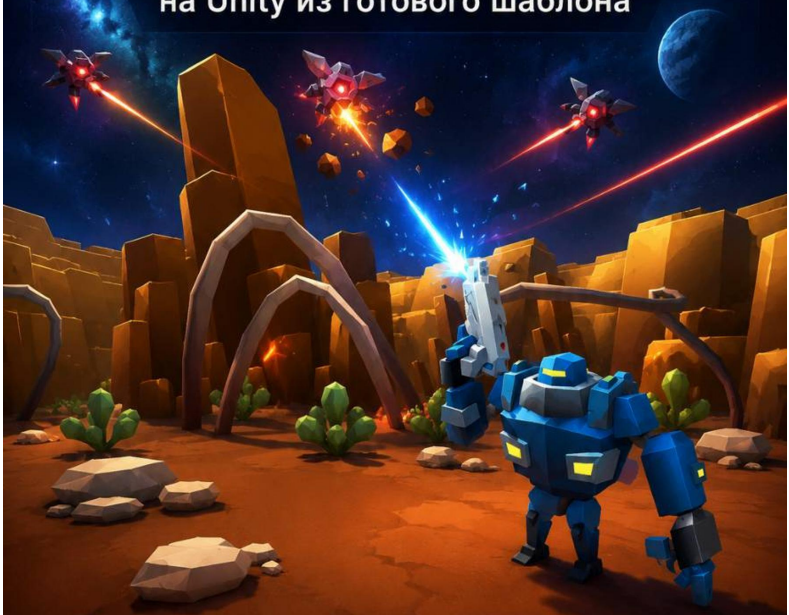
ШУТЕР КАК КОНСТРУКТОР

Собери и опубликуй свою игру на Unity из готового шаблона



Дмитрий Денисов

Дмитрий Денисов

Шутер как конструктор. Собери и опубликуй свою игру на Unity из готового шаблона

https://litres.ru/74106596
SelfPub; 2026

Аннотация

«Хочу делать игры» и «моя игра опубликована, и в неё играют» — между этими фразами не годы учёбы, а одна эта книга.

Никакого движка «с нуля». Вы берёте готовый шаблон шутера на Unity — и уже в первой главе собираете браузерную версию и выкладываете её на Яндекс.Играх, получая ссылку, которой можно поделиться с друзьями. Результат — с первых страниц, а не «когда-нибудь потом».

Дальше — шаг за шагом: разбираете проект изнутри, делаете свои уровни, врагов и предметы, учитесь мыслить как гейм-дизайнер (эту главу помог собрать гейм-дизайнер студии Targem Games), добавляете таблицы лидеров, облачные сохранения и достижения, занимаетесь балансом и монетизацией. А в финале — о суперсиле современного разработчика: как ускорять всё нейросетями — код, арт, 3D-модели, анимацию и идеи.

Практикум для начинающих: минимум теории, максимум результата. Для тех, кто устал «изучать геймдев» и хочет наконец сделать и выпустить свою игру.

Содержание

Шутер как конструктор. Собери и опубликуй свою игру на Unity из готового шаблона

Введение

Эта книга — о коротком пути от «хочу делать игры» до «моя игра опубликована, и в неё играют другие люди».

Есть распространённый миф: чтобы сделать игру, нужно сначала годами учить программирование, написать всё с нуля и только потом, может быть, увидеть результат. На самом деле индустрия так не работает. Веб-разработчики опираются на фреймворки, мобильные — на готовые SDK, а в геймдеве мы используем движки, плагины и шаблоны. Игра по своей сути — это **конструктор**: вы собираете проект из деталей — сцен, префабов, материалов, скриптов, анимаций и интерфейса.

Поэтому мы пойдём не «с нуля», а от готового шаблона шутера. В коробке уже лежат «мотор и колёса»: контроллер игрока, камера, система урона, враги, интерфейс, менеджер уровней. Наша задача — не изобретать велосипед, а быстро

получить работающий прототип и заняться самым интересным: своей механикой, своими уровнями, своим стилем — и публикацией.

Рис. В.1. Коллаж: стартовое окно игры и кадр геймплея шутера.

И это не обещание из далёкого будущего: игру, которую мы соберём, **уже можно открыть в браузере прямо сейчас** — https://yandex.ru/games/app/223547?draft=true&lang=ru. Поиграйте пару минут, чтобы увидеть цель, а потом вернёмся и соберём её вместе.

За семь глав мы пройдём весь цикл:

В **первой главе** — быстрый старт: за 15 минут проведём игру от проекта до публикации на Яндекс.Играх и получим

рабочую ссылку, которой можно поделиться с друзьями. Это «фишка» книги — результат с первых страниц.

Во **второй** установим Unity и импортируем шаблон, а заодно научимся играть осознанно — как разработчик, а не как обычный игрок.

В **третьей** разберём проект изнутри и начнём вносить изменения: соберём свой уровень, расставим врагов и предметы, добавим второй уровень и настроим переходы между сценами.

В **четвёртой** научимся думать как гейм-дизайнер: поставим цель, найдём уникальность игры (USP), разберём дизайн-документ — эту главу помог подготовить приглашённый гейм-дизайнер студии Targem Games.

В **пятой** соберём собственную браузерную версию и опубликуем её на Яндекс.Играх с помощью плагина YG2.

В **шестой** займёмся развитием: лидерборды, облачные сохранения, достижения, новые механики, баланс и монетизация.

А **седьмая глава** — бонус о суперсиле современного разработчика: как ускорять всю работу нейросетями — код, арт, 3D-модели, анимации и генерацию идей.

Игру, которую мы соберём, будем называть **DeepGalacticShooter**. Но это лишь рабочее название — к концу книги у вас будет свой проект, со своими уровнями и идеями.

Совет: эта книга — ускоренный спутник моего пошагово-

го практикума по игре **«Ловец драконов» (Dragon Picker)**, где игра собирается с самого нуля. Если вы хотите сначала понять все основы «по кирпичику», начните с неё. А если хотите быстрый результат и публикацию — вы держите в руках нужную книгу.

Не нужно бояться, что это «копипаста» или «не по-настоящему». Так работает индустрия. Вы получаете прочный каркас, а наполняете его собственными идеями. Поехали!

Игровая индустрия сегодня

Игровая индустрия развивается с неимоверной скоростью, и разработка игр становится всё более доступным и востребованным направлением. Прогресс технологий и растущая популярность видеоигр, охватывающих самые разные жанры и аудитории, открывают перед разработчиками огромные возможности. Игры уже давно перестали быть просто развлечением — они стали частью нашей культуры, способом общения и даже средством для обучения и тренировки навыков.

Разработка игр позволяет человеку раскрыть свой творческий потенциал, ведь в процессе создаются не просто программные продукты, а целые миры. Игры объединяют в себе такие дисциплины, как программирование, дизайн, математика, физика и психология. Это помогает разработчикам расширять кругозор и развивать навыки, которые могут пригодиться и в других областях жизни.

10 лет назад разработка игры требовала от человека глубоких знаний программирования и понимания множества сложных технологий, то сегодня этот процесс стал намного проще. В первую очередь это связано с развитием игровых движков — специализированных программных платформ, позволяющих создавать игры на различных устройствах.

Современные игровые движки предлагают обширный ин-

струментарий, который автоматизирует многие процессы, такие как анимация, физика, управление освещением и тенями. Это значит, что разработчикам больше не нужно писать код для каждого элемента игры вручную, что существенно экономит время и усилия. Более того, доступность движков с открытыми инструментами и обучающими материалами делает разработку игр доступной даже для новичков без серьезного опыта в программировании.

Игровой движок Unity

Один из популярных и доступных игровых движков сегодня — это Unity. За последние годы Unity стал настоящим стандартом для разработки игр благодаря своей гибкости и мощному функционалу. Считаю основным достижением этого движка является его открытость для взаимодействия комьюнити (сообществ), большого количества материалов и ресурсов для разработки игр. Можно не быть дизайнером, а использовать в своей игре самые разные готовые asset-паки, опубликованные в библиотеке assetstore.unity.com. Итак, основные достоинства Unity заключаются в следующем:

Доступность. Unity предлагает бесплатную версию для начинающих разработчиков, что делает его идеальным выбором для старта. Вы можете создать полноценную игру без начальных финансовых вложений, используя бесплатные инструменты и ресурсы.

Мультиплатформенность. Unity поддерживает создание игр для множества платформ, таких как Windows, Mac, Android, iOS, и даже игровые консоли. Это значит, что ваша игра может охватить большую аудиторию на разных устройствах.

Сообщество и ресурсы. У Unity огромная база знаний — документация, форумы, обучающие курсы и готовые решения. Это позволяет легко найти ответы на любые вопросы

и быстро освоить новые функции движка.

Поддержка 2D и 3D игр. Независимо от того, разрабатываете ли вы 2D-игру, например, шутер или платформер, или стремитесь создать масштабный 3D-проект, Unity предоставляет все необходимые инструменты для достижения поставленных целей.

Активное развитие. Команда разработчиков Unity регулярно обновляет движок, добавляя новые функции и улучшая его производительность, что делает его актуальным и удобным инструментом для создания современных игр.

Таким образом, Unity — это мощный, удобный и доступный инструмент, с которым можно начать разработку игры даже без опыта программирования, и это одна из причин, почему разработка игр сегодня доступнее, чем когда-либо.

Что нового в Unity 6

Недавний выход Unity 6 стал важной вехой в развитии этого популярного игрового движка, предложив разработчикам множество новых возможностей и усовершенствований. Это обновление продолжает тенденцию Unity к совершенствованию как инструментов для разработки игр, так и платформы для создания приложений в сфере дополненной и виртуальной реальности, кино и анимации. Рассмотрим основные новшества и ключевые изменения, которые принес Unity 6.

Оптимизация производительности и новые инструменты рендеринга. Одним из наиболее значительных обновлений в Unity 6 стало улучшение производительности. Движок теперь лучше поддерживает высокопроизводительные проекты, особенно с большим количеством объектов на сцене. Это особенно полезно для разработчиков 3D-игр с открытыми мирами или сложными симуляциями, где каждый кадр должен быть отрендерен максимально эффективно.

Unity 6 ввел новую систему **Render Pipeline**, которая позволяет разработчикам точно настраивать процессы рендеринга в зависимости от типа проекта. Например, для простых мобильных игр можно использовать **Universal Render Pipeline** (URP), который оптимизирован для производительности, а для AAA-проектов с высокими требованиями к графике доступен **High Definition Render Pipeline** (HDRP),

который поддерживает сложное освещение, тени и шейдеры.

Поддержка мультиплатформенной разработки. Unity уже долгое время славится своей мультиплатформенной поддержкой, и в Unity 6 этот аспект был еще больше усовершенствован. Обновление включает более глубокую интеграцию с платформами нового поколения, такими как PlayStation 5, Xbox Series X/S и новыми версиями мобильных операционных систем. Улучшена поддержка устройств на базе Apple Silicon, что повышает производительность приложений на Mac с чипами M1 и M2.

Кроме того, Unity 6 сделал важный шаг вперед в области облачных технологий, предложив разработчикам инструменты для более простого создания и интеграции игр с сервисами облачного гейминга. Это делает игры доступными на большем количестве устройств и упрощает разработку мультиплеерных проектов с синхронизацией между платформами.

Новшества для виртуальной и дополненной реальности (VR и AR). Одной из самых быстро развивающихся областей в игровой индустрии является виртуальная и дополненная реальность. Unity 6 расширяет возможности разработки под **VR и AR**, добавляя нативную поддержку для всех основных платформ, таких как **Oculus**, **HTC Vive**, **HoloLens** и **Magic Leap**.

Обновленный API для дополненной реальности позволяет разработчикам создавать еще более точные и стабильные

приложения, работающие с реальным миром. Появились новые функции для отслеживания рук и улучшенная система взаимодействия с объектами виртуальной реальности, что делает пользовательский опыт более естественным.

Новые инструменты для художников и аниматоров. Unity 6 предоставляет расширенные возможности для дизайнеров и аниматоров, упрощая работу с визуальными эффектами и анимацией персонажей. Новая система **Shader Graph** была значительно улучшена, что позволяет художникам создавать сложные шейдеры без необходимости написания кода. Это особенно полезно для небольших студий, где художники могут работать напрямую с визуальными эффектами, не задействуя программистов.

Существенно улучшена система анимации **Cinemachine**, которая позволяет создавать динамические камеры и сценические планы без написания кода. Это упрощает процесс создания кинематографичных роликов и кат-сцен в играх, а также улучшает работу с анимацией персонажей.

Динамическое освещение и улучшенная физика. Unity 6 предлагает новый инструмент для динамического освещения, который значительно улучшает внешний вид сцен, особенно для игр с большим количеством подвижных объектов. Внедрение Real-Time Global Illumination позволяет освещению динамически реагировать на изменения сцены в режиме реального времени, что создает более реалистичное взаимодействие света и объектов.

Что касается физики, Unity 6 улучшил систему физического взаимодействия между объектами, добавив более точное моделирование столкновений и деформаций. Это важно для разработчиков игр с продвинутыми физическими симуляциями, таких как гоночные игры или проекты с разрушаемыми окружениями.

Улучшенные инструменты для разработки пользовательских интерфейсов (UI). Создание удобных и функциональных интерфейсов для игр — это одна из ключевых задач разработчиков, и Unity 6 предложил новые инструменты, облегчающие эту работу. В частности, новый UI Builder позволяет разработчикам создавать интерфейсы с помощью визуальных инструментов, что значительно ускоряет процесс разработки и настройки элементов UI.

Была улучшена интеграция с новыми технологиями отображения, что помогает дизайнерам быстрее работать с адаптивными интерфейсами, которые одинаково хорошо отображаются на разных разрешениях экрана, будь то мобильные устройства или 4K-дисплеи.

Интеграция с машинным обучением и искусственным интеллектом. Unity 6 продолжает развивать инструменты для работы с машинным обучением и искусственным интеллектом (AI). Платформа теперь включает улучшенную поддержку **ML-Agents**, что позволяет использовать алгоритмы машинного обучения для создания сложных и реалистичных NPC или симуляций. Эти возможности могут ис-

пользоваться не только в играх, но и в обучающих или научных приложениях, где требуется моделировать поведение сложных систем.

Новая архитектура для гибкости и масштабируемости проектов. Unity 6 вводит обновленную модульную архитектуру, которая позволяет разработчикам легче управлять проектами любого масштаба. Внедрена концепция Packages — отдельных модулей, которые можно добавлять или удалять из проекта по мере необходимости. Это упрощает работу над проектами и снижает объем ненужного кода, что повышает стабильность и производительность.

Выход Unity 6 — это важный шаг вперед для разработчиков игр, предлагающий целый спектр новых инструментов и возможностей, которые делают создание игр и интерактивных приложений еще проще и эффективнее. Улучшенная производительность, расширенные возможности работы с графикой и физикой, а также поддержка новых технологий, таких как VR, AR и машинное обучение, обеспечивают Unity статус одного из самых гибких и мощных движков на рынке. Независимо от того, создаете ли вы простую 2D-игру или амбициозный 3D-проект с продвинутой графикой, Unity 6 предоставляет все необходимые инструменты для достижения успеха.

Игры, сделанные на Unity

Многие популярные и успешные игры были разработаны на движке Unity, что подчеркивает его гибкость и мощь. Вот несколько примеров известных игр, сделанных на Unity, которые получили признание в мире:

Ori and the Blind Forest (и продолжение **Ori and the Will of the Wisps**).

Эта игра стала известной благодаря потрясающей художественной стилистике, глубокой истории и великолепному геймплею. **Ori and the Blind Forest —** это приключенческая платформенная игра, которая сочетает в себе сложные голо-

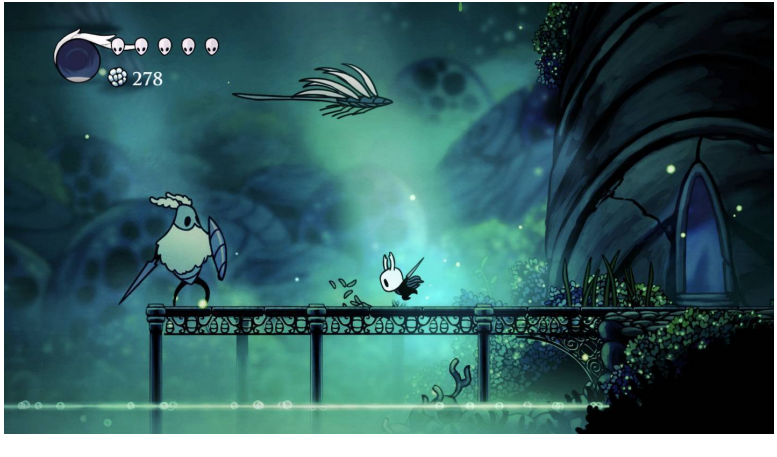
воломки, захватывающий дизайн уровней и красивую анимацию. Несмотря на то, что игра выглядит как произведение искусства, она работает на движке Unity, что доказывает его потенциал для создания визуально ошеломляющих проектов.

Cuphead (порт для Nintendo Switch)**.** Хотя оригинальная версия игры была разработана на другом движке, портирование **Cuphead** на Nintendo Switch было выполнено с помощью Unity. Это яркий пример того, как Unity позволяет адаптировать игры для новых платформ с сохранением оригинального качества. **Cuphead** привлек внимание своей уникальной графикой в стиле ретро-мультипликации 1930-х годов и хардкорным игровым процессом.

Изображение выглядит как мультфильм, Анимация,

Мультфильм, картина Автоматически созданное описание

Hollow Knight — еще один популярный представитель жанра "метроидвания", созданный на Unity. Игра привлекает внимание своим глубоким миром, сложным геймплеем и великолепной атмосферой. Она была высоко оценена как игроками, так и критиками за свое мастерство в дизайне и повествовании. Это одна из тех игр, которые показывают, что независимые разработчики могут создавать масштабные проекты на Unity.



Изображение выглядит как мультфильм, птица, Компьютерная игра, снимок экрана Автоматически созданное описание

Monument Valley. Эта инди-головоломка с уникальной графикой и иллюзиями в архитектурном стиле принесла

большую популярность благодаря своим оригинальным механикам и минималистичной эстетике.

Monument Valley была разработана на Unity и стала одной из самых успешных мобильных игр, завоевав множество наград за дизайн.

Subnautica — это игра в жанре survival с элементами открытого мира, в которой игроки исследуют подводные глубины инопланетного океана. Эта игра, разработанная на Unity, демонстрирует, как движок может поддерживать большие, открытые миры с реалистичной симуляцией физики и экосистем.

Изображение выглядит как риф, подводный, плавание, вода Автоматически созданное описание

Escape from Tarkov (частично на Unity). Хардкорный шутер с элементами симуляции и RPG, Escape from Tarkov, разрабатывается с использованием различных технологий, включая Unity для отдельных аспектов проекта. Игра привлекла внимание благодаря своей детализированной системе боя и реалистичности. Несмотря на то, что Unity не является основным движком для всего проекта, этот пример показывает его использование в сложных игровых средах.

Among Us. Игра, которая взорвала рынок в 2020 году, хотя была выпущена в 2018.

Among Us — это социальная многопользовательская игра, где игроки пытаются найти предателя среди команды. Ее простой, но увлекательный геймплей, а также кросс-платформенность (доступна на PC, мобильных устройствах и консолях) — отличный пример возможностей Unity для создания игр, которые могут стать культурным феноменом.

Rust — популярная многопользовательская онлайн-игра в жанре survival.

Изображение выглядит как одежда, на открытом воздухе, небо, оружие Автоматически созданное описание

Изначально разрабатывалась как клон DayZ, но в итоге стала самостоятельным проектом с уникальными механиками и интенсивными PvP-столкновениями. Благодаря поддержке больших карт, интерактивного окружения и высокодетализированной графики, Unity позволил разработчикам создавать сложные и большие многопользовательские миры.

Welcome Pack


вселенная
ирит-ртф

Меня зовут **Дмитрий Денисов**. Я работаю в Уральском федеральном университете, в институте радиоэлектроники и информационных технологий (ИРИТ-РТФ), руковожу лабораторией разработки игр и VR-решений, веду курсы по разработке игр и VR и делаю небольшие инди-проекты.

Несколько слов о том, где вообще учат «разработчиков игр». Речь не про онлайн-школы с громкими обещаниями из рекламы, а про высшее образование. Делать игры и интерактивные симуляторы в той или иной степени учат на ИТ-направлениях подготовки с шифрами 09.03.xx (бакалавриат) и 09.04.xx (магистратура). Стоит честно сказать: в дипломе не будет написано «гуру по разработке игр» — но именно на этих направлениях дают фундамент, без которого профессии не существует.

На уральском радиофаке (ИРИТ-РТФ), например, учась на бакалавриате, можно выбрать спецкурсы по разработке игр, пройти практику в екатеринбургской студии Targem Games или поступить в специализированную магистратуру «Инженерия мультимедийных систем». На просторах страны есть и другие вузы — учат ли там делать игры, стоит уточнять в учебных планах. Так что если после этой книги вы захотите пойти дальше и получить профильное образование — это реальный и достойный путь.

Установка Unity и первый проект (для новичков)

Этот раздел можно пропустить, если вы уже хоть раз что-то делали в Unity. Если открываете движок впервые — пройдите пошаговую инструкцию ниже: установим Unity и соберём первый простой проект, чтобы освоиться в редакторе. Уже работали в Unity? Смело переходите к Главе 1.

В современном мире разработка компьютерных игр стала одной из самых быстроразвивающихся и востребованных отраслей в IT. Среди множества инструментов для создания игр Unity выделяется как одна из самых мощных и гибких платформ. Этот игровой движок предоставляет пользователям возможность разрабатывать игры для различных платформ, начиная от мобильных устройств и заканчивая консолями, а также веб-приложениями. Благодаря своей доступности и широкому функционалу, Unity получил огромную популярность среди как начинающих разработчиков, так и профессионалов игровой индустрии.

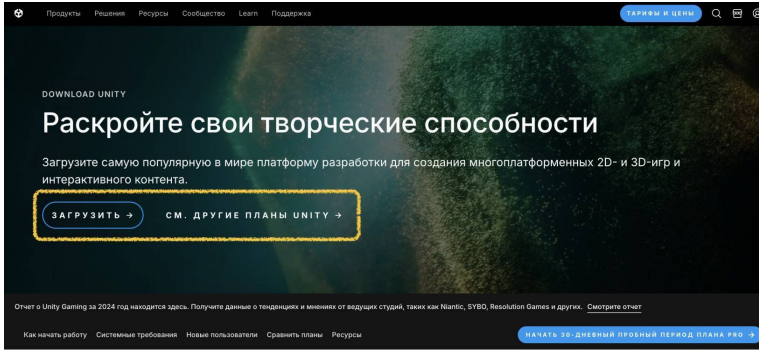
Основной задачей данного раздела является обучение начальным шагам работы с Unity и создание первого проекта. Мы рассмотрим установку Unity и Unity Hub — специального интерфейса, с помощью которого можно управлять про-

ектами и версиями редактора. Далее мы научимся создавать первый проект, добавлять игровые объекты и писать простые скрипты на языке программирования C#. Помимо этого, вы познакомитесь с основными элементами интерфейса Unity и их функциональными возможностями, что позволит вам уверенно чувствовать себя в среде разработки.

Для того чтобы полноценно освоить базовые навыки разработки на Unity, мы создадим простую сцену с минимальным игровым функционалом, научимся подключать скрипты и взаимодействовать с объектами в процессе игры. В ходе выполнения заданий будут продемонстрированы как визуальные, так и программные подходы к созданию игрового мира, а также будет рассмотрен базовый цикл разработки игры, начиная от установки среды разработки до создания простейшего игрового проекта.

Установка Unity и Unity Hub

1. Перейдите на портал unity.com в раздел для загрузки программного обеспечения: **unity.com/ru/download**. На этой странице при нажатии на кнопку «Загрузить» вы сможете загрузить версию для своей операционной системы.

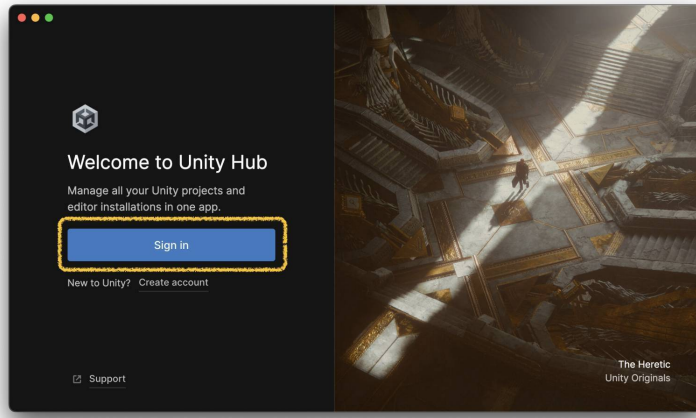


Изображение выглядит как текст, снимок экрана, программное обеспечение, Веб-сайт Автоматически созданное описание

2. Разрабатывать на Unity можно в основных операционных системах, что, несомненно, играет большую и важную

роль в популярности этой среды разработки игр. После завершения загрузки у вас на компьютере окажется установщик UnityHubSetup, по сути, это загрузчик (лаунчер) для ваших будущих проектов. Установка Unity Hub стандартная, дважды кликните по установщику UnityHubSetup и дождитесь окончания установки.

3. Запустите Unity Hub. При первом запуске система предложит вам войти или создать свой аккаунт. Так вы можете войти / создать свою учетную запись, выбрав в левой части Unity Hub – Sign in:



Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, программное обеспечение, текст Автоматически созданное описание

Процесс создания учетной записи для Unity стандартный, поэтому мы не будем здесь расписывать его подробно. Отдельно хочу обратить внимание, что следует использовать единую учетную запись для всех сервисов Unity, с которыми вы работаете. Таким образом, Unity предлагает широкий спектр сервисов и инструментов, которые покрывают все этапы разработки игр — от создания контента до обучения и публикации. Единая учетная запись предоставляет доступ ко всем сервисам через один логин, обеспечивая интеграцию между ними.

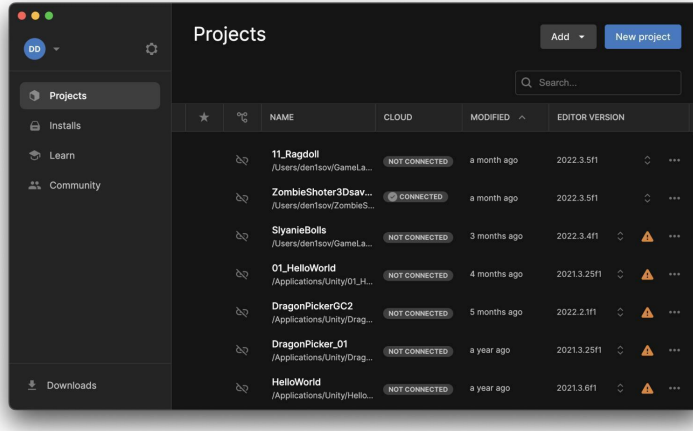
- **Asset Store (assetstore.unity.com):** здесь вы найдете всё, что может понадобиться для создания игры — 2D и 3D модели, анимации, звуковые эффекты, шейдеры, пакеты SDK, шаблоны проектов и другие полезные инструменты. Все приобретённые или скачанные ресурсы могут быть мгновенно добавлены в ваш проект в Unity.

- **Learn Unity (learn.unity.com):** Unity предлагает широкий выбор бесплатных учебных материалов и курсов, которые подойдут как для новичков, так и для продвинутых разработчиков. Здесь вы найдёте пошаговые инструкции по работе с Unity и разборы практических кейсов.

- **Unity Dashboard (cloud.unity.com):** в этой панели управления объединены различные аналитические и монетизационные инструменты, которые помогают отслеживать прогресс проекта, взаимодействие пользователей с игрой, а также управлять рекламой и покупками внутри приложения.

Таким образом, использование единой учетной записи для всех сервисов Unity не только упрощает процесс авторизации, но и делает разработку более эффективной, так как все нужные инструменты и ресурсы доступны в одном месте.

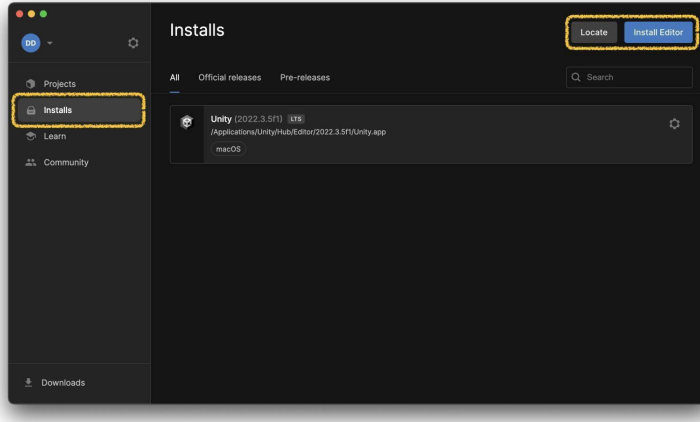
4. После того как вы создали и вошли в свой аккаунт Unity, откроется окно приложения Unity Hub. В центральной части приложения указаны проекты (Projects), с которыми вы работаете. Если вы используете Unity впервые, то это окно у вас должно быть пустым, однако очень скоро в нем начнут появляться созданные вами проекты, и Unity Hub будет выглядеть наполненным самыми разными проектами:



Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обес-

печение Автоматически созданное описание

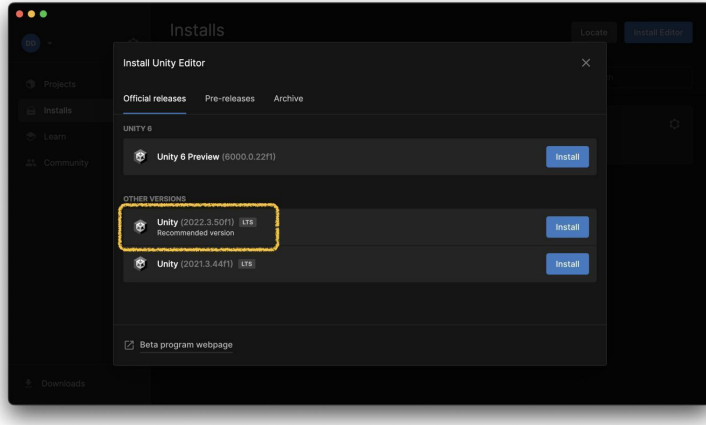
5. Теперь перейдем к установке редактора Unity. Оставаясь в Unity Hub нажмите кнопку Installs [1] в левом меню и далее - Install Editor [2]. Из Unity Hub можно запускать множество разных версий среды разработки Unity.



Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение Автоматически созданное описание

6. После этого откроется окно выбора версий Unity для установки. Рекомендуется установить последнюю стабильную версию Unity, если вы опытный пользователь. Для начинающих пользователей лучше использовать версию Unity, по которой составлено это руководство. **Рекомендуемая**

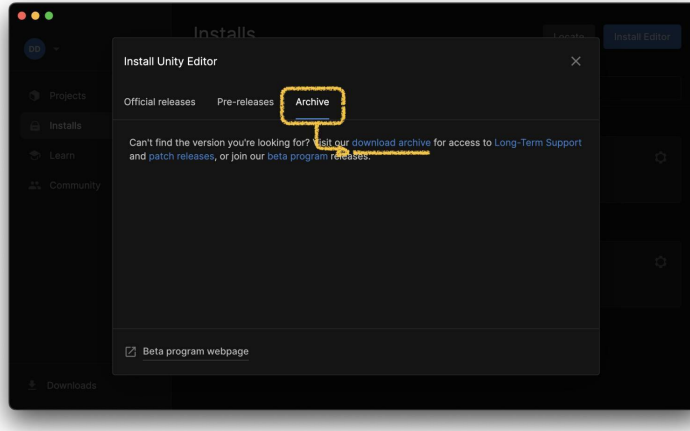
версия (Recommended Version) всегда является последней стабильной версией Unity, которая автоматически выделяется в списке. Для этой книги используйте последнюю стабильную версию **Unity 6** (она же будет отмечена как **Recommended Version**). Для установки, в окне **Install Unity Editor – Official Releases** выберите и установите **Recommended Version**. Чтобы начать установку, нажмите кнопку **Install**.



Изображение выглядит как снимок экрана, программное обеспечение, Мультимедийное программное обеспечение, текст Автоматически созданное описание

Если позднее вы захотите скачать любую другую версию Unity, перейдите в раздел Archive – download archive и най-

дите интересующую вас версию на сайте разработчика. Установка конкретной версии Unity может потребоваться в том случае, если вы нашли исходники некоторых сборок игр на таких ресурсах как github, либо купили исходники в unity asset store:



Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение Автоматически созданное описание

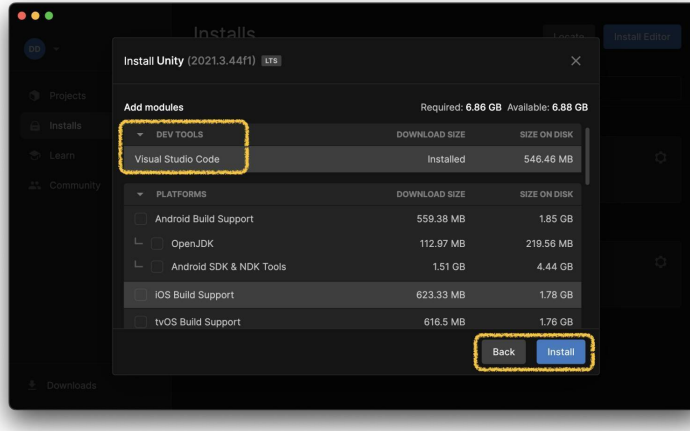
7. После того как вы нажали на кнопку Install, в следующем окне следует выбрать дополнительные модули. Напомню, что Unity позволяет создавать игры под самые разные платформы. Например, если в дальнейшем вы захотите сделать игру под мобильное устройство, то все что вам потре-

буется – это установить модули Android Build Support и iOS Build Support. На данном шаге нам потребуется установить среду разработки для работы с кодом. Для создания сценариев на Unity используется язык программирования C# и в самом верхнем списке вам предлагается установить Microsoft Visual Studio (если вы работаете на Windows) или Visual Studio for Mac (если вы работает на соответствующей операционной системе). Поставьте галочки напротив:

- модуля Visual Studio чтобы сразу скачать и установить среду для работы с кодом (поставьте флажок напротив модуля Visual Studio),

- WebGL Build Support, что позволит нам создавать сборку проекта под браузерные игры,

- нажмите кнопку Install:



Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение Автоматически созданное описание

Скачивание и установка модулей и среды разработки займет некоторое время, которое зависит от скорости вашего интернет-соединения

8. Когда скачивание завершится, произойдет автоматическая установка всех компонентов, и на этом процесс установки завершен. Если в дальнейшем вам понадобятся другие версии среды разработки Unity (например, вы найдете и захотите посмотреть готовые проекты, сделанные под более ранние версии среды разработки), - то вы всегда сможете открыть Unity Hub, перейти во вкладку Install и скачать недостающие версии Unity и модули, нажав кнопку Install Editor. Таким образом, Unity Hub является своего рода “точкой старта”, из которой происходит создание новых проектов (вкладка Projects), установка различных версий Unity (вкладка Installs) и т. д.

9. По итогу пошагового выполнения пунктов выше, у вас:

- должна быть установлена среда разработки Unity,

- среда для работы с кодом: Microsoft Visual Studio (для работы из-под Windows) или Visual Studio (для работы из-под Mac или Linux),

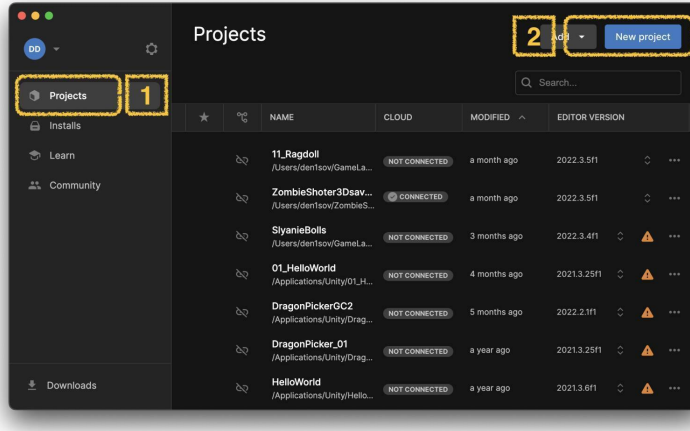
- создана учетная запись на сайте Unity.com. Не теряйте ее, так как эту учетную запись вы можете использовать для

работы в других сервисах Unity.

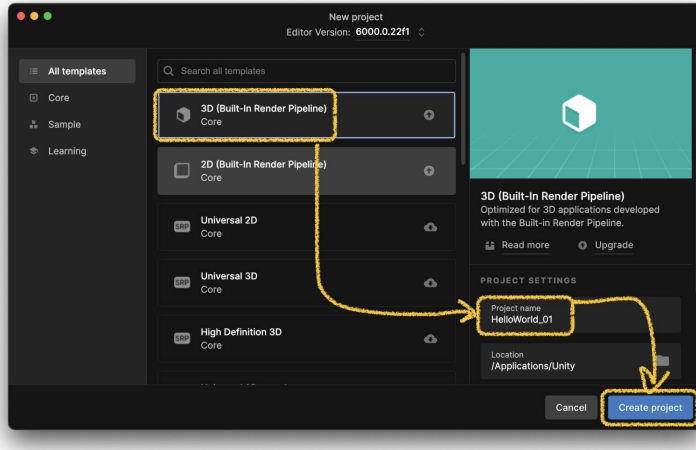
Первый проект: проверяем, что всё работает

Создадим простейшую сцену чтобы проверить корректность работы всех установленных программных пакетов. По традиции принято создавать программу, которая выводит сообщение «Hello World» в терминал. В нашем примере мы не только выведем сообщение, но и научимся взаимодействовать с объектами на среде Unity.

1. Чтобы создать первый проект на Unity, откройте Unity Hub и перейдите во вкладку Projects. Нажмите New project чтобы перейти в окно создания нового проекта:



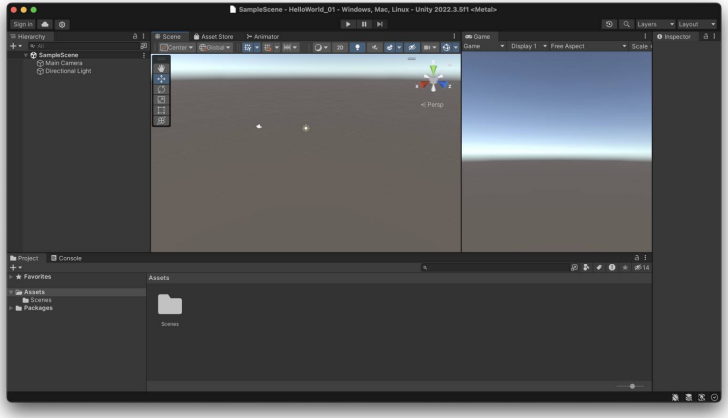
2. В появившемся окне нужно выбрать тип проекта - 3D, дайте имя новому проекту, например HelloWorld_01. Проверьте путь к папке, в которой будет создан проект (здесь скорее важно, чтобы вы осознанно указали папку для проекта и не потеряли его в дальнейшем). После этого нажмите Create project:



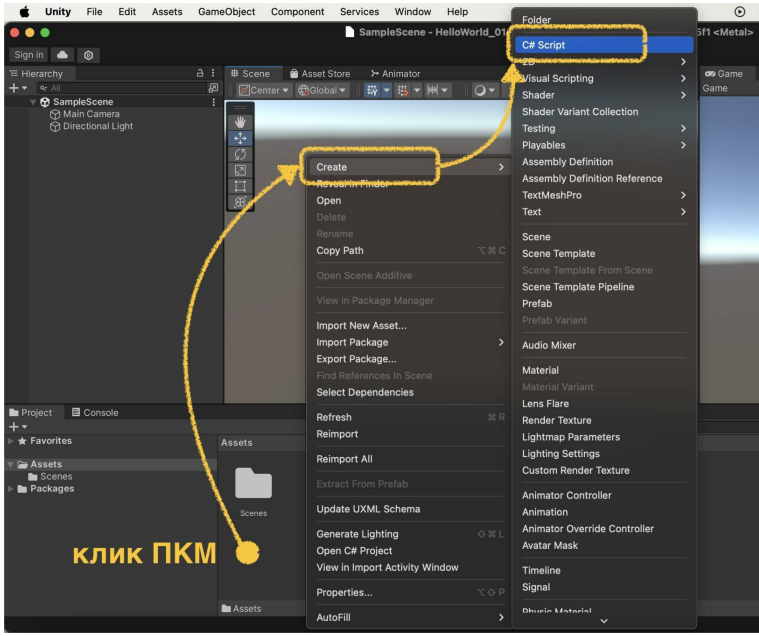
Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение Автоматически созданное описание

Проект будет создан и открыт в новом окне Unity. Первый запуск проекта может быть длительным, так как созда-

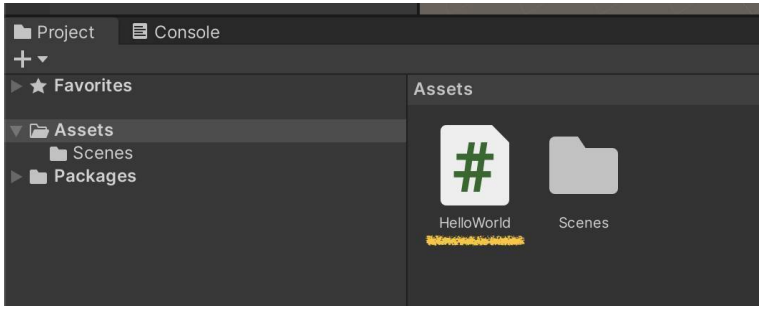
ются необходимые зависимости в библиотеках проекта. На рисунке ниже показано, как выглядит запущенная среда разработки. На данном этапе вам может показаться, что среда содержит довольно большое количество разнообразных и непонятных окон, но в дальнейшем мы разберемся, как они устроены и за что отвечают ее отдельные элементы:



4. Теперь мы готовы к тому, чтобы начать работу. Первым делом создадим новый C# Script-файл с простой командой, которая выводит сообщение “Hello World”. Для этого на панели Project перейдите в папку Assets, в данный момент в ней находится только одна папка с названием Scenes. Кликните правой кнопкой мыши внутри папки Assets и выберите из контекстного меню Create - C# Script, как показано ниже:

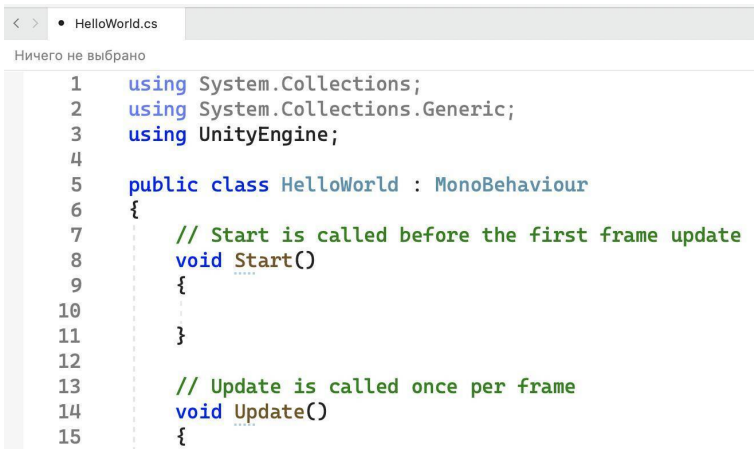


5. Назовите созданный скрипт-файл HelloWorld. Содержимое папки Assets после этого должно выглядеть так, как показано на рисунке ниже:



Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение Автоматически созданное описание

6. Откройте созданный файл HelloWorld.cs, кликнув по нему дважды. Файл автоматически откроется в Visual Studio, если этого не произошло автоматически, зайдите в Unity – Preferences (или Settings для Mac) – External Tools – External Script Editor – убедитесь, что выбрано Visual Studio и откройте скрипт-файл еще раз. Содержимое файла и вид среды разработки показаны на рисунке ниже:



7. Когда мы перейдём к разработке игры, программный код будет представлен как в виде скриншотов, так и в виде листинга (текста). Это сделает его удобным для восприятия. В случае использования электронной версии книги вы сможете легко копировать и вставлять фрагменты кода непосредственно в ваш проект, что ускорит процесс разработки. Обратите внимание на то, что внутри кода выше содержится два метода: void Start() и void Update().

- void Start() — метод, который вызывается один раз при запуске сцены в Unity. Все команды, написанные внутри фигурных скобок этого метода, будут выполнены при старте иг-

ры. Этот метод удобно использовать для начальной инициализации объектов, настройки переменных или загрузки данных.

- void Update () – метод, который вызывается каждый кадр в течение работы сцены. В него нужно писать функционал, который требует постоянного обновления. Например, в этот метод добавляют логику движения объектов, проверку ввода от игрока или обновление интерфейса.

8. Добавьте строку кода, которая будет выводить сообщение «*Hello World*». Для этого внутрь фигурных скобок метода void Start(), как показано в листинге ниже, нужно написать команду print:

HelloWorld.cs

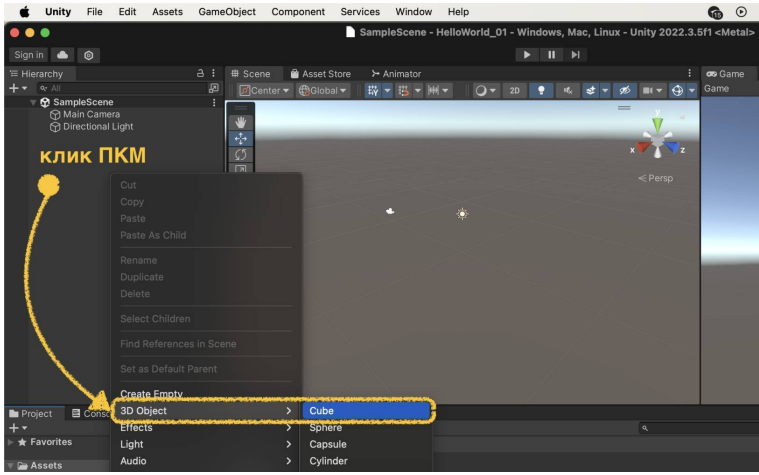
Assembly-CSharp.Player › HelloWorld › Start()

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class HelloWorld : MonoBehaviour
6   {
7       // Start is called before the first frame update
8       void Start()
9       {
10          print("Hello World");
11      }
12
13      // Update is called once per frame
14      void Update()
15      {
16
17      }
18  }
```

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере Автоматически созданное описание

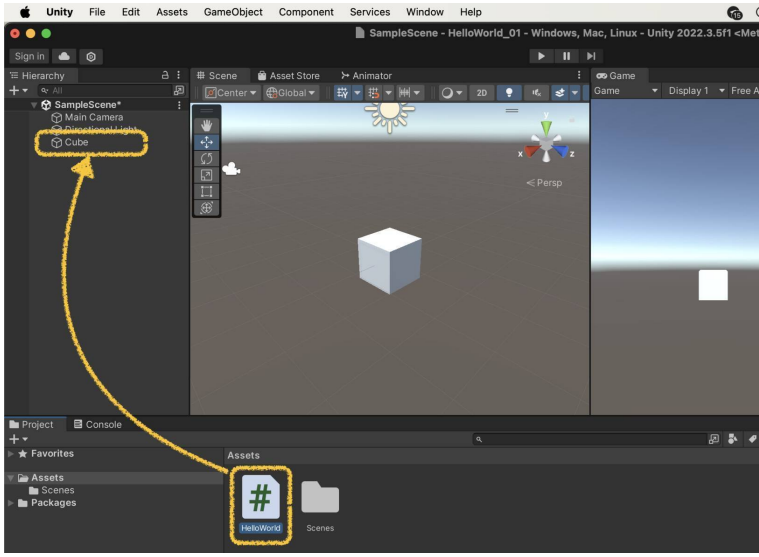
9. Скрипт-файл с названием HelloWorld.cs написан. Однако, чтобы он начал работать, нам следует его подключить к одному из игровых объектов внутри сцены Unity. Давайте создадим такой объект, например, простейший куб.

10. Все объекты на сцене находятся в окне иерархии объектов (Hierarchy в левой части среды разработки). Пока мы ничего не создали, но можете обратить внимание что на сцене уже существует камера (Main Camera), которая играет роль глаз игрока и освещением Direction Light, без которого на сцене было бы значительно темнее. Чтобы создать игровой объект “Куб”, кликните правой кнопкой мыши (ПКМ) внутри окна Hierarchy и в выпадающем меню выберите GameObject - 3D Object - Cube:



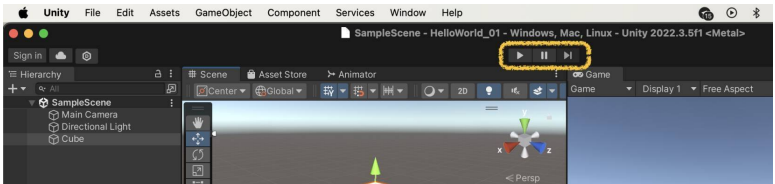
Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение Автоматически созданное описание

11. Таким образом, на сцене появится новый игровой объект Cube. Приблизить куб можно с помощью колесика мыши. Чтобы подключить скрипт HelloWorld.cs к объекту Cube, можно просто перетащить (зажав левую кнопку мыши) скрипт-файл на куб.

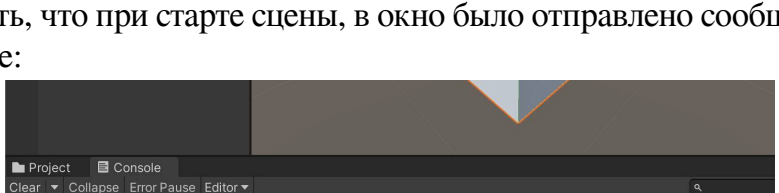


12. Теперь, если на сцене выделить объект Cube, кликнув по нему левой кнопкой мыши, то можно увидеть, что в правой части среды разработки (окно Inspector) к кубу в качестве компонента подключился файл HelloWorld.cs (Script-файл).

13. Теперь вы можете запустить сцену и проверить ее работу. Для этого нужно нажать кнопку Run в верхней центральной части среды разработки.



14. Обратите внимание, что на сцене статично висит куб и кажется, что ничего не происходит, но если перейти в окно Console (в нижней части среды разработки), то можно заметить, что при старте сцены, в окно было отправлено сообщение:



Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение Автоматически созданное описание

15. Остановите симуляцию сцены, нажав кнопку Run еще раз. Не забывайте выключать симуляцию сцены, так как изменения, которые вы вносите в редакторе, не будут сохраняться при запущенной сцене.

Вместо функции print, можно использовать функцию

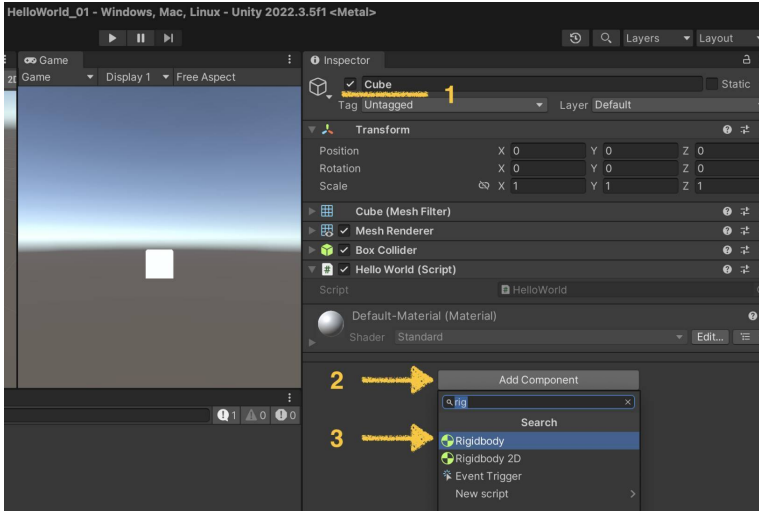
Debug.Log(), которая является частью движка Unity. Отличие функции Debug.Log() от функции print() заключается в том, что print() не позволяет увидеть какую-либо информацию, после сборки проект. То есть print() выводит информацию только в консоль среды разработки Unity, тогда как функция Debug.Log() выводит сообщение в специальный файл в папке проекта при запуске готовой игры, содержимое которого потом можно просмотреть. По сути, обе эти функции делают одно и тоже, но рекомендуют использовать именно Debug.Log(). В качестве эксперимента вы можете заменить функцию print("Hello World") в листинге выше на функцию Debug.Log("Hello World").

16. Обычным сообщением в окне консоли сложно удивить, особенно если речь идет о разработке игры. Поэтому давайте добавим еще немного функций для наглядности. Следует отметить, что многие моменты, связанные с разработкой игры, на себя берет Unity без необходимости написания кода. Иногда даже очень сложный функционал в игре можно реализовать просто настройками внутри среды разработки Unity. Продемонстрируем это на примере ниже.

17. Сделаем так, чтобы созданный 3D объект Cube при запуске сцены падал вниз. Для этого выделите объект Cube (клик левой кнопкой мыши в окне объектов Hierarchy), после этого в правой части среды разработки станет активно окно Inspector для выбранного игрового объекта - куба.

18. Содержание окна Inspector зависит от типа выбранно-

го объекта. В нем содержатся свойства объекта, его параметры, подключаемые Script-файлы и т. д. Нажмите в нижней части окна Inspector кнопку Add Component. После этого появится список с перечнем компонентов, которые могут быть подключены к выбранному объекту Cube. Найдите с помощью поиска компонент Rigidbody и кликните по нему левой кнопкой мыши так, чтобы он добавился в окно Inspector.

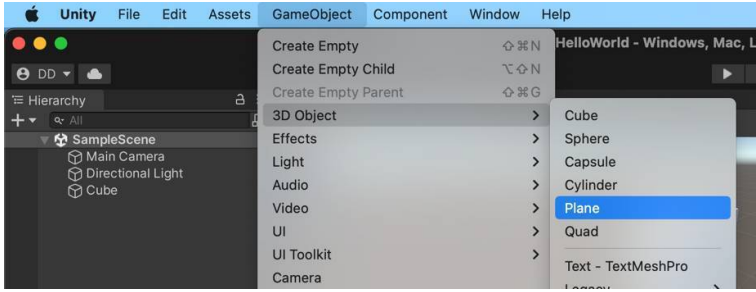


Изображение выглядит как Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение, Значок на компьютере Автоматически созданное описание

20. Компонент Rigidbody добавляет объекту свойства фи-

зики твердого тела, определенного в базовом движке Unity. Другими словами, если назначить этот компонент объекту, то он начнет вести себя в соответствии с законами механики: иметь массу, участвовать в упругих столкновениях, действовать на другие объекты с теми же свойствами и так далее. Запустите сцену еще раз (нажмите Run) и убедитесь, что теперь объект Cube начинает падать вниз. Если куб начал падать вниз, поздравляю, вы все сделали правильно. Можете отключить симуляцию сцены, нажав кнопку Run еще раз.

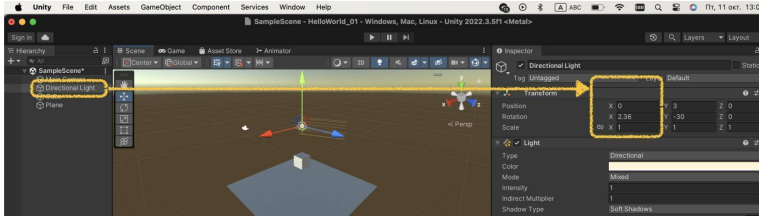
21. Создадим еще один объект - плоскость (Plane), которая будет ограничивать падение куба за пределы начальной сцены. Для этого выполните действия, которые уже выполнялись при создании объекта Cube, - в верхней части меню выберите GameObject - 3D Object – Plane (или клик ПКМ в окне иерархии объектов GameObject - 3D Object – Plane):



Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обес-

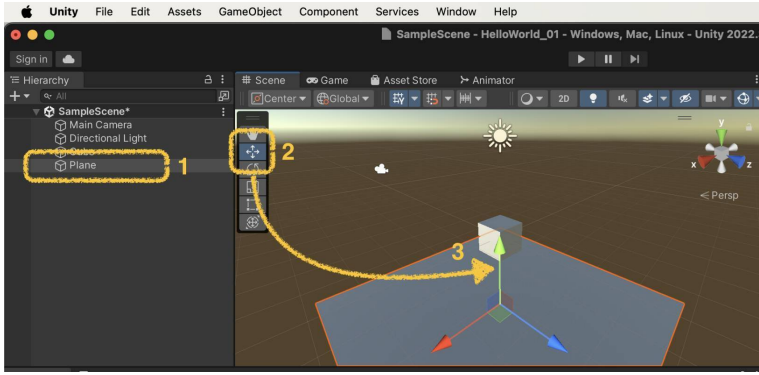
печение Автоматически созданное описание

22. Также обращу внимание на то, что, изменяя параметры объекта Transform, можно изменять его положение или ориентацию на сцене. Так, например, изменив угол падения света Direction Light, можно делать так чтобы на сцене «наступил вечер»:



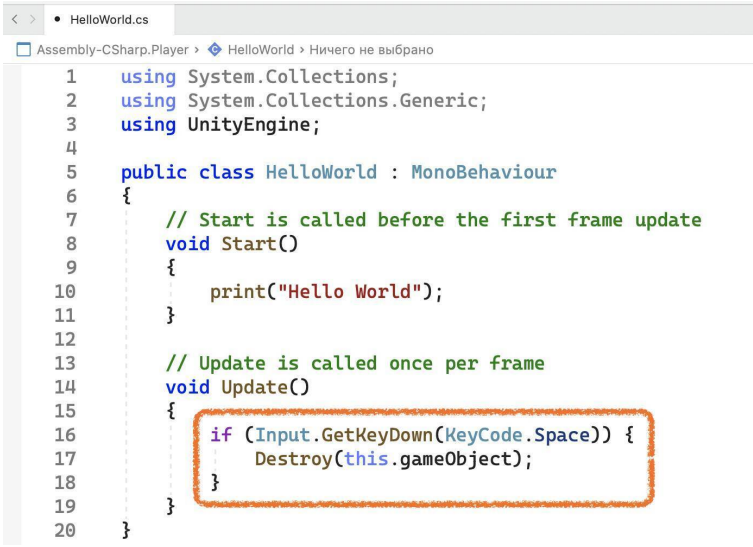
Изображение выглядит как Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение, снимок экрана Автоматически созданное описание

22. После создания плоскости переместим ее немного ниже уровня объекта Cube. Для этого выделите объект Plane в окне Scene и можете изменить значения Transform для перемещения, или выберите инструмент перемещения Move Tool, кликнув на ось Y сдвиньте плоскость Plane ниже куба. Используя Move Tool, вы можете двигать объекты на сцене:



23. Запустите сцену еще раз. Теперь объект куб (Cube) падает на плоскость (Plane) при старте сцены.

24. Теперь добавим немного интерактивности. Откройте скрипт-файл, который мы создали ранее с именем HelloWorld.cs и напишите туда небольшой функционал, который будет уничтожать объект Cube при нажатии клавиши пробел. В программном коде ниже показано содержимое файла HelloWorld.cs, а рамкой выделены новые строки кода, которые нужно ввести дополнительно:



Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница Автоматически созданное описание

В листинг были добавлены следующие строки кода:

- создается условие if, которое уничтожает объект с помощью команды Destroy при нажатии клавиши Space. При этом используется метод Input.GetKeyDown, который срабатывает, после того как игрок нажал клавишу. В скобках Destroy указана конструкция this.gameObject, которая означает что нужно удалить this (этот) игровой объект (game object), то есть тот самый к которому подключен скрипт-файл.

25. Запустите сцену и проверьте, что она работает следующим образом:

- В окно Console выводится сообщение *“Hello World”*;
- Куб (Cube) начинает падать;
- Куб падает на плоскость Plane и останавливается;
- При нажатии на клавишу пробел объект Cube удаляется.

Что дальше

После прохождения данного раздела вы получили общее представление о том, как работает игровой движок Unity и что необходимо для создания первого тестового проекта. Вы научились устанавливать Unity, создавать сцены, добавлять игровые объекты и писать простые скрипты на языке C#. Это лишь первые шаги на пути освоения игрового программирования, но они важны для построения более сложных и интерактивных проектов в будущем.

Теперь вы знаете, как использовать такие методы, как void Start() и void Update(), чтобы инициализировать объекты при запуске сцены и обрабатывать события каждый кадр. Вы также научились подключать скрипты к игровым объектам, добавлять физику в проект с помощью компонента Rigidbody и даже создавать простейшую интерактивность — например, удаление объекта при нажатии клавиши.

На этом этапе ваши навыки можно назвать основами игрового программирования на Unity. Дальнейшее освоение платформы предполагает более глубокую работу с игровыми объектами, скриптами и функционалом Unity. В следующих главах мы перейдем к созданию более сложных механик и изучению дополнительных возможностей движка. Понимание того, как работают базовые функции Unity, позволит вам продолжать развиваться и создавать более сложные и инте-

ресные игры.

После завершения всех пунктов рекомендуется вернуться в начало раздела и еще раз внимательно просмотреть всю последовательность действий. Попробуйте самостоятельно внести модификации в некоторые пункты на свой выбор. Так вы сможете более детально разобраться в устройстве взаимосвязей между объектами, скрипт-файлами и некоторыми элементами интерфейса Unity. Ниже приведен некоторый список возможных изменений в проекте Unity, который вы можете внести, опираясь на те инструкции, которые были даны в этом разделе:

- Сделайте так, чтобы в Console выводилось сообщение “Goodbye World”.

- Добавьте на сцену больше объектов произвольной формы, измените их размер, положение и ориентацию. При добавлении на объекты компонентов Rigibdoby они будут сталкиваться и падать.

- Создайте новый скрипт-файлы, чтобы разные объекты могли быть удалены со сцены при нажатии разных клавиш на клавиатуре.

- Перенесите строку кода *print("Hello World");* из фигурных скобок метода Start() в фигурные скобки метод Update(), и проверьте работу сцены. Что изменилось в выводе в командной строке Console? В качестве подсказки отметим, что метод Update обрабатывается каждый кадр, тогда как метод Start отрабатывает лишь один раз при старте сцены.

Глава 1. Быстрый старт от проекта до игры

Сначала результат — потом теория. Так интереснее.

Введение

Обычно учебники по разработке игр устроены так: сначала десятки страниц теории, потом ещё столько же про код — и только где-то к концу, если вы дошли, ваша игра наконец появляется в интернете. Мы сделаем наоборот.

Фишка этой книги — мгновенный результат. Уже в этой, самой первой главе вы пройдёте весь путь целиком: скачаете готовый проект, откроете его в Unity, соберёте браузерную версию и опубликуете её на Яндекс.Играх. В конце у вас будет рабочая **ссылка на игру**, которую можно отправить другу со словами «зацени, я сделал игру». Без долгой теории — по шагам, как по рельсам.

Зачем так? Потому что ничто так не мотивирует, как живой результат. Когда вы своими руками проведёте игру от проекта до публикации, всё остальное в книге — детальный разбор шаблона, изменение уровней, программирование механик — пойдёт совсем с другим настроем. Вы будете не «изучать абстракции», а улучшать то, что у вас уже работает и опубликовано.

Важно: это «экспресс-прогон». Здесь мы пройдём весь конвейер коротко, без подробных объяснений каждой кнопки — чтобы вы быстро увидели результат. Каждый шаг этой главы подробно разбирается дальше: установка Unity и шаблона — в Главе 2, разбор проекта — в Главе 3, сборка и пуб-

ликация во всех деталях — в Главе 5. Если на каком-то шаге застрянете — загляните в соответствующую главу и возвращайтесь.

Совет: для этой главы Unity уже должен быть установлен. Если его ещё нет — это минут десять: откройте Главу 2, дойдите до конца установки и возвращайтесь сюда.

А хотите увидеть результат прямо сейчас, не открывая Unity? Игра уже собрана и работает в браузере — откройте ссылку и поиграйте: https://yandex.ru/games/app/223547?draft=true&lang=ru. Это ровно та версия DeepGalacticShooter, которую вы соберёте и опубликуете своими руками к концу этой главы.

1.1 Что понадобится

Для быстрого старта нужно немного, и всё бесплатно:

Установленный Unity (Unity Hub + Unity Editor). Если ещё нет — см. Главу 2, там подробно описано как скачать, но вы просто можете найти на сайт unity.com и попробовать справится с задачей "добычи" инсталлера самостоятельно (в целом не сложнее чем скачать игру из Steam).

Аккаунт на Яндексе — обычная почта на yandex.ru (нужна для консоли разработчика).

Любой современный браузер на компьютере.

План на ближайшие 15 минут: скачать проект → открыть в Unity → собрать билд → залить в консоль Яндекс.Игр → получить ссылку. Поехали.

1.2 Скачиваем проект с Яндекс.Диска

Весь проект я собрал в один компактный архив (около 10 МБ) и выложил на Яндекс.Диск — специально для книги. Служебная папка Library с кэшем в архив не входит: Unity создаст её сам при первом открытии, поэтому файл такой лёгкий.

Ссылка на проект: https://disk.yandex.ru/d/MCnVTXw9IWNVcw

Откройте ссылку выше — она ведёт на архив DeepGalacticShooter.zip (рядом лежит и короткая текстовая инструкция по установке).

Нажмите **«Скачать»** — архив сохранится на компьютер. Положите его в удобное место и **распакуйте**: получится папка DeepGalacticShooter с подпапками Assets, Packages и ProjectSettings.

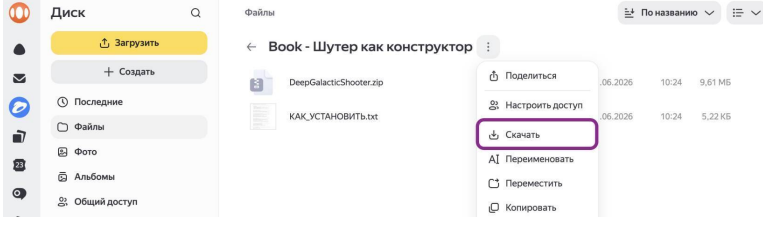


Рис. 1.1. Страница Яндекс.Диска с проектом игры и кнопкой «Скачать».

Совет: распакуйте архив в постоянную папку (не во «Временные» и не в «Загрузки»), потому что Unity будет открывать проект именно из этого места. Если папку потом переместить — просто добавите её в Unity Hub заново.

1.3 Открываем проект в Unity

Запустите **Unity Hub**. На вкладке **Projects** нажмите **Add → Add project from disk**.

Укажите распакованную папку проекта и нажмите **Open**. Проект появится в списке — кликните по нему, чтобы открыть в редакторе. Первый запуск может занять время: Unity импортирует ресурсы и собирает зависимости.

Рис. 1.2. Добавление проекта в Unity Hub: Add → Add project from disk.

Когда проект откроется, в окне **Project** появится папка Assets/Project_Shooter, а в папке **Scenes** — три сцены:

MainMenu, Level_1, EndScene. Откройте MainMenu и нажмите **Play**, чтобы убедиться, что игра запускается.

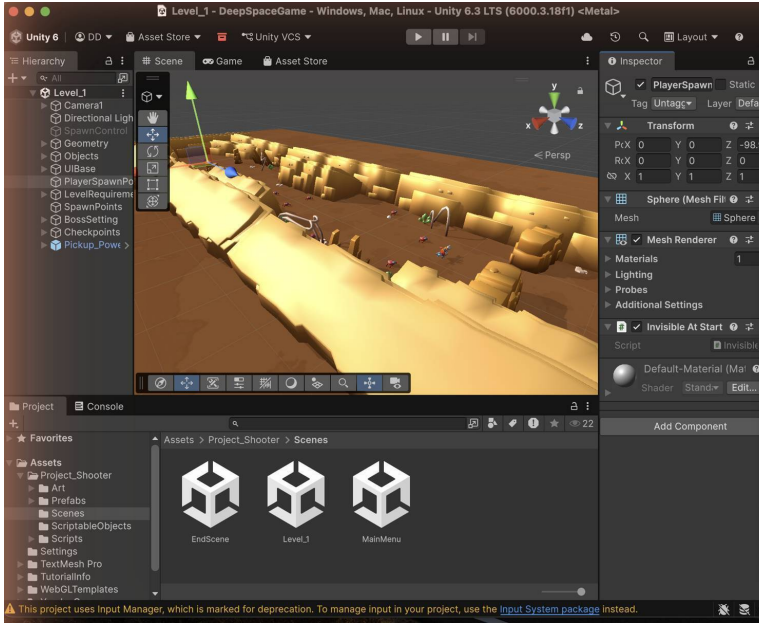


Рис. 1.3. Проект открыт в редакторе Unity: окна Scene, Game, Hierarchy, Project.

1.4 Собираем браузерный билд (WebGL)

Откройте **File → Build Profiles** (в более ранних версиях Unity — **Build Settings**). В списке платформ выберите **Web (WebGL)** и нажмите **Switch Platform**. Если платформа недоступна — согласитесь на установку модуля Web Build Support и подождите.

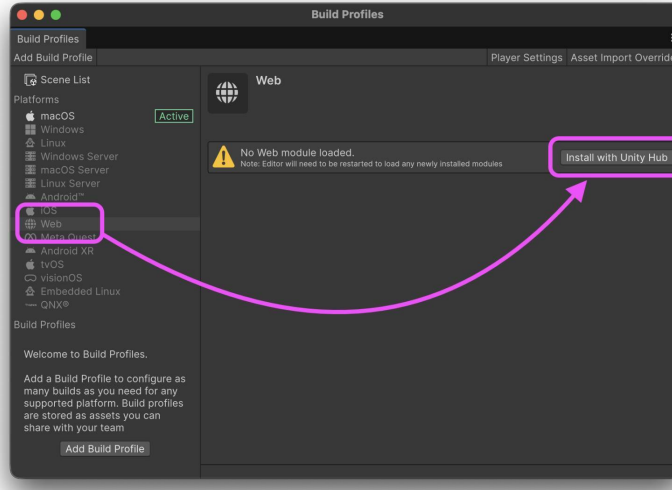


Рис. 1.4. Окно Build Profiles: выбрана платформа Web

(WebGL), кнопка Switch Platform.

Убедитесь, что в списке сцен сборки (**Scene List**) присутствуют MainMenu, Level_1 и EndScene, причём MainMenu стоит **первой**. Нажмите **Build**, укажите папку и назовите её, например, DeepGalacticShooterBuild. Дождитесь окончания сборки.

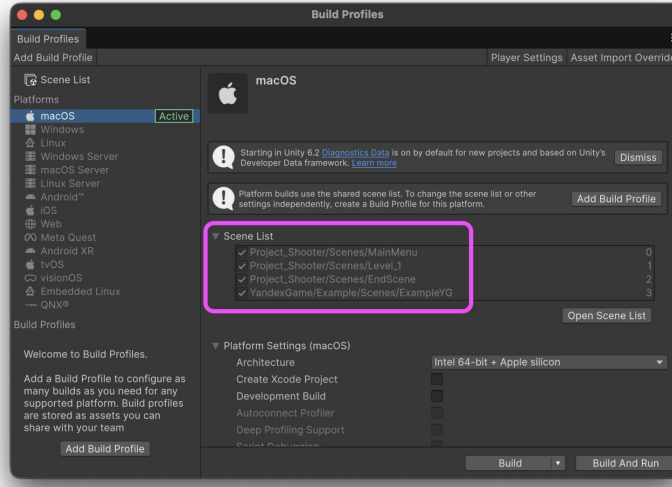


Рис. 1.5. Выбор папки сборки DeepGalacticShooterBuild и список сцен.

После сборки соберите содержимое папки DeepGalacticShooterBuild в один .zip-**архив** — именно архив

мы загрузим на Яндекс.Игры.

Совет: подробно про настройки сборки под WebGL (Player Settings, шаблон, сжатие) — в Главе 5. Для быстрого старта достаточно переключить платформу на WebGL и нажать Build: значения по умолчанию вполне рабочие.

1.5 Заливаем билд в консоль Яндекс.Игр

Перейдите на yandex.ru/dev/games, войдите под аккаунтом Яндекса и откройте **консоль разработчика**. При первом входе пройдите быструю бесплатную регистрацию разработчика.

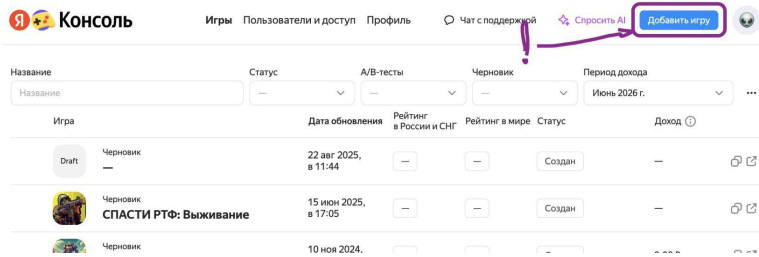


Рис. 1.6. Консоль разработчика Яндекс.Игр: кнопка «Добавить приложение».

Нажмите **«Добавить приложение»**, придумайте название (например, **DeepGalacticShooter**) и сохраните черновик. Игра появится во вкладке **«Приложения»** со статусом **Draft** (черновик).

Откройте черновик, найдите раздел **«Исходники»** и за-

грузите ваш .zip-**архив** со сборкой. Ниже, в поле **«Поддерживаемые платформы»**, выберите **«все десктопные браузеры»**. Сохраните черновик.

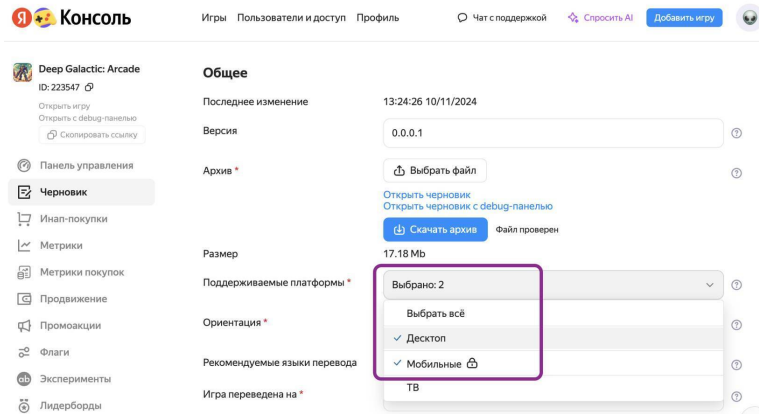


Рис. 1.7. Раздел «Исходники»: загрузка архива сборки и выбор поддерживаемых платформ.

1.6 Получаем ссылку и отправляем другу

Когда архив пройдёт автоматическую проверку, в черновике появится **ссылка на игру** (вида yandex.ru/games/app/...). Это рабочая ссылка на вашу публикацию.

Скопируйте её и отправьте другу — он откроет игру прямо в браузере, без установки чего-либо. Управление: бег — стрелки, стрельба — клавишей **Z**.

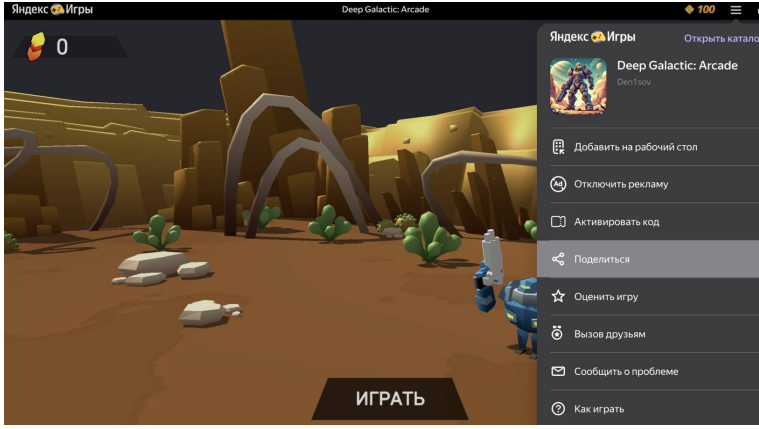


Рис. 1.8. Готовая ссылка на игру и запуск в браузере на Яндекс.Играх.

Готово! Вы собрали игру и опубликовали её. Если уло-

жились в 15 минут — поздравляю: вы только что прошли путь, на который у многих уходят недели.

Теория: жизненный цикл игры — от идеи до игроков

Вы только что собрали и опубликовали игру за пятнадцать минут — и, честно говоря, это маленькое чудо. Но за этими пятнадцатью минутами прячется большой и красивый процесс, который проходит каждая игра на свете, от инди-однодневки до многомиллионного блокбастера. Давайте посмотрим на него целиком, с высоты птичьего полёта, — чтобы вы понимали не только *что* нажимали, но и *где* в общей картине находились. Хорошая новость: вы уже прошли весь этот путь от начала до конца. Осталось только разложить его по полочкам, внести изменения чтобы сделать игру "оригинальной" и начать на ней зарабатывать :)

Пять этапов: идея → прототип → продакшн → билд → публикация (и потом — поддержка)

Любая игра рождается из **идеи**. Иногда это одна фраза: «космический шутер, где ты отстреливаешься от волн врагов и собираешь кристаллы». Идея — это ещё не игра, это направление, обещание самому себе. На этом этапе важно не утонуть в мечтах о «лучшей игре всех времён», а сформулировать суть в паре предложений: какой жанр, что делает игрок, что приносит удовольствие. Наш DeepGalacticShooter — это как раз такое обещание, обретшее форму.

Дальше идёт **прототип** — самая честная стадия. Здесь вы проверяете, *весело ли вообще играть*, ещё до красивой графики и музыки. Прототип может быть уродливым: кубик стреляет по кубикам — но если в этот момент уже интересно, идея жива. Если нет — лучше узнать это сейчас, на кубиках, а не через полгода полировки.

Затем — **продакшн**, основная и самая долгая часть. Это всё, что мы будем делать в следующих главах: настраивать оружие и врагов, рисовать уровни, добавлять интерфейс, звук, баланс. Прототип превращается в полноценную игру. Именно здесь живёт большая часть этой книги.

Когда игра готова (или хотя бы готова показаться людям),

наступает **сборка**, она же **билд** — превращение проекта в работающее приложение. И финальный аккорд — **публикация**: вы выкладываете игру туда, где её найдут игроки. В нашем случае — на Яндекс.Игры, в браузер.

А после публикации история не заканчивается — начинается **поддержка и обновления**: вы читаете отзывы, чините баги, докручиваете баланс, добавляете контент. Игра — это не камень, который высекли и забыли, а скорее сад, за которым ухаживают.

Разбираем каждую стадию подробнее: что она даёт

Пять этапов выше — это карта, на которой видны материки. Но если приблизиться, у каждого материка обнаруживаются свои города, и стоит пройтись по ним пешком. В «большом» геймдеве цикл обычно раскладывают чуть подробнее, и эти названия полезно знать — вы будете встречать их в статьях, вакансиях и постмортемах разработчиков.

Идея и концепт. Сначала идея живёт в голове, а потом её записывают. Документ, в котором сформулирована суть игры, называют *концептом* (или однострочным «pitch»): жанр, целевая аудитория, ключевая эмоция, главная механика, чем игра отличается от соседей по полке. Для инди это не толстый «дизайн-документ на 200 страниц», а скорее одна-две страницы, которые можно показать другу и спросить: «понятно, во что тут играют?». Что даёт стадия: *фокус*. Когда суть записана, легче отвечать «нет» всему, что в эту суть не вписывается.

Препродакшн (preproduction). Это этап «разведки боем»: вы ещё не строите всю игру, а проверяете рискованные места. Заработает ли управление? Весело ли стрелять? Потянет ли выбранная графика браузер? Здесь делают черновые наброски, маленькие технические пробы (их называют *spike*

— «укол», быстрый эксперимент ради одного вопроса) и собирают первые прототипы. Что даёт стадия: *снимает страх неизвестности*. К концу препродакшна вы знаете, что игра в принципе осуществима силами вашей команды.

Прототип и вертикальный срез. Прототип отвечает на вопрос «весело ли?». А когда веселье подтвердилось, делают **вертикальный срез** (vertical slice) — маленький, но *полностью готовый* кусочек игры: один уровень, доведённый до финального качества, с графикой, звуком, интерфейсом и балансом. Это как пробная порция в ресторане: по ней судят обо всём блюде. Наш Level_1 в шаблоне очень близок по духу к вертикальному срезу — это один уровень, который выглядит и играется «по-настоящему», хотя вся игра ещё не достроена.

Продакшн. Когда срез доказал, что «вот так это и должно ощущаться», начинается тиражирование: по образцу первого уровня делают остальные, добавляют врагов, оружие, боссов, наполняют игру контентом. Это самая долгая и самая «рабочая» стадия — много рутины, мало откровений. Что даёт стадия: *объём*. Из одного убедительного кусочка вырастает целая игра.

Альфа, бета и плейтест. *Альфа* — момент, когда в игре есть всё основное содержимое («feature complete»), но оно ещё сырое и в багах. *Бета* — когда новые фичи больше не добавляют, а только шлифуют и чинят. Между ними и вокруг них непрерывно идёт **плейтест** — наблюдение за

тем, как в игру играют другие люди. Что даёт стадия: *правду*. Только живой игрок покажет, что «очевидная» кнопка неочевидна, а «честный» враг кажется нечестным.

Релиз. Игра выходит к широкой аудитории. Для инди это часто не громкий «запуск ракеты», а тихая загрузка билда в магазин или на платформу — ровно то, что вы сделали в этой главе.

Пост-релиз и live-ops. После выхода игру поддерживают: патчи, баланс, новый контент, реакция на отзывы. Если игра живёт долго и постоянно обновляется, такую поддержку называют *live-ops* (live operations — «эксплуатация вживую»). Что даёт стадия: *жизнь после рождения*. Многие любимые игры стали любимыми не на релизе, а спустя десятки обновлений.

Не пугайтесь обилия слов. На инди-масштабе все эти стадии часто сливаются и проходятся за вечер: вы придумали врага (идея), накидали его кубиком (прототип), довели один экземпляр до ума (срез), наплодили таких же (продакшн), дали другу пострелять (плейтест) и выложили обновление (релиз). Большой цикл и маленький устроены одинаково — отличается только масштаб.

Итеративность: круг, а не прямая

Может показаться, что стадии идут строго по очереди: закончил одну — перешёл к следующей. На деле геймдев почти всегда *итеративен* — он движется кругами. Вы делаете маленькую версию, показываете её, узнаёте что-то новое и возвращаетесь назад, чтобы переделать. Прототип может отправить вас обратно к идее («а давай враги не летят, а ползут»). Плейтест посреди продакшна может заставить переписать управление. Это не провал и не хаос — это нормальный пульс разработки.

Сердце каждой итерации — **плейтест**. И здесь есть тонкость, которую новички узнают на собственных шишках: *смотреть, как человек играет, важнее, чем спрашивать его мнение*. Игрок может вежливо сказать «всё классно», но если вы видели, как он трижды промахнулся мимо кнопки старта или не понял, куда идти, — вы узнали в десять раз больше, чем из любых слов. Поэтому правило простое: дайте человеку поиграть, молчите, не подсказывайте и наблюдайте. А выводы делайте по действиям, а не по комплиментам.

Скоуп и коварный «feature creep»

Есть слово, от которого у опытных разработчиков слегка дёргается глаз, — **скоуп** (scope), то есть объём задуманного: сколько уровней, врагов, механик, режимов вы собираетесь сделать. Скоуп — это размер вашего обещания самому себе. И главная беда новичка не в том, что он мало задумал, а в том, что он задумал *слишком много*.

У этой беды есть имя — **feature creep** («расползание фич»): когда в проект потихоньку, по одной невинной идее за раз, добавляются всё новые возможности. «А давай ещё прокачку. И крафт. И мультиплеер. И сюжет с диалогами». Каждая идея по отдельности хороша, но вместе они топят проект: игра, которую можно было выпустить через месяц, не выходит и через год, потому что список «ещё бы добавить» растёт быстрее, чем вы успеваете его закрывать. Кладбище незаконченных инди-игр на 90% состоит именно из жертв feature creep, а не из плохих идей.

Лекарство — дисциплина скоупа. Запишите список фич, разделите его на «обязательно для выхода» и «потом, если будет время», и охраняйте первый список как сокровище. Хорошая привычка — спрашивать о каждой новой идее: «без этого игру нельзя выпустить?». Если можно — значит, это «потом».

Прикиньте сами: один аккуратный уровень, три типа вра-

гов и одно оружие — это игра, которую реально доделать и выпустить. «Открытый космос с сотней планет, экономикой и кооперативом» — это мечта, которая, скорее всего, никогда не доедет до игроков. Маленький законченный шутер бьёт большой ненаписанный по всем статьям.

Прототип и MVP: почему «выпускай рано, выпускай часто»

Здесь важно познакомиться с двумя понятиями, которые сэкономят вам месяцы. **Прототип** — это проверка идеи на «весело / не весело». А **MVP** (minimum viable product, «минимально жизнеспособный продукт») — это самая маленькая версия игры, которую *уже не стыдно показать игрокам*: в ней есть начало, середина и конец, она работает и в неё можно играть, пусть и без излишеств.

Главная мысль звучит почти как девиз: **«выпускай рано, выпускай часто»** (release early, release often). Смысл прост и слегка контринтуитивен. Начинающий разработчик хочет «доделать всё до идеала, а потом показать». Опытный — наоборот, стремится показать как можно раньше, пусть и сыроватое. Почему? Потому что пока игра лежит у вас на диске, вы гадаете, что понравится людям. А как только она попадает к живым игрокам, гадание сменяется фактами. Маленькая игра в руках игроков ценнее, чем шедевр в воображении.

В этой главе вы фактически сделали именно это: взяли готовый MVP и сразу выпустили его. Вы не ждали, пока освоите весь Unity, — вы получили живую ссылку уже сегодня. Это и есть правильный геймдев-рефлекс.

MVP и вертикальный срез: два разных «маленьких»

Выше мы уже познакомились с MVP. Теперь стоит развести два похожих, но разных понятия, которые легко спутать.

MVP — это самая маленькая игра *вширь*: в ней есть всё необходимое от старта до финала, но всё — по минимуму. Один уровень, один враг, базовый интерфейс, экран победы. Игру можно пройти от начала до конца — и этого достаточно, чтобы выпустить и собрать обратную связь. MVP отвечает на вопрос: «можно ли в это играть как в законченную игру?».

Вертикальный срез — это маленькая игра *вглубь*: один кусочек, доведённый до финального качества по всем «слоям» сразу (отсюда «вертикальный» — он прорезает все этажи: графику, звук, геймплей, полировку). Срез отвечает на вопрос: «а так это будет выглядеть и ощущаться в готовой игре?».

Грубо говоря, MVP показывает *форму* всей игры в дешёвом исполнении, а срез показывает *качество* одного куска. Наш шаблон удачно совмещает оба: цепочка MainMenu → Level_1 → EndScene — это полноценный MVP (есть начало, игра и финал), а сам Level_1 тянет на вертикальный срез — он играбелен и оформлен «по-настоящему».

Почему публиковаться рано — это про петли обратной связи

За девизом «выпускай рано» стоит не лень и не спешка, а холодный расчёт. Пока игра лежит у вас на диске, вы крутитесь внутри *одного* контура: придумал → сделал → сам же оценил. Беда в том, что вы — худший в мире судья собственной игры: вы знаете все правила наизусть, помните, где какая кнопка, и давно перестали замечать то, что новичка ставит в тупик. Ваш внутренний контур врёт.

Публикация замыкает *второй* контур — с живыми игроками. И этот контур говорит правду. Чем раньше вы его замкнёте, тем раньше начнёте принимать решения на фактах, а не на догадках. Это и называют **петлёй обратной связи**: выпустил версию → понаблюдал за реакцией → поправил → выпустил снова. Чем короче петля и чем чаще она прокручивается, тем быстрее игра становится хорошей. Разработчик, который выпустил десять сыроватых версий и десять раз послушал игроков, обгонит того, кто полгода в одиночку «полировал шедевр», — просто потому, что у первого было десять уроков от реальности, а у второго ни одного.

Есть и приятный побочный эффект, чисто человеческий: ранний релиз даёт *энергию*. Незаконченный проект давит чувством «ещё столько всего надо». А игра, которая уже жи-

вёт по ссылке и которой кто-то поиграл, наоборот заряжает: хочется сделать следующую версию лучше. Мотивация — топливо инди-разработки, и публикация заправляет бак.

Где в этом цикле находимся мы — и почему начали с конца

Теперь самое интересное: где во всём этом большом круге стоите *вы* прямо сейчас? Формально — в самом начале пути: впереди и разбор проекта, и изменение уровней, и написание механик, весь продакшн. Но на карте цикла вы только что сделали странную вещь — *шагнули сразу в финал*. Вы собрали билд и опубликовали игру раньше, чем что-либо в ней изменили.

Это сделано намеренно, и вот зачем. Обычно публикация — пугающая финальная стена, к которой новичок подходит измотанным, после месяцев работы, с мыслью «а вдруг не получится залить». Мы убрали эту стену в самом начале: теперь вы *знаете*, что путь до игроков проходим, потому что прошли его своими руками. Дальше, когда мы будем менять врагов и рисовать уровни, у вас за спиной уже будет рабочая публикация и понятная процедура «как выложить». Это меняет всё: вы не «учитесь ради далёкого результата», а улучшаете то, что уже живёт в интернете.

Так что наша книга устроена как одна большая итерация цикла. Глава 1 — это релиз «как есть» (чужой MVP, выпущенный без изменений). Главы 2–5 — это возвращение в продакшн: разбор, переделка под себя, проектирование и но-

вая сборка. А Глава 6 — пост-релиз и развитие: лидерборды, сохранения, достижения. Мы прокрутили петлю обратной связи один раз прямо на первой странице — чтобы дальше крутить её осознанно.

Что такое билд и зачем игру «собирают»

Пока вы работаете в Unity, проект существует как груда исходных файлов: сцены, скрипты на C#, текстуры, модели, настройки. Запускать это умеет только сам редактор. Чтобы игру смог запустить обычный человек без Unity, проект нужно **собрать** — то есть превратить в самостоятельное приложение. Этот процесс и его результат называют **билдом** (от англ. *build* — «собирать»).

Во время сборки Unity компилирует ваш код, упаковывает все ресурсы, оптимизирует их под выбранную платформу и складывает в папку, которую уже можно запустить или залить на сервер. По сути билд — это перевод вашего проекта с «языка разработчика» на «язык игрока». Именно папку DeepGalacticShooterBuild вы только что и получили.

Платформы, WebGL и сила первой обратной связи

Платформа публикации — это то, *где* живёт ваша игра: Windows, Android, iOS, PlayStation, Steam или браузер. Под каждую платформу собирается свой билд, со своими правилами и заморочками. И вот почему мы выбрали браузерную сборку (**WebGL**): это самый короткий путь от вас до игроков. Не нужно ничего устанавливать, не нужно проходить недели модерации в магазинах приложений, не нужен отдельный компьютер с нужной системой. Игрок просто открывает ссылку — и играет. Одна ссылка, которую можно отправить кому угодно в мессенджере, — это и есть магия WebGL.

И тут включается, пожалуй, самое ценное во всём цикле — **обратная связь от первых игроков**. Когда друг напишет «слушай, а враги слишком быстрые» или «классно, но кнопка не нажимается на телефоне», вы получите то, чего не даст никакая теория: взгляд со стороны. Первые игроки покажут, где игре весело, а где скучно или непонятно, — и именно их реакция направит ваш продакшн в нужную сторону. Поэтому мы и начали книгу с публикации: чтобы у вас как можно раньше появилась живая игра, которую можно показать, и живые люди, которые на неё ответят.

1.7 Что дальше

Давайте честно проговорим, что произошло — это важно для понимания всей книги.

Мы прошли весь конвейер целиком: **проект → Unity → билд → консоль → ссылка**. Пока на «готовом» проекте и без подробностей — но вы своими руками довели игру до публикации и убедились, что «выложить игру в интернет» проще, чем кажется. Теперь это для вас не страшный финал, а понятная процедура.

Дальше мы пройдём тот же путь **осознанно и детально**, превращая чужой шаблон в свою игру:

во **второй главе** установим Unity и подключим шаблон шутера;

в **третьей** разберём проект изнутри и начнём менять его под себя — уровни, врагов, предметы;

в **четвёртой** научимся думать как гейм-дизайнер и спроектируем свою игру на бумаге;

в **пятой** соберём собственную WebGL-версию и обновим ею тот же черновик, что вы создали сегодня;

в **шестой** добавим лидерборды, облачные сохранения, достижения и займёмся развитием игры.

Черновик, который вы только что завели, никуда не денется — в Главе 5 мы зальём в него уже **изменённую вами** сборку. Быстрый старт превратится в полноценный проект.

Выводы

В этой главе мы перевернули привычный порядок и начали с результата: вы скачали проект, собрали браузерную версию в Unity и опубликовали её на Яндекс.Играх, получив рабочую ссылку. Весь путь — от файлов проекта до игры, доступной другу по ссылке, — занял считаные минуты.

Это и есть главная идея книги: не «учиться годами ради далёкого результата», а быстро получить работающую игру и наращивать на неё своё. Теперь, когда вы уже видели финал, вернёмся к началу пути и соберём эту игру вдумчиво — со своим стилем, уровнями и механиками.

Закрепление главы
Что вы освоили

Эта глава была про самое смелое — провести чужой готовый проект DeepGalacticShooter до публикации по ссылке. Оглянитесь: вы уже умеете делать то, на что у многих уходят недели.

1. Скачивать и разворачивать готовый проект Unity с Яндекс.Диска и добавлять его в Unity Hub через **Add → Add project from disk**.

2. Открывать проект и проверять его на работоспособность: находить папку Assets/Project_Shooter, открывать сцену MainMenu и запускать игру кнопкой **Play**.

3. Собирать браузерный билд (WebGL): переключать платформу в **Build Profiles**, следить за порядком сцен (MainMenu первой, затем Level_1, EndScene) и получать готовую папку сборки.

4. Упаковывать сборку в .zip и заливать её в консоль разработчика Яндекс.Игр, создавать черновик приложения и выбирать поддерживаемые платформы.

5. Получать рабочую ссылку на игру и делиться ею — то есть проходить весь конвейер «проект → Unity → билд → консоль → ссылка» целиком.

6. Видеть жизненный цикл игры целиком — от идеи

и прототипа до релиза и поддержки — и понимать, почему публиковаться стоит рано.

Чек-лист

Пройдитесь по списку и убедитесь, что быстрый старт получился полностью:

1. Проект DeepGalacticShooter скачан, распакован в постоянную папку и добавлен в Unity Hub.

2. Проект открывается в редакторе, видна папка Assets/Project_Shooter, а в Scenes лежат три сцены — MainMenu, Level_1, EndScene.

3. Игра запускается по кнопке **Play** из сцены MainMenu, управление работает (бег — стрелки, выстрел — **Z**).

4. Платформа переключена на **Web (WebGL)**, в списке сцен MainMenu стоит первой, билд собран в папку DeepGalacticShooterBuild.

5. Содержимое папки сборки упаковано в .zip.

6. В консоли Яндекс.Игр создан черновик приложения, загружен архив и выбраны поддерживаемые платформы.

7. Получена рабочая ссылка вида yandex.ru/games/app/..., и игра открывается в браузере по этой ссылке.

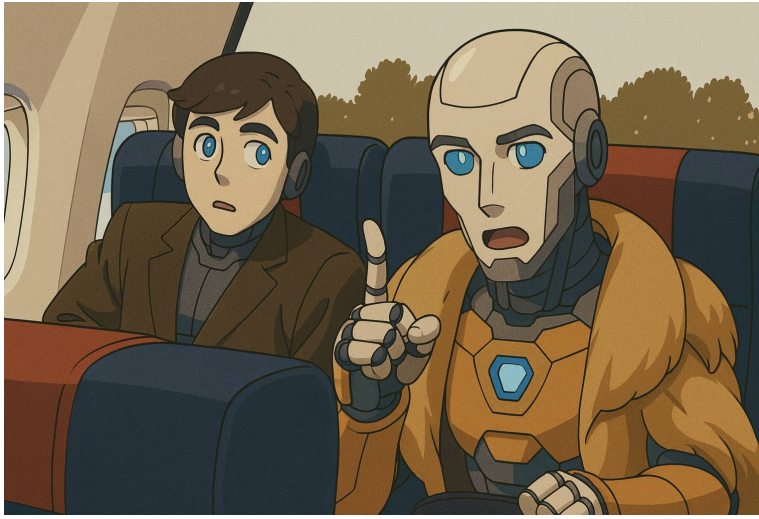
Задания для закрепления

Поделитесь и соберите первую обратную связь (лёгкое). Отправьте ссылку на свою публикацию двум-трём друзьям и попросите просто поиграть, ничего не подсказывая. Запишите в блокнот 3–5 их реакций: где было непонятно, что понравилось, где они застряли. Это ваш первый, ещё крошечный, контур обратной связи — тот самый, ради которого мы и публиковались так рано.

Пересоберите билд осознанно (средней сложности). Удалите папку DeepGalacticShooterBuild и .zip-архив и пройдите сборку заново, не подглядывая в шаги. Цель — чтобы маршрут «переключить платформу → проверить список сцен → Build → упаковать в zip» стал для вас привычным движением, а не разовым подвигом. Засеките время: уложиться стоит быстрее, чем в первый раз.

Обновите черновик новой сборкой (продвинутое). Зайдите в уже созданный черновик в консоли Яндекс.Игр и загрузите в него .zip со свежесобранным билдом из задания 2, заменив прежний. Убедитесь, что после автоматической проверки игра по ссылке по-прежнему открывается и запускается. Так вы отрепетируете именно ту операцию, которую проделаете в Главе 5, — обновление публикации уже изменённой вами игрой.

Глава 2. Установка Unity и шаблона шутера



— Мальчик, сосновый латте нам принеси. Мы игры делать будем!

Введение

В этой главе мы разберём, как установить Unity, настроить рабочее окружение и импортировать готовый шаблон шутера во вновь созданный проект. К концу этой главы вы уже сможете пройти первый уровень разрабатываемой игры прямо в игровом редакторе.

Кстати, ниже пара скриншотов — стартового окна и геймплея:

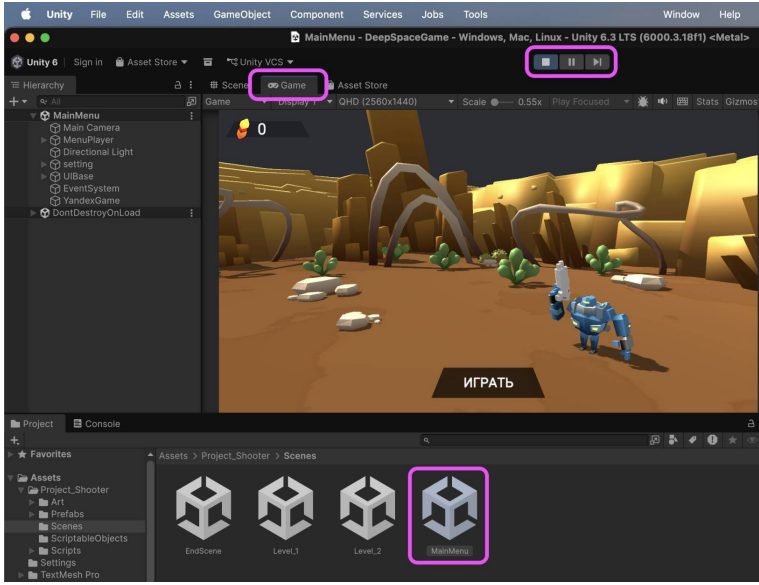


Рис. 2.1. Стартовое окно (главное меню) игры DeepGalacticShooter.

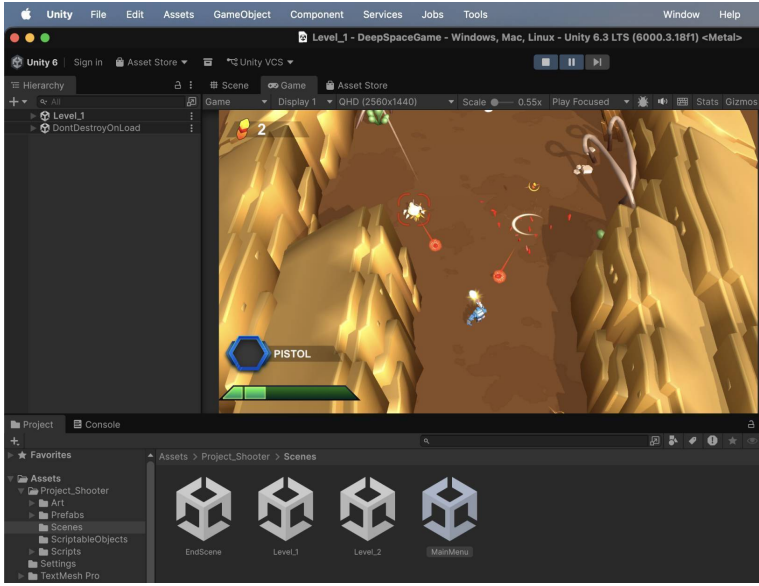


Рис. 2.2. Геймплей первого уровня — игрок, враги, интерфейс.

Начинать игру с шаблона в Unity — это не «читерство», а нормальный профессиональный подход. В современной разработке никто не пишет всё с нуля: веб-разработчики опираются на фреймворки (React, Django), мобильщики — на SDK и готовые компоненты, а в геймдеве мы используем движок, плагины и шаблоны. Это ускоряет работу, снижает

количество ошибок и позволяет сосредоточиться на геймдизайне и контенте.

Шаблон — это коробка, где уже лежат «колёсики, мотор и болтики»: базовый контроллер игрока, камера, система урона, враги, интерфейс, менеджер уровней. Вместо того чтобы неделями собирать «велосипед», вы стартуете с работающего прототипа и сразу двигаетесь к интересному — своей механике и визуальному стилю.

Что даёт старт с шаблона:

Скорость: за часы получаете то, что вручную заняло бы недели (инпуты, камера, меню, пауза, спавн врагов).

Архитектура по умолчанию: видите, как организованы сцены, папки, префабы, скрипты, и перенимаете паттерны.

Меньше багов: критически важные системы уже обкатаны и работают «из коробки».

Фокус на дизайне: время уходит не на инфраструктуру, а на геймплей, баланс и контент.

Учебный эффект: изучая готовый код, вы понимаете, как правильно стыковать системы (Input → Player → Camera → UI → Audio).

Реальный результат: быстрее доходите до сборки билда, тестов на друзьях и первых публикаций.

Совет: если вы всё же захотите узнать все тонкости создания игры с нуля, то у меня есть отдельная книга для начинающих — как собрать игру «Ловец драконов (Dragon Picker)», она пошаговая и уже опубликована на ЛитРес. Там

мы тренируемся на простом проекте: логика подбора объектов, счётчики, интерфейс — всё это пригодится и в шутере.

Итак, в этой книге мы соберём свой шутер на основе шаблона, отполируем геймплей и опубликуем игру на Яндекс.Играх — я пошагово покажу, как это сделать: от подготовки билда до загрузки, настроек витрины и проверок. Ваша цель — не просто «поиграться в редакторе», а выложить живую игру, в которую смогут играть другие. Погнали!

Теория: что такое игровой движок и как устроен Unity

Мы только что поставили на компьютер Unity — и прежде чем нырять в кнопки и меню, давайте на минуту остановимся и поймём, *что* именно мы установили. Слово «движок» звучит солидно и немного загадочно, но за ним прячется очень человеческая идея: не делать одну и ту же тяжёлую работу заново в каждой игре. Разберёмся, что движок берёт на себя, по каким законам он устроен внутри и почему Unity стал любимым инструментом миллионов разработчиков.

Что такое игровой движок и что он делает за вас

Представьте, что вы решили построить дом. Можно отлить собственные кирпичи, выковать гвозди, сплести провода — а можно взять готовые материалы и сосредоточиться на самом доме. **Игровой движок** — это и есть набор готовых «материалов и инструментов» для игр. Он берёт на себя задачи, которые одинаковы почти в любой игре, чтобы вы занимались тем, что делает вашу игру *вашей*.

Вот что движок делает за вас:

Рендеринг — рисует трёхмерную (или двумерную) картинку на экране: модели, свет, тени, эффекты. Вам не нужно знать, как видеокарта закрашивает каждый пиксель.

Физика — считает столкновения, гравитацию, отскоки. Пуля попадает во врага, ящик падает на пол — это всё физический движок.

Ввод — слушает клавиатуру, мышь, геймпад, касания экрана и переводит их в понятные события.

Звук — проигрывает музыку и эффекты, регулирует громкость, расставляет звуки в пространстве.

Сборка под платформы — упаковывает игру под Windows, Android, браузер и так далее (тот самый билд из Главы 1).

Без движка каждую из этих систем пришлось бы писать с нуля — это годы работы целых команд. С движком вы получаете их «из коробки» и сразу строите игру поверх.

Откуда вообще взялись движки

Чтобы прочувствовать, какое это благо, полезно знать, как жили до них. В восьмидесятые и ранние девяностые каждую игру писали практически с нуля и под конкретное «железо»: разработчик сам общался с видеочипом, сам считал, какие точки зажечь на экране, сам ужимал звук в скудную память. Перенести игру на другой компьютер означало переписать половину кода. Это было похоже на то, как если бы каждый строитель перед домом сначала изобретал бетон.

Перелом случился, когда разработчики заметили: от игры к игре *повторяется одно и то же*. Рисовать кадры, ловить нажатия клавиш, проигрывать звук, считать столкновения — всё это нужно почти любой игре, независимо от сюжета. Раз так, эту общую часть можно вынести в отдельный, многократно используемый «мотор». Само слово прижилось в эпоху первых трёхмерных шутеров: студии стали отделять «движок» (общую техническую начинку) от «контента» (уровней, врагов, графики) — и даже продавать движок другим студиям, чтобы те делали на нём свои игры. Так появилась сама идея: технологию пишут один раз, а игр на ней делают много.

Unity довёл эту идею до логического конца и добавил к ней две вещи, которые и сделали его таким популярным. Первая — *доступность*

Конец ознакомительного фрагмента.