью GDScript.....	55

Завершение игры .....	56
Подключение HUD к Main .....	56
Часть 5 – Завершающие штрихи .....	58
Визуальные эффекты .....	58
Звук .....	59
Усиления .....	59
Анимация монет .....	62
Препятствия .....	62
Итоги главы .....	64
<b>Глава 3. Space Rocks – создание классической 2D-аркады с физикой</b>	<b>65</b>
Технические требования .....	66
Настройка проекта .....	66
Физика твердых тел .....	67
Корабль игрока .....	69
Настройка тела и физики .....	70
Конечные автоматы .....	72
Управление кораблем игрока .....	74
Обработка выхода за границы игровой области .....	76
Реализация стрельбы .....	77
Создание астероидов .....	81
Настройка сцены .....	81
Астероиды разного размера .....	81
Создание экземпляров астероидов .....	82
Взрывающиеся астероиды .....	84
Создание пользовательского интерфейса .....	89
Компоновка .....	89
Скрипт пользовательского интерфейса .....	91
Код интерфейса главной сцены .....	93
Код игрока .....	94
Завершение игры .....	96
Обнаружение столкновений твердых тел .....	97
Пауза в игре .....	97
Противники .....	98
Движение по контуру .....	99
Сцена противника .....	101
Движение противника .....	101
Генерация противников .....	102
Стрельба и столкновения .....	103
Щит игрока .....	106
Звуковые и визуальные эффекты .....	109
Звуки и музыка .....	109
Частицы .....	110
След противника .....	113
Итоги главы .....	113
<b>Глава 4. Jungle Jump – создание 2D-платформера с бегом и прыжками</b>	<b>115</b>
Технические требования .....	116

Настройка проекта .....	116
Знакомство с кинематическими телами .....	118
Реакция на столкновение .....	118
Создание сцены игрока.....	119
Слои и маски столкновений .....	120
Об AnimationPlayer.....	120
Анимации .....	121
Форма столкновений .....	122
Завершение сцены игрока.....	123
Состояния игрока.....	123
Скрипт игрока .....	124
Движение игрока.....	125
Здоровье игрока .....	128
Предметы для сбора .....	129
Настройка сцены.....	129
Скрипт предметов.....	130
Проектирование уровня.....	131
Использование TileMap .....	132
Проектирование первого уровня .....	135
Добавление ловушек .....	137
Прокручивающийся фон .....	138
Добавление противников .....	141
Настройка сцены.....	141
Скрипт противника.....	142
Нанесение урона противнику .....	143
Скрипт игрока .....	144
Создание интерфейса.....	147
Настройка сцены.....	147
Скрипт HUD .....	148
Подключение HUD.....	149
Стартовый экран.....	150
Настройка сцены.....	150
Настройка главной сцены .....	151
Переход между уровнями .....	153
Сцена двери .....	153
Настройки экрана.....	155
Завершающие штрихи .....	155
Звуковые эффекты .....	155
Двойной прыжок.....	155
Частицы пыли.....	156
Лестницы .....	158
Движущиеся платформы .....	160
Итоги главы.....	162

## **Глава 5. 3D Minigolf – погружение в 3D и создание поля**

<b>для мини-гольфа.....</b>	<b>163</b>
Технические требования.....	163
Введение в 3D.....	164
Ориентация в трехмерном пространстве.....	164

3D-редактор Godot .....	165
Добавление 3D-объектов .....	166
Глобальное и локальное пространство .....	167
Трансформации .....	168
Меши .....	168
Камеры .....	170
Настройка проекта .....	172
Создание поля .....	173
Принцип работы GridMaps .....	173
Создание первой лунки .....	174
Окружение и освещение .....	175
Добавление лунки .....	176
Создание мяча .....	177
Тестирование мяча .....	177
Улучшение физики мяча .....	177
Создание интерфейса .....	179
Создание стрелки прицеливания .....	179
Реализация интерфейса .....	181
Создание скриптов .....	183
Скрипт интерфейса .....	183
Главный скрипт .....	184
Скрипт мяча .....	186
Тестирование .....	187
Улучшение системы прицеливания .....	187
Улучшение системы прицеливания (вариант 2) .....	188
Настройка камеры .....	190
Создание полноценного поля .....	193
Визуальные эффекты .....	194
Добавление материалов .....	195
Освещение и окружение .....	197
Итоги главы .....	199
<b>Глава 6. Infinite Flyer.....</b>	<b>200</b>
Технические требования .....	201
Настройка проекта .....	201
Настройка ввода .....	201
Сцена самолета .....	202
Формы столкновений .....	203
Скрипт самолета .....	204
Создание игрового мира .....	207
Объекты мира .....	207
Чанки .....	210
Главная сцена .....	214
Создание новых чанков .....	216
Усложнение .....	217
Столкновения .....	219
Топливо и счет .....	221
Стартовый экран .....	224
Звуковое оформление .....	225

Сохранение рекорда .....	225
О расположении файлов .....	226
Доступ к файлам .....	226
Дополнительные функции .....	228
Итоги главы .....	228
<b>Глава 7. Дальнейшие шаги и дополнительные ресурсы .....</b>	<b>229</b>
Работа со встроенной документацией Godot .....	229
Чтение документации API .....	231
Контроль версий и использование Git с Godot .....	232
Использование Blender с Godot .....	233
Подсказки для импорта .....	233
Работа с файлами .blend .....	234
Экспорт проектов .....	235
Шаблоны экспорта .....	236
Пресеты экспорта .....	236
Как экспортировать проект .....	237
Экспорт для различных платформ .....	237
Введение в шейдеры .....	238
Создание 2D-шейдеров .....	240
3D-шейдеры .....	245
Дополнительные материалы .....	247
Использование других языков программирования в Godot .....	247
C# .....	247
Другие языки и GDExtension .....	249
Помощь и ресурсы сообщества .....	250
Godot Recipes .....	251
Участие в развитии Godot .....	252
Участие в разработке движка .....	252
Написание документации .....	252
Финансовая поддержка .....	253
Итоги главы .....	253
<b>Заключение .....</b>	<b>254</b>
<b>Предметный указатель .....</b>	<b>255</b>

# Об авторе



**Крис Брэдфилд** (Chris Bradfield) работает в сфере интернет-технологий более 25 лет. Он участвовал в разработке ряда успешных MMO- и социальных онлайн-игр для издателей в Южной Корее и США. За время карьеры в игровой индустрии он успел поработать гейм-дизайнером, разработчиком, продуктовым менеджером и руководителем по развитию бизнеса.

В 2012 году он увлекся преподаванием и основал KidsCanCode, где обучает программированию учащихся средних и старших классов. Кроме того, Крис создает видео- и текстовые учебные материалы для студентов, изучающих разработку игр по всему миру.

Выражаем благодарность Кенни Влэгелсу (Kenney Vleugels, @kenneyNL) за 3D-тайлы для мини-гольфа, skorpio – за арт космического корабля, Луису Зуно (Luis Zuno, @ansimuz) и авторам Sunny Land за их работу.

# Предисловие

Эта книга – введение в игровой движок Godot и его новейшую версию, Godot 4. Godot 4 предлагает огромное количество функций и возможностей, представляя собой серьезную альтернативу дорогим коммерческим решениям. Для начинающих это удобный и эффективный способ освоить основы разработки игр. Опытным же разработчикам Godot предлагает мощный, гибкий и настраиваемый инструмент для воплощения творческих идей.

В основе книги лежит проектный подход. Вы последовательно создадите пять игровых проектов, а дополнительные ресурсы помогут глубже разобраться в принципах создания игр в Godot.

## Для кого эта книга

Эта книга предназначена для всех, кто хочет научиться создавать игры на современном игровом движке. Она станет полезным ресурсом как для начинающих, так и для опытных разработчиков. Предварительный опыт программирования приветствуется.

## Содержание книги

Эта книга предлагает проектно-ориентированный подход к изучению Godot. Пять игровых проектов выстроены по принципу «от простого к сложному», где каждый следующий опирается на концепции, изученные в предыдущем.

*Глава 1 «Введение в Godot 4.0».* Знакомит с концепцией игровых движков в целом и Godot в частности. Здесь вы найдете инструкции по установке Godot, обзор интерфейса редактора и рекомендации по эффективной работе с книгой.

*Глава 2 «Coin Dash – создание первой 2D-игры».* Небольшой 2D-проект, в котором вы научитесь создавать сцены, работать с системой нод, ориентироваться в редакторе и писать первые скрипты на GDScript.

*Глава 3 «Space Rocks – создание классической 2D-аркады с физикой».* Посвящена работе с физическими телами и созданию космического шутера в стиле Asteroids.

*Глава 4 «Jungle Jump – создание 2D-платформера с бегом и прыжками».* Создание платформера с боковой прокруткой в духе Super Mario Bros. Вы познакомитесь с кинематическими телами, анимационными состояниями и дизайном уровней с помощью тайловых карт.

*Глава 5 «3D Minigolf – погружение в 3D и создание поля для мини-гольфа».* Переносит освоенные 2D-концепции в трехмерное пространство. Вы поработаете с 3D-мешами, освещением и управлением камерой.

*Глава 6 «Infinite Flyer».* Посвящена дальнейшему изучению 3D-разработки. Вы узнаете о создании динамического контента, процедурной генерации и других продвинутых техниках.

*Глава 7 «Дальнейшие шаги и дополнительные ресурсы».* Предлагает темы для самостоятельного изучения после прохождения основных проектов, а также полезные ссылки и советы для расширения ваших навыков.

## Как извлечь из книги максимум пользы

Для лучшего понимания примеров кода желательно иметь общие представления о программировании. Идеально, если это будет опыт работы с современным языком с динамической типизацией, таким как Python или JavaScript. Если же вы только начинаете свой путь в программировании, будет полезно сначала изучить основы.

Godot запустится на большинстве современных компьютеров под управлением Windows, macOS или Linux.

Если вы читаете электронную версию книги, советуем набирать код вручную или использовать файлы из репозитория книги на GitHub (ссылка приведена в следующем разделе). Это поможет избежать возможных ошибок из-за копирования и вставки кода.

## Исходный код проектов

Исходный код для проектов доступен в репозитории GitHub по адресу <https://github.com/PacktPublishing/Godot-4-Game-Development-Projects-Second-Edition>. Репозиторий поддерживается в актуальном состоянии.

## Загрузка цветных изображений

Все скриншоты и схемы, использованные в книге, также доступны в виде отдельного цветного PDF-файла. Загрузить его можно по адресу: <https://packt.link/1Y2hq>.

## Принятые обозначения

В книге используется ряд принятых текстовых соглашений.

Код в тексте обозначает элементы кода, имена таблиц баз данных, папок, файлов, расширения файлов, пути, примеры URL-адресов и ввод пользователя. Например: «В Godot 4 появилась возможность импортировать файлы `.blend` напрямую в проект».

Блоки кода оформляются так:

```
shader_type canvas_item;

void fragment() {
    // Place fragment code here.
}
```

**Полужирным** выделяются названия элементов интерфейса: кнопки, вкладки, свойства, пункты меню. Пример: «Первым свойством является **Shader** (Шейдер), где можно выбрать **New Shader** (Новый шейдер). При этом откроется панель **Create Shader** (Создание шейдера)».

*Курсивом* выделяются ключевые термины при первом упоминании и важные смысловые акценты. Пример: «Важным понятием является *конечный автомат (Finite State Machine – FSM)*».

---

Советы и важные примечания выделяются таким образом.

---

# Глава 1

## Введение в Godot 4.0

Разработка игр – увлекательное и благодарное дело, будь то профессиональная карьера или хобби. Сейчас идеальное время для старта: современные инструменты и языки программирования позволяют создавать качественные игры и распространять их по всему миру легче, чем когда-либо прежде. Если вы читаете эту книгу, значит, вы готовы сделать первый шаг к созданию игры (или игр) своей мечты.

Эта книга представляет собой введение в игровой движок Godot, а именно в его версию 4.0, выпущенную в 2023 году. Godot 4 предлагает множество нововведений, что делает его мощной альтернативой платным коммерческим движкам. Начинающим разработчикам он позволяет легко освоить фундаментальные концепции разработки игр, а опытным служит гибким, расширяемым и *открытым* инструментом для воплощения творческих идей.

В основе книги лежит проектный подход: вы освоите ключевые аспекты движка, создавая пять игровых проектов. Это поможет сформировать глубокое понимание как теории разработки игр, так и ее практического применения в Godot. Вы познакомитесь с архитектурой движка и освоите важные техники для ваших будущих проектов.

В этой главе мы рассмотрим следующие темы:

- общие рекомендации по работе с книгой;
- что такое игровой движок;
- знакомство с Godot;
- загрузка и установка Godot;
- обзор пользовательского интерфейса Godot;
- ноды и сцены;
- написание скриптов в Godot.

### Общие рекомендации

В этом разделе собраны советы, основанные на педагогическом опыте автора. Имейте их в виду, работая с книгой, особенно если вы только начинаете свой путь в программировании.

Старайтесь выполнять проекты из книги в предложенном порядке. Содержание глав взаимосвязано: более поздние опираются на темы, рассмотренные в ранних. Если что-то забыли, просто вернитесь и повторите тему. Ваша главная цель – понять материал, а не поставить рекорд скорости.

Материала много, поэтому не отчаивайтесь, если что-то не дается сразу. Помните, что, как и в любом ремесле, для достижения мастерства нужны годы практики. Стать экспертом за одну ночь невозможно. Ключ к освоению сложных тем – повторение. Чем больше вы работаете с Godot, тем привычнее и понятнее он становится. Попробуйте перечитать одну из первых глав, когда закончите книгу. Вы удивитесь, насколько углубилось ваше понимание.

Если вы читаете электронную версию, не поддавайтесь искушению копировать код. Набирая его вручную, вы заставляете мозг работать активнее – подобно тому, как ведение конспекта на лекции помогает усвоить материал лучше, чем простое слушание, даже если вы никогда не вернетесь к этим записям. Для тех, кто печатает медленно, это также станет отличной тренировкой скорости. Коротко говоря, вы программист, поэтому привыкайте набирать код.

Одна из самых частых ошибок начинающих – браться за проекты, которые им не по силам. На старте очень важно ограничивать масштаб. Гораздо полезнее и результативнее завершить две-три небольшие игры, чем забросить один неподъемный проект.

Пять игр в этой книге строго следуют этой стратегии. Их объем сознательно ограничен – не только чтобы уместить материал в рамки урока, но и чтобы фокус оставался на отработке базовых навыков. В процессе разработки у вас наверняка появятся идеи для новых функций и игровых механик: «Что, если добавить улучшения для космического корабля?» или «А если научить персонажа отталкиваться от стен?».

Идеи – это прекрасно, но помните, что первостепенная задача – выполнить основной проект. Записывайте интересные мысли, но не позволяйте им вас отвлекать. Разработчики называют это «раздуванием функционала» – бесконечным добавлением функций, которое является самой распространенной причиной незавершенных проектов. Не попадайте в эту ловушку.

И наконец, не пренебрегайте перерывами. Не пытайтесь выполнить проект и тем более прочесть всю книгу за пару подходов. После изучения каждой новой концепции, и особенно после прочтения главы, дайте себе время осмыслить информацию. Это не только улучшит усвоение материала, но и сделает процесс обучения комфортнее и приятнее.

## Как учиться эффективно

Вот секрет, который поможет извлечь из книги максимум пользы и надежно закрепить навыки: завершив работу над главой и создав проект, удалите его и начните с нуля. Теперь постарайтесь воссоздать проект, не загляды-

вая в книгу. Если зайдете в тупик, обратитесь к книге, а затем продолжите самостоятельно. Если почувствуете себя уверенно, попробуйте добавить в игру что-то свое, например изменить геймплей или добавить новую механику.

Продельвайте это с каждым проектом, и вы заметите, что обращаетесь к книге все реже. Если вы сможете самостоятельно воссоздать все проекты из этой книги, значит, вы готовы к созданию собственных уникальных игр.

Помните об этих советах, читая следующие разделы. Далее вы узнаете, что такое игровые движки и почему разработчики выбирают их для создания игр.

## Что такое игровой движок?

Разработка игр – это сложный процесс, требующий широкого спектра знаний. Кроме того, для создания современной игры необходим солидный технологический фундамент. Представьте, что перед программированием вам пришлось бы сначала собрать компьютер и написать операционную систему. Примерно так выглядела бы разработка игр, если бы каждый раз приходилось начинать с чистого листа.

К тому же существует ряд общих задач, присущих каждой игре. Например, вне зависимости от жанра в любом проекте необходимо выводить изображение на экран. Если код для этого уже написан, гораздо эффективнее использовать его повторно, чем создавать заново. Именно для этого и существуют игровые фреймворки и движки.

*Игровой фреймворк (game framework)* – это набор библиотек со вспомогательным кодом для создания базовых компонентов игры. Однако он редко предоставляет все необходимые модули, и для их интеграции часто приходится писать много дополнительного кода. Вот почему создание игры на фреймворке обычно занимает больше времени, чем с использованием полноценного движка.

*Игровой движок (game engine)* – это комплекс инструментов и технологий, который значительно упрощает создание игр, избавляя разработчиков от необходимости изобретать велосипед для каждого нового проекта. Он предоставляет набор стандартных функций, реализация которых с нуля потребовала бы огромных затрат времени и сил.

Типичный игровой движок обычно включает следующие возможности:

- **рендеринг (2D и 3D)**: рендеринг – это процесс отображения игры на экране. Эффективный рендеринг подразумевает поддержку современных графических процессоров (*GPU – Graphics Processing Unit*), работу с высокими разрешениями, а также реализацию эффектов, таких как освещение, перспектива и настройка областей просмотра (вьюпортов), – и все это при условии высокой частоты кадров;
- **физика**: хотя это и базовая потребность, создание точного и стабильного физического движка – сложнейшая задача. Почти любой

игре нужна система обнаружения столкновений, а часто и продвинутая физическая симуляция. При этом найдется мало разработчиков, кто станет писать подобный код с нуля, особенно без соответствующего опыта;

- **кросс-платформенность:** на современном рынке большинство разработчиков стремятся выпускать игры сразу для нескольких платформ: ПК, консолей, мобильных устройств и/или веб-браузеров. Движок решает эту задачу, предлагая единый процесс экспорта для публикации на разных платформах без необходимости переписывать код и поддерживать несколько его версий;
- **единая среда разработки:** работая в одном знакомом интерфейсе над разными играми, разработчику не приходится осваивать новый рабочий процесс для каждого проекта.

Помимо перечисленного, в движок встроены инструменты для реализации сетевых функций, управления изображениями, звуком и анимацией, а также средства отладки и многое другое. Часто движки позволяют импортировать контент, созданный в других программах (например, для анимации или 3D-моделирования).

Использование движка позволяет разработчикам сосредоточиться на самой игре, а не на разработке ее базовой архитектуры. Небольшим командам и независимым разработчикам это дает возможность выпустить игру за год вместо трех или вовсе не забросить проект.

Сегодня на рынке представлены десятки популярных игровых движков, таких как Unity, Unreal Engine, GameMaker Studio и др. Важно понимать, что большинство из них – коммерческие продукты. Они могут не требовать первоначальных вложений, однако, если ваша игра начнет приносить доход, с большой вероятностью придется платить лицензионные отчисления и/или роялти. Вне зависимости от выбора движка первым делом внимательно изучите лицензионное соглашение, чтобы четко понимать условия использования движка и возможные издержки.

С другой стороны, существуют и некоммерческие движки с *открытым исходным кодом*, такие как Godot, которому и посвящена эта книга.

## Что такое Godot?

Godot – это полнофункциональный современный игровой движок, предоставляющий все описанные возможности и даже больше. Он распространяется бесплатно и имеет открытый исходный код под лицензией MIT. Это означает отсутствие взносов, скрытых издержек и роялти. Все, что вы создаете на Godot, на 100 % принадлежит вам. Для многих разработчиков это ключевое преимущество.

Если вы впервые сталкиваетесь с концепцией *открытого исходного кода* (*open-source*), это может показаться необычным. Однако Godot, как

и ядро Linux или браузер Firefox, не коммерческий продукт корпорации. Его создает и развивает сообщество энтузиастов, которые вкладывают время, знания и силы в улучшение движка, поиск ошибок и написание документации.

Для разработчиков Godot предлагает ряд важных преимуществ. Отсутствие коммерческой лицензии дает полную свободу в выборе способа и площадок для распространения игры. В отличие от многих коммерческих движков, которые могут ограничивать типы проектов или требовать приобретения дорогостоящей лицензии для определенных жанров (например, азартных игр), Godot не накладывает таких ограничений.

Открытый исходный код Godot обеспечивает прозрачность, недоступную в коммерческих движках. Это также означает, что, если возможностей движка недостаточно для ваших задач, вы можете модифицировать его исходный код и добавить нужный функционал. Кроме того, открытый код значительно упрощает отладку сложных проектов, поскольку вы получаете полный доступ к внутренней архитектуре движка.

Вы также можете напрямую участвовать в развитии Godot. Подробнее о том, как внести вклад в разработку движка, читайте в главе 7.

Теперь, когда вы понимаете, что такое Godot и как он помогает в создании игр, самое время приступить к работе. В следующем разделе вы узнаете, как скачать Godot и подготовить его к работе на вашем компьютере.

## Загрузка Godot

Чтобы скачать последнюю версию Godot, перейдите на официальный сайт <https://godotengine.org/> и нажмите кнопку **Download Latest** (Скачать последнюю версию). Эта книга написана для версии 4.0. Если вы скачаете версию с дополнительными цифрами в названии (например, 4.0.3) – не беспокойтесь, это актуальный релиз с исправлениями.

На странице загрузки представлены два варианта: стандартная версия и версия .NET. Версия .NET предназначена для разработки на языке C#. Выбирайте стандартную сборку, если не планируете его использовать. Все проекты в этой книге созданы в стандартной версии движка.



Рис. 1.1. Страница загрузки Godot

Чтобы установить приложение Godot, распакуйте скачанный архив в удобное место, например в папку Programs или Applications. Запустите Godot двойным кликом. После этого откроется окно **Project Manager** (Диспетчер проектов), с которым вы познакомитесь в следующем разделе.

## Альтернативные способы загрузки

Помимо загрузки с официального сайта, Godot можно установить на компьютер и другими способами. Функциональность движка при этом не меняется. Ниже приведены альтернативные варианты:

- **Steam:** если у вас есть учетная запись в Steam, можете установить Godot через его десктопный клиент. Найдите движок в магазине Steam и следуйте инструкциям по установке. Запускать Godot можно будет прямо из библиотеки Steam;

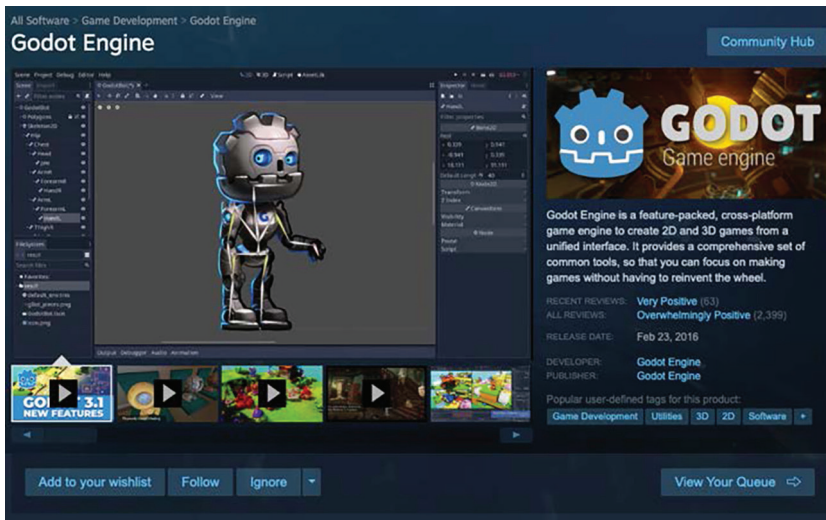


Рис. 1.2. Godot в Steam

- **itch.io:** также Godot доступен на популярной платформе itch.io. Это онлайн-магазин для независимых разработчиков игр и авторов контента. Найдите Godot через поиск на сайте и загрузите его по предоставленным ссылкам;
- **менеджеры пакетов:** если вы пользуетесь одним из приведенных ниже менеджеров пакетов, Godot можно установить стандартным для вашей системы способом. Подробности смотрите в документации вашего пакетного менеджера. Godot доступен:
  - в Homebrew (macOS),
  - Scoop (Windows),
  - Snap (Linux).

## Версии для Android и веб-браузеров

Также на сайте доступны для загрузки версии Godot для Android и веб-браузеров. На момент написания книги они помечены как «экспериментальные» и могут быть нестабильными или иметь ограниченную функциональность. Для работы и особенно для обучения рекомендуется использовать десктопную версию Godot (Windows, macOS или Linux).

Поздравляем! Теперь Godot установлен на вашем компьютере. В следующем разделе вас ждет обзор интерфейса редактора. Вы узнаете, как устроены основные окна и панели, с которыми предстоит работать.

## Обзор интерфейса Godot

Как и большинство игровых движков, Godot предоставляет единую среду разработки, в которой можно работать с кодом, графикой, звуком и другими ресурсами проекта. В этом разделе вы познакомитесь с ее основными элементами. Постарайтесь запомнить ключевые термины из этого раздела, ведь они будут использоваться на протяжении всей книги при описании работы в редакторе.

### Диспетчер проектов

Окно **Project Manager** – это стартовый экран, который вы видите при запуске Godot.

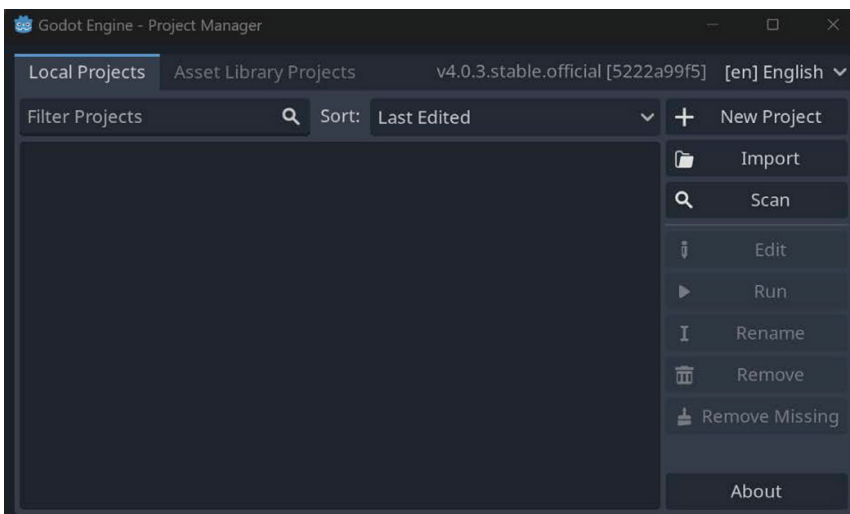


Рис. 1.3. Окно Project Manager

## Первый запуск Godot

При первом запуске у вас еще не будет созданных проектов. Скорее всего, появится всплывающее окно с предложением ознакомиться с официальной библиотекой примеров. Нажмите кнопку **Cancel** (Отмена), чтобы закрыть его. После этого появится окно **Project Manager**, как показано на рис. 1.3.

В этом окне отображается список ваших проектов Godot. Вы можете выбрать любой и нажать **Run** (Запустить) или **Edit** (Редактировать). Чтобы создать новый проект, нажмите **New Project** (Новый проект).

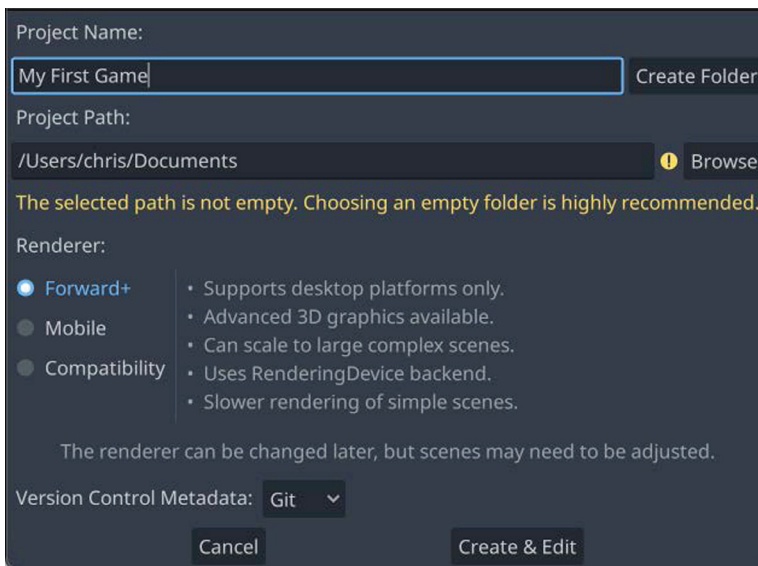


Рис. 1.4. Настройки нового проекта

Здесь можно задать имя проекта и выбрать папку для его сохранения. Важно: каждый проект Godot – это отдельная папка на вашем компьютере, и все файлы проекта должны находиться именно в ней. Такой подход упрощает распространение: достаточно заархивировать эту папку, и любой другой пользователь Godot сможет открыть проект без потери данных.

## Рендерер

При создании нового проекта вам также будет предложено выбрать **Renderer** (Рендерер). Доступны три варианта: от продвинутого режима для современных ПК с мощной видеокартой до оптимизированных для

мобильных устройств и устаревших компьютеров. Этот параметр можно изменить позже в настройках проекта, поэтому оставьте выбор по умолчанию. Если в будущем вы планируете создавать игры для мобильных платформ или специфического железа, обратитесь к официальной документации Godot. В ней содержится вся информация о производительности и настройках рендеринга. Подробные сведения и ссылки вы найдете в главе 7.

### **Выбор имени для проекта**

Давая имя новому проекту, следуйте нескольким простым правилам – это поможет избежать проблем в будущем. Давайте проекту описательное название. Например, Арена битвы волшебников – гораздо более предпочтительный вариант имени, чем *Игра № 2*. Спустя время вы уже не вспомните, что скрывалось под этим номером, поэтому название должно быть как можно более информативным.

Следите за именем папки проекта и файлов внутри нее. Некоторые операционные системы чувствительны к регистру символов и различают написание `My_Game` и `my_game`, а другие – нет. Это может привести к проблемам при переносе проекта с одного компьютера на другой. По этой причине многие разработчики используют стандартизированную схему именования, например не ставят пробелы в именах файлов, а разделяют слова символом `_`. Независимо от выбранной схемы главное – соблюдать единообразие.

После создания папки проекта нажмите кнопку **Create & Edit** (Создать и редактировать), чтобы открыть новый проект в редакторе. Попробуйте сделать это сейчас и создайте проект с именем `test_project`.

---

### **Консольное окно**

При запуске Godot в Windows дополнительно открывается консольное окно, в котором отображаются служебные сообщения, предупреждения и ошибки, генерируемые движком и проектом. В macOS и Linux такого отдельного окна по умолчанию нет, однако аналогичный вывод можно увидеть, если запустить Godot из терминала.

---

## **Окно редактора**

На рис. 1.5 представлен скриншот главного окна редактора Godot. Именно здесь вы будете проводить основную часть времени при создании проектов. Интерфейс редактора разделен на несколько областей (секций), каждая из которых отвечает за определенные функции. Подробное описание и названия каждой области приведены под рис. 1.5.

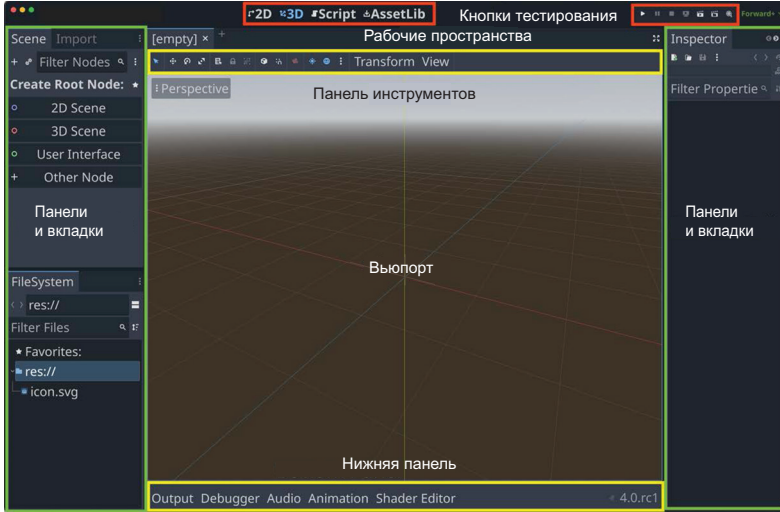


Рис. 1.5. Окно редактора Godot

Центральную часть окна редактора занимает *вьюпорт* (*viewport*). Здесь вы будете видеть и редактировать элементы вашей игры в процессе работы.

В верхней центральной части окна расположены *рабочие пространства* (*workspaces*), которые позволяют переключаться между различными аспектами разработки. Доступны режимы **2D**, **3D** и **Script**. **AssetLib** (Библиотека ресурсов) – это раздел, где можно загружать дополнения и примеры проектов от сообщества Godot. Подробнее об использовании библиотеки ресурсов можно узнать в главе 7.

На рис. 1.6 представлена *панель инструментов* (*toolbar*) для текущего рабочего пространства. Ее иконки меняются в зависимости от типа объекта, с которым вы работаете.



Рис. 1.6. Иконки панели инструментов

В правом верхнем углу находится *область тестирования* (*playtest area*) с кнопками для запуска игры и взаимодействия с ней во время выполнения.



Рис. 1.7. Кнопки области тестирования

Слева и справа расположены *панели* (*docks*) с *вкладками* (*tabs*), которые нужны для просмотра и выбора игровых объектов, а также для настройки их свойств. В нижней части левой панели находится вкладка **FileSystem**

(Файловая система), в которой отображаются все файлы проекта. Щелкая по папкам, вы можете открывать их и просматривать содержимое. Все ресурсы в проекте расположены относительно корневой папки проекта, которая в Godot обозначается специальным путем: `res://`. Например, путь к файлу может выглядеть так: `res://player/player.tscn`. Это означает, что файл `player.tscn` находится в папке `player`.

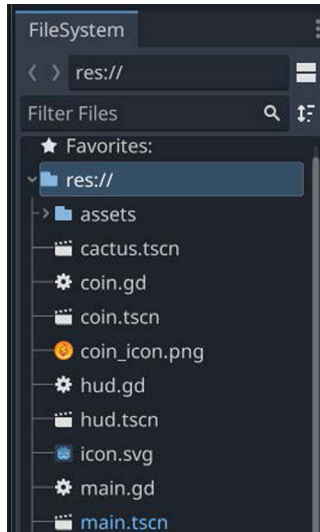


Рис. 1.8. Вкладка **FileSystem**

В верхней части левой панели расположена вкладка **Scene** (Сцена). Здесь отображается текущая сцена, над которой вы работаете во вьюпорте (подробнее о сценах – после рис. 1.9):

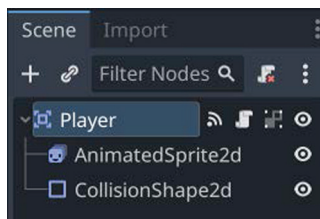


Рис. 1.9. Вкладка **Scene** (Сцена)

Справа находится панель **Inspector**, где можно просматривать и изменять свойства игровых объектов.

Работая над проектами из этой книги, вы детально изучите функциональность этих элементов и освоите навигацию в интерфейсе.

К этому моменту вы уже должны уверенно ориентироваться в структуре окна редактора Godot и запомнить названия его основных частей. Они будут постоянно использоваться в последующих главах. Вы стали на шаг

ближе к завершению введения и началу разработки своей первой игры. Однако перед этим необходимо разобраться с элементами, представленными на рис. 1.9. Это ноды, и в следующем разделе вы узнаете, что это такое.

## Ноды и сцены

*Ноды (nodes)* – это основные строительные блоки для создания игр в Godot. Каждая нода – это отдельный объект, который предоставляет определенную игровую функцию: одни отображают изображение, другие проигрывают анимацию, третьи представляют 3D-модель. Ноды обладают набором свойств для настройки их поведения. Выбор нод для вашего проекта зависит от необходимой функциональности. Это модульная система, предназначенная для гибкой сборки игровых объектов.

Ноды организуются в виде дерева, где одни ноды являются дочерними относительно других. Одна нода может иметь множество дочерних элементов, но только один родительский. Такое иерархическое дерево и называется сценой.

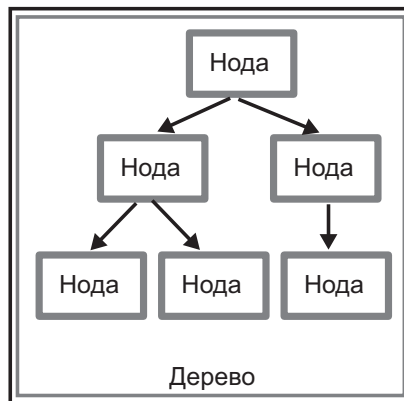


Рис. 1.10. Древоподобная структура нод

*Сцены (scenes)* в Godot служат для создания и организации различных игровых объектов. Например, вы можете создать сцену игрока, которая включает все ноды и скрипты, отвечающие за управление персонажем и его отображение. Отдельно можно создать сцену карты игры, определив на ней препятствия, которые игроку предстоит преодолеть, и предметы, которые игроку нужно будет собирать. Затем отдельные сцены объединяются в единую игровую сцену.

Хотя ноды обладают множеством свойств и функций, их поведение можно существенно расширить с помощью скриптов. Скрипт, прикрепленный к ноде, позволяет наделить ее дополнительной логикой, выходящей за рамки стандартных возможностей. Например, нода `Sprite2D` отображает изображение, но, чтобы оно двигалось или исчезало по клику, потребуется написать скрипт, реализующий такое поведение.

Ноды – это мощные инструменты, и их понимание является ключевым навыком для создания игровых объектов в Godot. Однако сами по себе ноды предоставляют лишь базовые возможности. Определение игровой логики и правил, по которым живут объекты вашего мира, остается за вами. В следующем разделе вы узнаете, как это делается с помощью скриптового языка Godot.

## Создание скриптов в Godot

Godot поддерживает два официальных языка: *GDScript* и *C#*. *GDScript* – это встроенный язык, тесно интегрированный с движком и наиболее простой в освоении. Для тех, кто предпочитает *C#*, доступна отдельная версия сборки движка.

При этом сам Godot написан на C++. Это позволяет опытным разработчикам при необходимости расширять функциональность движка напрямую, создавая модули на C++ для достижения максимального контроля и производительности. Подробнее об использовании других языков и расширении движка вы можете узнать в главе 7.

Во всех проектах этой книги используется *GDScript*. Для большинства задач это оптимальный выбор: язык создан для быстрой разработки и глубоко интегрирован с *программным интерфейсом приложения (API – Application Programming Interface) Godot*.

### О языке GDScript

Синтаксис *GDScript* во многом близок к Python. Если вы уже знакомы с Python, то и *GDScript* покажется вам знакомым. Тем, кто хорошо владеет другим языком с динамической типизацией (например, JavaScript), освоить *GDScript* также будет несложно. Python часто рекомендуют как первый язык для начинающих, и *GDScript*, созданный на его основе, унаследовал эту доступность.

Эта книга предполагает, что у вас уже есть базовый опыт программирования. Тем, кто никогда не писал код, будет немного сложнее. Освоение игрового движка – непростая задача, а одновременное изучение основ программирования – серьезный вызов. Если код в этой книге вызовет у вас затруднения, возможно, стоит сначала пройти вводный курс по Python или JavaScript.

Как и Python, *GDScript* – язык с динамической типизацией. Это означает, что тип переменной определяется автоматически, а для обозначения блоков кода используются отступы. Главное преимущество *GDScript* для разработки игр заключается в его тесной интеграции с движком Godot, что позволяет писать меньше кода, ускоряя разработку и снижая вероятность ошибок.

Вот небольшой пример скрипта на *GDScript*, который заставляет спрайт перемещаться слева направо с заданной скоростью:

```
extends Sprite2D
var speed = 200

func _ready():
    position = Vector2(100, 100)
func _process(delta):
    position.x += speed * delta
```

Если вы уже работали с языками высокого уровня, этот код покажется знакомым. Не волнуйтесь, если сейчас он не до конца понятен. В следующих главах вы напишете много кода, и каждый шаг будет подробным описанием.

## Итоги главы

В этой главе вы узнали, что такое игровой движок, и познакомились с Godot. Кроме того, вы уже скачали и запустили Godot на своем компьютере!

Вы освоили ключевые термины интерфейса редактора, которые будут использоваться на протяжении всей книги, а также узнали о нодах и сценах – строительных блоках любой игры в Godot.

Вы получили советы по работе с проектами из книги и общему подходу к разработке игр. Если в процессе возникнут трудности, всегда можно вернуться к разделу «Общие рекомендации». Материала для изучения много, и это нормально, если не все будет понятно сразу. Вас ждет создание пяти разных игр, каждая из которых поможет вам лучше понять принципы работы Godot.

Теперь вы готовы перейти к следующей главе и приступить к созданию своей первой игры в Godot.

# Глава 2

## Coin Dash – создание первой 2D-игры

В этом проекте вы шаг за шагом создадите свою первую игру на движке Godot. Вы познакомитесь с интерфейсом редактора, принципами организации проекта и создадите простую 2D-игру, используя ключевые типы нод.

### Почему следует начинать с 2D?

Если говорить кратко, создание 3D-игр сложнее. Однако базовые принципы работы движка Godot, которые вам предстоит освоить, во многом схожи для 2D и 3D. Поэтому оптимальной стратегией будет сначала уверенно освоить рабочий процесс на примере 2D, и тогда переход к 3D пройдет гораздо проще. Возможность поработать с 3D у вас будет в следующих главах этой книги.

Не пропускайте эту главу, даже если у вас уже есть опыт в разработке игр. Хотя многие концепции могут быть вам знакомы, этот проект покажет ключевые возможности и подходы к проектированию в Godot, что критически важно для дальнейшего изучения движка.

Игра, которую вы создадите в этой главе, называется Coin Dash. В ней вы будете управлять персонажем, который должен собрать как можно больше монет за отведенное время. Вот как выглядит готовый проект:

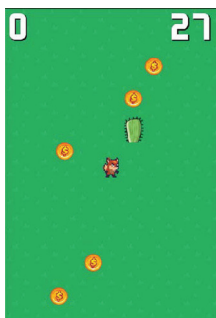


Рис. 2.1. Готовая игра

В этой главе мы рассмотрим следующие темы:

- настройка нового проекта;
- создание анимаций персонажа;
- управление движениями персонажа;
- использование ноды Area2D для обнаружения столкновений;
- использование нод Control для отображения информации;
- взаимодействие между игровыми объектами с помощью сигналов.

## Технические требования

Скачайте игровые ресурсы по адресу ниже и распакуйте их в папку вашего нового проекта: <https://github.com/PacktPublishing/Godot-4-Game-Development-Projects-Second-Edition/tree/main/Downloads>.

Полный код из этой главы также доступен на GitHub по адресу <https://github.com/PacktPublishing/Godot-4-Game-Development-Projects-Second-Edition/tree/main/Chapter02%20-%20Coin%20Dash>.

## Настройка проекта

Запустите Godot и в окне **Project Manager** нажмите кнопку **+ New Project** (+ Новый проект).

Сначала создайте папку проекта. В поле **Project Name** (Имя проекта) введите **Coin Dash** и нажмите **Create Folder** (Создать папку). Важно хранить каждый проект в отдельной папке, чтобы файлы не перемешивались. Затем нажмите **Create & Edit**, и проект откроется в редакторе Godot.

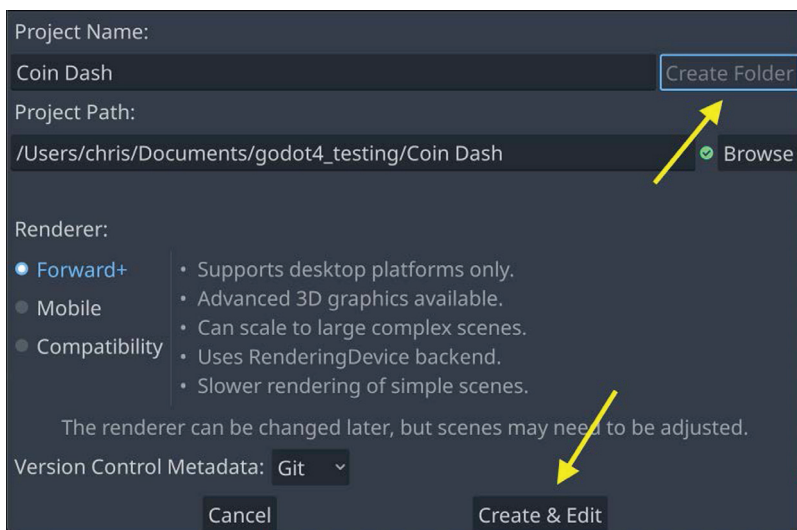


Рис. 2.2. Окно создания нового проекта

В этом проекте вы создадите три независимые сцены: персонажа игрока, монету и интерфейс для отображения счета и времени. Затем вы объедините их в главной сцене игры (подробнее в главе 1). В больших проектах рекомендуется создавать отдельные папки для ресурсов и скриптов каждой сцены. Для нашей небольшой игры можно сохранять все сцены и скрипты непосредственно в корневой папке проекта, которая в Godot обозначается как `res://`, где `res` – сокращение от `resources` (ресурсы). Все файлы вашего проекта будут находиться относительно этого пути `res://`. Файлы проекта отображаются на панели **FileSystem** в левом нижнем углу редактора. Поскольку проект новый, он будет пустым, за исключением файла `icon.svg` (иконка Godot).

Арт-ресурсы и звуки для игры – так называемые *аксеты* (*assets*) – можно скачать в виде ZIP-архива по адресу <https://github.com/PacktPublishing/Godot-Engine-Game-Development-Projects-Second-Edition/tree/main/Downloads>. Распакуйте содержимое архива в корневую папку вашего нового проекта.

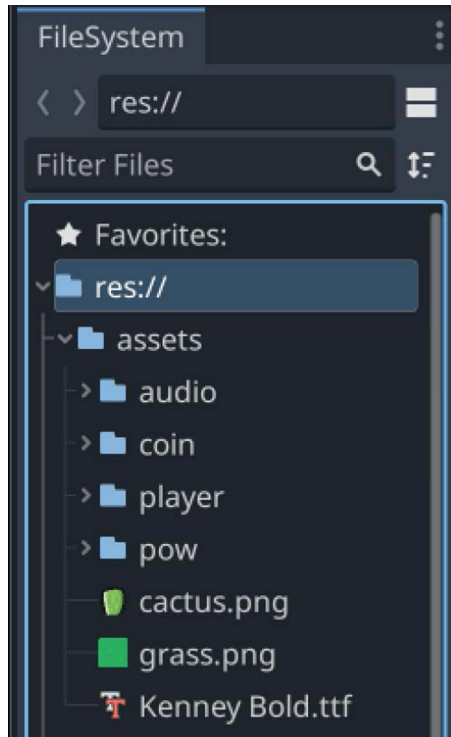


Рис. 2.3. Вкладка **FileSystem**

Например, изображения монеты находятся по пути `res://assets/coin/`.

Поскольку игра будет отображаться в портретном режиме (высота больше ширины), необходимо настроить игровое окно.

В верхнем меню выберите пункт **Project > Project Settings** (Проект > Настройки проекта). Откроется окно настроек:

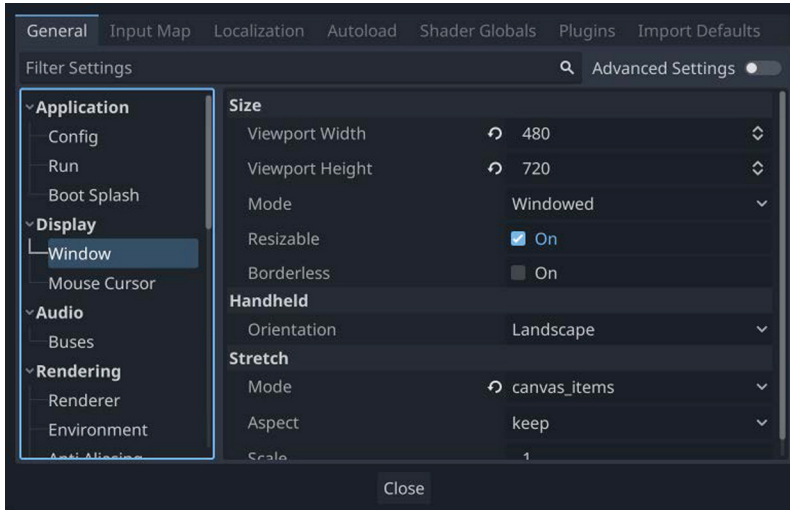


Рис. 2.4. Окно **Project Settings**

Найдите раздел **Display > Window** (Отображение > Окно) и установите **Viewport Width** (Ширина вьюпорта) равным 480 и **Viewport Height** (Высота вьюпорта) равным 720, как показано на рис. 2.4. В этом же разделе, в подразделе **Stretch** (Растяжение), задайте для параметра **Mode** (Режим) значение `canvas_items`, а для **Aspect** (Соотношение сторон) значение `keep`. Это обеспечит корректное масштабирование всех элементов игры при изменении размера окна без искажений. Для фиксированного размера окна можно снять флажок **Resizable** (Изменяемый размер) в подразделе **Size** (Размер).

Отлично! Проект настроен, и теперь можно приступать к созданию игры. Поскольку объекты в ней будут находиться и перемещаться в 2D-пространстве, важно разобраться в системе 2D-координат и ее применении в вашей игре. Именно этому посвящен следующий раздел.

## Векторы и 2D-координатные системы

Этот раздел представляет собой краткое введение в 2D-координатные системы и векторную математику в контексте разработки игр. Векторная математика – важнейший инструмент для разработчика игр. Для более глубокого погружения в тему вы можете пройти курс линейной алгебры от Khan Academy (<https://www.khanacademy.org/math/linear-algebra>).

В двумерном пространстве положение точек определяется с помощью декартовых координат. Оно задается парой чисел, например (4, 3), где первое число соответствует положению по оси  $x$ , а второе – по оси  $y$ . Таким образом можно описать любую точку на плоскости.

В Godot, как и во многих других инструментах компьютерной графики, принята ориентация осей, где  $x$  направлена вправо, а  $y$  вниз:

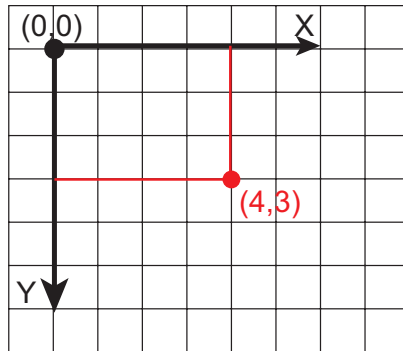


Рис. 2.5. 2D-координатная система

### В школе учили не так!

Если вы новичок в компьютерной графике или разработке игр, вам может показаться непривычным, что положительная ось  $y$  направлена вниз, а не вверх, как вас, вероятно, учили на уроках математики. Однако такая ориентация широко распространена в приложениях для компьютерной графики.

### Векторы

Позицию  $(4, 3)$  также можно рассматривать как смещение относительно точки  $(0, 0)$ , или начала координат. Представьте себе стрелку, направленную из начала координат в эту точку:

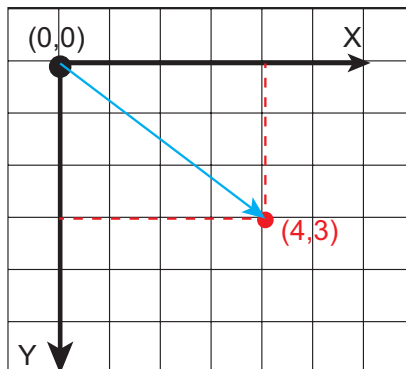


Рис. 2.6. 2D-вектор

Эта стрелка и является *вектором*. Он содержит важную информацию: положение точки, его *длину* ( $m$ ) и *угол* ( $\theta$ ) относительно оси  $x$ . Такой вектор точнее называть *позиционным вектором*, поскольку он описывает конкрет-

ное положение в пространстве. В более широком смысле векторы могут представлять собой перемещение, ускорение или любую другую величину, обладающую размером и направлением.

В Godot векторы применяются повсеместно, и вы будете использовать их в каждом проекте этой книги.

Теперь, когда вы понимаете основы 2D-координатного пространства и роль векторов в позиционировании объектов, можно переходить к практике. В следующем разделе вы создадите игрового персонажа и примените новые знания для управления его движением.

## Часть 1 – Сцена игрока

Первой сценой, которую вы создадите, будет объект игрока. Одно из преимуществ отдельной сцены для персонажа (и других объектов) – возможность тестировать ее независимо, еще до создания остальных частей игры. Такой модульный подход становится еще ценнее по мере роста проекта. Изолированное хранение объектов упрощает их отладку, модификацию и даже полную замену, не затрагивая другие компоненты игры. Кроме того, это обеспечивает переиспользуемость: сцену игрока можно будет перенести в другую игру, и она будет работать так же.

Сцена игрока в вашем проекте должна выполнять следующие действия:

- отображать персонажа и его анимации;
- реагировать на ввод пользователя, перемещая персонажа;
- обнаруживать столкновения с другими игровыми объектами, такими как монеты или препятствия.

### Создание сцены

Чтобы создать новую сцену, нажмите **Add > Create a New Node** (Добавить > Создать новую ноду) (сочетание клавиш **Ctrl+A**) и выберите **Area2D**. Затем щелкните по имени ноды и измените его на **Player**. Сохраните сцену, выбрав в меню **Scene > Save Scene** (Сцена > Сохранить сцену) (сочетание клавиш **Ctrl+S**).

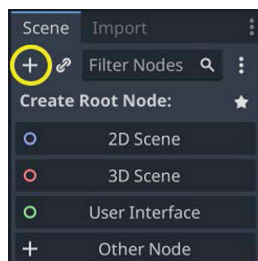


Рис. 2.7. Добавление ноды

На вкладке **FileSystem** появился файл `player.tscn`. В Godot все сохраненные сцены используют расширение `.tscn` – это собственный формат файлов движка. Буква *t* в названии означает *текст* (*text*), поскольку такие файлы являются текстовыми. При желании вы можете открыть его во внешнем текстовом редакторе и посмотреть, но редактировать вручную не рекомендуется, поскольку это может привести к повреждению файла.

Вы создали *корневую ноду* сцены. Именно она определяет основную функциональность объекта. Мы выбрали `Area2D`, потому что это 2D-нода, которая может перемещаться в 2D-пространстве и обнаруживать пересечения с другими объектами. Это позволит обрабатывать сбор монет и взаимодействие с другими объектами. Выбор типа ноды для конкретного игрового объекта – это ваше первое важное решение при проектировании игровых объектов.

Прежде чем добавлять дочерние ноды, рекомендуется заблокировать родительскую ноду, чтобы случайно не сместить или не масштабировать их. Выберите ноду `Player` и наведите курсор на значок рядом с замком – **Group Selected Node(s)** (Группировать выбранные ноды):

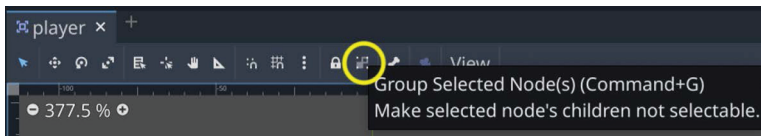


Рис. 2.8. Включение группировки нод

В подсказке указано: **Make selected node's children not selectable** (Сделать дочерние элементы выбранной ноды недоступными для выбора). Эта функция – именно то, что нужно, она помогает избежать случайных ошибок. Нажмите кнопку, и такой же значок появится рядом с именем ноды игрока:

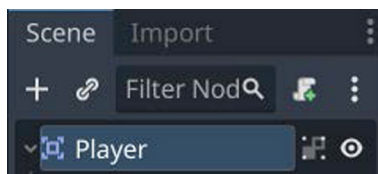


Рис. 2.9. Иконка группировки нод

Эту процедуру рекомендуется выполнять при создании каждой новой сцены. Смещение или масштабирование дочерних нод может привести к ошибкам, которые будет сложно исправить.

## Анимация спрайтов

Нода `Area2D` предназначена для обнаружения пересечений с другими объектами, но не отображает графику. Чтобы показать изображение персонажа, добавьте к ноде игрока дочернюю ноду `AnimatedSprite2D`. Она будет отвечать за отображение и анимации персонажа. Обратите внимание на значок

предупреждения рядом с нодой. `AnimatedSprite2D` требует ресурс `SpriteFrames`, в котором хранятся кадры анимации. Чтобы создать его, найдите свойство **Frames** (Кадры) в окне **Inspector** и нажмите на `<empty>`. В появившемся меню выберите **New SpriteFrames** (Новый `SpriteFrames`):

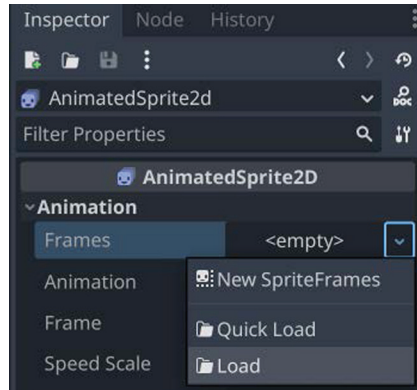


Рис. 2.10. Добавление ресурса `SpriteFrames`

Далее нажмите на появившуюся надпись `SpriteFrames`, и в нижней части экрана откроется новая панель:

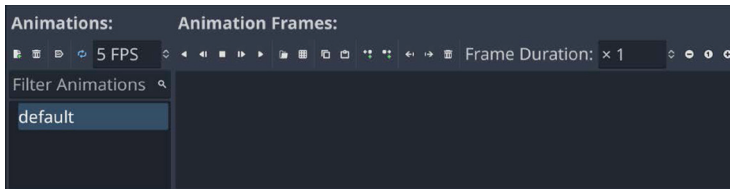


Рис. 2.11. Панель `SpriteFrames`

Слева расположен список анимаций. Щелкните по анимации `default` и переименуйте ее в `run`. Затем нажмите кнопку **Add Animation** (Добавить анимацию) и создайте вторую анимацию с именем `idle`, а третью – `hurt`.

На панели **FileSystem** найдите изображения `run`, `idle` и `hurt` в папке `res://assets/player/` и перетащите их на соответствующие анимации:

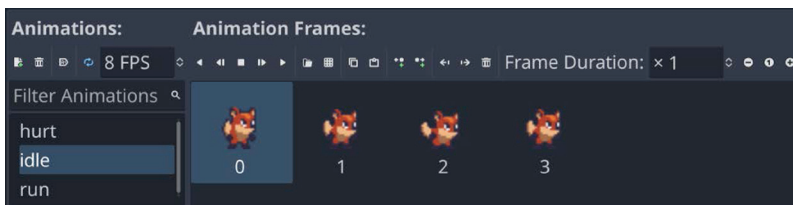


Рис. 2.12. Настройка анимаций персонажа

Скорость каждой новой анимации по умолчанию равна 5 кадрам в секунду. Этого мало, поэтому для каждой анимации установите параметр **Speed** (Скорость) равным 8.

Чтобы увидеть анимации в действии, нажмите кнопку (▶) **Play** (Воспроизвести). Список анимаций появится в окне **Inspector** в выпадающем меню свойства **Animation** (Анимация). Выберите одну из них для просмотра:

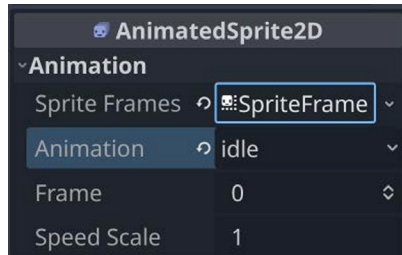


Рис. 2.13. Свойство **Animation**

Вы можете настроить автоматическое воспроизведение анимации при загрузке сцены. Для этого выберите в списке анимацию `idle` и нажмите кнопку **Autoplay on Load** (Автовоспроизведение при загрузке):

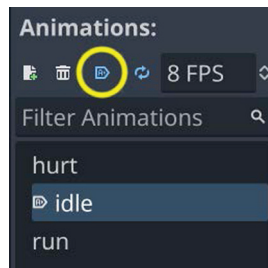


Рис. 2.14. Настройка автовоспроизведения анимации

Позже вы напишете код, который будет переключать анимации в зависимости от действий игрока. Но сначала необходимо завершить настройку остальных нод персонажа.

Изображение персонажа выглядит слишком мелким. Чтобы увеличить его, задайте для параметра **Scale** (Масштаб) ноды `AnimatedSprite2D` значение (2, 2). Оно находится в разделе **Transform** (Трансформация) окна **Inspector**.



Рис. 2.15. Настройка **Scale**