UNITY ДЛЯ ИГР

ЭЛЕКТРОННАЯ КНИГА

ПРАКТИЧЕСКОЕ РУКОВОДСТВО ПО РАЗРАБОТКЕ ИГР В UNITY

ИЗДАНИЕ ДЛЯ UNITY 2020 LTS

Unity®

Содержание

Первые шаги

Путь к выпуску игры на новой платформе и новом движке может показаться дорогой в тысячу миль, но любое стоящее дело начинается с первого шага. А в любом путешествии полезно иметь карту.

Как опытный специалист отрасли, вы уже сталкивались с препятствиями, о которых пойдёт речь: проектированием и планированием, разработкой и тестированием, а затем выпуском и поддержкой.

Это руководство предназначено для опытных разработчиков, которые переходят на Unity или возвращаются к нему после долгого перерыва. Работаете ли вы в одиночку как независимый разработчик или в известной игровой студии, основная платформа Unity возьмёт на себя значительную часть работы. Вы сможете сосредоточиться на том, что умеете лучше всего: создавать увлекательные интерактивные проекты реального времени.

Unity стал лучше, чем когда-либо. Мы приглашаем вас познакомиться с множеством новых возможностей, призванных сделать разработку игр ещё быстрее, эффективнее и увлекательнее.

Более 1,5 миллиона активных создателей ежемесячно выбирают Unity, чтобы разрабатывать впечатляющие игры и выпускать их на самых разных устройствах. Наша легко расширяемая платформа объединяет художников, дизайнеров и разработчиков.

Это руководство поможет вам быстро освоить богатый набор возможностей Unity и интуитивно понятные рабочие процессы. Разрабатываете ли вы ролевой боевик для ПК и консолей или новую мобильную головоломку на основе физики, Unity поможет воплотить ваш замысел.

Обзор ключевых возможностей

Визуальный редактор

Unity Editor предоставляет комплексный интерфейс, который позволяет быстро повторять циклы прототипирования и тестирования и превращать концептуальный прототип в рабочую сборку.

Компонентная архитектура

Каждый объект сцены Unity представляет собой GameObject. Вы дополняете объекты компонентами - модульными, многократно используемыми частями, которые можно сочетать для получения нужной функциональности. Камера в Unity - это просто GameObject с компонентом Camera, а источник света - GameObject с компонентом Light. Такая компонентная система проста, гибка и хорошо масштабируется.

C#

В Unity используется C# - отраслевой стандарт и один из самых распространённых языков программирования в мире. C# - объектно-ориентированный типобезопасный язык с широкой поддержкой сообщества и корпоративного сектора.

Если вы уже знакомы с C-подобными языками или Java, освоить скрипты Unity будет несложно. Если стандартных возможностей Unity Editor недостаточно для вашего проекта, можно написать собственные скриптовые компоненты, которые будут органично работать со встроенными.

Unity поддерживает несколько IDE, включая Visual Studio, VS Code и JetBrains Rider. Подробности приведены ниже в разделе «Поддержка интегрированных сред разработки (IDE)».

Работа с Prefab

Не создавайте одно и то же дважды, если можно создать один раз и использовать повторно: так вы сэкономите время и повысите производительность во время выполнения. С помощью Prefab можно сохранять настроенные GameObject как ресурсы библиотеки, а затем создавать их экземпляры во время выполнения. Этот рабочий процесс поддерживает варианты: можно создавать своего рода «подклассы» Prefab и переопределять отдельные части «шаблона». Из небольших Prefab также можно собирать сложные вложенные Prefab, чтобы изменения сразу распространялись на сцену.

Unity Hub

Unity Hub - приложение-оболочка для установки Unity и управления версиями движка, лицензиями и проектами. Оно связывает проекты с конкретными версиями Unity, поэтому обновление движка не нарушит рабочий процесс и не создаст локальных конфликтов. Unity Hub также предоставляет удобный доступ к полезным учебным материалам и активному сообществу Unity.

Конвейеры рендеринга

В Unity можно выбрать один из нескольких готовых конвейеров рендеринга, которые помогут

выводить графику на экран. Universal Render Pipeline (URP)

ориентирован на широкую совместимость с платформами и максимальную производительность, а

High Definition Render Pipeline (HDRP) обеспечивает высокую детализацию изображения на ПК и

консолях. Шейдеры можно создавать на HLSL или с помощью узлового инструмента Shader Graph, не требующего написания кода.

HDRP предназначен для создания высокодетализированной графики.

Мультиплатформенная разработка

Unity следует принципу «создать один раз, развернуть где угодно». Наша цель - помочь вам охватить как можно более широкую аудиторию и развивать свою интеллектуальную собственность вместе с отраслью. Мы уже помогли добиться этого тысячам студий, включая Unknown Worlds, создателей серии Subnautica.

Создав контент в Unity Editor, вы можете выпустить его более чем на 20 платформах, включая PlayStation, Xbox, Nintendo Switch, ПК и мобильные устройства. С помощью Unity можно также создавать сборки для WebGL и XR-платформ, например Oculus Quest.

Пакеты и расширяемость

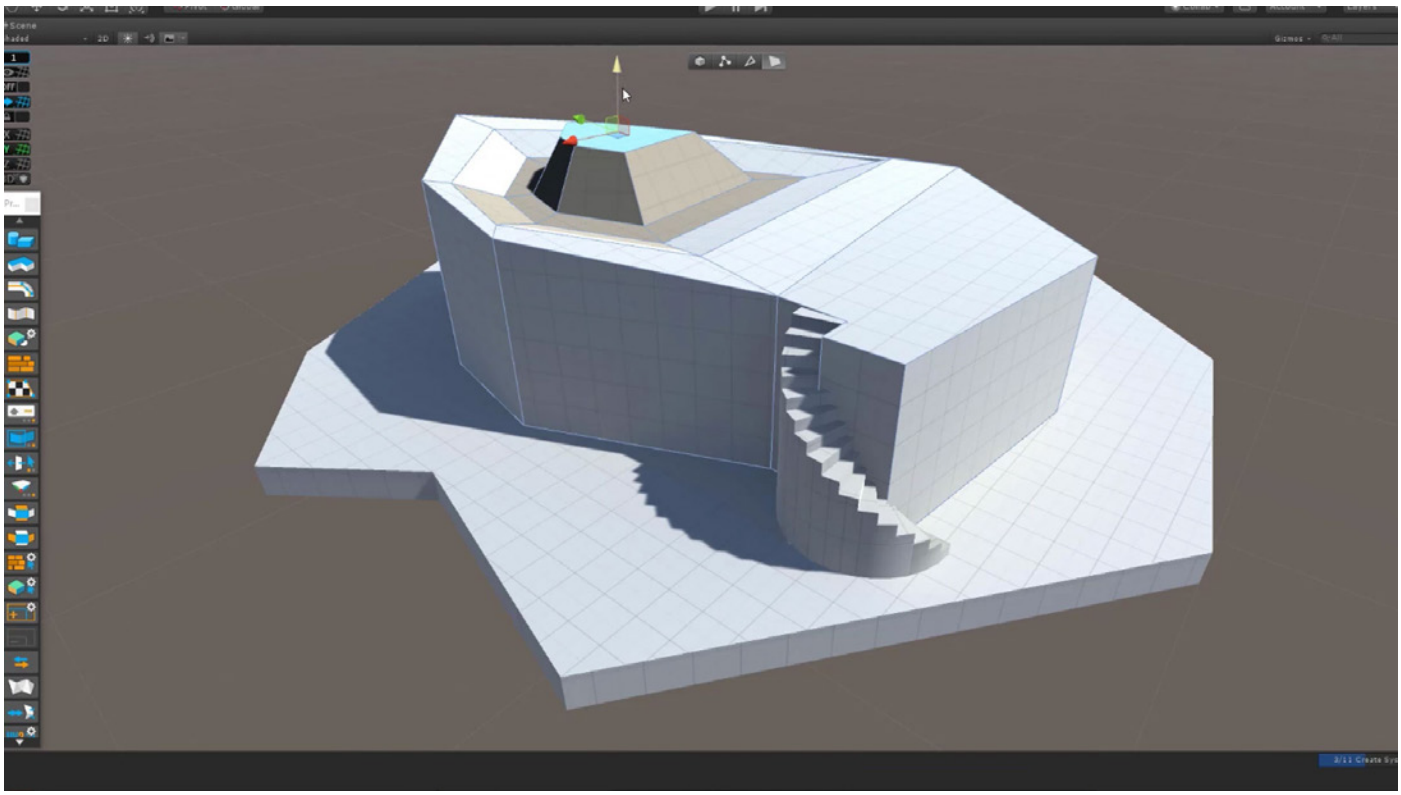
Package Manager позволяет находить и обновлять функции независимо от Unity Editor. Unity поддерживает репозиторий пакетов, расширяющих основные возможности платформы.

Не перегружайте проект и импортируйте только необходимое. Пакеты позволяют знакомиться с функциями со статусом Preview или Experimental до их готовности к рабочему применению и заранее планировать использование новых технологий в будущих проектах.

Создание игрового мира

Unity включает набор инструментов для создания окружения. ProBuilder сочетает средства 3D-моделирования и проектирования уровней: с его помощью можно моделировать простую геометрию и создавать блокинг игровых уровней.

Для 2D-проектов в Unity предусмотрены такие инструменты, как Sprite Shape и Tilemap, которые помогают компоновать двумерное окружение. Художники по окружению могут создавать природные ландшафты с помощью Terrain и рисовать по ним встроенными кистями.



ProBuilder помогает создавать прототипы уровней.

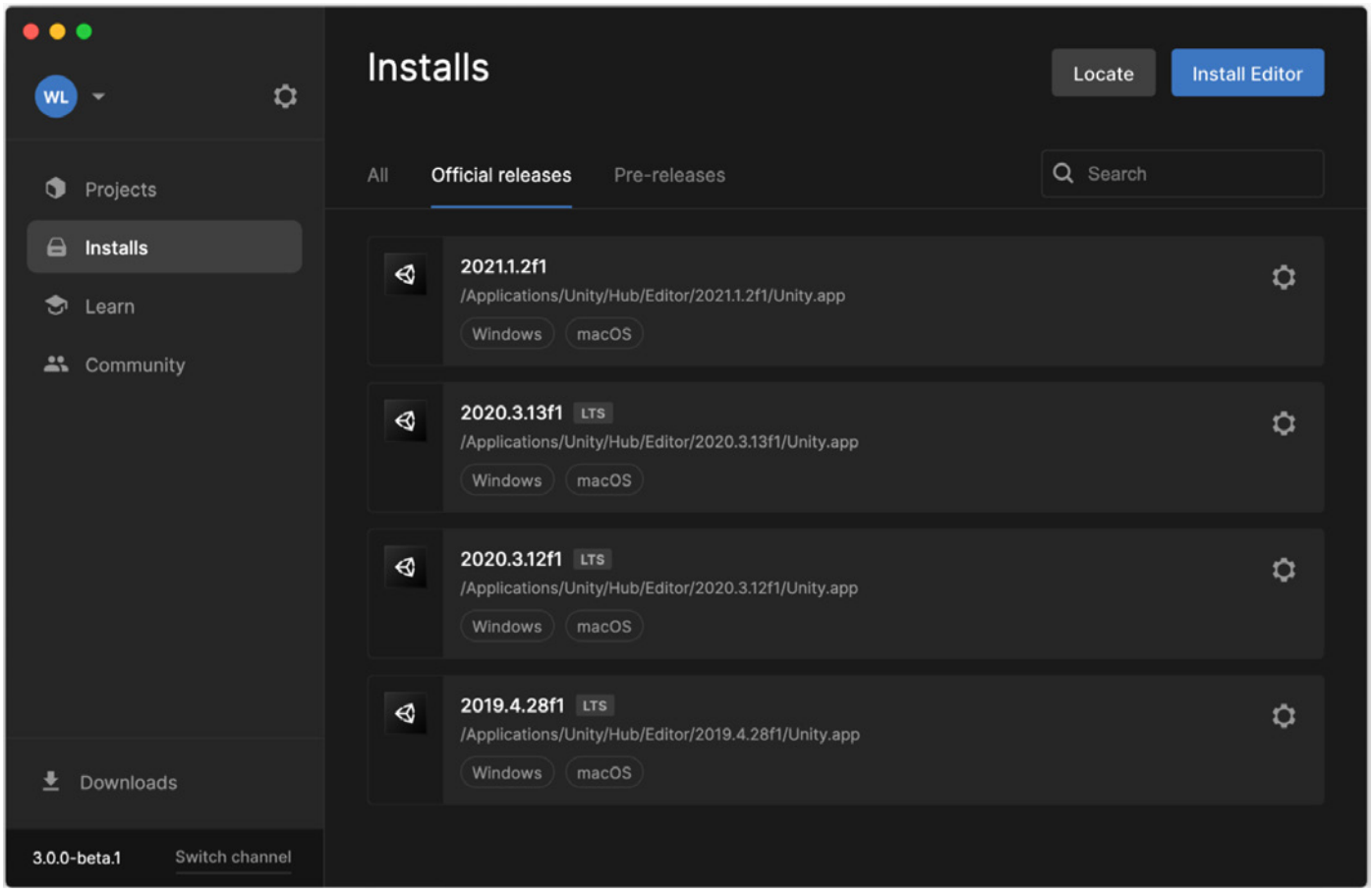
Активное сообщество

Unity широко распространён и располагает активной базой пользователей. На основной платформе Unity разработчики создали и монетизировали более половины из 1000 самых популярных мобильных игр в Apple App Store и Google Play. Благодаря сообществу, объединяющему более миллиона активных создателей в месяц, вам доступна обширная база знаний по вопросам разработки на официальных форумах Unity и в других интернет-сообществах.

Unity Hub

Начните знакомство с Unity с Unity Hub - инструмента для управления всеми проектами и установленными версиями Unity. В Unity Hub можно установить одну или несколько версий Unity Editor, создать новый проект или открыть существующий.

Чтобы получить Unity Hub, откройте сайт Unity и выберите Download Hub. Установите Unity через интерфейс, а перед запуском движка активируйте индивидуальную или командную лицензию. Для лицензий Unity Plus и Pro также требуется действительный серийный номер.



Unity Hub помогает управлять проектами и установленными версиями Unity.

В Unity Hub доступны разделы Projects, Installs, Learn и Community.

В разделе Installs можно выборочно устанавливать разные версии Unity, включая предварительные и официальные выпуски. Добавляйте только нужные модули, чтобы точно настраивать поддержку конкретных платформ и инструментов разработки.

Для рабочих проектов рекомендуется использовать текущую версию Unity с долгосрочной поддержкой (LTS). Unity 2020.3 LTS предлагает следующее:

- стабильная основа для проектов, готовящихся к выпуску;

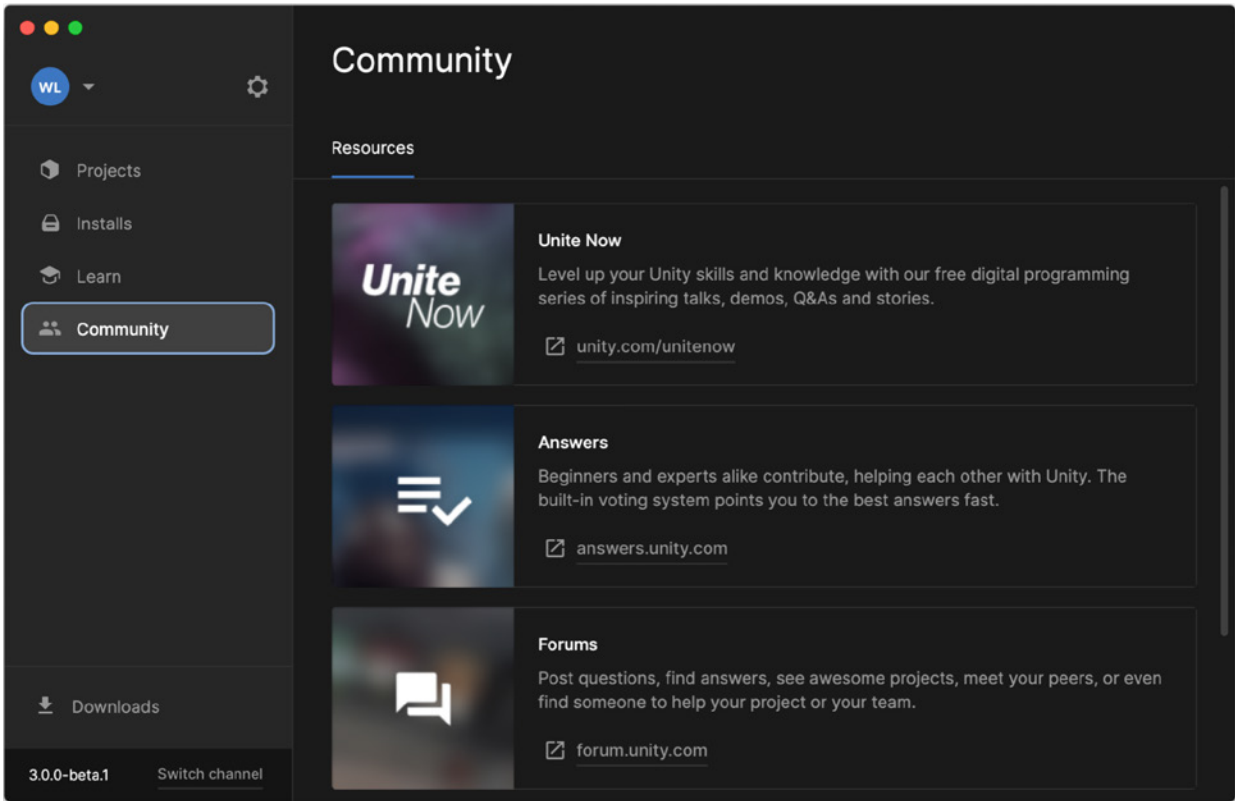
- обновления раз в две недели до середины 2022 года, затем ежемесячные обновления до марта 2023 года, то есть в течение двух лет после первоначального выпуска;

Обновления версий LTS включают только исправления, повышающие удобство использования и стабильность движка.

На экране Projects отображаются активные проекты. Для каждого проекта задайте целевую платформу и закрепите конкретную версию Unity, чтобы обновление не внесло ошибки, способные остановить работу.

Экран Learn предоставляет доступ к множеству учебных материалов и руководств, которые помогут освоиться в движке.

На экране Community доступны другие популярные ресурсы, например Unity Blog и выступления Unite Now. Если у вас возник технический вопрос, задайте его или прочитайте обсуждение на страницах Answers либо Forums.



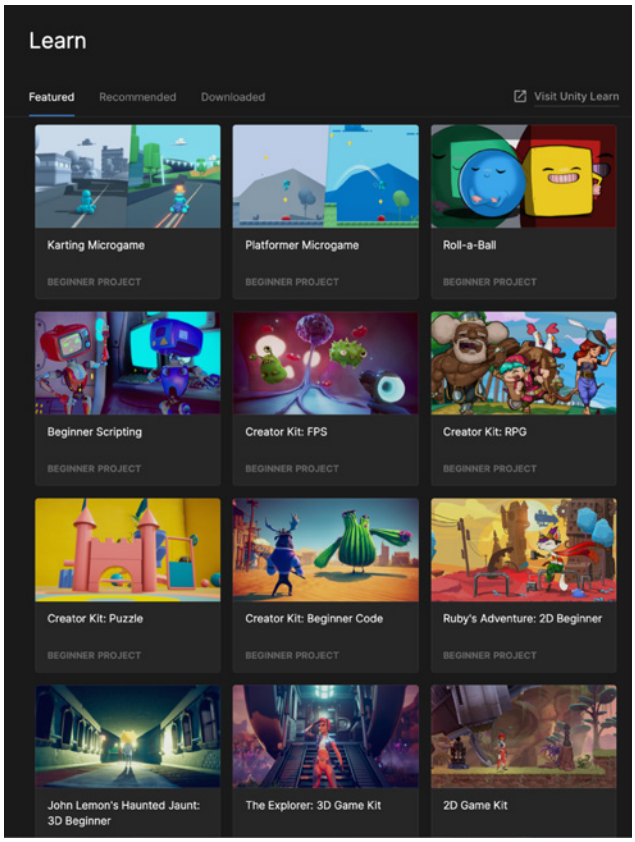
Экран Community содержит ссылки на несколько интернет-ресурсов.

Примеры проектов

Вероятно, лучший способ познакомиться со средой разработки - заглянуть в вертикальный срез игрового прототипа. Такие примеры доступны в нескольких местах и помогут вам быстрее начать изучение Unity.

Учебные проекты

Экран Learn в Unity Hub предоставляет доступ к разнообразным руководствам и учебным материалам, включая множество примеров проектов, которые можно загрузить и импортировать непосредственно в Unity.



Изучите стартовые проекты на экране Learn.

Creator Kits и Microgames можно рассматривать как мини-шаблоны с документированными примерами кода. Многие из этих проектов рассчитаны на начинающих и не требуют написания кода, но опытные разработчики могут изучить их скрипты и ресурсы. Исследование и изменение проектов Learn - хороший способ познакомиться с распространёнными приёмами разработки проектов на Unity.

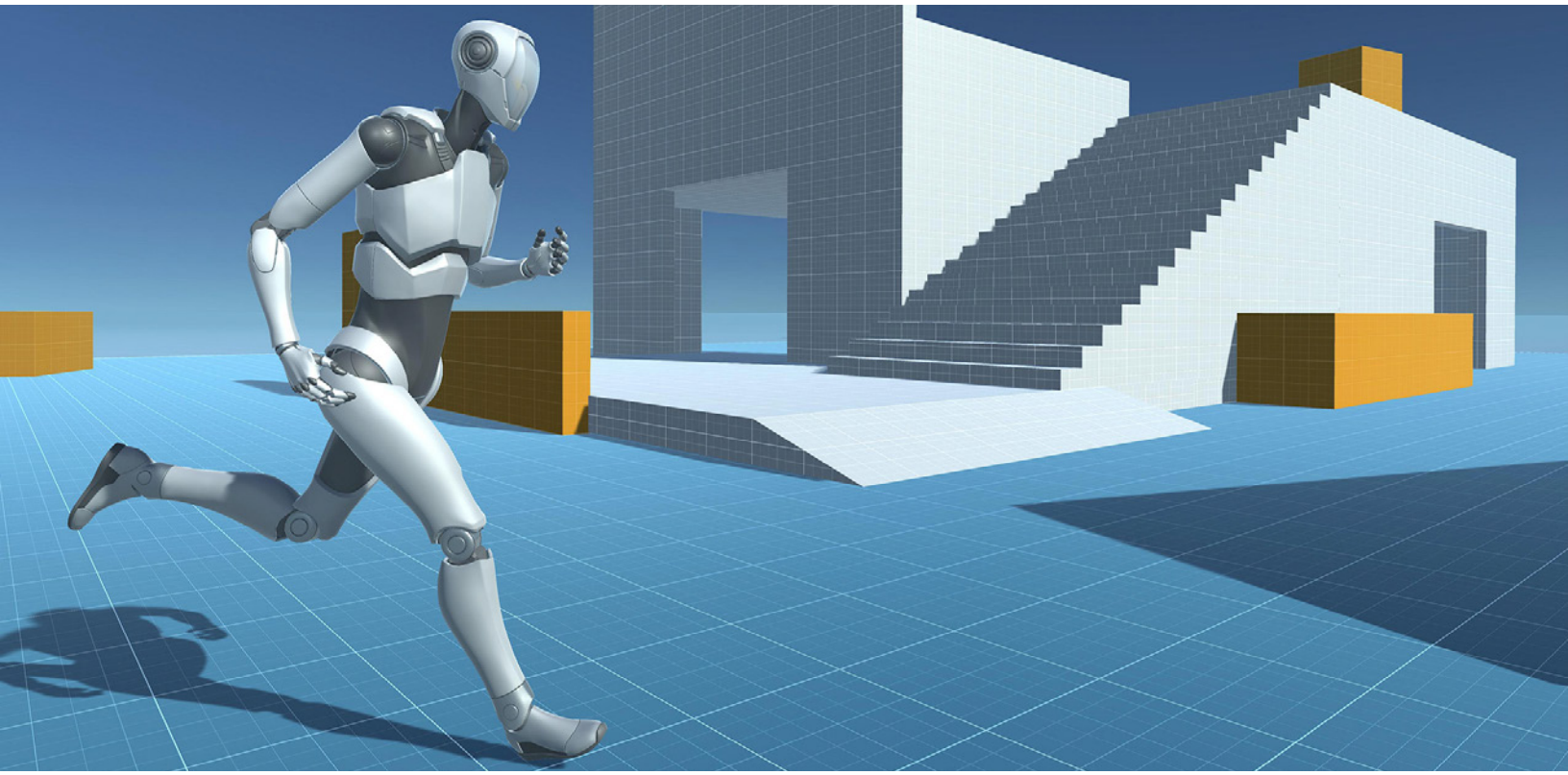
Unity позволяет сохранить изменённый проект, а затем найти его в Unity Hub и продолжить работу. Исходные проекты без изменений останутся доступными на экране Learn.

Ресурсы Unity Asset Store

Unity Asset Store - это площадка сообщества с бесплатными и платными ресурсами. Здесь можно найти сторонние инструменты и графические материалы почти для всего, что команда не хочет разрабатывать самостоятельно. Доступны ресурсы для готовых проектов и прототипов, расширения Unity Editor, инструменты для скриптов и даже почти готовые к выпуску проекты. Многие пакеты содержат подробные шаблоны, демонстрирующие важнейшие аспекты разработки на Unity и способные сэкономить дни или недели работы.

В Asset Store представлены десятки тысяч ресурсов, и каждый день появляются новые пакеты. Если вы только начинаете, хорошей отправной точкой станут следующие пакеты Unity:

- Пакет Starter Assets содержит бесплатные облегчённые базовые контроллеры персонажа от первого и третьего лица. Они показывают, как управлять камерой с помощью пакетов Cinemachine и Input System. Starter Assets - First Person Controller и Starter Assets - Third Person Controller демонстрируют создание прототипа контроллера персонажа на тестовой сцене.



Starter Assets - Third Person Controller

- Lost Crypt - 2D Sample Project: демонстрационный 2D-проект с боковой прокруткой, показывающий совместную работу специализированных 2D-инструментов Unity. В проекте представлены 2D Animation, 2D Sprite Shape, 2D Tilemaps, 2D Lights, Shader Graph for 2D, Secondary Textures и постобработка с Volume. Подробнее о проекте рассказывается в этой публикации блога.

Lost Crypt демонстрирует 2D-возможности Unity.

Если вы только начинаете работать с Unity, ознакомьтесь также с подборкой лучших ресурсов для новых пользователей от команды Asset Store.

Репозиторий Unity на GitHub

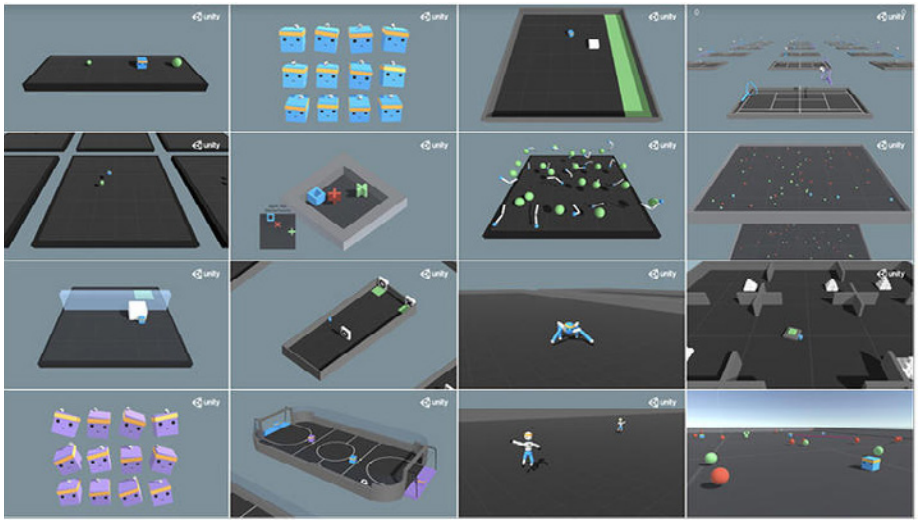
В репозитории Unity Technologies на GitHub размещён ряд проектов, демонстрирующих отдельные функции и темы. Не все они работают с последней версией Unity, но через Unity Hub можно установить подходящую версию. Многие репозитории всё ещё имеют статус Experimental или Preview, поэтому позволяют оценить новые возможности для следующего рабочего проекта.

Вот несколько
примечательных проектов:

- Boss Room - небольшой пример кооперативной игры, созданный на основе новой экспериментальной библиотеки сетевого кода и MLAPI. Проект знакомит с принципами и шаблонами организации многопользовательского игрового процесса. В нём доступен один уровень подземелья и четыре класса персонажей, с помощью которых можно отрабатывать сражения со стилизованными противниками и боссом, давшим проекту название.

Boss Room

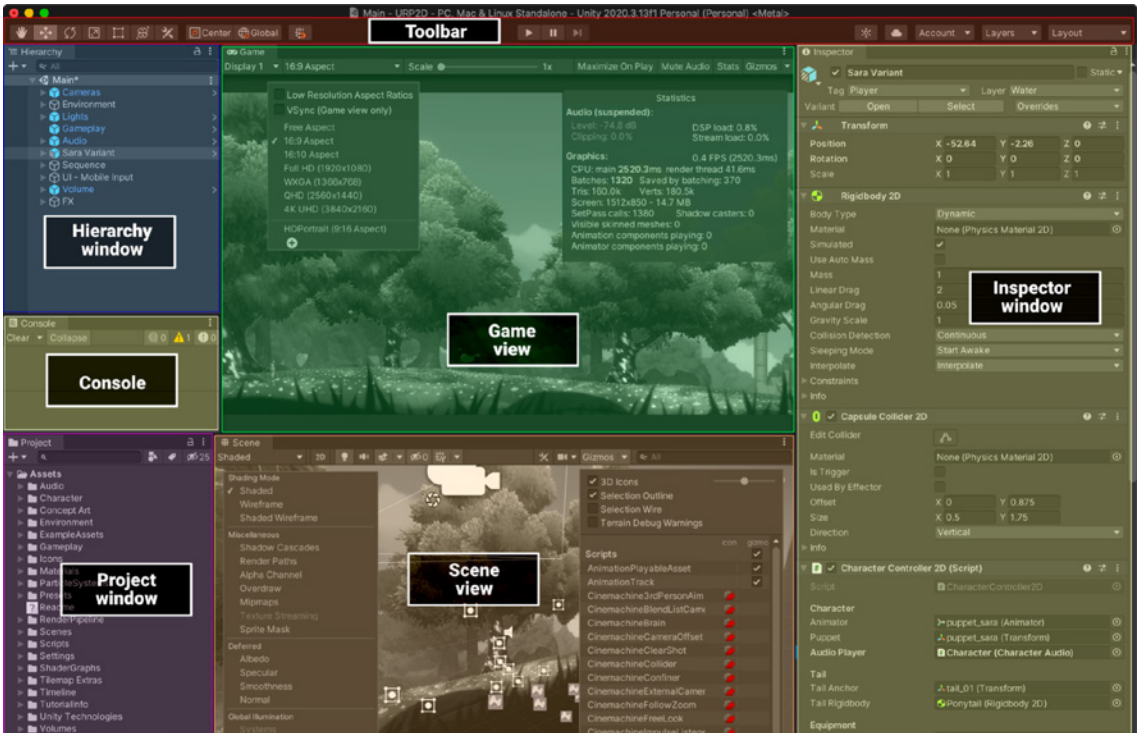
- Unity Machine Learning Agents Toolkit (ML-Agents) - проект с открытым исходным кодом, который позволяет обучать интеллектуальных агентов в играх и симуляциях для 2D-, 3D- и VR/AR-приложений. Исследователи могут использовать предоставленный Python API, чтобы применять к игровым агентам обучение с подкреплением, обучение по примерам и другие методы машинного обучения.



Инструментарий
ML-Agents

Интерфейс Unity Editor

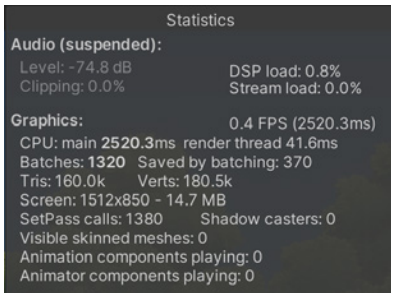
Unity предлагает гибкий модульный пользовательский интерфейс. Если вы раньше не работали с Unity Editor, начните с этого краткого обзора основных окон.



Основные окна Unity Editor

Основные окна

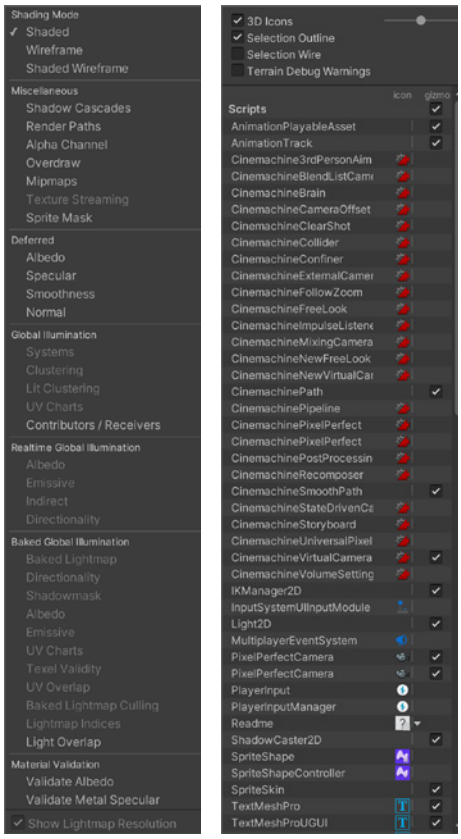
Окно Game: здесь можно предварительно увидеть изображение с основной камеры игры и оценить, как игрок будет взаимодействовать с игрой. В правом верхнем углу находится окно Rendering Statistics со статистикой графики и звука, которая помогает оптимизировать приложение. Элемент Aspect Ratio в левом верхнем углу позволяет проверить несколько разрешений экрана или задать собственное целевое разрешение.



Статистика рендеринга

Aspect Ratio (соотношение сторон)

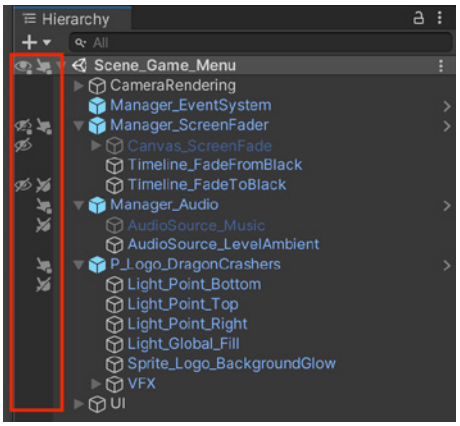
Окно Scene: здесь показаны объекты, составляющие каждую сцену. Именно в этом окне Unity Editor вы в основном будете размещать модели, источники света, физические тела и другие элементы. На панели управления можно изменить Draw Mode, что полезно для поиска неполадок и быстрой диагностики. Меню Gizmos управляет значками и накладной графикой отдельных GameObject.



Меню Draw Mode и Gizmos

Окно Hierarchy: здесь показаны объекты активной сцены и родительско-дочерние связи между ними.

Hierarchy предоставляет удобные средства переопределения видимости объектов и возможности выбирать их в окне Scene. Эти настройки интерфейса не влияют на окно Game. Они упрощают навигацию по сложным сценам с большим количеством GameObject.



Переопределение видимости и возможности выбора объектов в окне Scene

Окно Inspector: здесь отображаются свойства и настройки выбранных объектов, а также доступны широкие возможности настройки. Можно открыть отдельное окно Inspector для конкретного GameObject, ресурса или компонента, чтобы упростить сравнение игровых переменных и объектов сцены. Inspector позволяет изменять эти свойства во время выполнения, не обращаясь каждый раз к коду, благодаря чему тестирование становится интерактивнее, а работа идёт быстрее.

Окно Project: Unity хранит все ресурсы, из которых создаётся игра (Prefab, текстуры, модели, анимации и скрипты), в виде файлов на диске. Окно Project предоставляет удобный доступ к этим файлам. Его можно представить как каталог или библиотеку содержимого, где находятся игровые ресурсы. Для поиска конкретных ресурсов или их типов используйте строку Search.

Панель инструментов: она расположена в верхней части Unity Editor и закреплена в строке заголовка окна приложения. Кнопки панели управляют режимом Play и позволяют проверить работу публикуемого приложения. Инструмент Hand перемещает область просмотра сцены, а Move, Rotate, Scale, Rect Transform и Transform служат для изменения GameObject.

Окно Console: используется для отладки и отображает ошибки, предупреждения и другие сообщения, создаваемые Unity (см. класс Debug). Всё, что Unity или ваш код выводит в Console, также записывается в файл журнала.

Режим Play в Unity Editor

В окне Game отображается изображение с одной или нескольких камер приложения; оно показывает итоговый вид опубликованного приложения. Любые изменения, внесённые в режиме Play, временны и сбрасываются после выхода из него. Интерфейс Unity Editor затемняется, напоминая, что изменения в режиме Play не сохраняются.

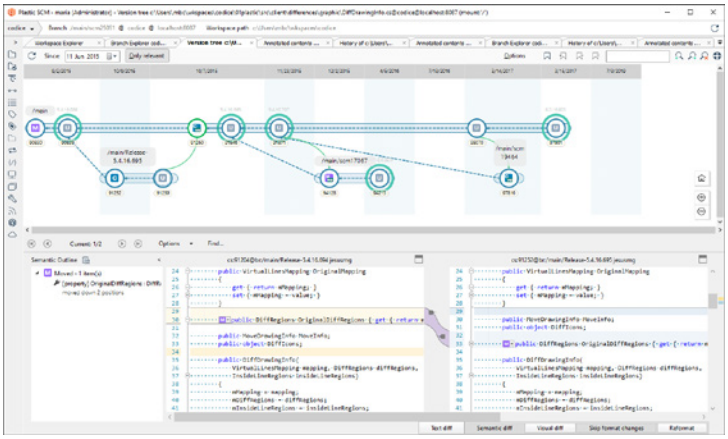
Помните: чтобы управлять тем, что игрок видит во время выполнения, потребуется одна или несколько камер. Дополнительные сведения приведены на странице компонента Camera.

Более подробные сведения о каждой части интерфейса можно найти в соответствующем обзоре Unity Learn.

Контроль версий

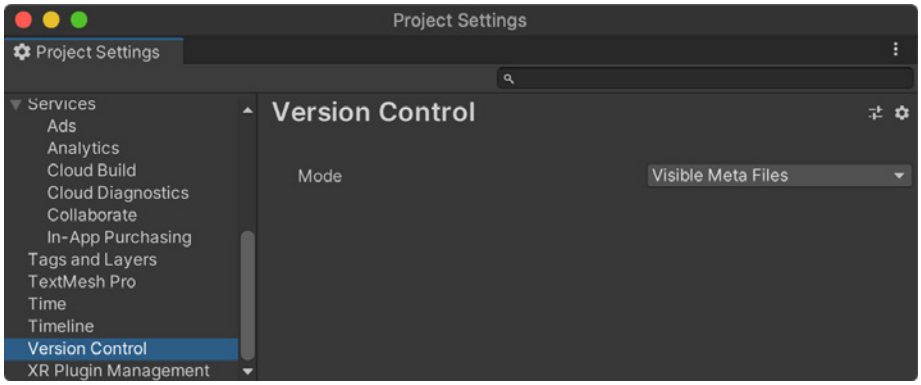
Будь то Plastic SCM, Git, Perforce или другая система, Unity позволяет выбрать решение для контроля версий, которое лучше всего подходит вам и вашей команде.

Unity Editor поддерживает встроенную интеграцию с двумя ведущими системами контроля версий: Perforce и Plastic SCM. Для работы с любой из них в проекте Unity необходимо предварительно настроить соответствующий сервер.



Plastic SCM упрощает контроль версий благодаря лаконичному интерфейсу.

Перед началом работы с системой контроля версий откройте Project Settings > Editor. Убедитесь, что для параметра Asset Serialization Mode выбрано значение Force Text. Unity использует сериализацию для загрузки ресурсов с диска и их сохранения. В режиме Force Text файлы сцен хранятся в текстовом формате, что упрощает слияние изменений в системе контроля версий.



Version Control в окне Project Settings

Откройте Project Settings > Editor > Version Control. В зависимости от используемой системы контроля версий выберите режим Plastic SCM, Perforce или Visible Meta Files (для внешней системы контроля версий, например Git).

Plastic
SCM

Plastic SCM

Plastic SCM - рекомендуемая система контроля версий для разработки игр на Unity. Она особенно удобна при работе с большими двоичными файлами (более 500 МБ), поэтому графические ресурсы можно контролировать столь же полно, как и исходный код.

Plastic SCM надёжно создаёт резервные копии как графических, так и программных ресурсов. Кроме того, понятный визуальный интерфейс упрощает ветвление и управление версиями.

Вот некоторые ключевые преимущества
Plastic SCM:

- Скорость: в большой кодовой базе ветви создаются в Plastic SCM значительно быстрее, чем в Perforce или Git.

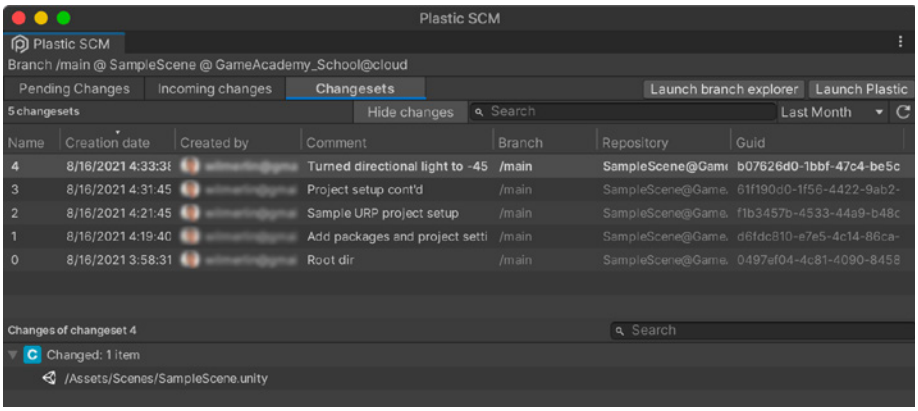
- Простота для пользователей, не работающих с кодом: художники команды могут использовать режим Gluon. Этот упрощённый рабочий процесс позволяет получить рабочую копию своей части проекта, внести изменения и зафиксировать их, скрывая ненужные элементы интерфейса.

- Облачный хостинг: Plastic SCM предлагает собственное решение Plastic Cloud Edition, рассчитанное на полностью распределённые команды. Если команда не хочет обслуживать собственные серверы и предпочитает хранить всё в сети, попробуйте Plastic Cloud Edition.

- Распределённый контроль версий: настройте отдельные репозитории в удалённых офисах, чтобы команды всегда работали «локально». Это рабочий процесс, похожий на Git, но рассчитанный на больший масштаб. Графический интерфейс позволяет отправлять и получать изменения, а также разрешать удалённые конфликты.

- Окно сравнения: Plastic SCM предоставляет полноценный интерфейс для просмотра изменённых, добавленных и удалённых файлов. В графический интерфейс также входит отдельный инструмент сравнения изображений, который помогает сопоставить две версии ресурса текстуры.

При работе с Plastic SCM откройте окно Plastic SCM через меню Window > Plastic SCM, чтобы просмотреть файлы в списке изменений.



Окно Plastic SCM

На вкладке Pending changes перечислены все локальные изменения, ожидающие фиксации в системе контроля версий. Вкладка Incoming changes позволяет просматривать все входящие изменения и конфликты, а также обновлять локальный проект. При изменении проекта в правом верхнем углу окна Plastic SCM появляется уведомление Incoming changes.

Дополнительные сведения об окне контроля версий приведены в документации по интеграции систем контроля версий. Также можно посмотреть видео «Контроль версий для игр с Plastic SCM» с конференции Unite Now или начать с книги Plastic SCM.

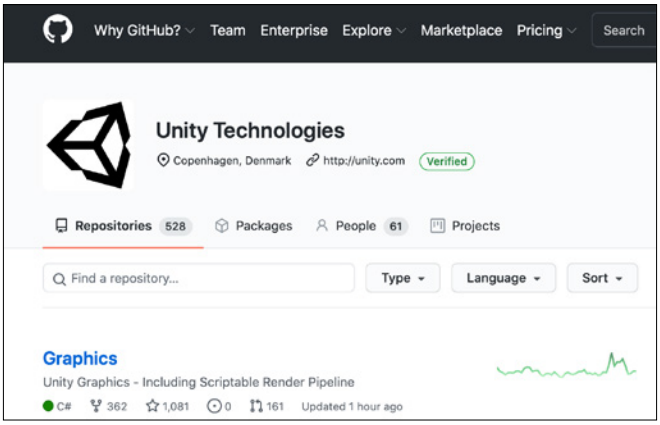
Git

Разработчики Unity могут использовать и внешние системы контроля версий, например Git. Однако сначала проект придётся настроить вручную.

Чтобы использовать Git с Unity, создайте пустой локальный репозиторий и при необходимости синхронизируйте его с облаком через GitHub. Обязательно подключите Git LFS (Large File Storage), чтобы эффективнее контролировать версии крупных ресурсов, например графики и звука. Unity поддерживает файл .gitignore, который поможет определить, что следует добавлять в репозиторий Git, а что нет.

Для более удобной работы с хостингом GitHub установите плагин GitHub for Unity. Это расширение с открытым исходным кодом позволяет просматривать историю проекта, экспериментировать в ветвях, фиксировать изменения и отправлять код на GitHub, не выходя из Unity.

Также обратите внимание на удобные графические клиенты, например GitKraken, SourceTree или GitHub Desktop.



Создание репозитория с помощью GitHub.

Пошаговая настройка Unity для работы с Git показана в видео «Введение в контроль версий», где на примере проекта объясняются основы использования графического клиента GitHub.

Расширение GitHub for Unity

Perforc
e

Если вы выбрали Perforce в качестве системы контроля версий, управляйте кодом, графикой и ресурсами игрового движка с помощью Helix Core. Helix Core бесплатен для команд до пяти пользователей и 20 рабочих пространств.

Perforce обеспечивает хорошую производительность даже для удалённых команд, участники которых находятся в разных странах. Многие крупные и независимые игровые студии используют Perforce как основную систему контроля версий.

Для
начала:

- Выполните настройку, описанную в документации Unity Version Control.

- Ознакомьтесь с руководством по настройке Perforce Helix Core для Unity.

- Подробные сведения о Perforce Helix Core приведены в документации Perforce.

Организация проекта

По мере роста проекта необходимо поддерживать такой уровень организации, который позволит масштабировать его с учётом размера команды и требований приложения. Эти общие рекомендации помогут заложить базовую структуру проекта и сцен.

Окно Project

В окне Project отображаются все файлы, связанные с проектом. Это каталог содержимого, где находятся ресурсы и другие файлы приложения.

Unity хранит исходные файлы непосредственно в проекте вместе с отдельными файлами .meta. В файлах .meta содержатся связанные с ресурсом данные, необходимые движку и Unity Editor.

Unity также импортирует каждый ресурс в оптимизированный формат, который движок использует во время выполнения. Обработанные ресурсы помещаются в папку Library: она служит кэшем, и добавлять её в систему контроля версий не нужно.

В окне Project есть несколько элементов интерфейса, упрощающих навигацию:

- Щёлкните правой кнопкой мыши, чтобы открыть контекстное меню часто используемых команд: создания и импорта ресурсов, показа полного пути на диске и других.

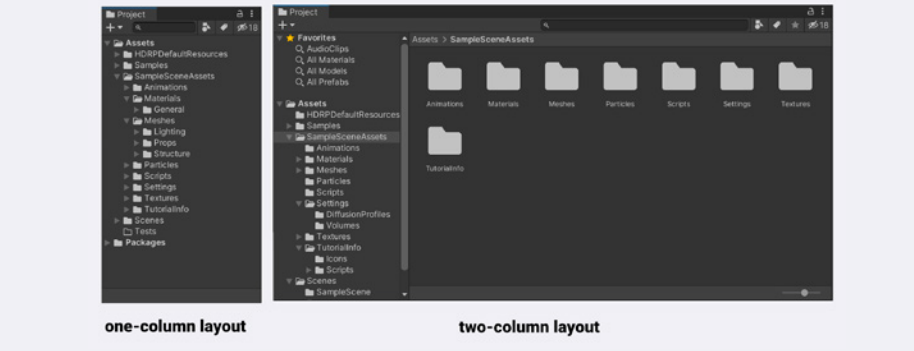
- По мере роста проекта используйте поле Search для поиска ресурсов. Чтобы найти ресурсы определённого типа, примените фильтр с синтаксисом t:. Например, t:Material покажет все ресурсы материалов в проекте. Это упрощает навигацию по крупным проектам.

- Перетащите часто используемую папку в область Favorites в верхней части интерфейса. Результат поиска также можно сохранить в Favorites с помощью кнопки Save Search.

- Можно изменить и компоновку самого окна. Откройте меню More Items в правом верхнем углу и выберите One

Column Layout или Two Column Layout. В двухколоночном режиме есть

дополнительная панель с визуальным предпросмотром каждого файла.

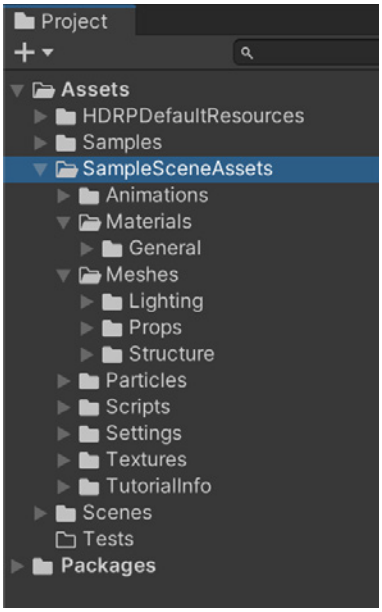


Одноколоночный и двухколоночный режимы

В окне Project находится папка Assets. Она содержит ресурсы, используемые при сборке игры. Если проект создан из шаблона, внутри должны быть подпапки для нескольких распространённых типов ресурсов. Большинство этих папок определяет пользователь, однако Unity резервирует некоторые имена для специальных целей. Обязательно ознакомьтесь со списком специальных имён папок.

Ниже перечислены распространённые подпапки для организации проекта. Конкретный набор зависит от предпочтений команды и особенностей проекта. Главное - будьте последовательны: составьте руководство по стилю и придерживайтесь его.

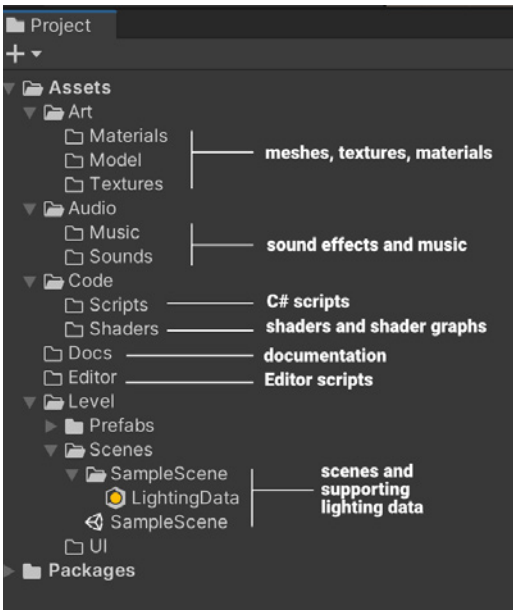
Анимация	В этой папке хранятся клипы анимации и файлы их контроллеров, а также ресурсы Timeline для внутриигровых кинематографических сцен и данные рига для процедурной анимации.
Аудио	К звуковым ресурсам относятся аудиоклипы и микшеры, используемые для сведения эффектов и музыки.
Editor	Здесь находятся скриптовые инструменты для Unity Editor, которые не включаются в целевую сборку.
Шрифты	Шрифты, используемые в игре.
Gizmos	Значок Gizmo помогает увидеть GameObject в окне Scene или Game, особенно если у него нет меша. Файлы изображений для таких значков храните в папке Gizmos.
Материалы	Эти ресурсы описывают свойства затенения поверхности.
Меши	Здесь хранятся модели, созданные во внешнем приложении для создания цифрового контента (DCC).
Частицы	Управляйте в Unity симуляциями частиц, созданными с помощью Particle System или Visual Effect Graph.
Prefab	Многократно используемые GameObject с заранее настроенными компонентами. Добавляйте их на сцену, чтобы собирать уровни и элементы игрового процесса.
Скрипты	Здесь находится весь написанный пользователями код игрового процесса.
Сцены	Unity хранит небольшие функциональные части проекта в ресурсах сцен. Обычно они соответствуют игровому уровню или его части.
Настройки	Здесь хранятся параметры конвейеров рендеринга HDRP и URP.
Шейдеры	Эти программы выполняются на GPU как часть графического конвейера.
Текстуры	Файлы изображений могут содержать текстуры материалов и поверхностей, элементы интерфейса и карты освещения со сведениями об освещении.
ThirdParty	Ресурсы из внешних источников, например Asset Store, храните отдельно. Так сторонние ресурсы и скрипты проще обновлять. Их фиксированную структуру иногда нельзя изменять.



Пример сцены из шаблона HDRP содержит несколько папок ресурсов.

На странице руководства «Поддерживаемые типы ресурсов» распространённые ресурсы описаны подробнее. В качестве примера эффективной организации папок можно использовать проекты Template или Learn. Вы не обязаны ограничиваться перечисленными именами папок, но этот список станет хорошей отправной точкой, которую можно расширять по мере роста проекта.

Разумеется, структуру папок можно адаптировать к потребностям конкретного проекта и предпочтениям команды, как показано на рисунке ниже.



Организуйте папки в соответствии с потребностями проекта, но, выбрав структуру, придерживайтесь её.

Структура папок и именование

Единственного правильного способа организовать проект нет, однако мы рекомендуем придерживаться следующих общих правил:

- Документируйте соглашения об именовании и структуру папок.

Руководство по стилю и/или шаблон проекта упрощают поиск и организацию файлов.

- Какое бы соглашение об именовании вы ни выбрали, обязательно соблюдайте

его последовательно. Не отступайте от выбранного руководства по стилю или шаблона. Если правила именования всё же нужно изменить, обработайте и переименуйте все затронутые ресурсы одновременно с помощью скрипта.

- Не используйте пробелы в именах файлов и папок. Инструменты командной строки Unity

могут некорректно работать с путями, содержащими пробелы.

- Отделяйте области тестирования и экспериментов. Создайте отдельную папку

для сцен, не предназначенных для выпуска, и экспериментов. Подпапки с именами пользователей помогут разделить рабочую область между участниками команды.

- Не создавайте лишних папок в корне. Обычно храните материалы

в папке Assets. Не создавайте дополнительные папки в корне проекта без крайней необходимости.

Сцены

В Unity вы работаете с содержимым в сценах. Они содержат объекты игры и могут представлять главное меню, отдельные уровни и многое другое. В каждой сцене размещаются окружение, препятствия и декорации, обычно соответствующие одному уровню игры. Так приложение можно проектировать и собирать по частям, сохраняя модульность.

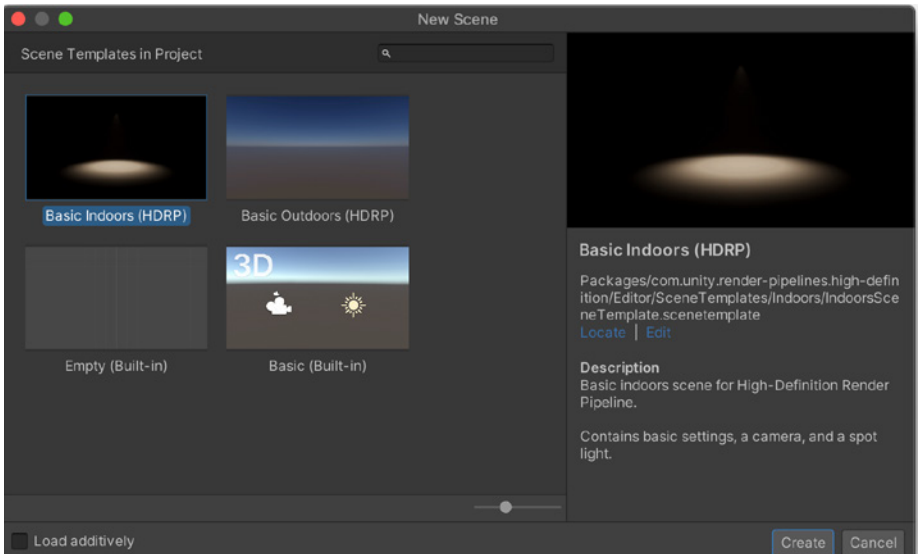
Файлы сцен сами являются ресурсами и хранятся на диске. Если для Asset Serialization выбран режим Force Text, они представлены текстовыми файлами; в противном случае по умолчанию используется двоичный формат.

Сцена часто представляет уровень игры или его часть. Демонстрационные и простые игры могут состоять всего из одной сцены, но в большинстве коммерческих игр для каждого уровня используется отдельная сцена со своим окружением, персонажами, пользовательским интерфейсом и другими элементами.

В проекте можно создать любое количество сцен, однако их структура способна существенно повлиять на производительность. Подробнее об организации сцен и производительности рассказывается в нашем руководстве по оптимизации производительности на мобильных устройствах.

Чтобы представить разные части игры и наполнить приложение содержимым, потребуется создавать, загружать и сохранять сцены. Типичный «поток сцен» предполагает загрузку другой сцены по событию. Например, после щелчка по элементу интерфейса сцена меню может загрузить основную игровую сцену. Сцены можно загружать по одной либо разделить элементы между ними и использовать аддитивную загрузку.

При создании новой сцены Unity позволяет выбрать один из шаблонов сцен. Например, HDRP 3D Sample Scene поставляется с несколькими шаблонами. Можно определить собственные шаблоны сцен, чтобы ускорить работу и предоставить всем участникам команды одинаковый исходный набор ресурсов и настроек.



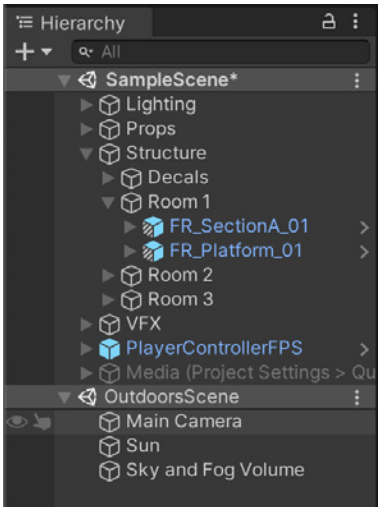
Шаблоны сцен в HDRP

Unity предоставляет SceneManagement API для загрузки сцен и управления ими из скриптов. Подробнее о потоке сцен рассказывается в этом руководстве Learn.

Окно Hierarchy

Окно Hierarchy отображает все GameObject в загруженной сцене. Среди них могут быть модели, камеры и экземпляры Prefab. Чтобы изменить родительско-дочерние связи GameObject, просто перетащите его.

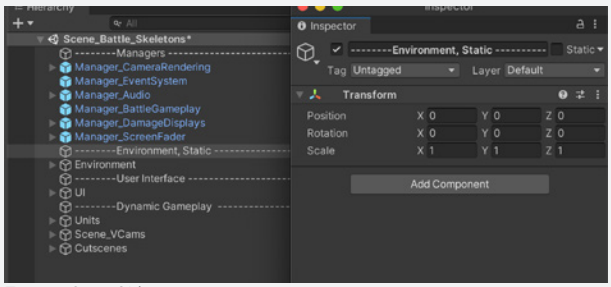
При добавлении или удалении объектов в окне Scene они также добавляются в Hierarchy или удаляются из него, и наоборот. Во время выполнения Hierarchy может показывать несколько загруженных сцен, каждая со своими GameObject.



Пример сцены из шаблона HDRP содержит несколько папок ресурсов.

Общие рекомендации по сценам и иерархиям

- Используйте именованные пустые GameObject как разделители. Тщательно организуйте сцены, чтобы объекты было легко находить. Не создавайте лишние GameObject: каждый из них влияет на производительность. Соотносите удобство структуры с затратами и не усложняйте иерархию без необходимости.



Пустые GameObject служат разделителями иерархии.

- Размещайте служебные Prefab и пустые GameObject в начале мировых координат. Если Transform не задаёт позицию объекта, оставляйте его в точке (0, 0, 0): это упрощает код и преобразования между локальными и мировыми координатами.

- Располагайте поверхность мира на уровне y = 0. Тогда объекты проще ставить на землю. Для игровой логики, ИИ и физики рассматривайте мир как двумерное пространство в плоскости XZ.

- Разделяйте динамические и статические объекты. Объекты, создаваемые во время выполнения, удобно хранить под пустым контейнером. Неподвижную геометрию уровня разместите в другой части иерархии: так проще применять разные методы освещения, например lightmap и Light Probe.

- Правильно задавайте родительско-дочерние связи. Группируйте объекты по назначению и руководствуйтесь здравым смыслом: например, колёса должны быть дочерними объектами кузова автомобиля. Избегайте лишней вложенности - более плоская иерархия обычно производительнее.

Стандарты именования

Хотя единого стандарта именования GameObject не существует, примите во внимание следующие правила и рекомендации для своего проекта.

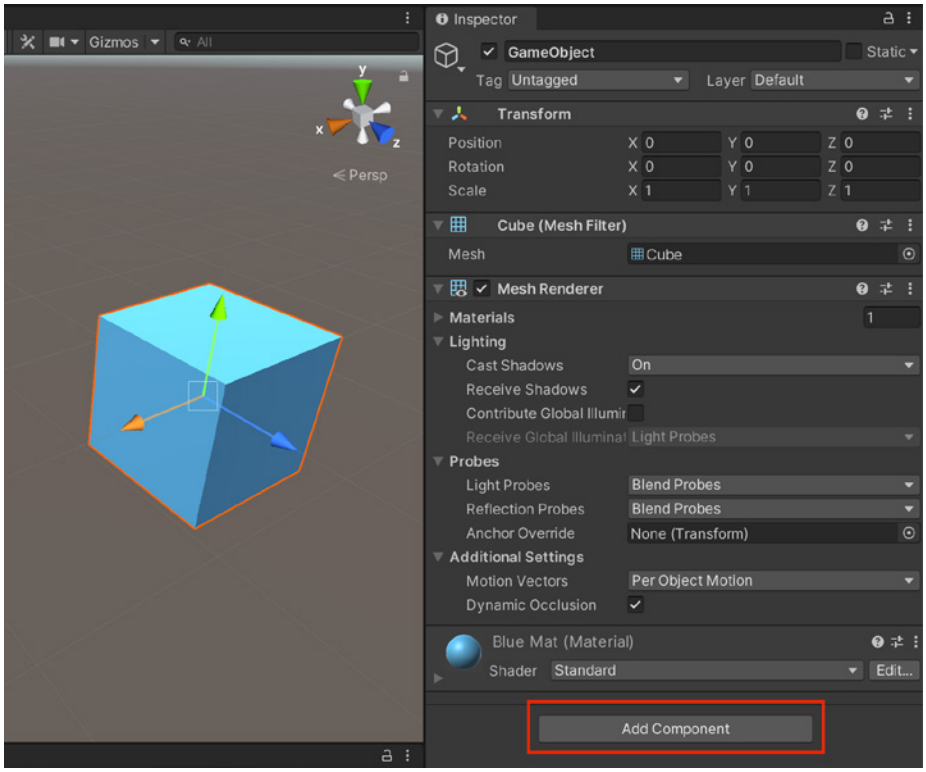
Правило	Пример
Используйте понятные имена без сокращений. Выбирайте варианты, которые легко произнести, запомнить и понять другому участнику команды.	largeButton, LargeButton или leftButton Не: lButton
Используйте camelCase или PascalCase. Не ставьте пробелы в именах объектов: это повышает читаемость и точность набора.	OutOfMemoryException, dateTimeFormat Не: Outofmemoryexception, datetimeformat
Не злоупотребляйте подчёркиваниями и дефисами. Они полезны для вариантов и состояний; начальное подчёркивание ставит имя первым по алфавиту.	Активные состояния: EnterButton_Active EnterButton_Inactive Карты: Foliage_Diffuse Foliage_Normalmap Уровни детализации: Building_LOD1 Building_LOD0
Числовые суффиксы используйте только для элементов последовательности.	Для пути: Node0, Node1, Node2 и т. д.
Следуйте правилам именования из дизайн-документа.	HighSpellTower RedDragonLair

Как и любые правила именования, выберите подходящий для вашей команды вариант и применяйте его последовательно.

GameObject и компоненты

Игровой процесс и интерактивность в Unity создаются с помощью GameObject и компонентов. Это основные составляющие проекта Unity. Их можно создавать через интерфейс или с помощью скриптов.

Все элементы в иерархии игры являются GameObject. По сути, GameObject представляют собой пустые контейнеры, которые сами по себе ничего не делают. Они могут обозначать в игре самые разные сущности: персонажа, элемент окружения, эффект частиц и многое другое.



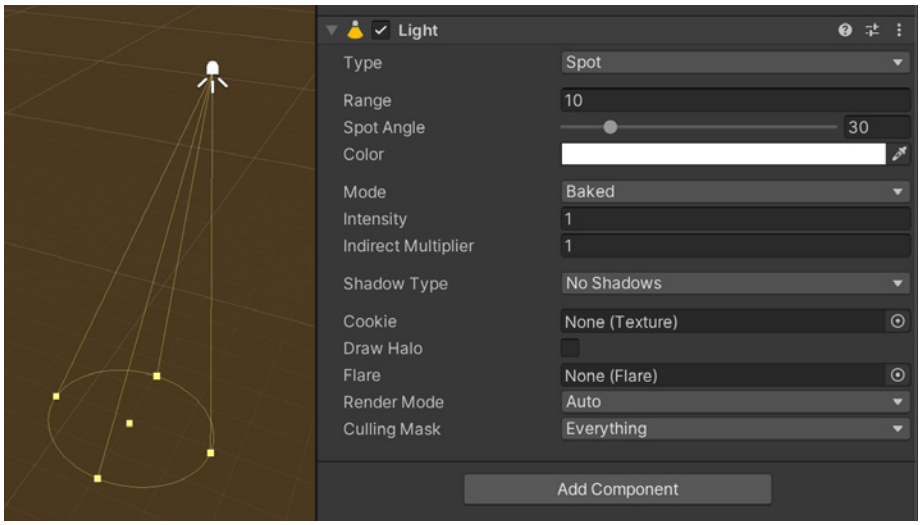
Добавление компонента изменяет функциональность GameObject.

Чтобы GameObject стал функциональной частью игры, к нему нужно добавить специальные компоненты. Каждый GameObject может содержать несколько компонентов. Они часто работают совместно друг с другом или обмениваются данными с компонентами других GameObject.

В зависимости от требуемой функциональности к GameObject добавляют разные сочетания компонентов. С помощью скриптов можно создавать и собственные компоненты.

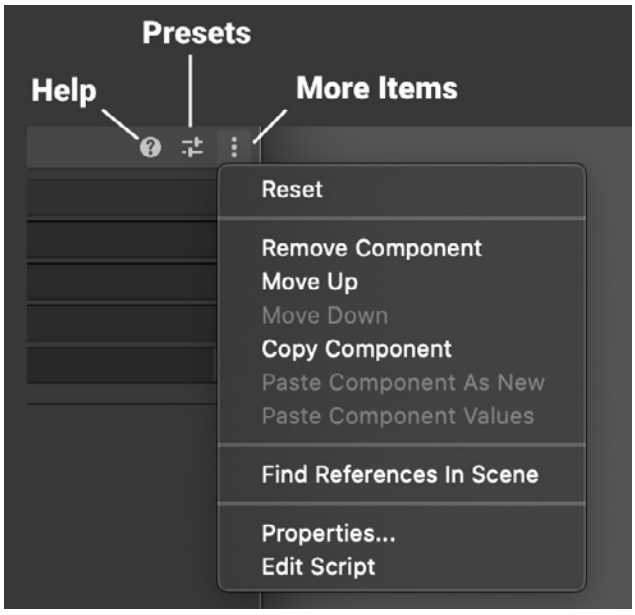
Inspector

У компонентов может быть любое число редактируемых свойств, или переменных, значения которых изменяются в окне Inspector. Эти переменные также можно изменять из пользовательских скриптов.



В окне Inspector измените дальность, цвет и интенсивность источника света из примера.

Используйте Inspector для ввода и сброса значений. Значок Help обеспечивает быстрый доступ к документации, а функция Presets позволяет быстро сохранять и загружать значения. Через меню More Items можно менять порядок компонентов, а также копировать и вставлять значения.



Используйте меню Help, Preset и More Items для управления компонентами.

Компонент Transform

У каждого GameObject в Unity есть компонент Transform, задающий его положение, поворот и масштаб. Удалить этот компонент нельзя. Для любого дочернего GameObject значения Transform в Inspector отображаются в локальных координатах относительно Transform родительского объекта.

Масштаб Transform определяет размер меша в Unity относительно его размера в приложении для 3D-моделирования. Это важно для физического движка, который считает, что одна единица мирового пространства равна одному метру.

Как и многие 3D-приложения, Unity внутренне представляет повороты кватернионами, но для удобства редактирования показывает в Inspector эквивалентные углы Эйлера.

Советы по работе с Transform

- Перед добавлением дочернего объекта обнулите Transform родителя: задайте

позицию и/или поворот родительского Transform в (0, 0, 0), прежде чем сделать другой Transform дочерним. При изменении родительско-дочерних связей Unity компенсирует существующие значения, чтобы дочерний объект сохранил прежнее положение в мировом пространстве.

Если Transform родительского объекта сохраняет значения по умолчанию, локальные координаты дочернего объекта совпадут с глобальными. Это упрощает настройку.

- Используйте правильный режим Tool/Pivot. При перемещении, вращении и

масштабировании следите за настройками инструментов на Toolbar. В Unity можно переключаться между режимами Pivot и Center: манипулятор Gizmo будет расположен соответственно в опорной точке модели или в центре выделения. Также учитывайте переключение между системами координат Local и Global.

Настройки инструментов

- Если объект должен находиться или стоять на полу, разместите его pivot у

основания, а не в центре. Это позволяет точно ставить персонажей и объекты на пол. В 3D-проекте так также удобнее работать в виде сверху на плоскости XZ.

- Все меши должны быть ориентированы в одном направлении. Это касается

мешей персонажей и других объектов, у которых есть направление вперёд. Единообразие значительно упрощает многие вычисления во время выполнения.

- Сразу задайте правильный масштаб. Создавайте ресурсы так, чтобы

все они импортировались с коэффициентом масштаба 1 и значением 1 по осям масштаба X, Y и Z.

Сверяйте масштаб с кубом (GameObject > 3D Object > Cube).

Если масштабирование необходимо, избегайте неоднородного масштаба корневого Transform. Иначе при добавлении дочерних Transform возможны неожиданные результаты.

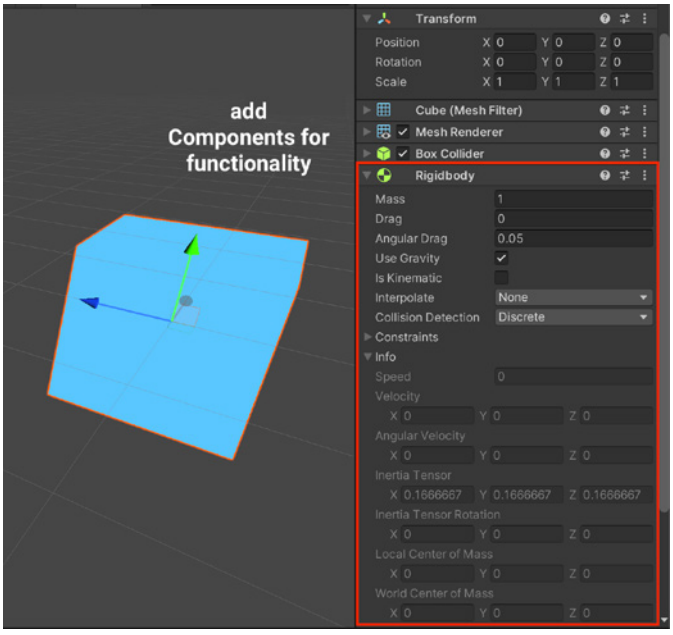
- Если для симуляции физики используются компоненты Rigidbody, моделируйте из расчёта 1 единица = 1 метр. В противном случае скорректируйте масштаб с помощью

параметра Mesh Scale Factor в Import Settings модели. Крупные объекты кажутся падающими в замедленном темпе, поскольку скорость воспринимается относительно: при одинаковой скорости в мировых единицах движение мыши и слона выглядит по-разному из-за различий в размерах их тел.

Симуляция физики может быть корректной, даже если воспринимаемая скорость кажется неверной. Подробнее о влиянии масштаба меша на физику см. на справочной странице компонента Rigidbody.

Встроенные компоненты

Добавляйте компоненты к выбранному GameObject через меню Components. Например, **если добавить Rigidbody (Add Component > Physics > Rigidbody) к** 3D-кубу, его свойства появятся в Inspector.



Добавляйте встроенные компоненты, чтобы расширять функциональность GameObject (в этом примере - Rigidbody).

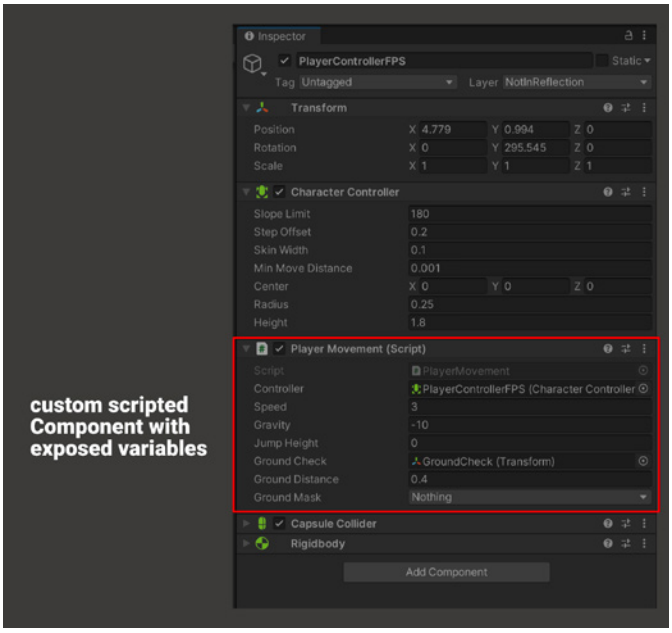
Нажмите Play, пока пустой GameObject всё ещё выбран: физический движок Unity заставит его падать под действием силы тяжести. Компонент Rigidbody наделил прежде пустой GameObject функциональностью.

Комбинации компонентов позволяют создавать уникальный игровой процесс. Встроенные компоненты предоставляют звук, рендеринг, навигацию ИИ и другие возможности. По мере дальнейшего знакомства с Editor вы откроете ещё больше функций.

Пользовательские компоненты

Хотя встроенные компоненты добавляют GameObject множество возможностей, вам неизбежно потребуется реализовывать собственное поведение с помощью Unity Scripting API. Пользовательские скриптовые компоненты могут вызывать игровые события, проверять столкновения, применять физику, реагировать на пользовательский ввод и делать многое другое.

Когда вы создаёте скрипт и прикрепляете его к GameObject, он появляется в Inspector этого GameObject как встроенный компонент. Во время проверки игрового дизайна в Editor его значения можно настраивать. После сохранения и компиляции в проекте скрипты становятся компонентами.



Пример пользовательского скриптового компонента

Скрипты, прикреплённые к GameObject, наследуются от встроенного класса MonoBehaviour. Взаимодействие MonoBehaviour друг с другом лежит в основе игрового процесса в Unity.

Unity Scripting API подробно описывает все классы, доступные в UnityEngine и UnityEditor. Рекомендуем сначала ознакомиться с руководством, а затем переходить к более глубокому изучению.

Prefab

Система Prefab в Unity позволяет создавать, настраивать и хранить GameObject как многократно используемый ресурс, сохраняя все его компоненты, значения свойств и дочерние GameObject. Ресурс Prefab служит в проекте шаблоном для создания новых экземпляров Prefab в сцене.

Если вы хотите многократно использовать GameObject, настроенный определённым образом, преобразуйте его в Prefab.

Использование Prefab

Экземпляры Prefab можно использовать для многих GameObject в иерархии. Prefab может представлять всё, что требуется воспроизводить многократно: персонажа игрока или NPC, объект окружения либо часть геометрии уровня. Распространённые примеры использования Prefab:

- Ресурсы окружения: если одно и то же дерево или здание нужно несколько раз использовать на уровне, преобразуйте соответствующие меши в Prefab.

- Неигровые персонажи (NPC): персонаж определённого типа может появляться

в игре несколько раз и на разных уровнях. С помощью переопределений можно различать такие экземпляры по поведению, внешнему виду или звукам.

- Снаряды и усиления: используйте Prefab, если во время выполнения нужно создавать экземпляры GameObject, которых изначально не было в сцене. Например, лазерное оружие может при каждом выстреле создавать экземпляр эффекта частиц.

- Главный персонаж игрока: Prefab игрока можно разместить

в начальной точке каждого уровня игры или в отдельной сцене, загружаемой аддитивно.

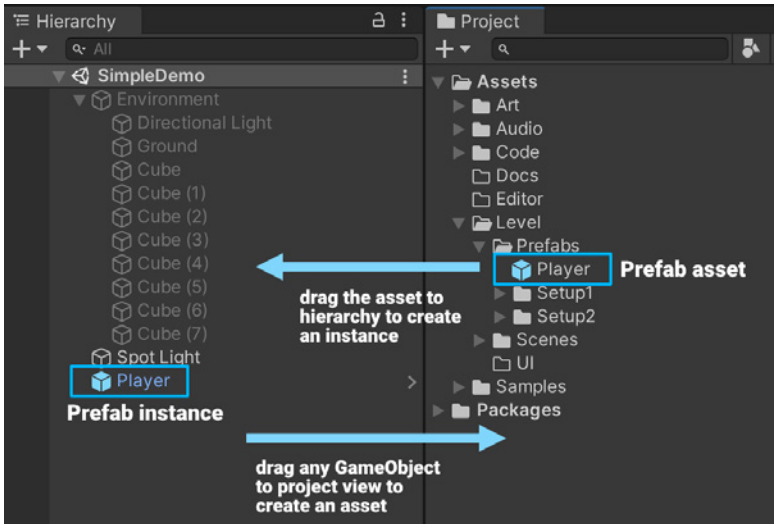
Изменения можно автоматически синхронизировать с Prefab, что упрощает редактирование ресурсов и распространение изменений на сцену.

Создание ресурсов и экземпляров Prefab

Чтобы создать ресурс Prefab, перетащите GameObject из окна Hierarchy в окно Project. GameObject вместе с его компонентами и дочерними GameObject станет новым ресурсом в окне Project.

В зависимости от одноколоночного или двухколоночного режима окна Project ресурсы Prefab отображаются значком синего куба либо миниатюрой GameObject.

Аналогично, чтобы создать экземпляр Prefab, перетащите ресурс Prefab из окна Project в окно Hierarchy или Scene.



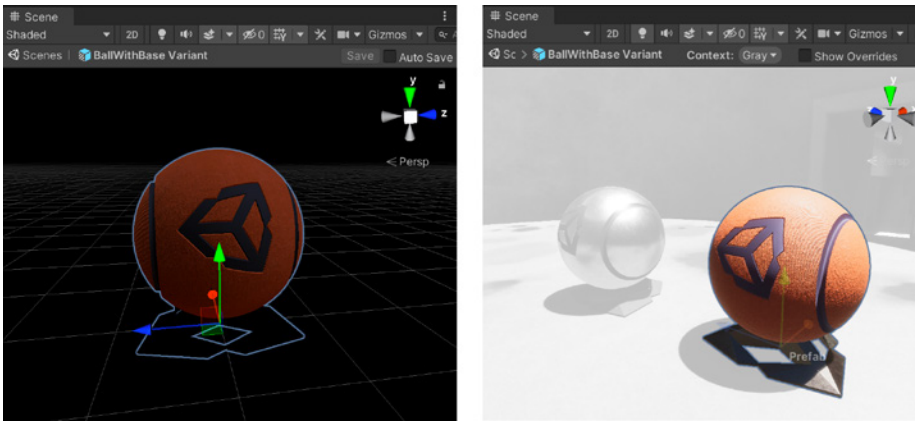
Создание ресурса и экземпляра Prefab

Изменения, внесённые в ресурс Prefab на диске, автоматически применяются к его экземплярам в сцене. Поэтому не нужно повторять одну и ту же правку в каждой копии ресурса.

Это не означает, что все экземпляры Prefab должны быть одинаковыми. Настройки отдельных экземпляров можно переопределять, чтобы они отличались друг от друга или от исходного ресурса.

Режим Prefab

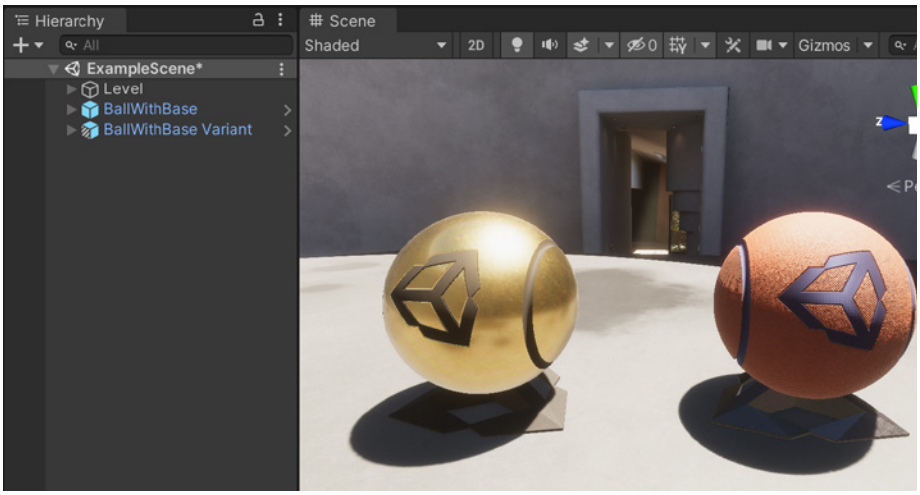
Для редактирования ресурса Prefab его можно открыть в специальном режиме Prefab. Этот режим позволяет просматривать и изменять содержимое ресурса Prefab изолированно либо в контексте других GameObject сцены. Изменения, внесённые в режиме Prefab, применяются ко всем экземплярам этого Prefab.



Режим Prefab: изолированно (слева) и в контексте (справа)

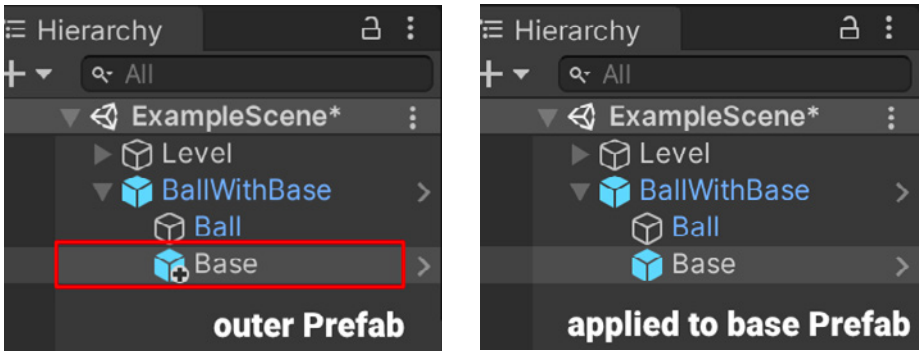
Варианты Prefab и вложенные Prefab

Можно создавать варианты Prefab - осмысленные модификации базового Prefab. Это похоже на наследование от базового ресурса с переопределением только тех частей, которые требуется изменить.



Варианты одного базового Prefab

Вкладывайте Prefab в другие Prefab, чтобы создавать сложные иерархии объектов, которые удобно редактировать на разных уровнях. Поверх значка экземпляра вложенного Prefab в Hierarchy отображается значок плюса. Он указывает, что данный экземпляр переопределён во внешнем Prefab.



Вложенные Prefab; «внешний» Prefab отмечен значком плюса

Исходные ресурсы Prefab можно редактировать независимо, распространяя изменения на вложенные Prefab.

Package Manager

Для большей гибкости разработки Unity предоставляет некоторые возможности в виде дополнительных пакетов. Они поставляются отдельно от основной части Unity Editor, поэтому могут обновляться чаще. Система управления пакетами позволяет устанавливать только инструменты, необходимые конкретному проекту, и тем самым сохранять сборку компактной. Пакеты различаются по назначению: они могут содержать любые сочетания ресурсов, шейдеров, текстур, плагинов, значков и скриптов, расширяющих разные части проекта.

Управляйте доступными пакетами в окне Package Manager (Window > **Package Manager).**

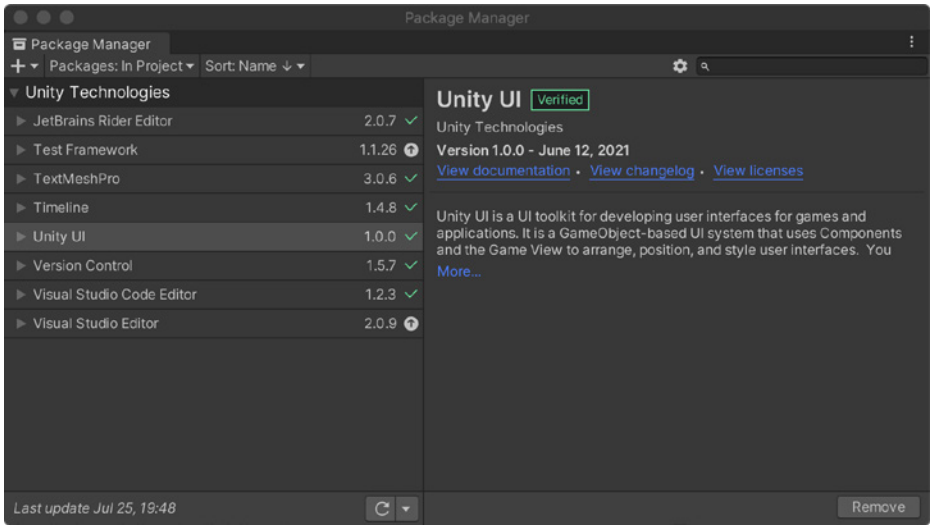
Помимо встроенных пакетов Unity и пакетов из Unity Registry можно устанавливать пользовательские пакеты с диска, по URL GitHub или из Asset Store:

- Нажмите кнопку, чтобы установить пакет непосредственно по URL Git или по локальному пути.

- В раскрывающемся меню Packages выбирается содержимое списка (Unity Registry, In Project, My Assets или Built-in), а меню Sort упорядочивает пакеты по имени или дате.

- Для пакетов Asset Store в разделе My Assets доступны дополнительные параметры фильтрации.

- Поле Search позволяет найти пакет по имени. Для управления пакетами проекта используйте кнопки Download, Import и Update.



Package Manager

Опытные разработчики могут также использовать PackageManager API для создания собственных общих библиотек. Краткое введение представлено в видео Creating Custom Packages in Unity.

Программирование

Код в Unity позволяет настраивать и контролировать практически все аспекты игры, создавать визуальные инструменты для команды и даже изменять работу самой Unity. Unity использует C# - современный объектно-ориентированный язык, широко применяемый в индустрии программного обеспечения.

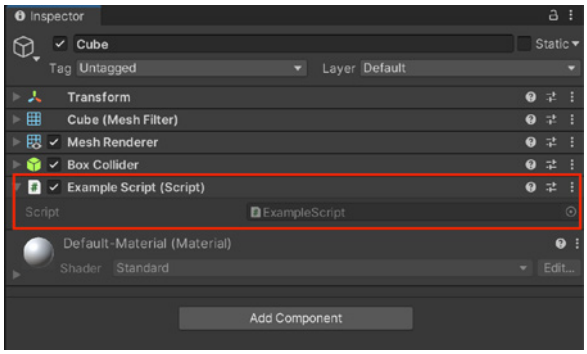
Пользовательские компоненты и MonoBehaviour

Каждый тип GameObject имеет набор компонентов по умолчанию. Например, пустой GameObject изначально содержит компонент Transform. Этот набор можно расширять пользовательскими компонентами, которые связывают вашу игровую логику непосредственно с объектами в игровых сценах. Можно также дать дизайнерам возможность настраивать значения и поведение через поля компонентов.

Для этого создайте скрипты, а затем добавьте их к GameObject как компоненты. Каждый такой скрипт наследуется от встроенного класса MonoBehaviour.

Класс MonoBehaviour можно представить как шаблон для создания компонента нового типа. Каждый раз, когда вы прикрепляете скриптовый компонент к GameObject, по этому шаблону создаётся новый экземпляр соответствующего компонента.

Скрипты создаются прямо в Unity. Выберите Assets > Create > C# Script либо щёлкните правой кнопкой мыши и выберите Create > C# Script, чтобы создать новый C#-скрипт на диске. Имя файла должно совпадать с требуемым именем класса. Перетащите скрипт на объект в Hierarchy - ExampleScript появится в Inspector как компонент.



Новый C#-скрипт

Примечание. Если изменить имя класса внутри файла, но не имя самого файла, прикреплённый пользовательский компонент может работать неправильно. Unity автоматически создаёт класс ExampleScript, наследующийся от MonoBehaviour.

Текущий шаблон скрипта Unity создаёт класс с двумя функциями:
Start и Update.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ExampleScript : MonoBehaviour
{
    // Start is called before the first frame update
   void Start()
   {
   }

// Update is called once per frame
   void Update()

  {
  }

}
```

Unity вызывает функцию Start до начала игрового процесса и до первого вызова Update. Поэтому Start идеально подходит для настройки переменных, чтения параметров и установления связей с другими GameObject.

Функция Update обрабатывает код, выполняемый в каждом кадре: например, перемещение, запуск действий и реакцию на пользовательский ввод.

Update и Start - лишь две функции-события MonoBehaviour. Эти встроенные методы выполняются в установленном порядке. Переопределяя функции-события MonoBehaviour, вы формируете игровой процесс и основной игровой цикл.

Чтобы познакомиться с основами программирования в Unity, изучите нашу документацию.

Инициализация объектов

Опытных программистов может удивить, что объект инициализируется не с помощью конструктора. Объекты конструирует Unity Editor, причём это происходит не в момент начала игрового процесса. Попытка определить конструктор для скриптового компонента нарушает штатную работу Unity и может вызвать ошибки.

Жизненный цикл и структура MonoBehaviour

Игровые движки работают на основе бесконечного цикла, который обрабатывает пользовательский ввод, обновляет состояние игры и выводит изображение на экран. В Unity PlayerLoop - низкоуровневый класс, лежащий в основе игрового движка. Он управляет рядом подсистем, отвечающих за инициализацию и покадровые обновления.

Чтобы взаимодействовать с PlayerLoop, скрипты наследуются от базового класса MonoBehaviour. Понимание его работы необходимо для создания игрового процесса. На этой блок-схеме показаны функции-события MonoBehaviour и порядок их выполнения в течение жизненного цикла скрипта.

Вот основные этапы игрового цикла при работе с MonoBehaviour:

- Первая загрузка сцены

- Editor

- Перед обновлением первого кадра

- Между кадрами

- Порядок обновления

- Цикл обновления анимации

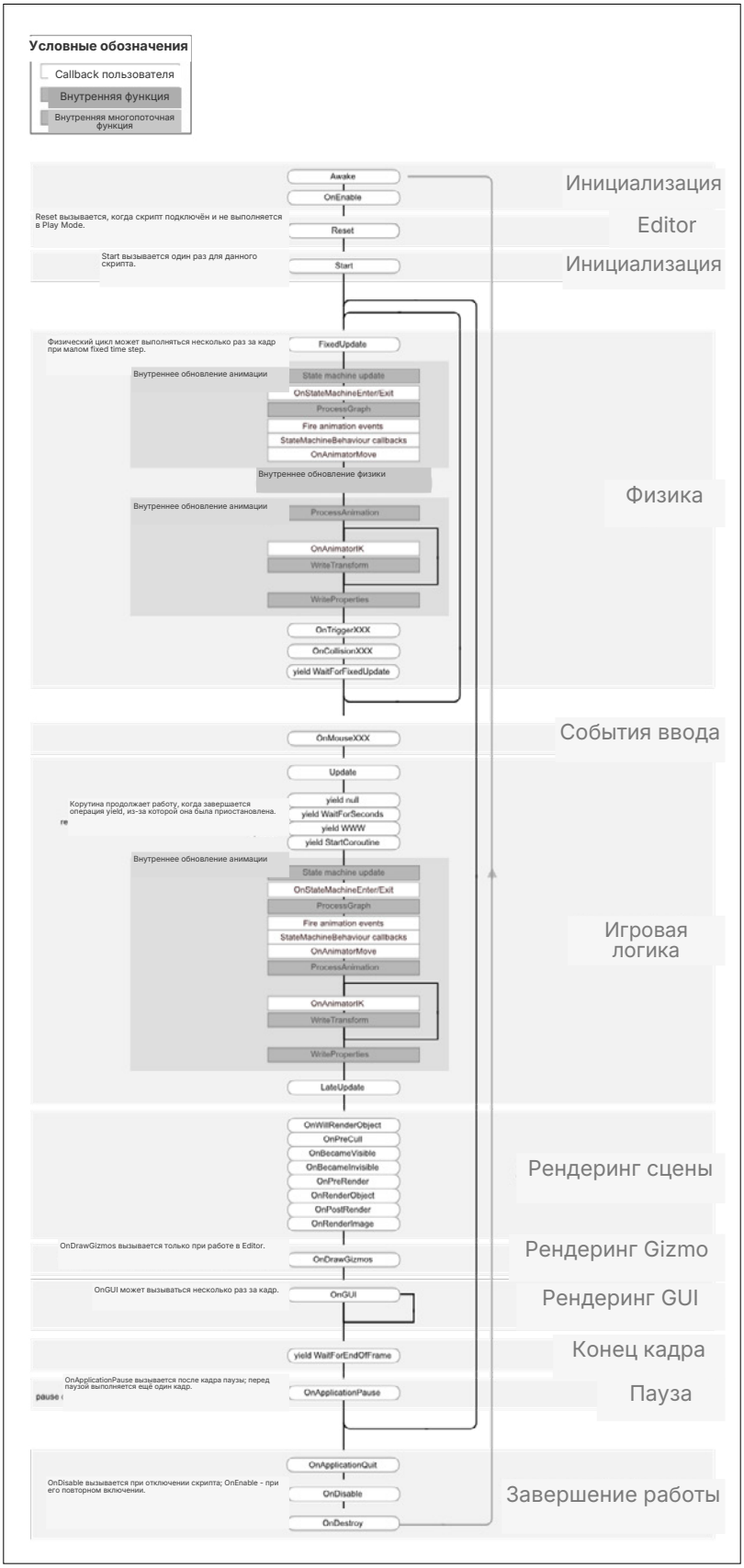
- Рендеринг

- Корутины

- При уничтожении объекта

- При выходе

Опытные пользователи могут даже создавать собственные PlayerLoop и PlayerLoopSystems. Однако рекомендуем начать с изучения класса MonoBehaviour и приведённых ниже наиболее распространённых классов.



Жизненный цикл
MonoBehaviour

Другие советы по скриптам

Бэкенд скриптинга Unity основан на .NET Framework. При разработке скриптов для Unity используйте следующие возможности C#:

- Пространства имён. Классы в Unity должны иметь уникальные имена. Когда над одним проектом работают несколько программистов, распространённые имена вроде Controller могут вызвать конфликты. Пространства имён помогают избежать этого и упорядочить типы данных. При необходимости используйте директиву using, чтобы сократить префикс пространства имён.

    Например, можно создать отдельные пространства имён для Player и Enemy. Тогда Player.Controller и Enemy.Controller смогут безопасно существовать в одном проекте.

    Подробнее см. на странице руководства Namespaces.

- Корутины. Иногда требуется запустить действие, которое будет выполняться на протяжении нескольких кадров: например, переместить объект из точки A в точку B за определённое время или постепенно изменить его цвет. Обычная функция выполняется до конца и возвращает управление в том же кадре, поэтому реализовать логику, распределённую во времени, сложнее.

    Корутина может приостановить выполнение, вернуть управление Unity, а в следующем кадре продолжить с места остановки. Корутины позволяют выполнять игровую логику в течение заданного времени. Например, через корутину часто реализуют паузу на определённый интервал. Корутина также может ожидать завершения других корутин; её можно сочетать с циклом while для ожидания условия. Корутины могут работать подобно функции-событию Update в MonoBehaviour, но дополнительно позволяют управлять интервалом обновления.

    Подробнее см. на странице руководства Coroutine.

- Атрибуты. Это маркеры, которые размещают над классом, свойством или функцией, чтобы указать особое поведение.

    Например, атрибут Range превращает числовое поле в ползунок в Inspector, а Tooltip добавляет к полю всплывающую подсказку. В C# имена атрибутов заключаются в квадратные скобки.

    Полный список атрибутов приведён в Scripting API.

Распространённые классы

Приступив к написанию скриптов в Unity, изучите наиболее важные встроенные классы. Этот список не исчерпывающий, но поможет начать знакомство с Unity. Полный перечень и дополнительные сведения приведены в Scripting API.

Класс	Описание
GameObject	Тип объектов, которые могут существовать в сцене.
MonoBehaviour	Базовый класс, от которого наследуется каждый скрипт Unity.
Object	Базовый класс для всех объектов, на которые Unity может ссылаться в Editor.
Transform	Управляет положением, поворотом и масштабом GameObject, а также его родительско-дочерними связями.
Vectors	Классы для представления и обработки 2D-, 3D- и 4D-точек, линий и направлений.
Quaternion	Класс для абсолютных и относительных поворотов и операций с ними.
ScriptableObject	Контейнер для хранения больших объёмов данных.
Time	Позволяет измерять и контролировать время и частоту кадров проекта.
Mathf	Набор математических функций, включая тригонометрические и логарифмические.
Random	Средства генерации распространённых типов случайных значений.
Debug	Помогает визуализировать в Editor сведения о выполняемом проекте.
Gizmos and Handles	Средства рисования линий и фигур в Scene и Game, а также создания интерактивных манипуляторов.

Управление памятью

Unity поддерживает C# - отраслевой стандарт языка программирования, в некоторых отношениях похожий на Java и C++. C# относится к управляемым языкам: он автоматически выделяет и освобождает память, снижает риск утечек памяти и решает другие задачи управления памятью.

В некоторых языках, например C++, программист сам выделяет и освобождает блоки памяти в куче соответствующими вызовами функций. Автоматическое управление памятью в C# требует меньше кода, чем явное выделение и освобождение, и значительно снижает вероятность утечки памяти - ситуации, когда память выделена, но впоследствии не освобождена.

Типы значений и ссылочные типы

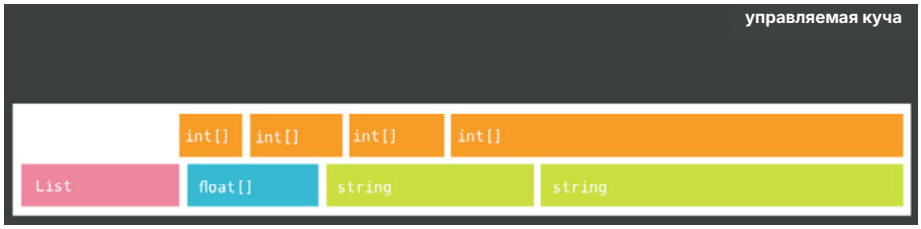
При вызове функции Unity резервирует для неё область памяти и копирует значения её параметров. Типы значений - например, целые числа, числа с плавающей запятой и логические значения - занимают лишь несколько байтов. Unity хранит их непосредственно и копирует при передаче параметров.

Другие типы данных, такие как объекты, строки и массивы, относятся к ссылочным. Они занимают больше места, и регулярно копировать их было бы неэффективно. Поэтому Unity хранит их данные в куче и обращается к ним через указатели. Если достаточно структуры (типа значения), она может быть эффективнее класса (ссылочного типа), содержащего те же данные.

Хотя явно выделять и освобождать память не требуется, необходимо понимать устройство управляемой кучи и её влияние на производительность игрового приложения.

Сборка мусора

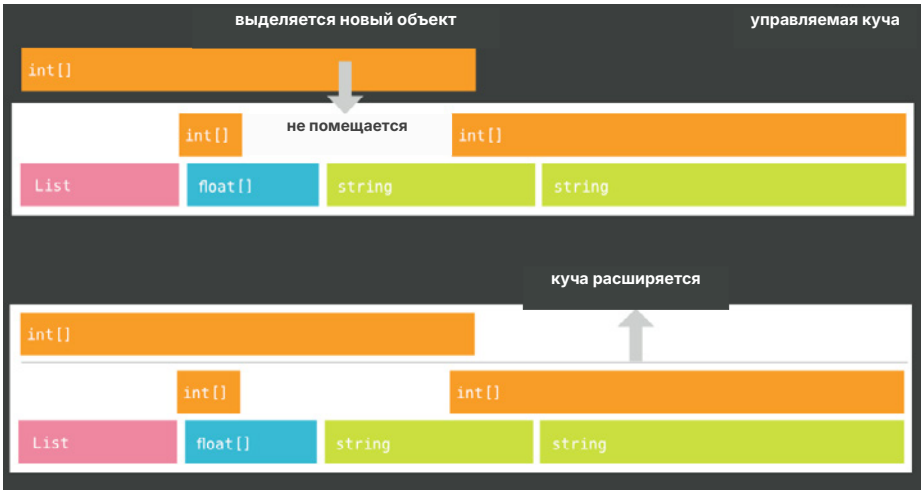
Блоки памяти в куче считаются «живыми», пока они используются и на них существуют активные ссылки.



Распределитель управляемой памяти автоматически выделяет память в куче.

По мере работы игры ссылки на блок памяти могут исчезать: GameObject уничтожаются, переменным присваиваются новые значения и т. д. Когда на блок памяти больше ничего не ссылается, распределитель управляемой памяти может безопасно использовать его повторно.

Распределитель периодически ищет свободные области между живыми блоками памяти. Поиск и освобождение неиспользуемой памяти называется сборкой мусора, или сокращённо GC. Когда игровому приложению требуются новые блоки памяти, распределитель выделяет их из этих свободных областей.



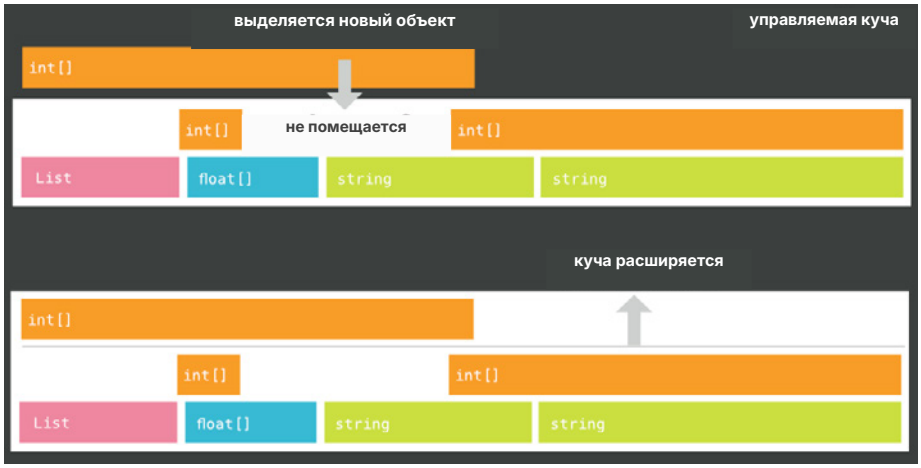
Сборка мусора освобождает неиспользуемую память, но приостанавливает выполнение кода скриптов.

В Unity используется сборщик мусора Boehm-Demers-Weiser. Во время сборки мусора Unity останавливает выполнение программного кода и возобновляет его только после завершения работы сборщика мусора.

Эта пауза может задержать выполнение приложения. Длительность зависит от объёма памяти, который должен обработать сборщик мусора, и от целевой платформы игры: от долей миллисекунды до сотен миллисекунд.

Для приложений реального времени, включая игры, это может стать серьёзной проблемой. Паузы сборщика мусора, называемые скачками GC, способны вызвать подёргивания игрового процесса. Хотя сборка мусора в основном незаметна, она требует значительного процессорного времени.

Учитывайте также, что алгоритм сборщика мусора Boehm не выполняет уплотнение: он не перемещает существующие объекты в памяти, чтобы закрыть промежутки между ними. Это может привести к фрагментации памяти. Если новый объект не помещается в существующие промежутки, распределителю может потребоваться расширить кучу, что способно снизить производительность.



Учитывайте, что куча может расширяться.

В Unity не следует запускать сборщик мусора чаще, чем необходимо. Иначе во время выполнения приложение может зависать или подтормаживать. В этой публикации блога приведены советы по оптимизации, которые помогают уменьшить влияние сборки мусора.

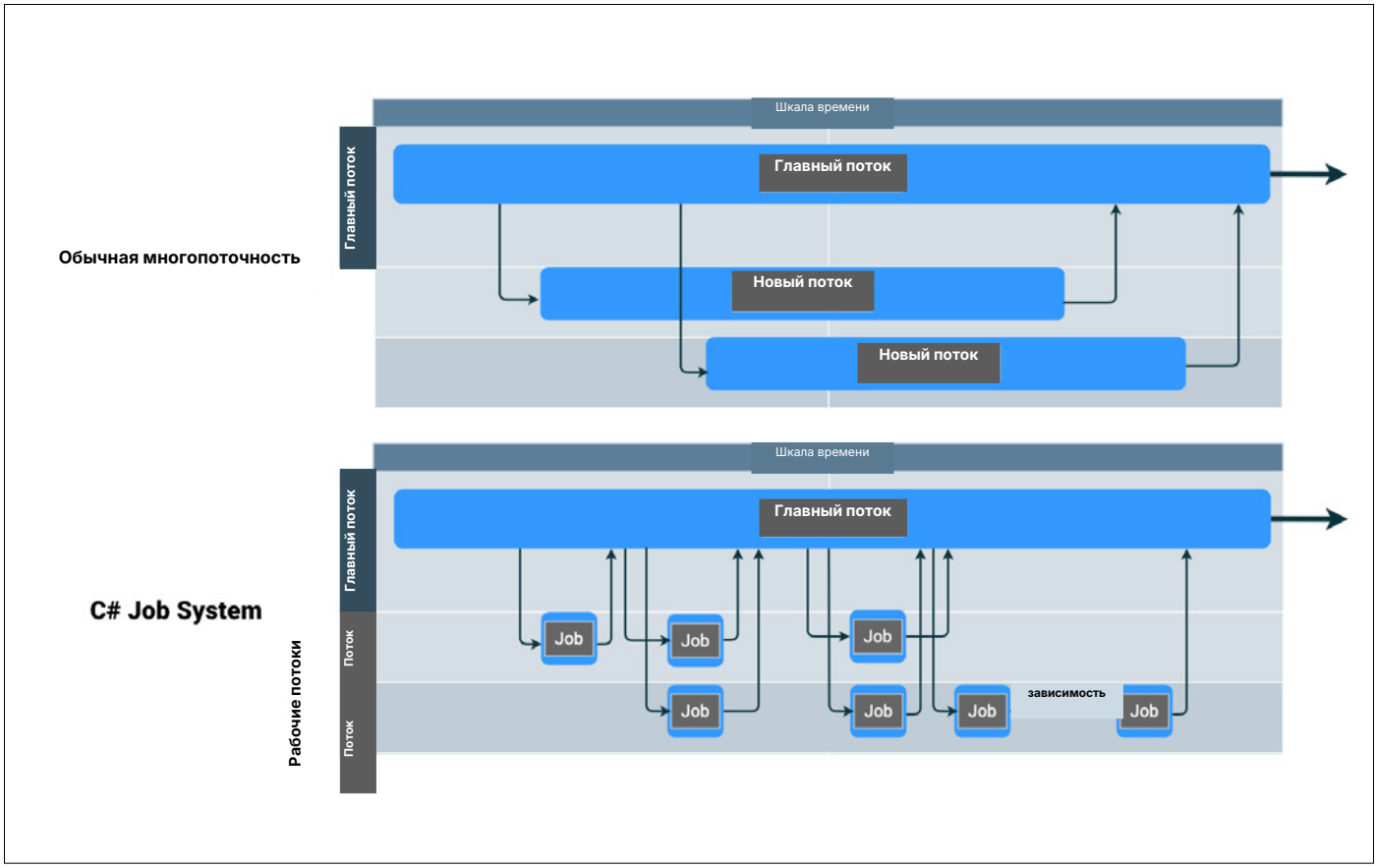
Unity также предлагает необязательный инкрементный сборщик мусора, распределяющий работу GC между несколькими кадрами. Сейчас эта функция имеет статус Experimental; подробности приведены в этой публикации блога.

Подробнее об управлении памятью и сборке мусора см. в разделах документации Unity Understanding Automatic Memory Management и Understanding the managed heap. Также доступно руководство Memory Management in Unity на сайте Learn.

Многопоточность: C# Job System и компилятор Burst

Современные процессоры имеют несколько ядер, но для их использования приложению нужен многопоточный код. Job System в Unity позволяет разбивать крупные задачи на небольшие части, которые параллельно выполняются на дополнительных ядрах процессора. Это может значительно повысить производительность.

В традиционной многопоточной программе один поток выполнения процессора - главный - создаёт другие потоки для обработки задач. После завершения работы эти дополнительные рабочие потоки синхронизируются с главным.



При традиционном подходе потоки создаются и уничтожаются. В C# Job System небольшие задания выполняются в пуле потоков.

Этот подход хорошо работает, если имеется несколько длительных задач. Но он менее эффективен для игрового приложения, которому обычно приходится обрабатывать множество коротких задач с частотой 30-60 кадров в секунду.

Поэтому Unity применяет несколько иной подход к многопоточности - C# Job System. Вместо создания множества короткоживущих потоков работа разбивается на небольшие единицы, называемые заданиями.

Задания помещаются в очередь, которая планирует их выполнение в общем пуле рабочих потоков. JobHandles позволяют задавать зависимости и обеспечивают правильный порядок выполнения. Чтобы система безопасности могла предотвращать состояния гонки, задания работают с копиями данных. Затем Native Containers передают результаты обратно в главный поток.

Job System дополняет компилятор Burst. С помощью LLVM Burst преобразует байт-код IL/.NET в оптимизированный нативный код. Чтобы получить к нему доступ, достаточно добавить пакет Burst через Package Manager. Burst позволяет разработчикам Unity сохранить удобство использования подмножества C# и при этом повысить производительность.

Бэкенды скриптинга в Unity

В Unity доступны два бэкенда скриптинга: Mono и IL2CPP (Intermediate Language To C++). Они используют разные методы компиляции:

- Mono использует JIT-компиляцию (just-in-time) и компилирует код по мере необходимости во время выполнения.

- IL2CPP использует AOT-компиляцию (ahead-of-time) и компилирует всё приложение до его запуска.

IL2CPP - разработанный Unity бэкенд скриптинга, который при сборке проектов для некоторых платформ можно использовать вместо Mono. Он способен повысить производительность и уменьшить размер сборки, но часто ценой увеличения времени сборки.

При сборке проекта с IL2CPP Unity преобразует IL-код скриптов и сборок в C++, а затем создаёт нативный двоичный файл для выбранной платформы, например .exe, .apk или .xap.

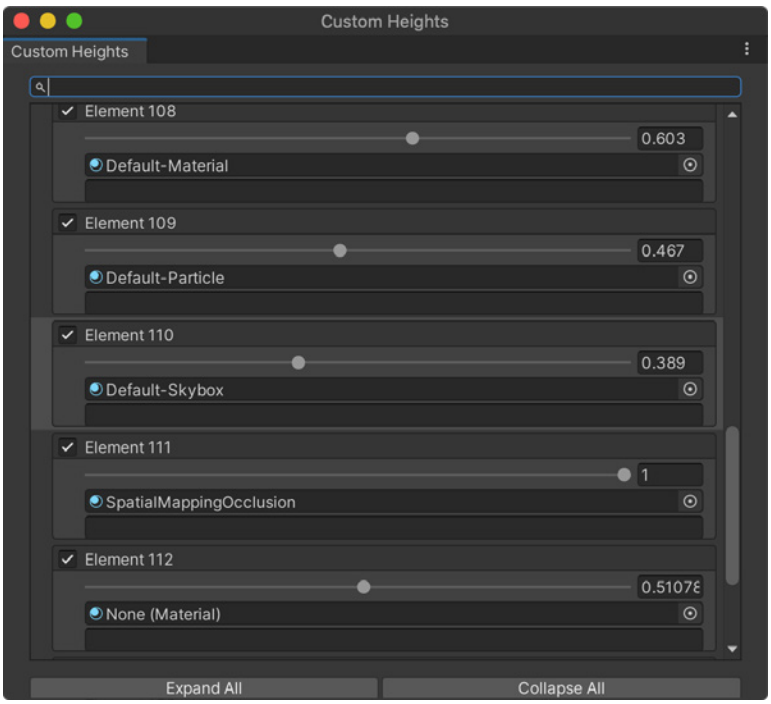
Обратите внимание: для сборки под iOS и WebGL доступен только бэкенд скриптинга IL2CPP.

Подробнее об использовании IL2CPP см. в серии публикаций The Unity IL2CPP и на странице Building a project using IL2CPP.

Скрипты для Editor

Чтобы работать эффективнее, среду разработки можно адаптировать к конкретным потребностям команды и проекта.

Если нужен специализированный рабочий процесс, Editor можно расширить собственными инспекторами и окнами. Они могут работать так же, как встроенные окна Inspector, Scene и другие. С помощью пользовательских Property Drawers можно также определять, как отображаются свойства.



Пользовательское окно Editor

Odin Inspector and Serializer

Odin Inspector and Serializer позволяет сократить время работы с API EditorWindow. Это сторонний инструмент партнёра Unity Verified Solutions Partner, доступный в Unity Asset Store. Odin содержит более 100 готовых атрибутов, позволяющих создавать пользовательские редакторы без ручного написания и сопровождения GUI-кода.

Чтобы создать пользовательское окно Editor с помощью Odin, унаследуйте свой класс от OdinEditorWindow и снабдите поля, свойства и методы атрибутами.

Вот лишь некоторые задачи, которые можно решать с помощью Odin:

- Настраивать компоновку с помощью групповых атрибутов, например TabGroup и ToggleGroup

- Сериализовать поля, например словари, которые обычно недоступны во встроенном Inspector Unity

- Легко создавать кнопки в окне Inspector, добавляя к методам атрибуты Button

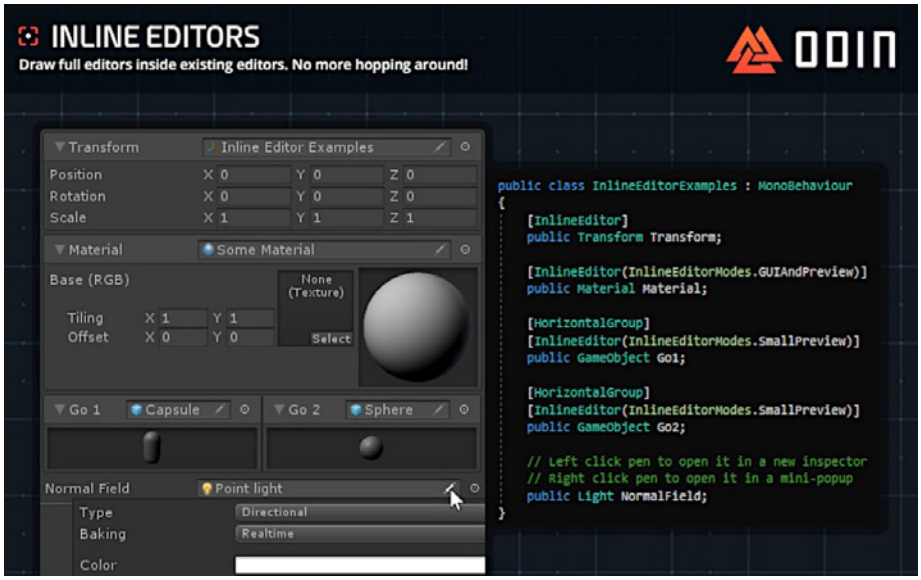
- Изменять статические члены для тестирования и отладки; например, вызывать статический метод с любыми аргументами непосредственно из Inspector

- Создавать пользовательские редакторы для окна Inspector с помощью атрибутов

- Писать фрагменты C# с выражениями атрибутов непосредственно внутри атрибутов, уменьшая объём шаблонного кода

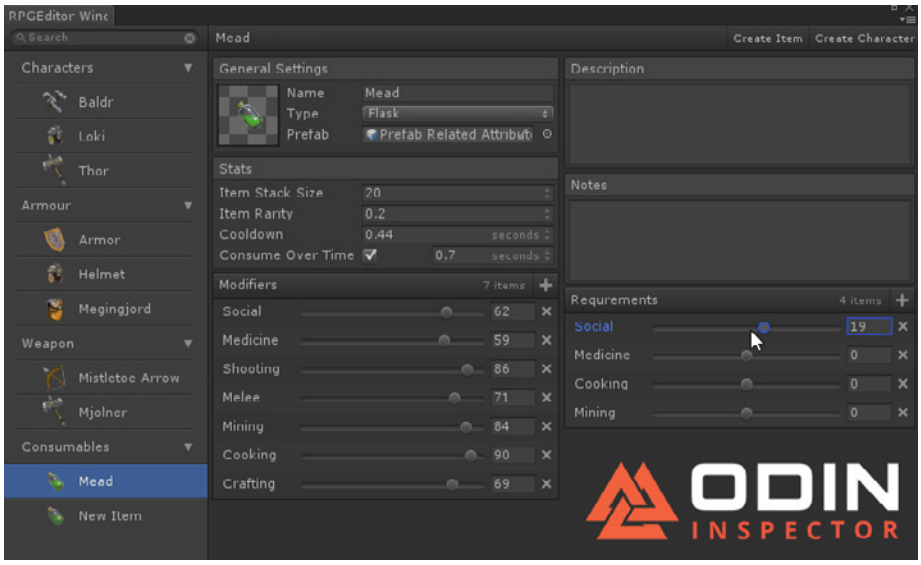
- Проверять пользовательский ввод с помощью таких атрибутов, как Required, ValidateInput и ChildGameObjectsOnly

Например, с помощью скрипта можно создать Inspector такого вида:



Пример, созданный в Odin Inspector

Вот пример окна Editor, созданного с помощью Odin:



Окно RPG-редактора, созданное в Odin

Odin Inspector доступен в редакциях Personal и Enterprise через Unity Asset Store.

Поддержка интегрированных сред разработки (IDE)

Unity поддерживает несколько IDE, поэтому вы можете работать в предпочитаемой среде разработки. Visual Studio по умолчанию устанавливается вместе с Unity в Windows и macOS. Выберите редактор скриптов в настройках (Unity > Preferences > External

Tools > External Script Editor).

Unity изначально поддерживает следующие IDE:

- Visual Studio - IDE по умолчанию для Unity в Windows и macOS. В Windows вместе с Unity также устанавливается Visual Studio 2019 Community, а в macOS - Visual Studio for Mac.

- Visual Studio Code (Windows, macOS, Linux) - бесплатный, лёгкий и настраиваемый редактор кода с открытым исходным кодом, известный своей скоростью и гибкостью. Подробнее об использовании VS Code с Unity см. в материале Unity Development with VS Code.

- JetBrains Rider (Windows, macOS, Linux) построен на основе ReSharper и включает большинство его возможностей. Подробнее см. в документации JetBrains по Rider for Unity.

Если выбранный текстовый редактор не входит в этот список и не поддерживается изначально, его может потребоваться настроить для разработки в Unity. Например, сообщество создало множество пакетов - плагинов, расширений и дополнений - для использования Sublime Text с Unity.

Шаблоны скриптов

При создании пользовательских компонентов вы можете заметить, что вносите одни и те же изменения в каждый новый C#-скрипт. Например, может требоваться автоматически удалять функцию-событие Update или добавлять пространство имён по умолчанию. Чтобы сократить ручную работу, настройте шаблон скрипта под текущую задачу.

Unity использует шаблоны из каталога ресурсов ScriptTemplates:

- Windows:

C:\Program Files\Unity\Editor\Data\Resources\ScriptTemplates

- macOS:

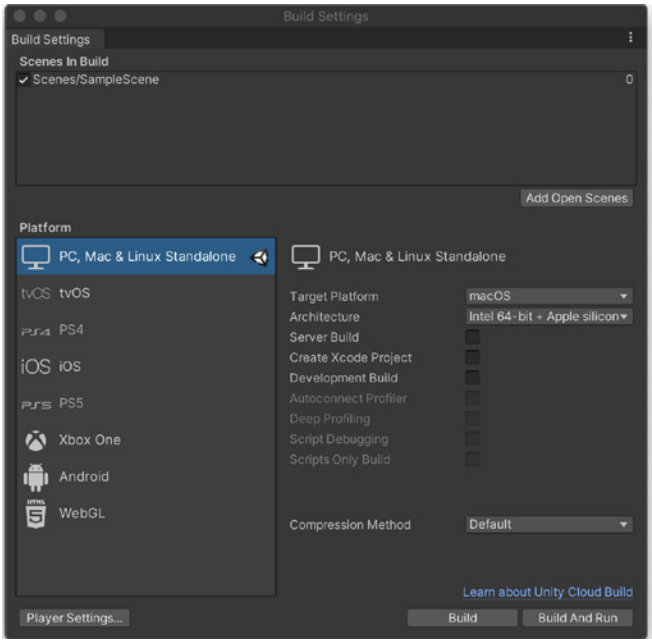
/Applications/Hub/Editor/[version]/Unity/Unity.app/Contents/Resources/ScriptTemplates

При необходимости откройте и отредактируйте эти файлы шаблонов, а затем перезапустите Unity Editor, чтобы применить изменения. Обязательно создайте резервные копии как исходных, так и изменённых файлов шаблонов.

Сборка и публикация

Одно из преимуществ Unity - возможность развёртывания на разных платформах. Чтобы изменить параметры публикации сборки приложения, откройте File > Build Settings.

Добавьте нужные сцены в список Scenes in Build. Сцена, служащая точкой входа в приложение, должна иметь индекс 0.



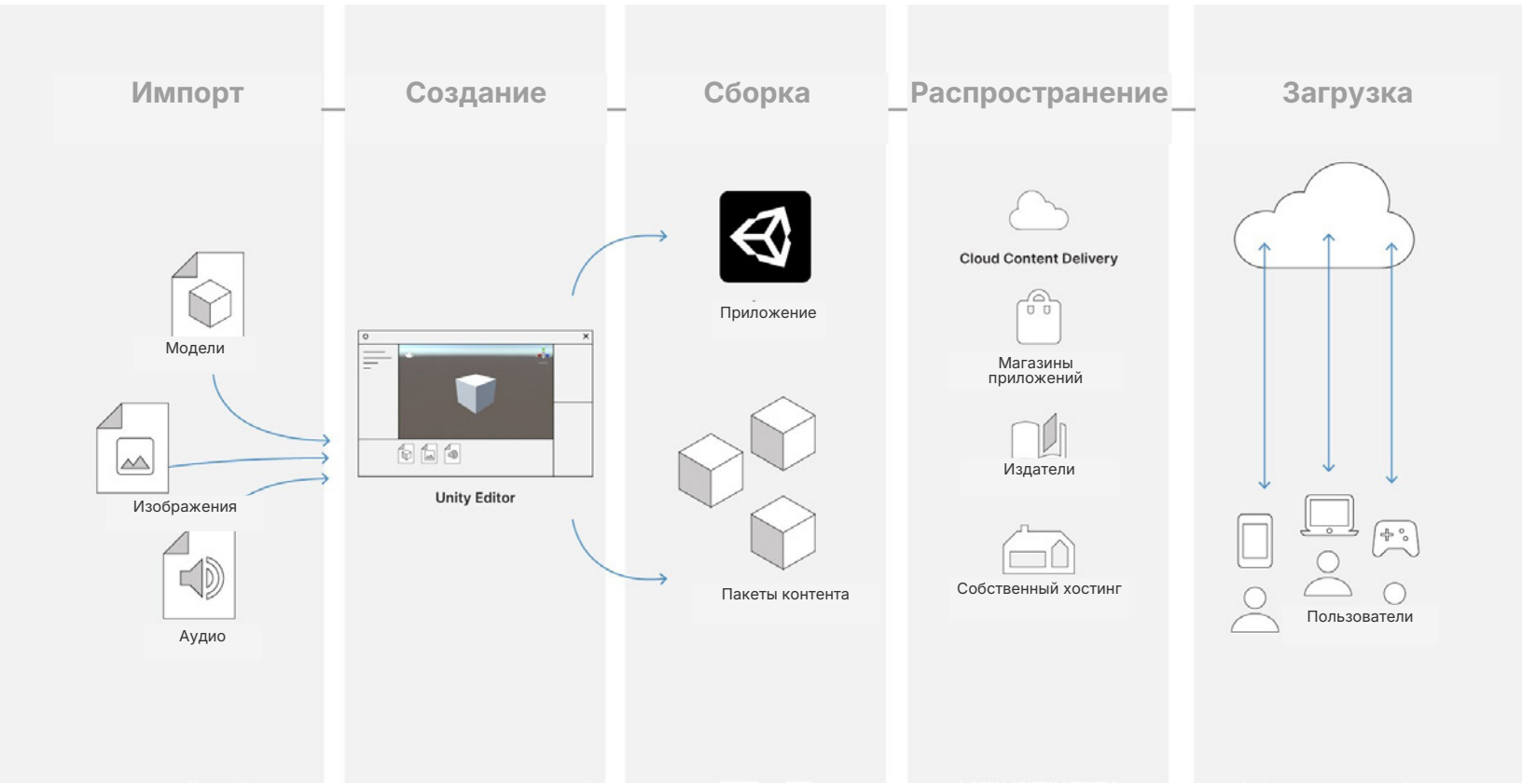
Build Settings

Доступные платформы перечислены в нижней левой части окна. Чтобы установить поддержку других целевых платформ, используйте Install with Unity Hub. Можно также открыть экран Installs в Unity Hub и с помощью Add Module добавить нужные модули к любой версии Unity. Для некоторых консолей, включая PlayStation, Nintendo Switch и Xbox, требуются дополнительные модули от издателя соответствующей платформы.

При сборке приложения учитывайте различия между платформами. Например, мобильные устройства располагают меньшими объёмами хранилища и памяти, а также менее мощными процессорами, поэтому сильнее подвержены скачкам GC при автоматическом управлении памятью. Графически требовательную консольную игру нельзя просто перенести на iOS или Android - во всяком случае, без значительной оптимизации под конкретную платформу.

При сборке для нескольких платформ учитывайте и требования к вводу. Мобильным устройствам с сенсорными экранами и акселерометрами нужны иные настройки, чем платформам с контроллером или клавиатурой и мышью. Новая Input System позволяет обрабатывать пользовательский ввод с разных устройств.

Платформозависимая компиляция позволяет с помощью директив препроцессора разделять код скриптов по целевым платформам. Используйте их вместе с директивой компилятора #if или классом ConditionalAttribute. Если для выпуска на определённой платформе нужно добавить или удалить элемент UI либо объект - например, исключить кнопку Quit из приложения для iOS, - платформозависимая компиляция упрощает такую логику.



Рабочий процесс сборки и публикации

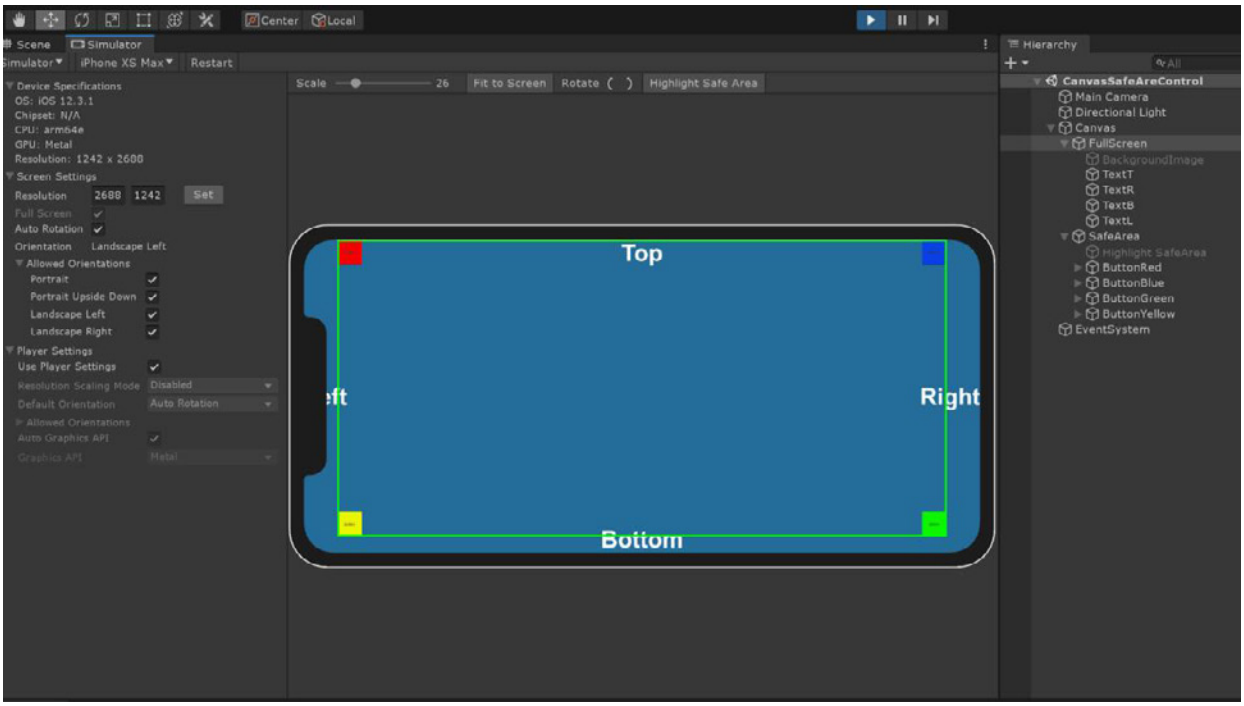
Device Simulator

При сборке для мобильных устройств установите Device Simulator через Package Manager. Он позволяет просматривать приложение на имитируемом мобильном устройстве и заранее проверять форму экрана, разрешение и ориентацию.

Чтобы открыть представление Device Simulator:

- В левом верхнем углу окна Game находится раскрывающееся меню. Оно переключает представления Game и Device Simulator.

- В верхнем меню выберите Window > General > Device Simulator.



Device Simulator

Device Simulator предназначен прежде всего для просмотра компоновки приложения и проверки базовых взаимодействий.

Представление Device Simulator не имитирует производительность и особенности рендеринга устройства, например скорость процессора или доступный объём оперативной памяти.

Для начала посмотрите видео Simulate your Game with Device Simulator in Unity.

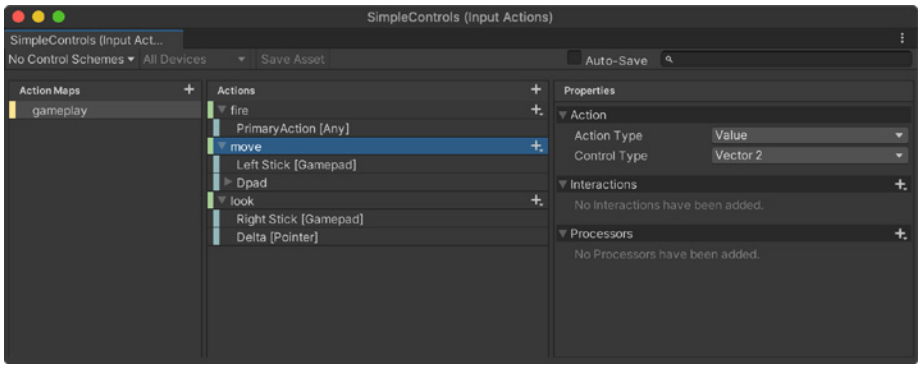
Ввод

В Unity есть две основные системы обработки ввода: Input System, устанавливаемая через Package Manager, и встроенный класс Input.

Input System

Input System ориентирована на простоту использования, гибкость и единообразную работу на разных устройствах и платформах. Она заменяет встроенный класс Input.

Хотя ввод по-прежнему можно получать непосредственно от устройства, преимущества Input System особенно заметны, когда для управления взаимодействием игрока создаётся набор абстрактных действий. Затем можно создать actionMap - набор привязок и действий. Карта действий связывает эти действия с конкретными устройствами.



Input System доступна через Package Manager.

Чтобы адаптировать управление к новым устройствам, создавайте дополнительные карты действий, а не жёстко привязывайте код к определённым кнопкам устройства, клавишам клавиатуры и т. п. Настройте функции обратного вызова и действия один раз, а затем добавляйте новые сопоставления для поддержки других устройств.

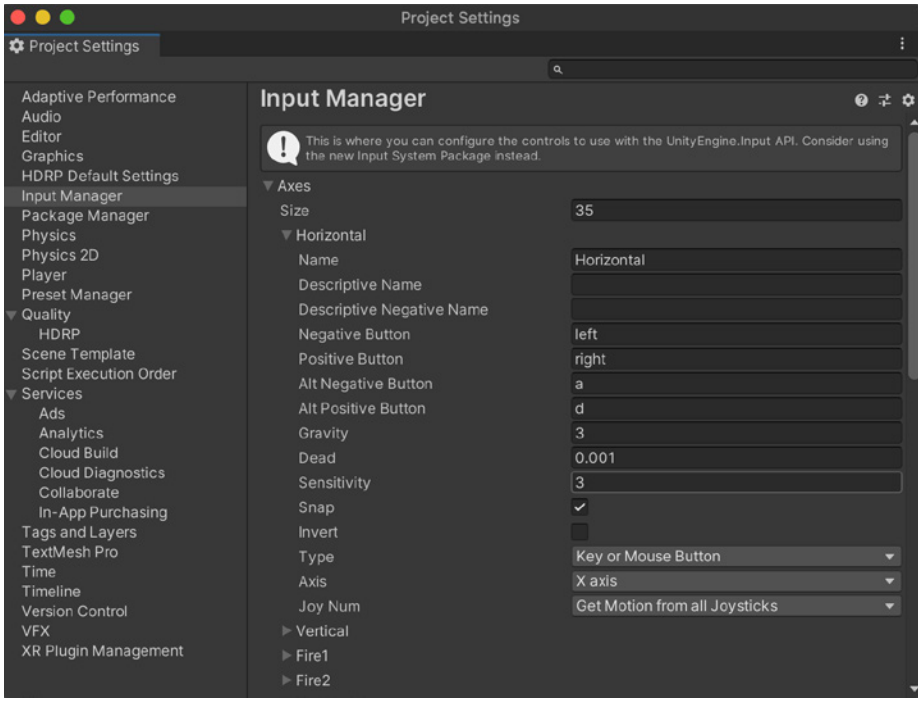
Карты действий также позволяют переключать схему ввода в зависимости от игрового контекста. Например, управление игроком может работать по-разному при вождении, ходьбе и беге.

Обязательно ознакомьтесь с кратким руководством по Input System и установите примеры, доступные через Package Manager.

Встроенный класс Input

В качестве альтернативы используйте встроенный класс Input и **Input Manager (Edit > Project Settings > Input Manager) с виртуальными осями** для клавиш, кнопок и других устройств ввода. Он также поддерживает мультитач-ввод и данные акселерометра на мобильных устройствах.

Обратите внимание: при включении пакета Input System описанный выше встроенный Input Manager отключается.



Встроенный Input Manager

Ниже перечислены полезные классы и методы для работы с вводом:

Input	Читает оси Conventional Game Input и предоставляет доступ к мультитач-вводу и акселерометру на мобильных устройствах.
Input.GetAxis	Возвращает сглаженное значение виртуальной оси axisName: от -1 до 1 для клавиатуры и джойстика; для мыши - текущее смещение, умноженное на чувствительность оси.
Input.GetAxisRaw	Аналог Input.GetAxis без сглаживания.
Input.GetButton	Возвращает true, пока виртуальная кнопка buttonName удерживается.
Input.GetButtonDown	Возвращает true в кадре нажатия виртуальной кнопки buttonName.
Input.GetKey	Возвращает true, пока пользователь удерживает клавишу name.
Input.GetKeyDown	Возвращает true в кадре начала нажатия клавиши name.
Input.touches	Доступный только для чтения список касаний за предыдущий кадр.
Touch	Описывает состояние пальца, касающегося экрана.
KeyCode	Перечисляет варианты клавиш, кнопок мыши и джойстика.

Обзор встроенной системы ввода см. на странице Input Manual.

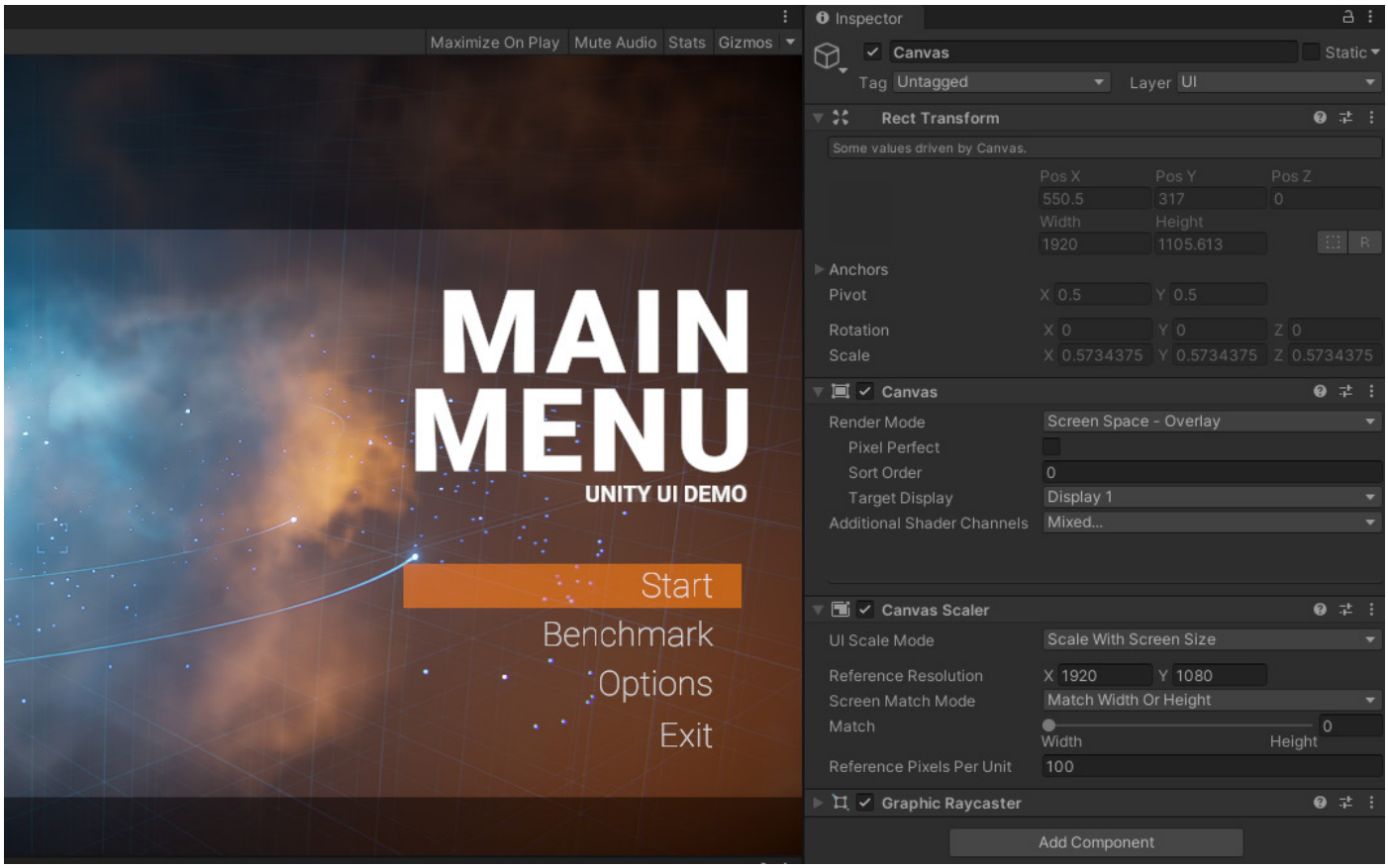
Пользовательский интерфейс

В Unity реализовано несколько систем UI: для внутриигрового интерфейса и для скриптов Editor. В зависимости от задачи можно использовать одну или несколько из них.

Unity UI

Unity UI (иногда называемая UGUI) предлагает основанный на GameObject подход к редактированию и размещению элементов интерфейса. Эта система предназначена для разработки пользовательских интерфейсов игр и приложений. Для компоновки, позиционирования и оформления интерфейса используются компоненты и окно Game. Unity UI нельзя использовать для создания или изменения интерфейса самого Unity Editor.

В руководстве по Unity UI объясняется, как размещать элементы интерфейса и настраивать взаимодействие с ними.



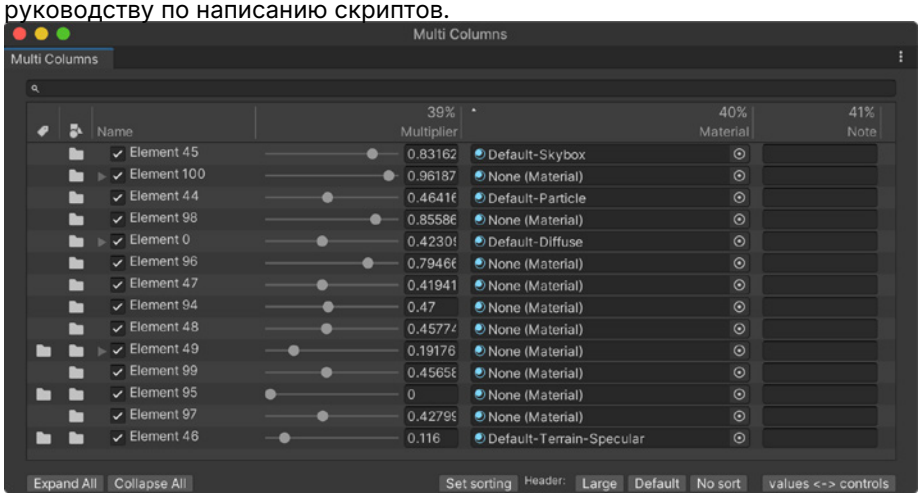
Unity UI использует основанный на GameObject подход к элементам интерфейса.

Immediate Mode GUI (IMGUI)

Immediate Mode GUI (IMGUI) позволяет создавать собственные инспекторы и окна Editor. Элементы IMGUI создаются кодом в специальной функции MonoBehaviour с именем OnGUI. Эта система, как правило, не предназначена для обычных внутриигровых интерфейсов.

Код отображает интерфейс в «непосредственном режиме» и выполняется в каждом кадре. Постоянных объектов GameObject нет, кроме объекта, к которому прикреплён код OnGUI. Элементы управления GUI отрисовываются и обрабатываются одним вызовом функции.

Если вам требуется IMGUI, следуйте этому руководству по написанию скриптов.



Интерфейс Editor, созданный с помощью Immediate Mode GUI (IMGUI)

Профилирование и оптимизация

После создания отличной игры следующая задача - обеспечить её высокую производительность. Unity включает набор инструментов профилирования и оптимизации, помогающих максимально эффективно использовать ресурсы целевой платформы.

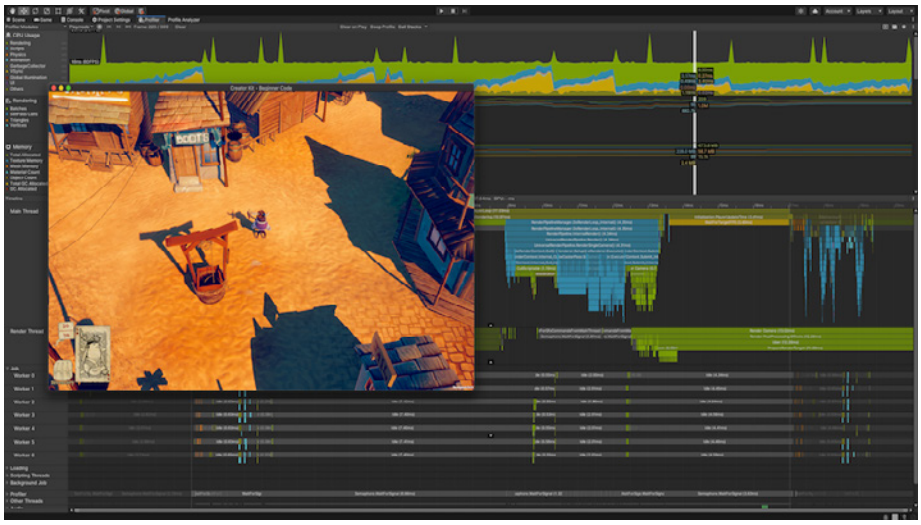
Unity Profiler

Unity Profiler предоставляет сведения о производительности приложения. Для наиболее точных результатов запускайте его на целевой платформе, для которой планируете публиковать сборку. Так вы получите достоверные данные о времени выполнения операций, влияющих на производительность приложения.

Profiler также можно запускать непосредственно из Unity Editor в режиме Play. Однако в таком режиме точность ниже: он подходит для быстрой проверки, но не для итоговой оценки.

Profiler помогает выявить потенциальные направления оптимизации приложения: CPU, GPU, память, рендерер, физику и аудио. Последовательно оптимизируйте эти области. Так можно точно определить, как на работу приложения влияют код, ресурсы, настройки сцены, рендеринг камерой и настройки сборки.

Profiler отображает результаты в виде графиков, на которых хорошо видны пики нагрузки.



Unity Profiler

Unity предоставляет множество маркеров Profiler, помогающих понять, на что приложение тратит время. ProfilerRecorder также может использовать эти маркеры для получения полезных временных показателей кадра.

Ниже перечислены наиболее распространённые маркеры:

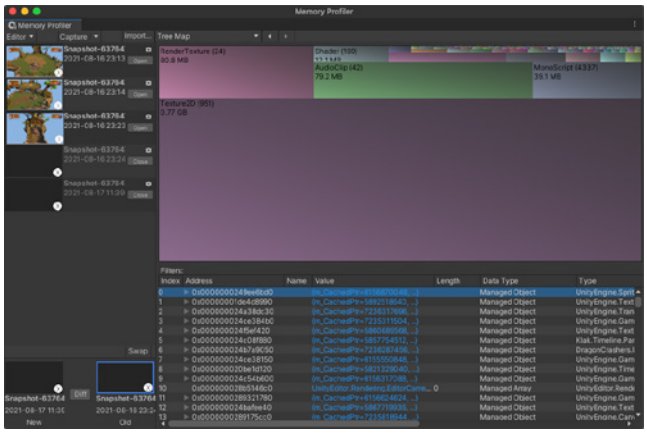
Маркер Profiler	Краткое описание
Основные маркеры главного потока	Выборки основного игрового цикла PlayerLoop и EditorLoop при профилировании в Editor.
Маркеры обновления скриптов	Выборки MonoBehaviour.Update (Update, FixedUpdate, LateUpdate) и корутин.
Маркеры рендеринга и VSync	Участки обработки данных для GPU или ожидания завершения его работы.
Маркеры бэкенда скриптинга	Работа бэкендов Mono и IL2CPP; полезны при диагностике сборки мусора и выделения памяти.
Маркеры многопоточности	Сведения о синхронизации потоков и Job System без измерения циклов CPU.
Маркеры физики	Высокоуровневые маркеры Physics.Contacts, Physics.TriggerEnterExits, Physics.UpdateBodies и другие.
Предупреждения о производительности	Вызовы, которых следует избегать в критичных к производительности участках.

Чтобы открыть окно Profiler, выберите Window > Analysis > Profiler. Подробное описание окна приведено в документации Profiler.

Memory Profiler

Memory Profiler помогает находить области проекта Unity и Unity Editor, в которых можно сократить использование памяти. Он даёт обзор выделений нативной и управляемой памяти, а также ссылок между ними, из-за которых эти выделения остаются в памяти.

Используйте Memory Profiler, чтобы снимать, исследовать и сравнивать снимки памяти, выявляя утечки и фрагментацию.

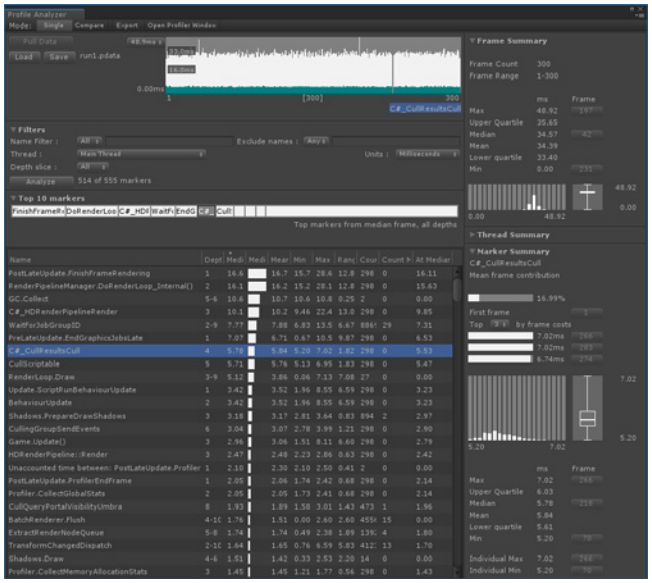


Главное представление Memory Profiler в режиме Tree Map

Profile Analyzer

Profile Analyzer агрегирует и визуализирует данные кадров и маркеров из набора кадров Unity Profiler, помогая понять их поведение. В Profile Analyzer можно сопоставить два набора данных рядом друг с другом; это дополняет анализ одного кадра, доступный в Unity Profiler.

Инструкции по работе с Profile Analyzer см. в документации этого окна.



Profile Analyzer

Чтобы лучше разобраться в оптимизации Unity для мобильных платформ, прочитайте публикацию в блоге Optimize your mobile game performance. Также можно скачать полную электронную книгу с рекомендациями экспертов Unity по оптимизации мобильных игр.

Отладка и тестирование игрового процесса

Unity отлично подходит для настройки и отладки. В любой момент можно перейти в режим Play в Unity Editor и наблюдать за всеми переменными игрового процесса во время работы приложения.

В режиме Play окно Game показывает предварительный вид итогового опубликованного приложения. Можно изменять сцену Unity на лету, экспериментировать, в любой момент приостанавливать выполнение и пошагово выполнять код по одному оператору.

Для тестирования игрового процесса обычно создают чит-команды, позволяющие:

- Открывать уровни, персонажей, предметы и т. п.

- Отключать врагов и/или элементы игрового процесса.

- Включать и отключать GUI.

- Давать игроку неуязвимость.

- Добавлять или вычитать время, деньги, коллекционные предметы и т. п.

Отладка игрового процесса

Игровое тестирование - не точная наука, но полезно учитывать следующие рекомендации:

- Реализуйте горячие клавиши для вывода мировых координат игрока. Это поможет определить, возникают ли конкретные ошибки в определённом месте уровня.

- В небольшой команде создайте тестовый Prefab для каждого участника, а настройки и параметры отладки считывайте из файла, не добавленного в систему контроля версий. Так участники команды не смогут случайно зафиксировать тестовые параметры в репозитории или изменить рабочую сцену.

- Поддерживайте тестовую сцену-песочницу со всеми элементами игрового процесса: например, со всеми врагами и интерактивными объектами. Так функциональность можно проверять без прохождения всей игры.

- Создайте набор чит-команд, доступных в Editor. Добавьте атрибут MenuItem к статическому методу, который проверяет Application.isPlaying, а затем выполняет нужную логику.

Дополнительные советы по отладке

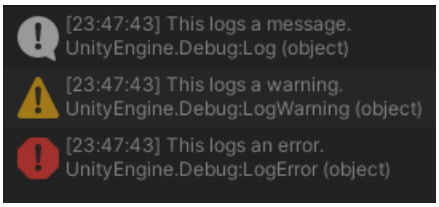
Unity также содержит класс Debug, который помогает визуализировать информацию во время работы Editor. С его помощью можно выводить сообщения и предупреждения в окно Console, где отображаются ошибки, предупреждения и другие сообщения Unity.

Debug также позволяет рисовать вспомогательные линии в окнах Scene и Game и приостанавливать режим Play в Editor из скрипта.

- Приостановка выполнения с помощью Debug.Break полезна, если нужно проверить значения в Inspector, а вручную поставить приложение на паузу трудно.

- Сообщения в Console можно разделять по уровням серьёзности

с помощью Debug.Log, Debug.LogWarning и Debug.LogError.



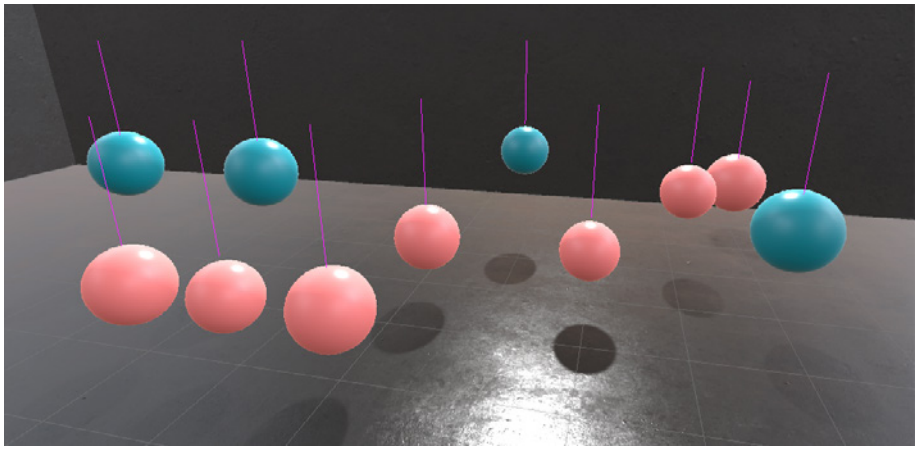
Сообщения, предупреждения и ошибки в Console

- При вызове Debug.Log можно передать объект в качестве контекста. Если щёлкнуть сообщение в Console, Unity выделит соответствующий GameObject в окне Hierarchy.

- Используйте Rich Text для разметки сообщений Debug.Log. Это помогает сделать отчёты об ошибках в Console нагляднее.

- Отлаживаете физику? Debug.DrawLine и Debug.DrawRay помогают

визуализировать трассировку лучей.



Debug.DrawLine

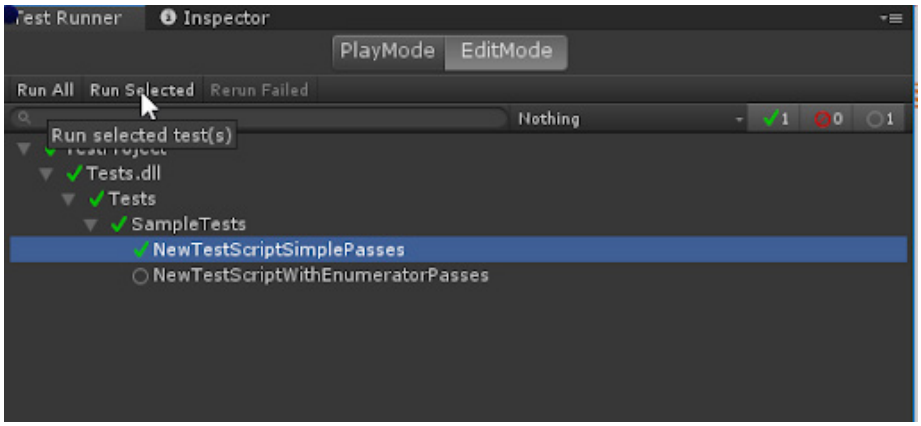
Unity Test Framework (UTF)

Unity Test Framework (ранее Unity Test Runner) позволяет создавать автоматизированные тесты и проверять, что код работает как задумано. Создавайте модульные тесты для отдельных логических фрагментов кода и применяйте разработку через тестирование по мере развития проекта.

UTF позволяет тестировать код как в режиме Edit, так и в режиме Play. Тесты также можно запускать в целевых сборках Standalone, Android, iOS и других. UTF расширяет NUnit - открытую библиотеку модульного тестирования для языков .NET.

Чтобы открыть Test Framework, выберите Window > General > Test Runner. Следуя инструкциям, настройте рабочую папку и Assembly Definition, а затем добавьте модульные тесты в Test Assembly. Это поможет быстрее изолировать ошибки и научиться писать код, удобный для тестирования.

Подробнее о Test Framework см. в публикации блога Performance Benchmarking in Unity и документации Unity Test Framework.



Unity Test Framework

Эффекты частиц

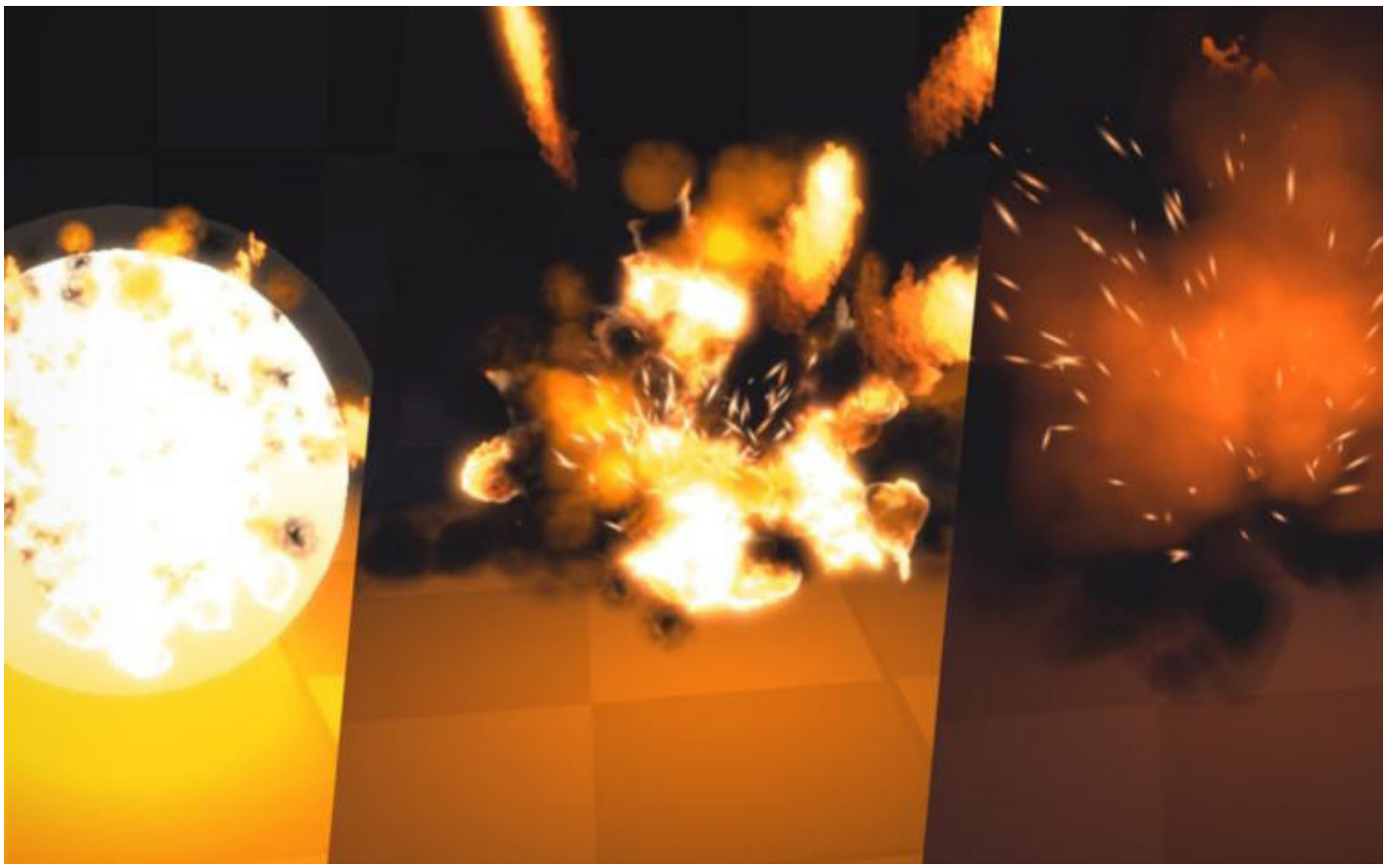
Для создания эффектов дыма, жидкостей и пламени Unity предлагает два решения для моделирования частиц:

- Particle System может моделировать на CPU тысячи частиц. Модули компонента Particle System задают готовые варианты поведения. Для создания одного эффекта часто объединяют несколько объектов GameObject с компонентами Particle System.

Класс ParticleSystem поддерживает Built-in Render Pipeline и URP. Через ParticleSystem можно определять саму систему и отдельные частицы из скрипта.

Particle System может взаимодействовать с физической системой Unity и компонентами Collider в сцене, но не имеет доступа к буферам кадров.

Чтобы ознакомиться с примерами встроенной Particle System, загрузите пакет Particle Pack из Asset Store.



Простая симуляция эффекта с помощью Particle System

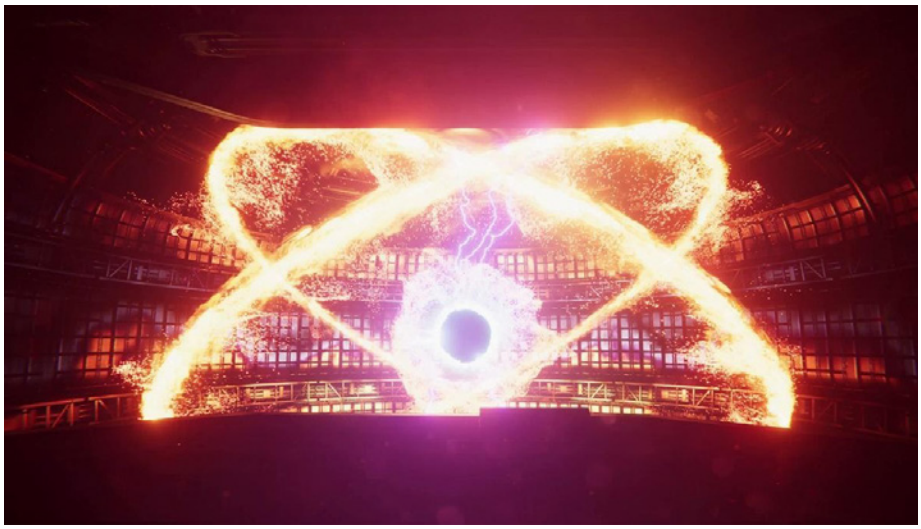
- Visual Effect Graph переносит вычисления на GPU с помощью вычислительных шейдеров. Это позволяет моделировать миллионы частиц, способных взаимодействовать с буферами цвета и глубины.

Рабочий процесс построен вокруг гибко настраиваемого графового редактора.

Visual Effect Graph не имеет доступа к физической системе, но может взаимодействовать со сложными ресурсами: Point Cache, Vector Field и Signed Distance Field.

Visual Effect Graph работает только на платформах, поддерживающих вычислительные шейдеры и HDRP; поддержка URP пока имеет статус Preview. Через интерфейс событий можно отправлять пользовательские события и передавать связанные данные для обработки графом. Компонент Visual Effect также предоставляет API управления воспроизведением.

Unity поддерживает два проекта на GitHub с примерами применения Visual Effect Graph в реальных проектах: Visual Effect Graph Samples и Spaceship Demo. Они показывают, как создавать множество высококачественных эффектов.



Миллионы частиц на экране, созданные с помощью Visual Effect Graph

При выборе одной из двух систем учитывайте совместимость устройств. Большинство ПК и консолей поддерживают вычислительные шейдеры, но многие мобильные устройства - нет. Для мобильных платформ сейчас рекомендуется встроенный Particle System. Если целевая платформа поддерживает вычислительные шейдеры, в проекте Unity можно использовать оба типа моделирования частиц.

Физика

Для 3D-проектов Unity использует движок NVIDIA PhysX, а для 2D-проектов - отдельный двумерный движок. Встроенная физическая система Unity предоставляет стандартные отраслевые решения для взаимодействия Rigidbody, соединений и сил. Методы Physics API позволяют выполнять трассировку лучей и объёмов, проверять пересечения, а также обрабатывать триггеры и столкновения.

3D-физик
а

Встроенный физический движок Unity включает в себя несколько полезных компонентов:

- Rigidbody позволяет физической системе управлять движением объекта. На Rigidbody действует сила тяжести, а доступные методы позволяют прикладывать к нему силу или крутящий момент. Если компонент Collider активен, соответствующий GameObject реагирует на физические столкновения.

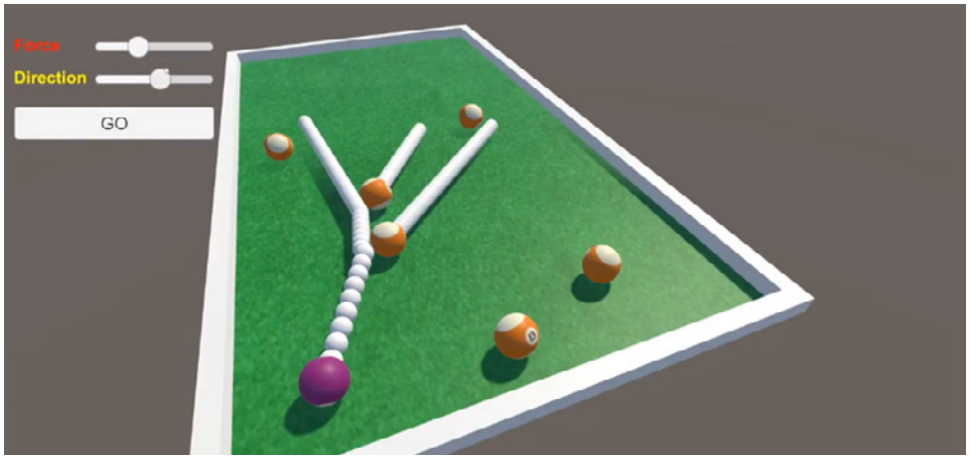
- Компоненты Collider (Box Collider, Sphere Collider, Capsule Collider и Mesh Collider) представляют примитивные объёмы или объёмы мешей, задающие границы физических столкновений.

- Компоненты Joint (Fixed Joint, Hinge Joint, Spring Joint и Character Joint) позволяют соединять два компонента Rigidbody и моделировать различные ограничения движения: разрушаемое соединение, вращение вокруг одной оси, упругое соединение, шаровой шарнир и т. п.

- CharacterController позволяет управлять движением с учётом столкновений без использования Rigidbody.

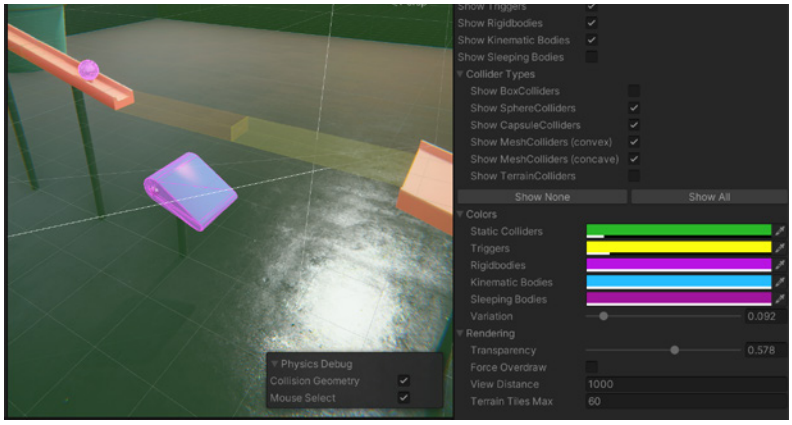
Компонент Rigidbody поддерживает спекулятивный и основанный на sweep-проверке режимы непрерывного обнаружения столкновений для быстро движущихся объектов.

Используйте несколько физических сцен для управления сложными физическими контекстами или обхода их ограничений. Например, можно моделировать несколько физических сцен, чтобы прогнозировать столкновения и траектории объектов GameObject.



Используйте несколько сцен для прогнозирования столкновений и траекторий.

Unity также предоставляет окно Debug Visualization (Window > Analysis > Physics Debugger). В нём можно раскрашивать физические компоненты по категориям и включать или отключать их отображение. Это помогает быстро диагностировать проблемы с Collider и другие физические сценарии в сцене.



Окно Physics Debugger

Обновляйте физику в методе MonoBehaviour.FixedUpdate, который вызывается через фиксированные интервалы времени. Класс Physics содержит статические методы для физических взаимодействий. Кроме того, класс Collider предоставляет события для обработки столкновений и триггеров. Ниже перечислены методы, часто используемые при работе с физикой.

Классы и методы	Описание
MonoBehaviour.FixedUpdate	Обновляет физические расчёты с фиксированной частотой, задаваемой в Edit > Settings > Time > Fixed Timestep.
Physics.Raycast, Physics.Linecast, Physics.BoxCast, Physics.CapsuleCast, Physics.SphereCast	Проверяют пересечение Collider с лучом, линией или объёмом.
Physics.OverlapBox, Physics.OverlapSphere, Physics.OverlapCapsule	Проверяют, находится ли Collider внутри заданного объёма.
Collider.OnCollisionEnter, Collider.OnCollisionStay, Collider.OnCollisionExit	События начала, продолжения и завершения контакта Collider.
Collider.OnTriggerEnter, Collider.OnTriggerStay, Collider.OnTriggerExit	События триггера при начале, продолжении и завершении контакта; физического столкновения не создают.

2D-физика

В Unity есть отдельный физический движок, оптимизированный для 2D-физики. Большинство её компонентов - двумерные аналоги компонентов 3D-физики.

К именам соответствующих компонентов добавлено «2D»: например, Physics2D, Rigidbody2D, Collider2D, Joint2D и т. п. В остальном они ведут себя так же, как аналогичные 3D-классы.

Загрузите проект PhysicsExamples2D с примерами применения и посмотрите сопутствующее вводное видео.

Подробнее о встроенной реализации 3D- и 2D-физики читайте в документации Physics. Советы по оптимизации 3D-физики и её настроек приведены в видео Physics Performance Optimization.

Аудио

Аудиосистема Unity предоставляет расширенные возможности воспроизведения звука в трёхмерном пространстве. Unity также может записывать звук с любого доступного микрофона для использования во время игры, хранения или передачи.

Компоненты аудиосистемы устроены по аналогии с реальными устройствами. Добавьте AudioSource к объекту GameObject, чтобы он мог воспроизводить аудиофайл AudioClip. Компонент AudioListener в другой части сцены, обычно на основной Camera, воспринимает этот звук.

Благодаря этому аудиосистема может воспроизводить эффект Доплера, когда источник и слушатель движутся относительно друг друга. Другие эффекты, например эхо, моделируются с помощью Audio Filters.

AudioMixer сводит различные источники звука, применяет к ним эффекты и выполняет мастеринг.

AudioMixer

Ниже перечислены наиболее распространённые классы аудиокомпонентов.

Класс	Описание
AudioClip	Импортированный аудиофайл, доступный скриптам. Даёт аудиосистеме доступ к закодированным данным; метаданные могут загрузиться раньше самих данных.
AudioSource	Компонент GameObject для воспроизведения звука в трёхмерной среде.
AudioListener	Подобен микрофону и воспринимает звуки вокруг себя; в сцене используется один.
AudioMixer	Цепь обработки сигнала для изменения громкости и высоты тона, применения эффектов и мастеринга.

Конвейеры рендеринга и графика

Без графики игра не будет полной. До настройки освещения сцен необходимо выбрать один из конвейеров рендеринга. Конвейер выполняет последовательность операций, преобразующих содержимое сцены в изображение на экране.

Конвейеры 3D-рендеринга

Unity предоставляет три готовых конвейера рендеринга с разными возможностями и характеристиками производительности. Можно создать и собственный конвейер.

- Built-in Render Pipeline - универсальный конвейер рендеринга с ограниченными возможностями настройки. Он используется по умолчанию.

- Universal Render Pipeline (URP) - готовый конвейер Scriptable Render

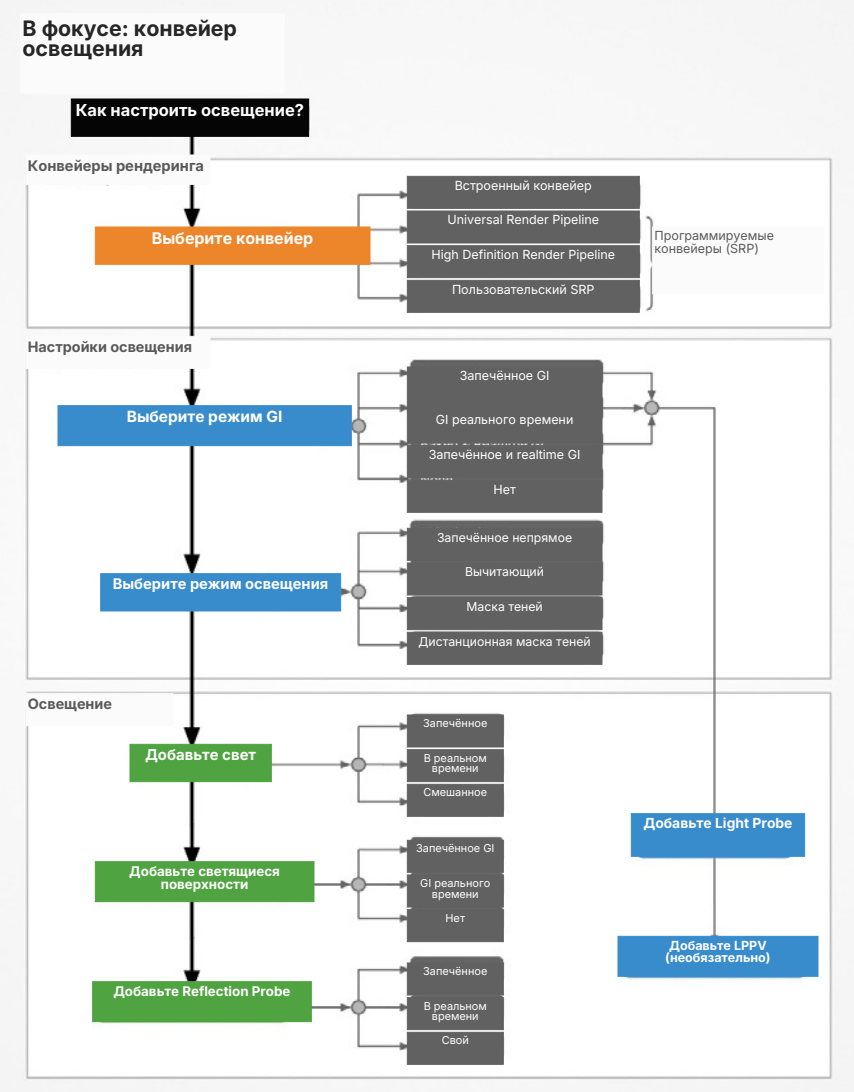
Pipeline (SRP). URP предлагает удобные для художников процессы создания оптимизированной графики. Выбирайте URP, если один проект Unity должен работать на нескольких платформах - от мобильных устройств и настольных компьютеров до консолей.

URP добавляет графические возможности и функции рендеринга, которых нет в Built-in Render Pipeline. Чтобы сохранить производительность, он использует компромиссные решения, снижающие вычислительную нагрузку освещения и затенения. URP работает на всех платформах, поддерживаемых Unity, и масштабируется с учётом их возможностей. Выбирайте URP для мобильных устройств, Nintendo Switch или Oculus Quest.

- High Definition Render Pipeline (HDRP) - ещё один готовый конвейер Scriptable

Render Pipeline, рассчитанный на производительное оборудование: ПК, Xbox и PlayStation. HDRP обеспечивает наиболее качественные физически корректные освещение и рендеринг из доступных в Unity. Выбирайте HDRP, если стремитесь к фотореализму.

HDRP поддерживает разные типы источников света (Spotlight Pyramid/Box, Realtime Tube и Realtime Rectangular) и расширенные виды отражений: зонды со смешиванием, плоские отражения и отражения в экранном пространстве. Точно настраивайте материалы с помощью высококачественных шейдеров и эффектов, а для диагностики рендеринга используйте инструменты отладки конвейера.



Выберите конвейер рендеринга на раннем этапе планирования проекта.

И HDRP, и URP включают:

- Shader Graph - инструмент для создания шейдеров в визуальном узловом редакторе без написания кода HLSL. Благодаря этому разработчики могут передать часть работы над шейдерами художникам или техническим художникам.

- Демонстрационные сцены с примерами настройки освещения, материалов и шейдеров.

> Несколько заранее настроенных ресурсов Render Pipeline Asset позволяют быстро переключать уровни качества графики, используя параметры, уже оптимизированные для соответствующего конвейера.

- Новейший Post Processing Stack: URP и HDRP включают собственные системы постобработки, оптимизированные для соответствующих конвейеров. Они позволяют применять полноэкранные фильтры через удобный для художников интерфейс.

- Утилита Render Pipeline Debug с инструментами визуализации для диагностики проблем освещения, затенения, теней и других эффектов.

URP и HDRP работают поверх Scriptable Render Pipeline - тонкого слоя API, позволяющего планировать и настраивать команды рендеринга с помощью скриптов C#. Такая гибкость даёт возможность изменять почти любую часть конвейера. На основе SRP можно создать и собственный конвейер рендеринга.

Built-in Render Pipeline настраивается не так гибко, как URP или HDRP, но поддерживает множество платформ. Для работы с ним необходимо настроить пути рендеринга и расширить его возможности с помощью буферов команд и функций обратного вызова.

Короткометражный фильм The Heretic демонстрирует передовые графические возможности HDRP.

Более подробное сравнение доступных конвейеров см. в материале Render pipelines in Unity.

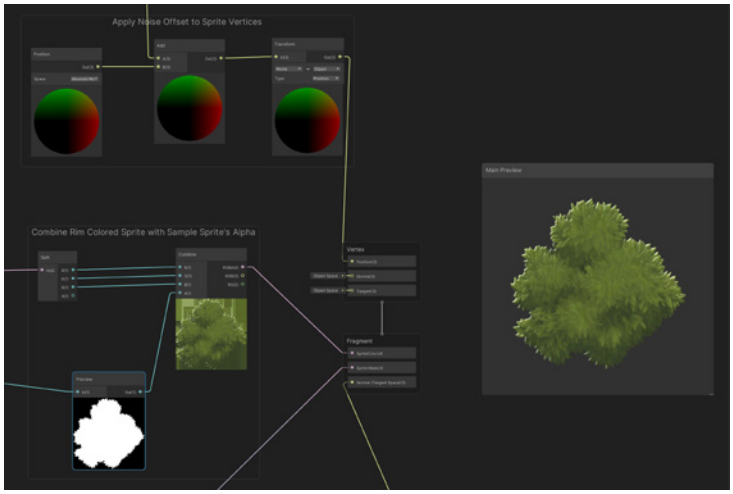
Конвейер 2D-рендеринга

URP также включает 2D Renderer. Если вы создаёте игру на основе спрайтов, он предоставляет инструменты, специально предназначенные для создания 2D-проектов:

- 2D Lights могут иметь разные формы: Freeform, Sprite, Parametric, Point и Global. Они могут взаимодействовать с текстурами карт нормалей и масок, если эти текстуры назначены как Secondary Texture в Sprite Editor. Это позволяет создавать сложные эффекты освещения.

Источники 2D Light в примере проекта Lost Crypt

- Shader Graph содержит главные узлы Lit Sprite и Unlit Sprite. Они упрощают набор входов узла, позволяя передавать подходящие узлы Sample Texture2D для 2D-рендеринга.



2D Shader Graph

- Пакет 2D Pixel Perfect содержит компонент Pixel Perfect Camera, благодаря которому пиксельная графика остаётся чёткой при разных разрешениях и стабильной в движении.

Рекомендуем начать с одной из доступных 2D-демонстраций, например Lost Crypt или Dragon Crashers. Эти примеры покажут, как объединить 2D-инструменты в полноценный проект реального времени.

Создание игрового мира

Игре нужны уровни. Следующие инструменты помогут их создавать.

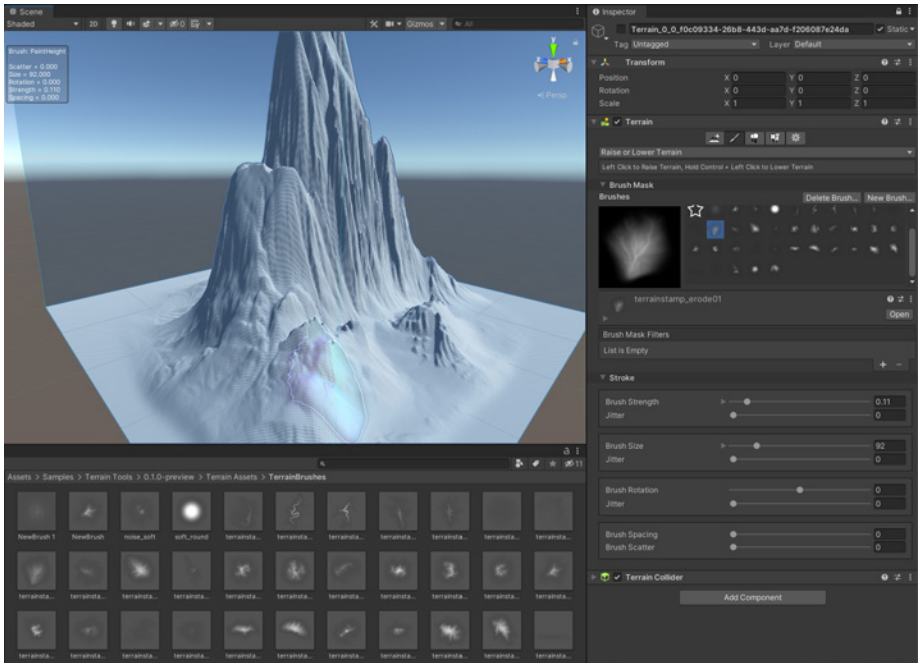
ProBuilder и Polybrush

Unity включает дополнительный пакет ProBuilder - упрощённый инструмент 3D-моделирования и проектирования уровней. Он позволяет быстро создавать прототипы сооружений, пользовательскую геометрию столкновений, триггерные зоны и навигационные меши. ProBuilder оптимизирован для простой геометрии и чернового блокинга уровней.

Polybrush позволяет скульптить меши, смешивать текстуры, раскрашивать вершины и расставлять объекты на уровнях. Вместе с ProBuilder он образует полный набор инструментов проектирования уровней. Для дальнейшей доработки моделей можно организовать двусторонний обмен с выбранным DCC-пакетом, например Maya или Blender.

Инструменты Terrain

Для создания сложного природного 3D-окружения Unity Editor содержит набор инструментов Terrain, доступный как пакет в Unity 2021.2 и новее. В Editor можно создать несколько плиток Terrain, изменить высоту и внешний вид ландшафта инструментами-кистями, а также добавить деревья и траву.

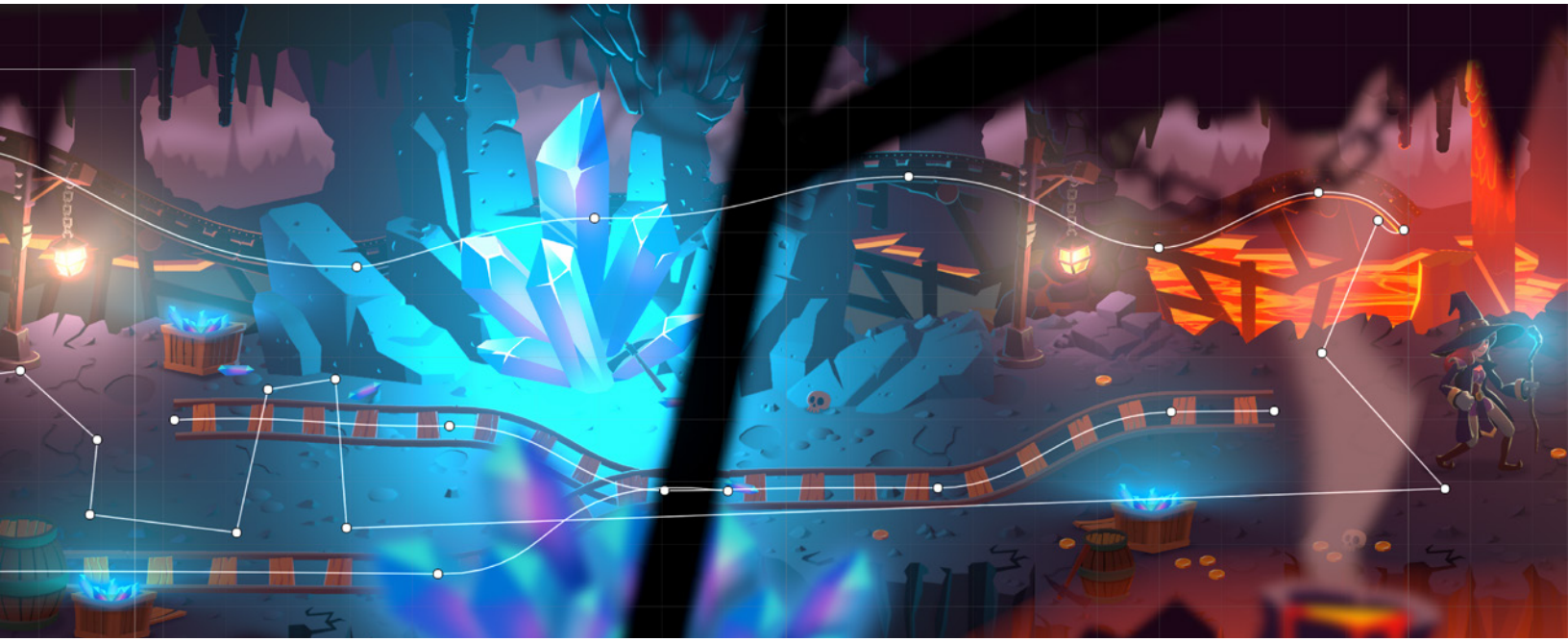


Инструменты Unity Terrain

2D Tilemap и Sprite Shape

Работаете в 2D? С помощью Tilemap можно создавать большие миры на основе сетки, включая шестиугольное и изометрическое окружение. Система хранит и обрабатывает ресурсы Tile для построения 2D-уровней. Затем можно настроить Tile Palette и рисовать 2D-карты Tilemap различными кистями.

Sprite Shape позволяет создавать насыщенное органичное 2D-окружение произвольной формы с помощью наглядного и понятного рабочего процесса. Инструмент динамически размещает спрайты вдоль сплайнов с учётом заданных диапазонов углов. Спрайты на сплайне автоматически деформируются, а их варианты переключаются в зависимости от угла контура.



Инструмент Sprite Shape позволяет создавать органичное 2D-окружение.

Unity Asset Store

Unity Asset Store предлагает разнообразные ресурсы: от текстур, анимаций и моделей до примеров целых проектов, учебных материалов и расширений Editor. Unity Technologies и участники сообщества публикуют ресурсы на этой активно развивающейся площадке. Пакеты ресурсов можно загрузить прямо в проект Unity через Package Manager.

Чтобы открыть Asset Store в интернете, войдите в учётную запись Unity. Там доступны различные ресурсы для 2D, 3D и написания скриптов:

- В разделе Templates можно загрузить учебные проекты и стартовые наборы.

- Раздел Audio содержит библиотеку звуковых файлов для обогащения пользовательского опыта: фоновый звук, музыку, речь и звуковые эффекты.

- В разделах 2D и 3D представлено множество вариантов окружения, реквизита и персонажей во всевозможных художественных стилях и жанрах для наполнения игровых уровней.

- Если времени на разработку мало, просмотрите разделы Tools и Editors.

- Найдите системы с расширенными возможностями - от AI до Visual Scripting.

Импорт ресурса

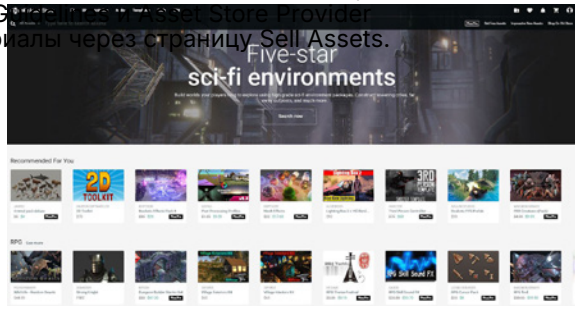
После приобретения бесплатного или платного ресурса он появится в вашей учётной записи Unity.

В Package Manager (Window > Package Manager) выберите фильтр My Assets,

а затем загрузите нужные ресурсы, связанные с вашей учётной записью, и импортируйте их в текущий проект.

Стать издателем

Вы можете стать издателем в Asset Store, продавать свои работы - 3D-модели, расширения Editor, аудио и другие ресурсы - и получать дополнительный доход параллельно с разработкой игры. Для начала создайте учётную запись Asset Store Publisher, ознакомьтесь с Submission Guidelines и Asset Store Provider Agreement. Отправьте материалы через страницу Sell Assets.



Площадка Asset Store для ресурсов сторонних разработчиков

Учебные ресурсы

Используйте обширную документацию и учебные ресурсы Unity, чтобы глубже освоить разработку.

Документация

Unity Manual и Scripting API - наиболее полные руководства по Unity. Unity Manual охватывает практически все возможности и инструменты, а его документация постоянно обновляется.

У пакетов есть собственные сайты с документацией. Последняя версия всегда доступна через Unity Manual, а документацию для конкретной версии пакета можно открыть в окне Package Manager.

Unity Learn

На Unity Learn всем доступно более 750 часов бесплатных учебных материалов по запросу и в прямом эфире. Там можно задавать вопросы, получать советы и работать над реальными проектами при поддержке экспертов Unity и других авторов.

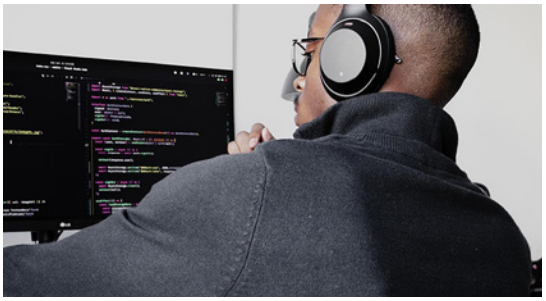
Блог Unity

В блоге Unity публикуются актуальные новости, обновления и сведения о выпусках программного обеспечения. Короткие и понятные статьи охватывают всё: от новых технологий и событий сообщества до бесед с создателями игр на Unity. Художники, программисты, технические художники и дизайнеры найдут в блоге советы и идеи сообщества для профессионального роста.

Профессиональное обучение

Unity Professional Training предоставляет более 200 часов материалов в формате On-Demand Training. Годовая подписка открывает полный доступ к отобранным курсам, коротким видео, проверочным заданиям, практическим испытаниям и сертификатам.

Эти материалы созданы опытными методистами совместно с инженерами и продуктовыми командами Unity. Поэтому ваша команда всегда имеет доступ к самым актуальным учебным материалам по новейшим технологиям Unity. Чтобы начать, ознакомьтесь с каталогом курсов.



Unity Professional Training

Следующие шаги

Надеемся, это руководство поможет вам понять, как Unity приближает вас к творческой цели. Не забывайте: сам путь к цели - уже половина удовольствия.

Unity предлагает инструменты и сервисы, обеспечивающие поддержку на всём пути разработки игры - от большой идеи до большого успеха. Если вы готовы двигаться дальше, начните работать с Unity Pro уже сегодня или обратитесь к одному из наших экспертов, чтобы узнать, чем мы можем помочь.

unity.com