РАЗРАБОТКА ИГРЫ НА UNITY

с нуля до публикации

Дмитрий Денисов

Дмитрий Денисов

Разработка игры на Unity. С нуля до публикации

«Автор»
2021

Денисов Д. В.

Разработка игры на Unity. С нуля до публикации /
Д. В. Денисов — «Автор», 2021

ISBN 978-5-532-94186-1

Это руководство по разработке игры, в котором шаг за шагом описывается процесс её создания на бесплатном движке Unity. Каждый ваш шаг будет сопровождать скриншот с понятным описанием. Вы научитесь создавать игровые объекты и описывать логику их работы, создавать элементы ландшафта, настраивать камеру, графический интерфейс, спецэффекты, звуковые эффекты и многое другое. Книга подойдет для тех, кто только знакомится с основами разработки игр на движке Unity и для тех, кто уже имеет некоторое представление об этом движке, но еще ни разу не публиковал свою игру в открытом доступе. Практикум регулярно обновляется, поэтому вы можете не переживать о том, что это электронное издание потеряет свою актуальность. Последнее обновление произошло летом 2023 года. Игра, описанная в этом практикуме, прошла модерацию и опубликована на платформе Яндекс игры, ссылку вы найдете на первых страницах книги. Создайте свою игру и опубликуйте на платформе для миллионов игроков с возможностью монетизации!

ISBN 978-5-532-94186-1

Содержание

Дмитрий Денисов
Разработка игры на Unity.
С нуля до публикации

Введение

Игровая индустрия сегодня

Компьютерные игры давно стали частью нашей культуры, через них можно донести мысль до широких масс, они развивают интеллект, реакцию и позволяют пережить целый спектр самых разнообразных эмоций. Игровая индустрия является одной из динамичных и успешных отраслей развлечений в современном мире. Сегодня она охватывает широкий спектр игровых форматов, платформ и моделей монетизации, и продолжает привлекать миллионы игроков по всему миру.

Разработка игр представляет огромные возможности для разработчиков, игроков и инвесторов. Она продолжает привлекать талантливых профессионалов, стимулирует инновации в технологиях и развлечениях, и продолжает быть ключевым фактором в развитии современной культуры и развлекательной сферы.

Разработка игр способствует развитию широкого спектра навыков, она требует комбинации навыков программирования, дизайна, математики, логики и многих других областей знаний. Разработчики игр также учатся работать в команде, улучшать свои коммуникационные навыки и эффективно сотрудничать с другими специалистами. Весь этот опыт развивает критическое мышление, аналитические способности и творческое мышление.

Игры имеют потенциал вдохновлять, развлекать и объединять людей. Они могут стать источником радости и развлечения для миллионов игроков по всему миру. Разработчики игр могут создавать уникальные и неповторимые эмоциональные переживания, расширяя возможности и границы игрового мира.

Станете вы профессиональным разработчиком или нет – выбор за вами. В любом случае геймдев – это многогранная профессия, в которой найдется место каждому.

О чем эта книга

В этой книге дано полноценное руководство по разработке игры на Unity с нуля. Без лишней воды и теории, только практика, потому что лучше один раз сделать что-то самостоятельно (даже по руководству), чем прочитать множество гайдов и посмотреть сотни видео на YouTube, но ничего не сделать. В отличие от большого количество книг по разработке и программированию, в этой вы найдете только ту информацию, которая непосредственно относится к созданию вполне конкретного игрового прототипа.

Это значит, что в книге не будет раздела по основам разработки на C# – языке программирования, который используется для написания сценариев в Unity. С другой стороны, вы получите информацию по созданию необходимых скриптов в создаваемой вами игре, а в приведенных листингах будут даны комментарии. Это позволит сконцентрироваться на изучении только тех функций среды разработки, или возможностях языка программирования, которые нужны для выполнения поставленной задачи.

Автор убежден, что важнее заложить базовые знания, дав реализовать свой первый проект. А копнуть глубже и разобраться в тонкостях поможет множество других гайдов, статей, учебников и руководств из официальной документации, “разбросанной” по просторам интернета.

В этой книге вы изучите основы работы с Unity – кроссплатформенной средой разработки компьютерных игр. И опубликуете свою первую игру на платформе yandex.ru/games. Под термином кроссплатформенности понимается возможность сборки готового игрового проекта под разные операционные системы и устройства: PC, iOS, Android, PS, WebGL (браузерные игры) и т. д. Материал ориентирован на новичков, которые только начинают работу в Unity. Поэтому мы начнем с установки необходимого программного обеспечения и создадим первую простую сцену.

Как итог изучения материалов вы сможете опубликовать браузерную игру на платформе WebGL, таких как simmer.io, itch.io и Yandex-игры (yandex.ru/games). Все что нужно для выгрузки игры на хостинг, – это собрать её билд и загрузить архив на хостинг. Хостингов для игр существует большое множество и принцип для многих одинаков. О том как собрать билд и загрузить на хостинг рассказывается в последней главе. **Ознакомиться с демо-версией игры**, которую мы сделаем, можно по ссылке: **yandex.ru/games/app/209669**. Скриншот стартовой сцены игры показан ниже:

Материал книги регулярно обновляется, последнее обновление произошло весной 2023 года. Задать вопросы в случае возникновения трудностей при работе с книгой можно в группе tg: t.me/BigDigitalCourse, дополнительные ресурсы к книге опубликованы на сайте автора: bigdigital-gamelab.ru/book. Успехов в разработке вашей первой игры на движке Unity.

Профессия игродела

Для начала следует отметить, что разработка игр – это целый мир, объединяющий множество профессий и специалистов. Большое заблуждение считать, что игры делают только программисты. И если ваша жизнь не заладилась с разработкой, то это вовсе не повод считать, что в геймдеве вам не место. В создании игр помимо разработчиков программного обеспечения участвуют дизайнеры, художники, аниматоры, композиторы, сценаристы, звукорежиссеры, тестировщики, геймдизайнеры и многие другие специалисты. Каждая из этих профессий имеет свою уникальную роль и вклад в процесс разработки игры.

Программисты как правило отвечают за создание игровой логики, механик, искусственного интеллекта, пользовательского интерфейса и других технических аспектов игры. Они работают с игровым движком, разрабатывают инструменты для удобства работы других специалистов и реализуют игровые функции.

Геймдизайнеры отвечают за создание игрового опыта, геймплея, уровней, сюжета и баланса игры. Они разрабатывают игровые концепции, проектируют игровые механики, задают правила и цели игры. Дизайнеры также тесно сотрудничают с художниками и аниматорами для создания визуального стиля игры.

Художники игр создают визуальные элементы игры, включая персонажей, фоны, объекты и интерфейс. Они отвечают за концепт-арт, моделирование персонажей и объектов, текстурирование, освещение и анимацию. Художники вносят в игру визуальную привлекательность и уникальный стиль.

Аниматоры игр придают жизнь персонажам и объектам игры. Они создают анимацию движения, взаимодействия и эффектов. Аниматоры работают в тесном сотрудничестве с дизайнерами и художниками для достижения плавной и реалистичной анимации в игре.

Композиторы игр создают музыкальное сопровождение и звуковые эффекты, которые обогащают атмосферу игры и передают эмоции игрокам. Они создают музыку, звуки окружающей среды, диалоги персонажей и звуковые эффекты для игровых событий.

Тестировщики игр выполняют тестирование и отладку игры для выявления ошибок, проблем с производительностью и других технических проблем. Они играют в игру, проводят тестовые сценарии и сообщают о найденных проблемах разработчикам.

Если у вас появится желание освоить какую-либо профессию в области разработки игр, то наиболее широкий спектр веток развития в этой сфере представлен в школе school-xyz.com. Исключительно с ознакомительной точки зрения приведу несколько ссылок с описанием вариантов развития вас как специалиста в геймдеве:

– для желающих создавать 3D-модели, трёхмерные объекты, сцены или персонажей – school-xyz.com/3d-professions

– для тех, кто мечтает проектировать игры, создавать игровые миры как Кодзима и разрабатывать уникальные механики – school-xyz.com/gamedesign-professions

– если вы хорошо рисуете или хотите освоить мастерство создания 2D-арта, скетчей и концепта, возможно вам подойдет направление иллюстратора – school-xyz.com/2d-professions

– для тех хочет погрузиться в разработку, создавать прототипы и не только – school-xyz.com/code-professions

– ну и отдельно стоит отметить, что все вышеперечисленные компетенции так же, как и любые команды в IT работают по отлаженным и эффективным бизнес-моделям – school-xyz.com/business-professions

Какую бы профессию вы ни выбрали, все перечисленные специалисты сотрудничают вместе, чтобы создать уникальный игровой опыт для игроков. Разработка игр – это коллективный труд, а создание такие популярных AAA проектов занимает десятки тысяч человеко-часов.

Каждая профессия играет важную роль в достижении общей цели – создания захватывающей и качественной игры. Ну а первым шагом в выборе может стать изучение этого практикума по разработке игр, который заканчивается публикацией вашего первого проекта на открытом веб-ресурсе.

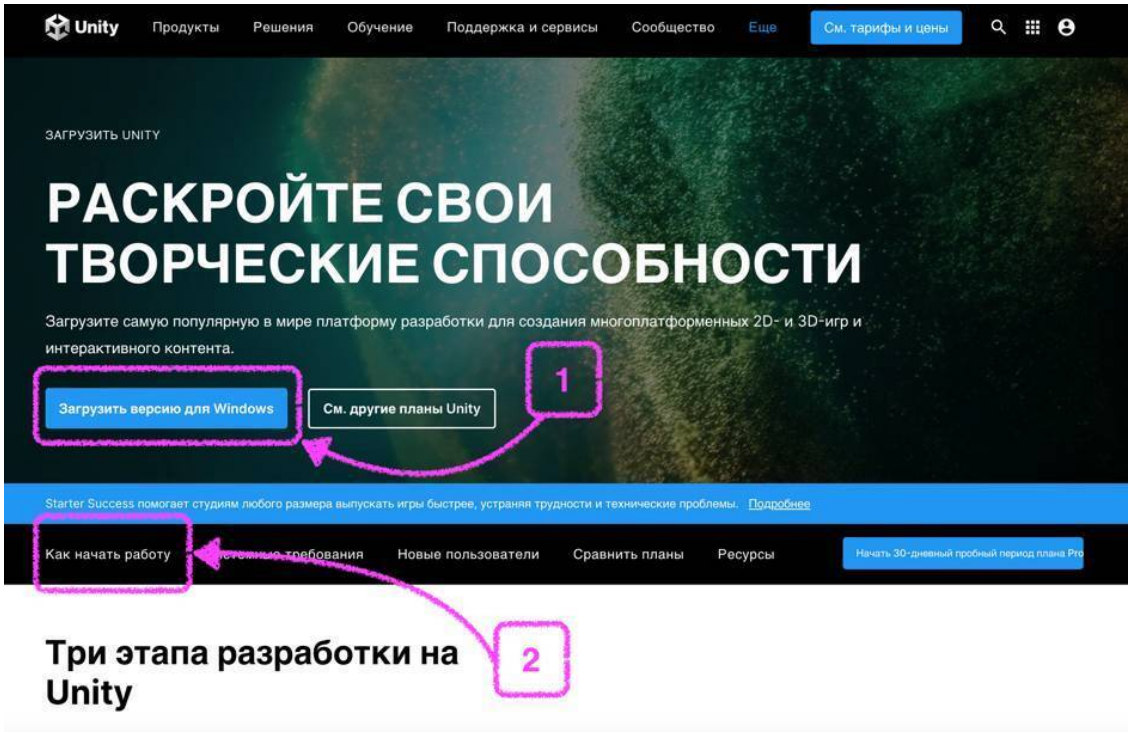
Глава 1. Установка необходимого программного обеспечения

1.1 Установка среды разработки

1. Перейдите на портал unity.com в раздел для загрузки программного обеспечения: unity.com/ru/download. На этой странице можно:

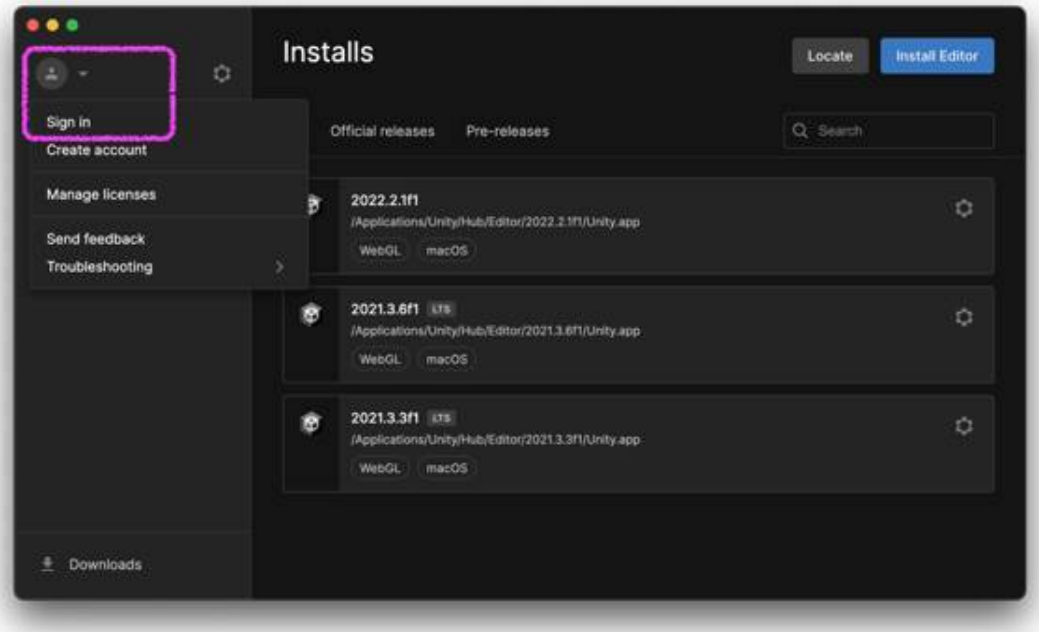
– загрузить версию для Windows [1]

– либо выбрать на той же странице «Как начать работу» [2] и скачать версию для MacOS или Linux.



2. Разрабатывать на Unity можно в основных операционных системах, что, несомненно, играет большую и важную роль в популярности этой среды разработки игр. После завершения загрузки у вас на компьютере окажется установщик UnityHubSetup, по сути, это загрузчик (лаунчер) для ваших будущих проектов. Установка Unity Hub стандартная, дважды кликните по установщику UnityHubSetup и дождитесь окончания установки.

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



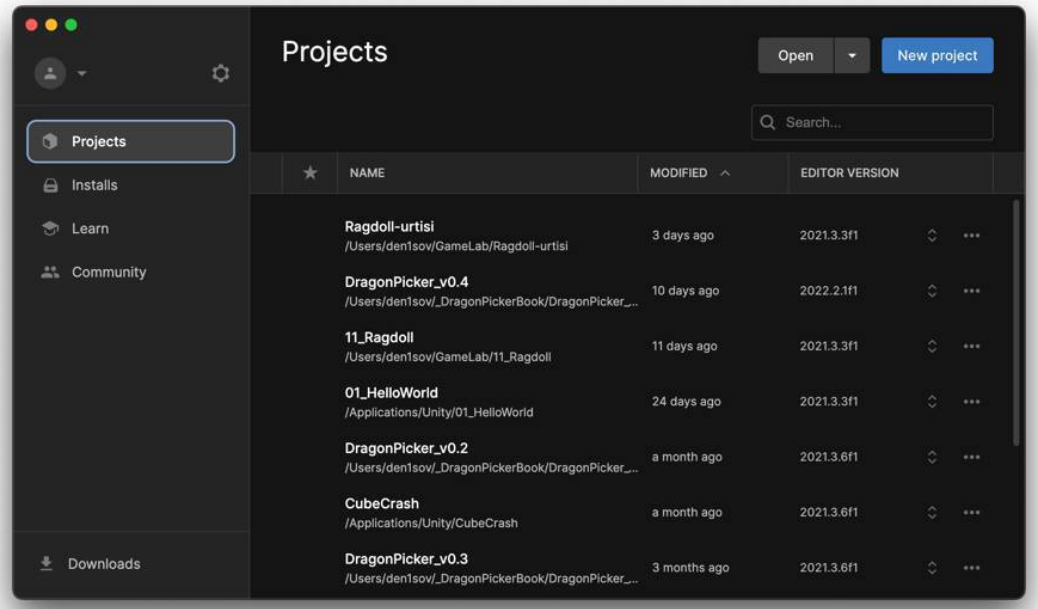
Процесс создания учетной записи для Unity стандартный, поэтому мы не будем здесь расписывать его подробно. Отдельно хочу обратить внимание, что следует использовать единую учетную запись для всех сервисов Unity, с которыми вы работаете. Так как Unity имеет широко развитую инфраструктуру и сеть порталов, облегчающих работу с этим движком:

– assetstore.unity.com – каталог 2D- и 3D-моделей, SDK, шаблонов и инструментов для разработки игр.

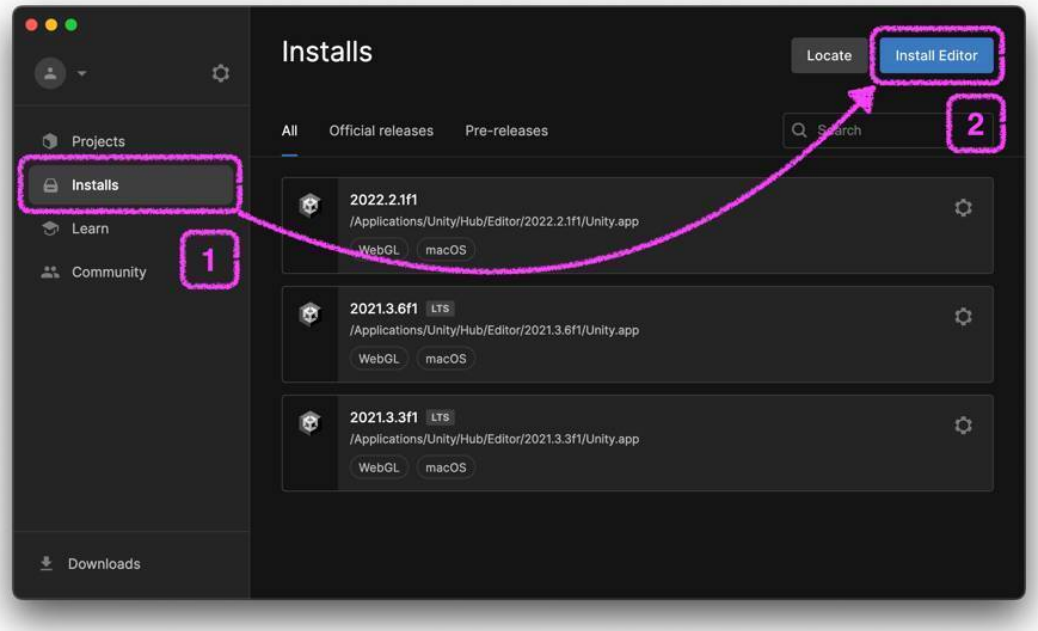
– learn.unity.com – портал с бесплатными учебными материалами, курсами и пошаговыми инструкциями для изучения работы в Unity.

Единая учетная запись облегчит навигацию и работу в сервисах Unity и позволит в пару кликов добавлять модели, найденные в каталоге assetstore.unity.com, в ваш проект в среде разработки.

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

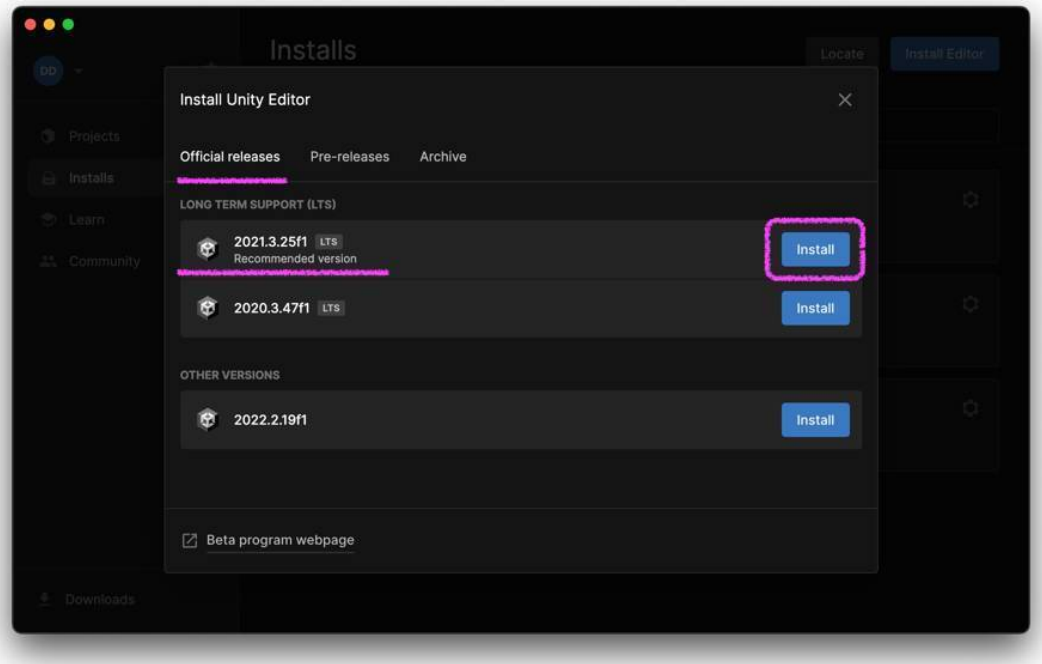


5. Теперь перейдем к установке редактора Unity. Оставаясь в Unity Hub нажмите кнопку Installs [1] в левом меню и далее – Install Editor [2]. Из Unity Hub можно запускать множество разных версий среды разработки Unity.

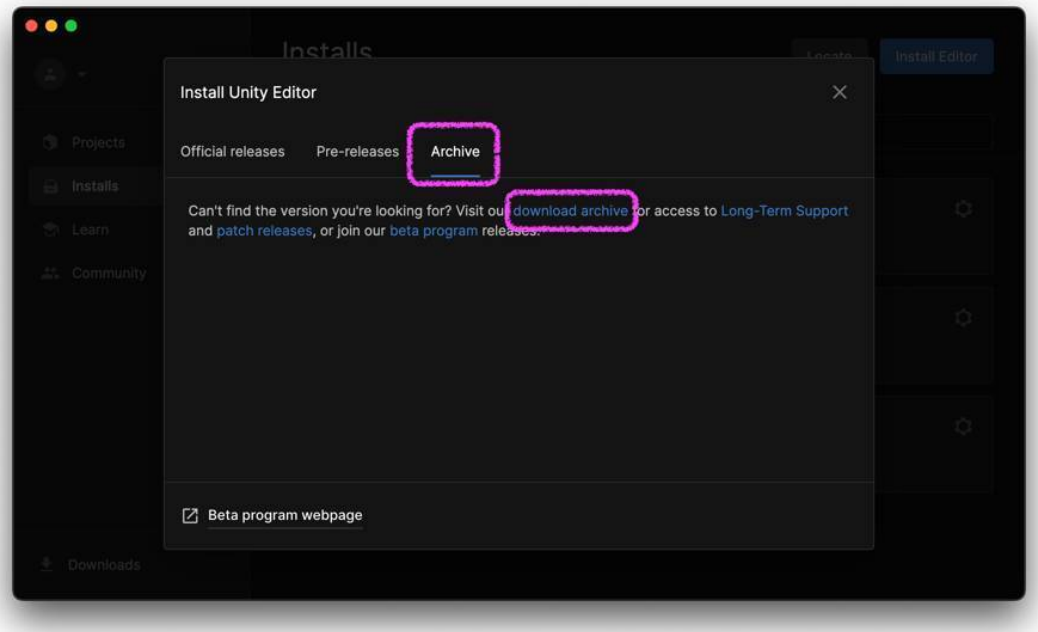


6. После этого откроется окно выбора версий Unity для установки. Для начинающих пользователей лучше устанавливать ту версию Unity, по которой составлено данное руководство. Если же вы являетесь опытным пользователем, то можете выбрать последний рекомендованный разработчиком релиз (Recommended Version) последней версии (как правило выбран по умолчанию). На момент обновления материалов книги в качестве рекомендованной версии

указана версия Unity 2021.3.25f1. Для начала установки в окне Install Unity Editor – Official Releases найдите и установите версию среды разработки 2021.3.25f1. Чтобы начать установку нажмите кнопку Install:



Если позднее вы захотите скачать любую другую версию Unity, перейдите в раздел Archive – download archive и найдите интересующую вас версию на сайте разработчика:

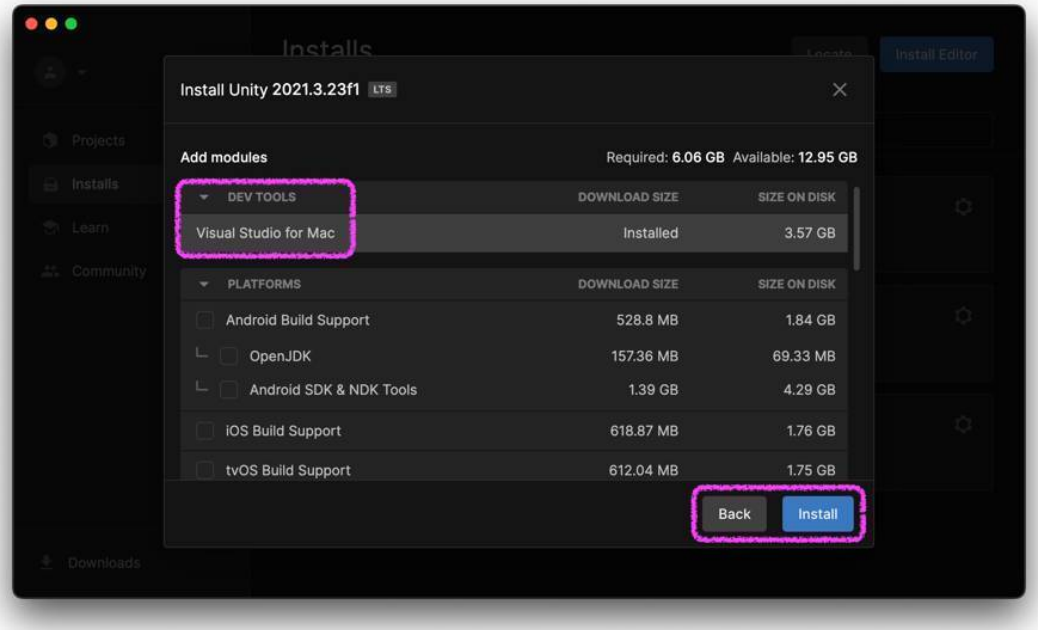


7. После того как вы нажали на кнопку Install, в следующем окне следует выбрать дополнительные модули. Напомню, что Unity позволяет создавать игры под самые разные платформы. Например, если в дальнейшем вы захотите сделать игру под мобильное устройство, то все что вам потребуется – это установить модули Android Build Support и iOS Build Support. На данном шаге нам потребуется установить среду разработки для работы с кодом. Для создания сценариев на Unity используется язык программирования C# и в самом верхнем списке вам предлагается установить Microsoft Visual Studio (если вы работаете на Windows) или Visual Studio for Mac (если вы работает на соответствующей операционной системе). Поставьте галочки напротив:

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

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

– нажмите кнопку Install:



Скачивание и установка модулей и среды разработки займет некоторое время.

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

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

– должна быть установлена среда разработки Unity,

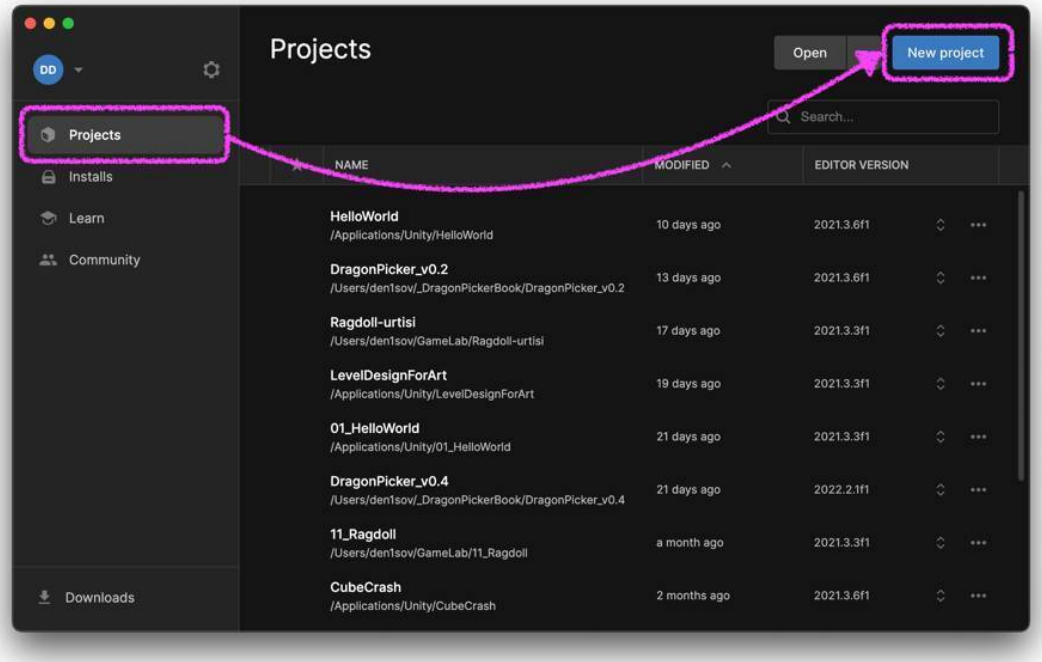
– среда для работы с кодом: Microsoft Visual Studio (для работы из-под Windows) или Visual Studio (для работы из-под Mac или Linux),

– создана учетная запись на сайте Unity.com. Не теряйте ее, так как эту учетную запись вы можете использовать для работы в других сервисах Unity.

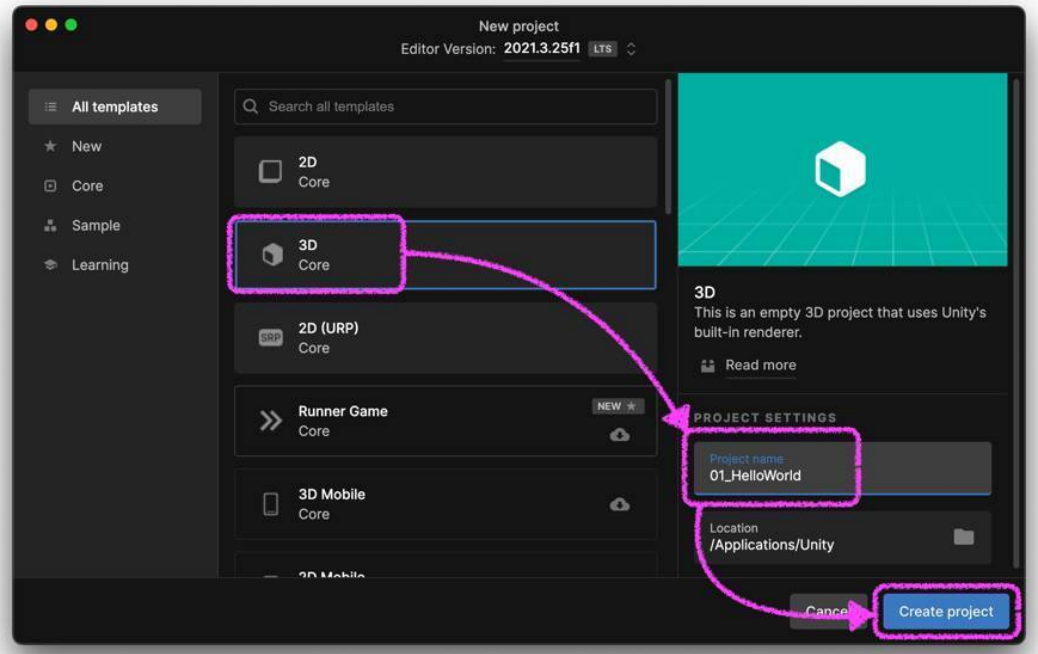
1.2 Первый проект и настройка среды разработки

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

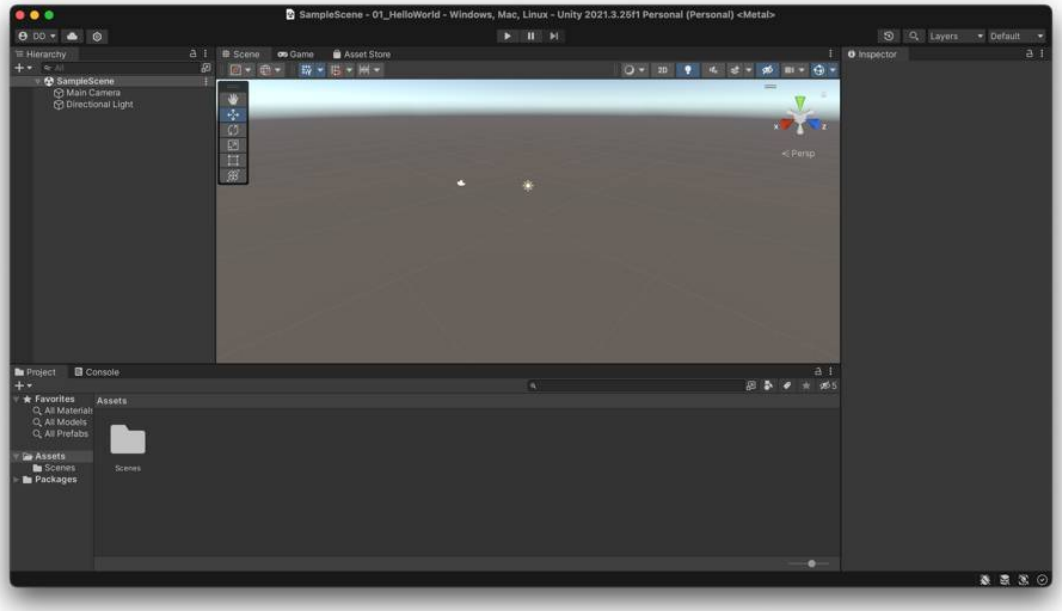
1. Чтобы создать первый проект на Unity, откройте Unity Hub и перейдите во вкладку Projects. Нажмите New project чтобы перейти в окно создания нового проекта:



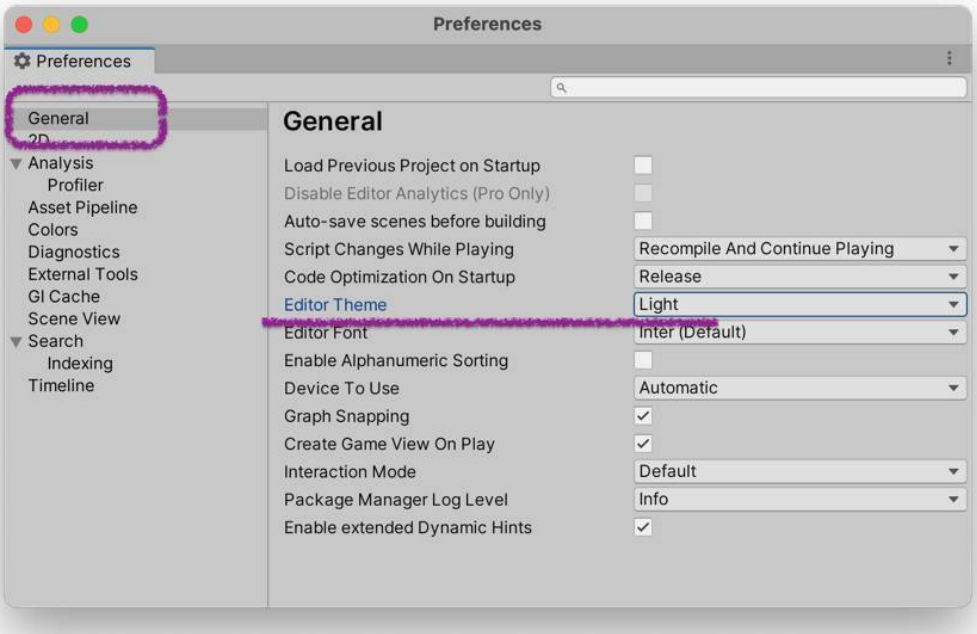
2. В появившемся окне нужно выбрать тип проекта – 3D, дайте имя новому проекту, например 01_HelloWorld. Проверьте путь к папке, в которой будет создан проект (здесь скорее важно, чтобы вы осознанно указали папку для проекта и не потеряли его в дальнейшем). После этого нажмите Create project:



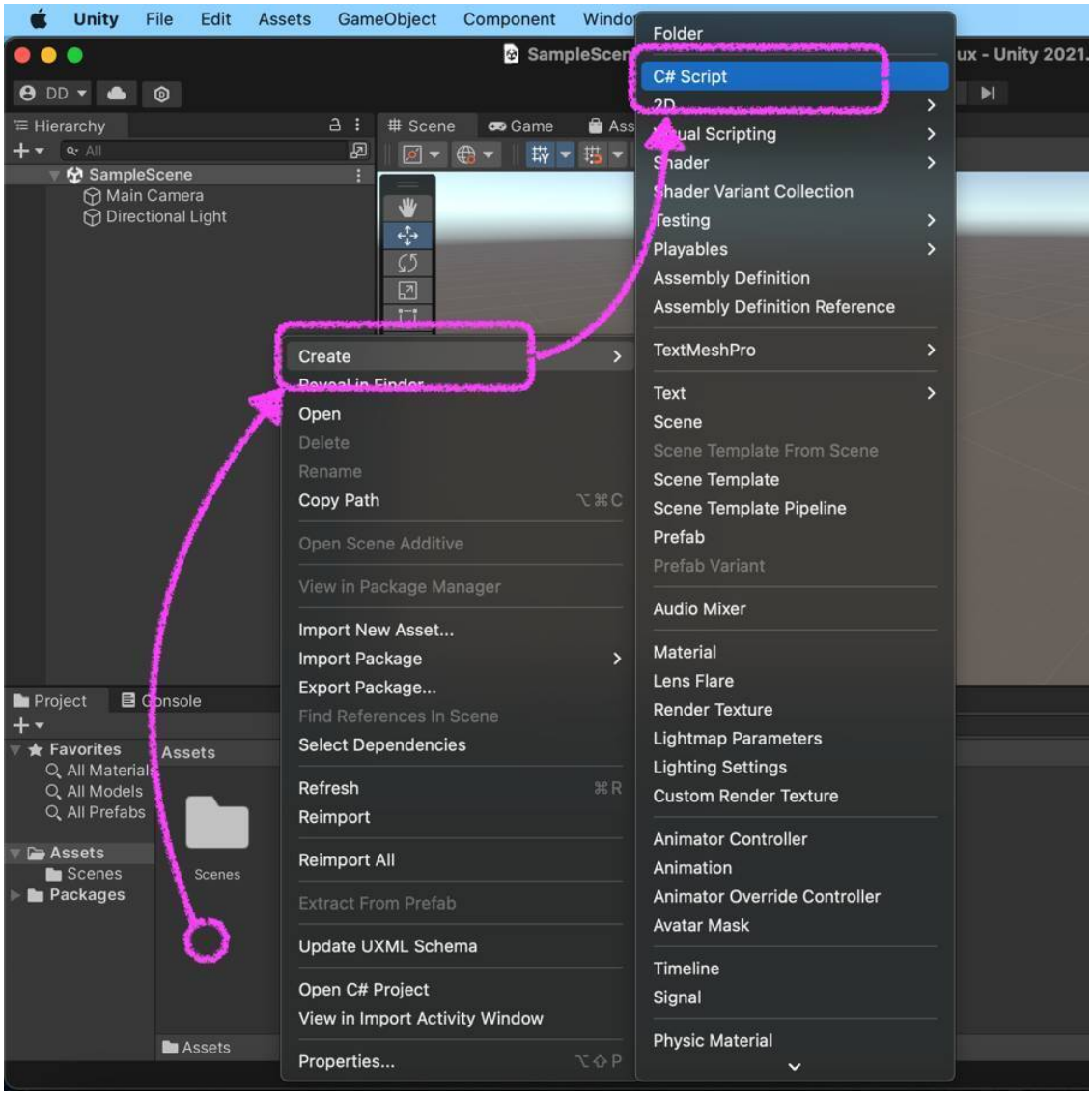
Проект будет создан и открыт в новом окне Unity. Первый запуск проект может быть длительным, так как создаются необходимые зависимости в библиотеках проекта. На рисунке ниже показано, как выглядит запущенная среда разработки. На данном этапе вам может показаться, что среда содержит довольно большое количество разнообразных и непонятных окон, но в дальнейшем мы разберемся, как они устроены и за что отвечают ее отдельные элементы:



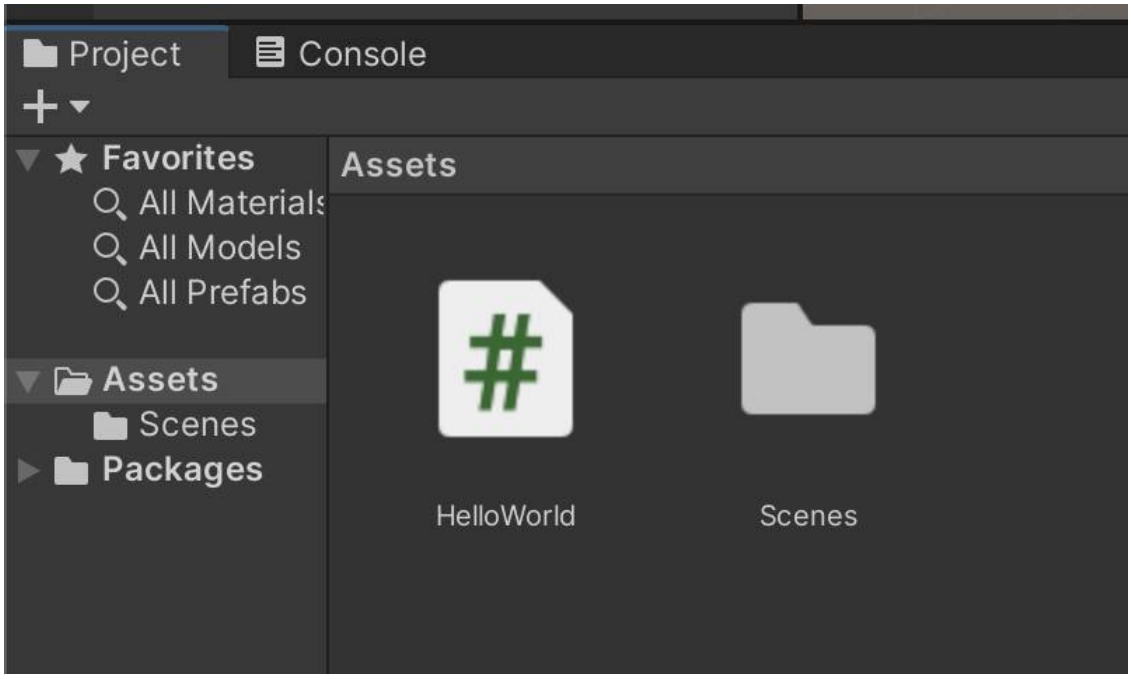
3. Если вы по какой-то причине захотите установить «светлую» тему, в верхнем меню перейдите во вкладку Unity – Preferences (или Settings для Mac) – General – Editor Theme – Light.



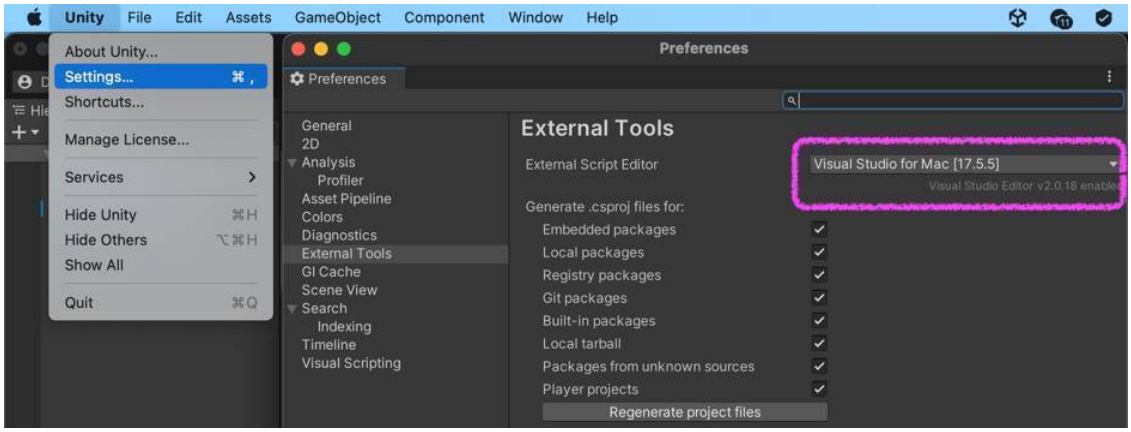
4. Теперь мы готовы к тому, чтобы начать работу. Первым делом создадим новый C# Script-файл с простой командой, которая выводит сообщение “Hello World”. Для этого на панели Project перейдите в папку Assets, в данный момент в ней находится только одна папка с названием Scenes. Кликните правой кнопкой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



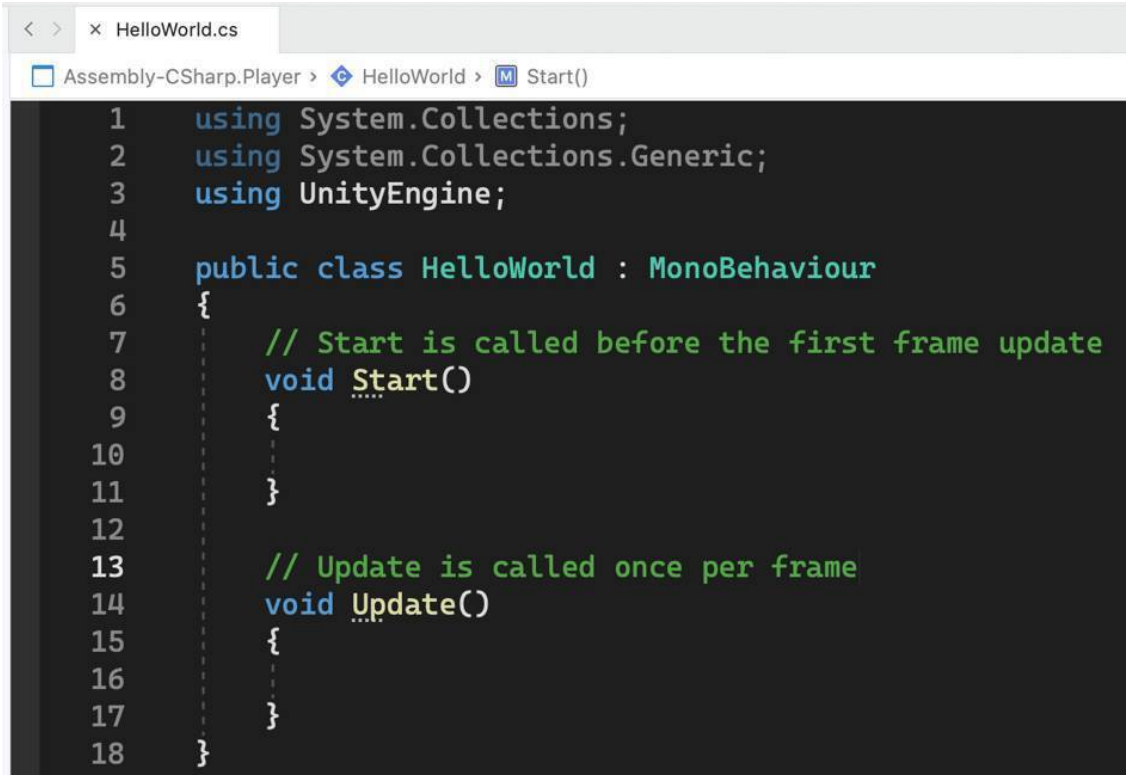
5. Назовите созданный скрипт-файл HelloWorld. Содержимое папки Assets после этого должно выглядеть так, как показано на рисунке ниже:



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



Содержимое файла и вид среды разработки показаны на рисунке ниже:

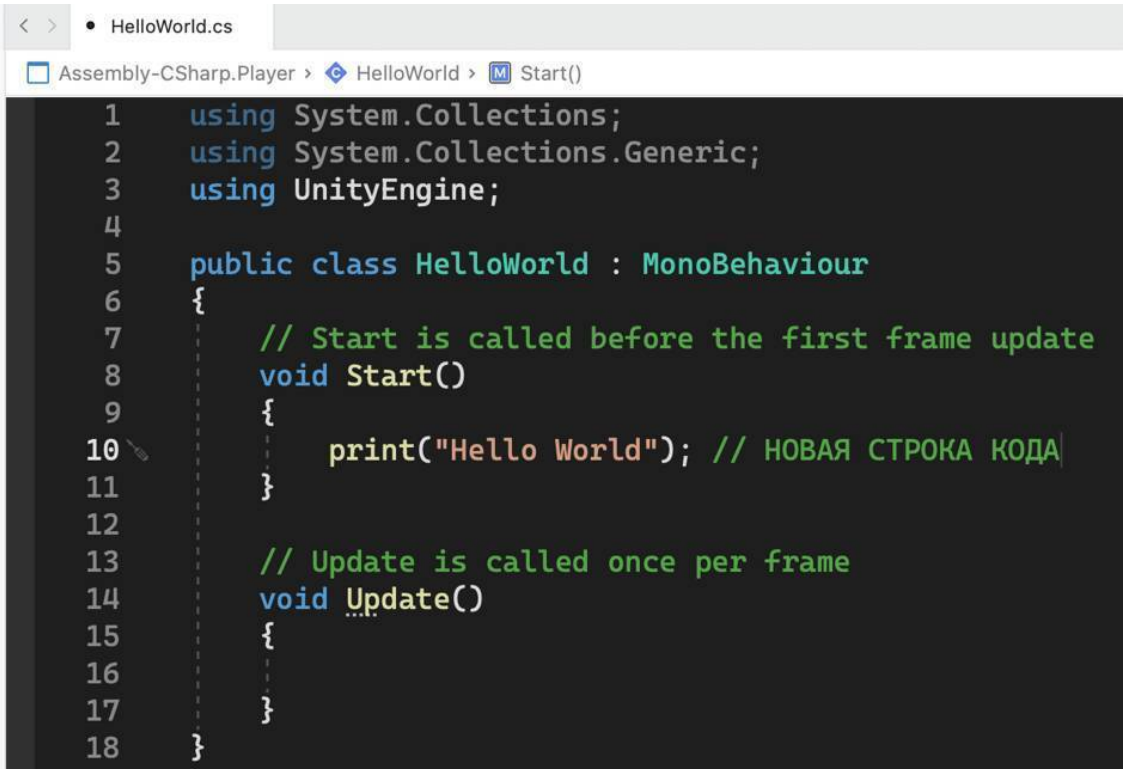


7. Когда мы перейдём к разработке игры, программный код будет приводиться в виде скриншотов, и в виде листинга (то есть в виде текста). Так его будет удобнее воспринимать, а в случае использования электронной версии издания – копировать и вставлять части кода в свой проект. Обратите внимание на то, что внутри кода содержится два метода: void Start() и void Update().

– void Start() – это метод, который запускается при старте игры в Unity. Это значит, что команды, написанные внутри фигурных скобок этого метода, отработают один раз при запуске сцены в Unity.

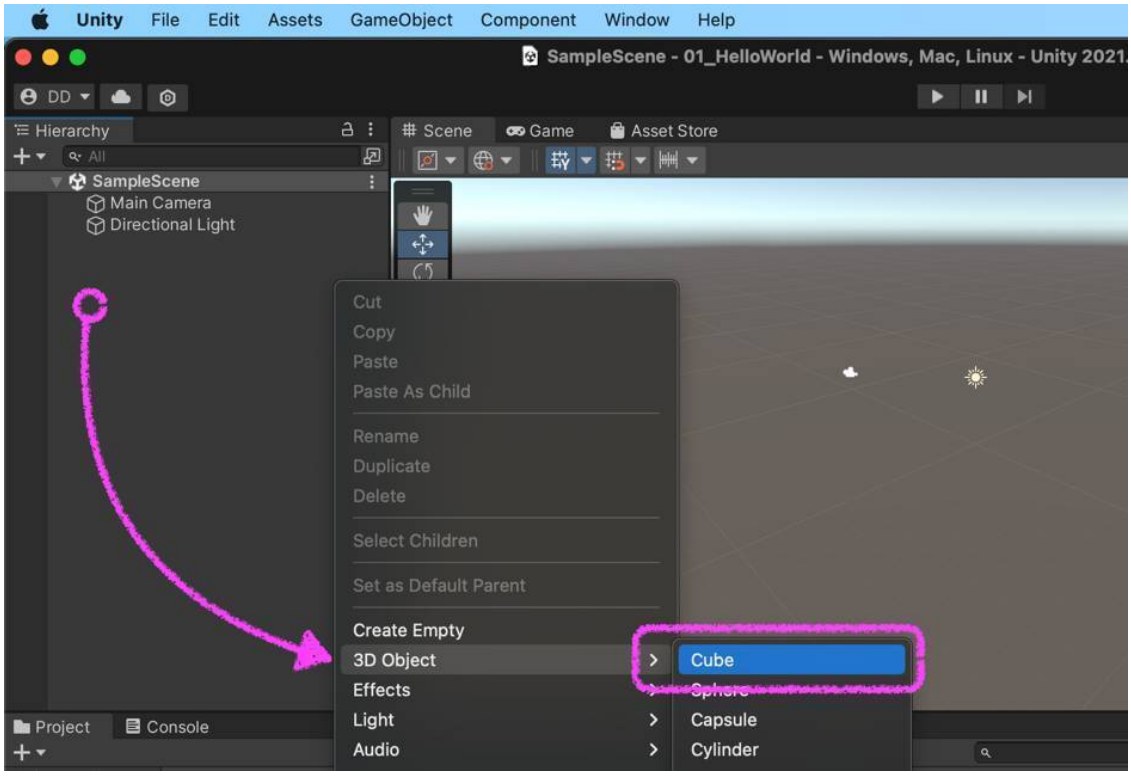
– void Update () – это метод, который запускается каждый кадр на сцене. Другими словами, в метод Update() следует писать тот функционал, который требует регулярного обновления в процессе игры.

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

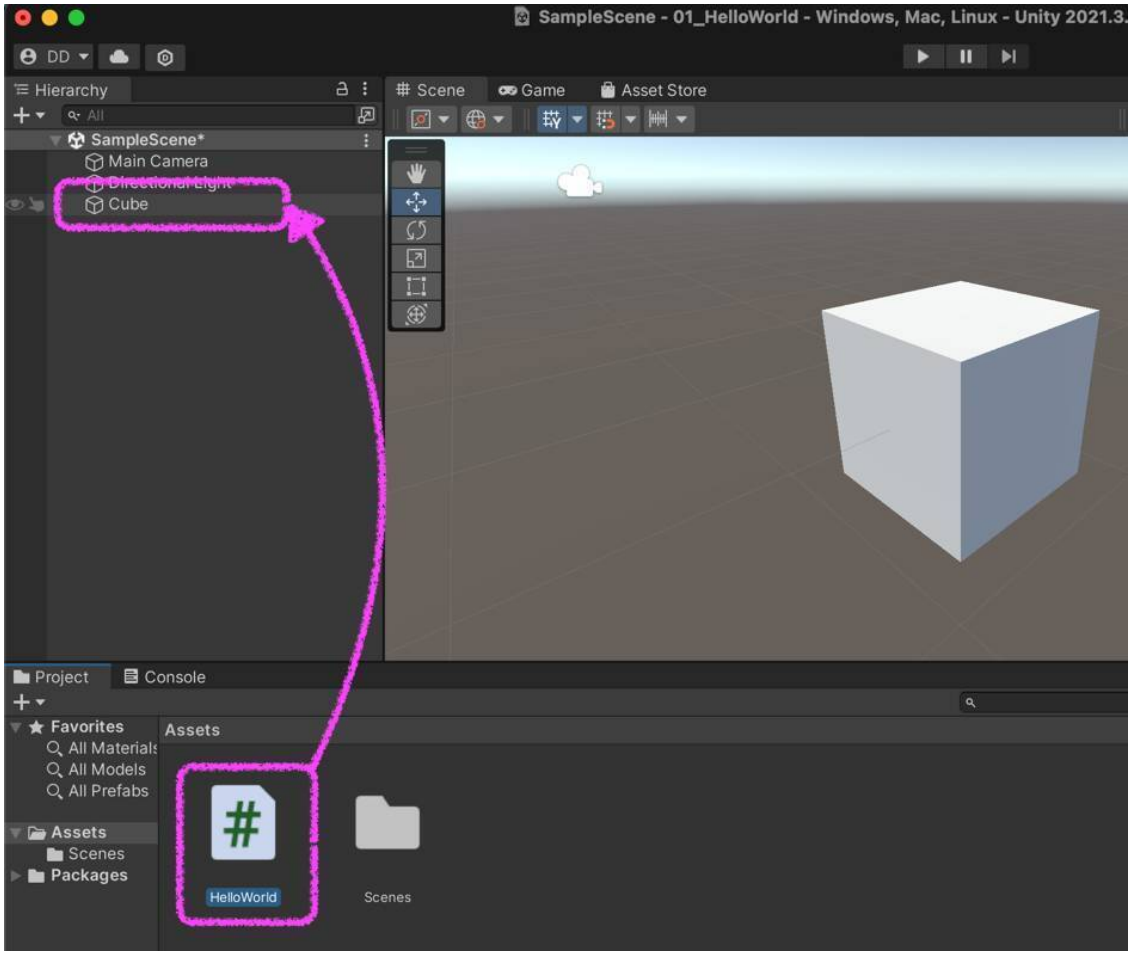


9. Скрипт-файл с названием HelloWorld.cs написан. Однако, чтобы он начал работать, нам следует его подключить к одному из игровых объектов внутри сцены Unity. Давайте создадим такой объект, например, простейший куб.

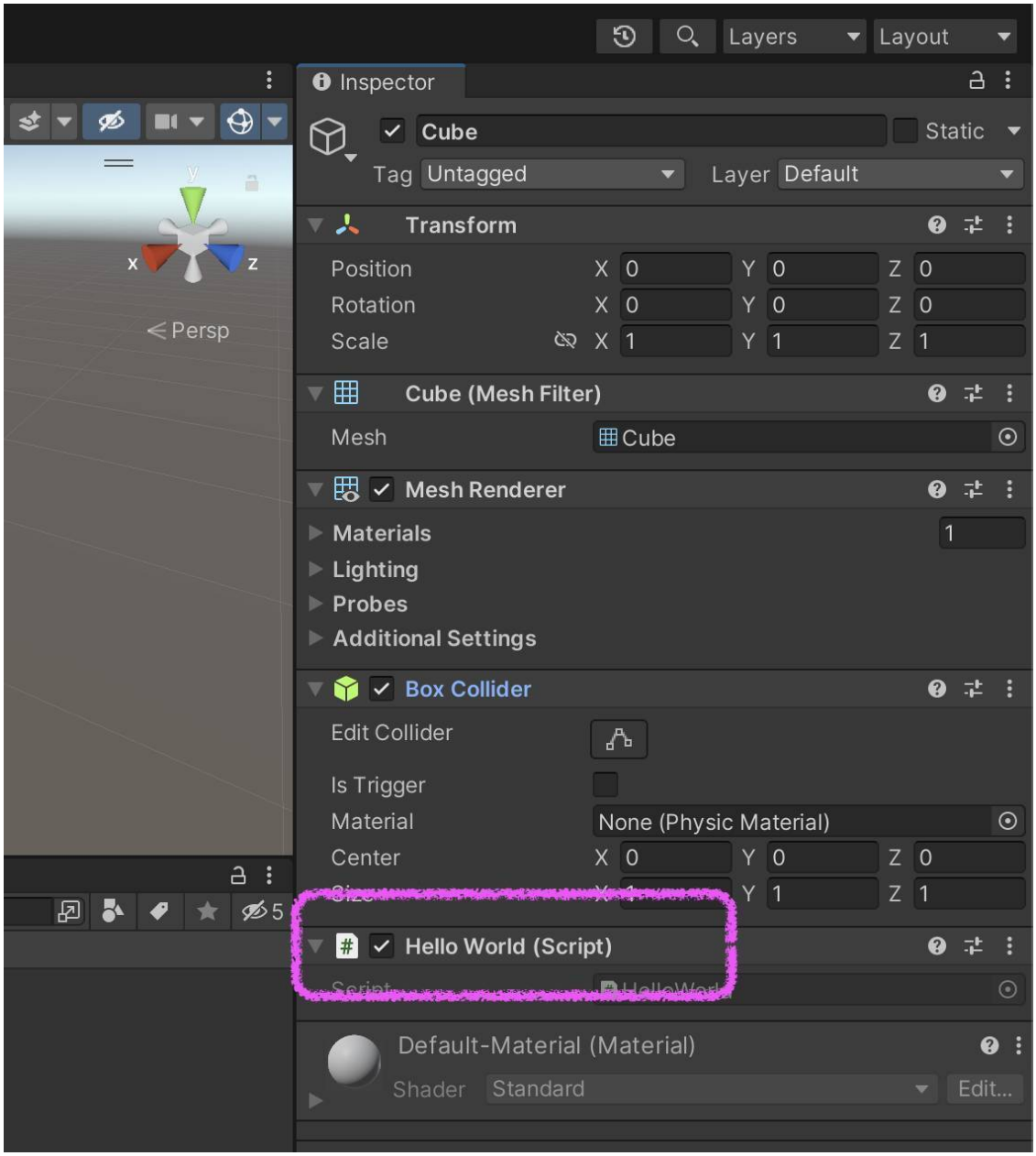
10. Все объекты на сцене находятся в окне иерархии объектов (Hierarchy в левой части среды разработки). Пока мы ничего не создали, но можете обратить внимание что на сцене уже существует камера (Main Camera), которая играет роль глаз игрока и освещением Direction Light, без которого на сцене было бы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



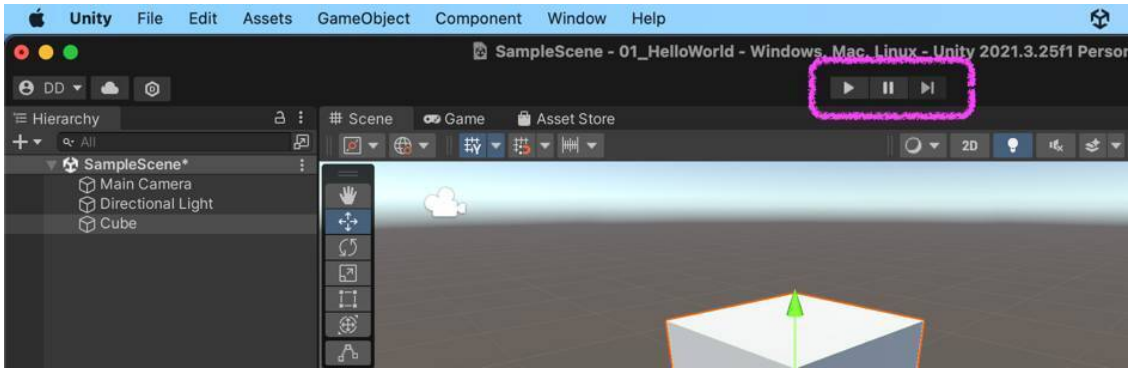
11. Таким образом, на сцене появится новый игровой объект Cube. Приблизить куб можно с помощью колесика мыши. Чтобы подключить скрипт HelloWorld.cs к объекту Cube, можно просто перетащить (зажав левую кнопку мыши) скрипт-файл на куб.



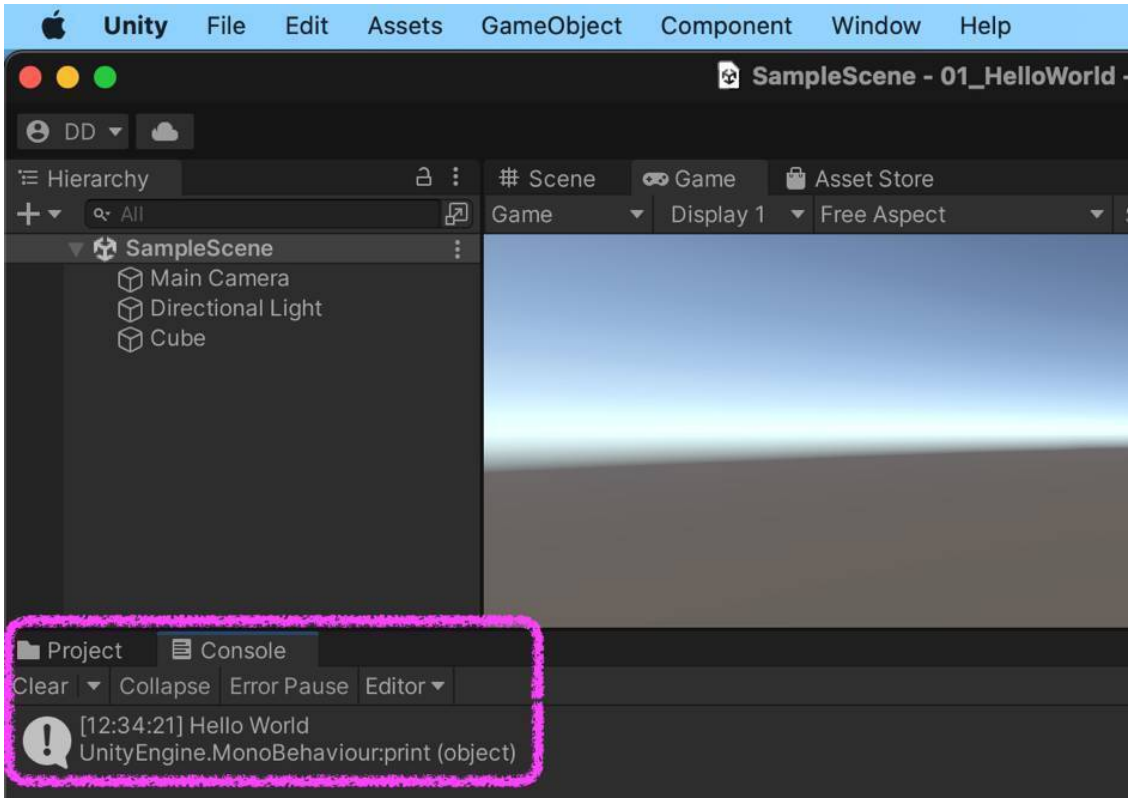
12. Теперь, если на сцене выделить объект Cube, кликнув по нему левой кнопкой мыши, то можно увидеть, что в правой части среды разработки (окно Inspector) к кубу в качестве компонента подключился файл HelloWorld.cs (Script-файл):



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



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



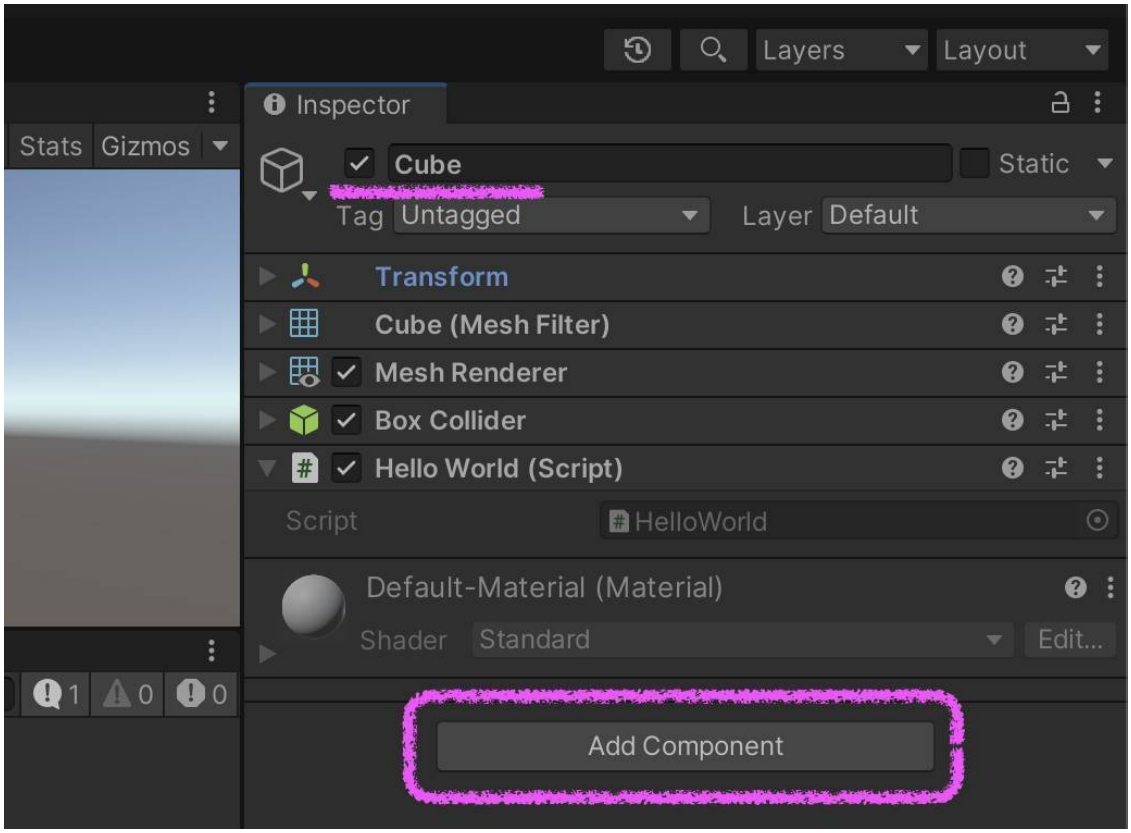
15. Остановите симуляцию сцены, нажав кнопку Run еще раз. Не забывайте выключать симуляцию сцены, так как изменения, которые вы вносите в редакторе, не будут сохраняться при запущенной сцене.

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

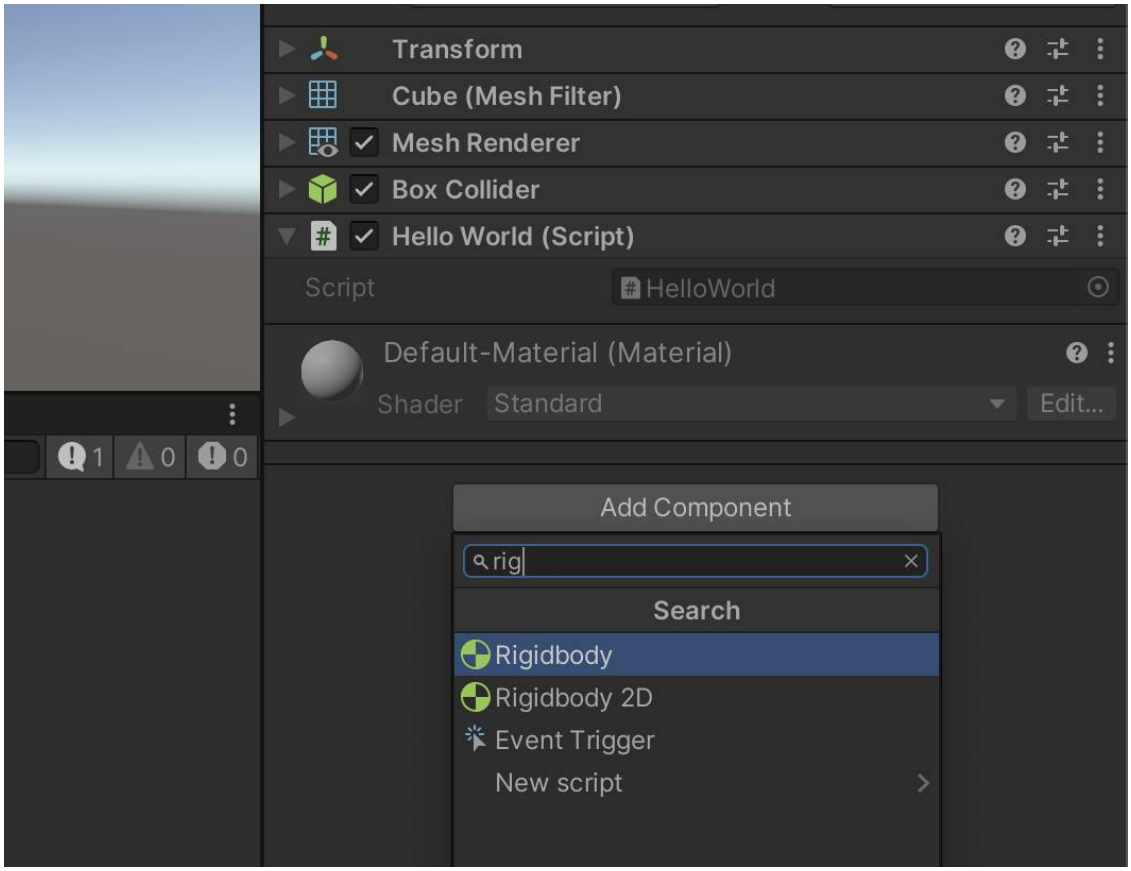
16. Обычным сообщением в окне консоли сложно удивить, особенно если речь идет о разработке игры. Поэтому давайте добавим еще немного функций для наглядности. Следует отметить, что многие моменты, связанные с разработкой игры, на себя берет Unity без необходимости написания кода. Иногда даже очень сложный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

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

18. Содержание окна Inspector зависит от типа выбранного объекта. В нем содержатся свойства объекта, его параметры, подключаемые Script-файлы и т. д. Нажмите в нижней части окна Inspector кнопку Add Component:



19. После этого появится список с перечнем компонентов, которые могут быть подключены к выбранному объекту Cube. Найдите с помощью поиска компонент Rigidbody и кликните по нему левой кнопкой мыши так, чтобы он добавился в окно Inspector.

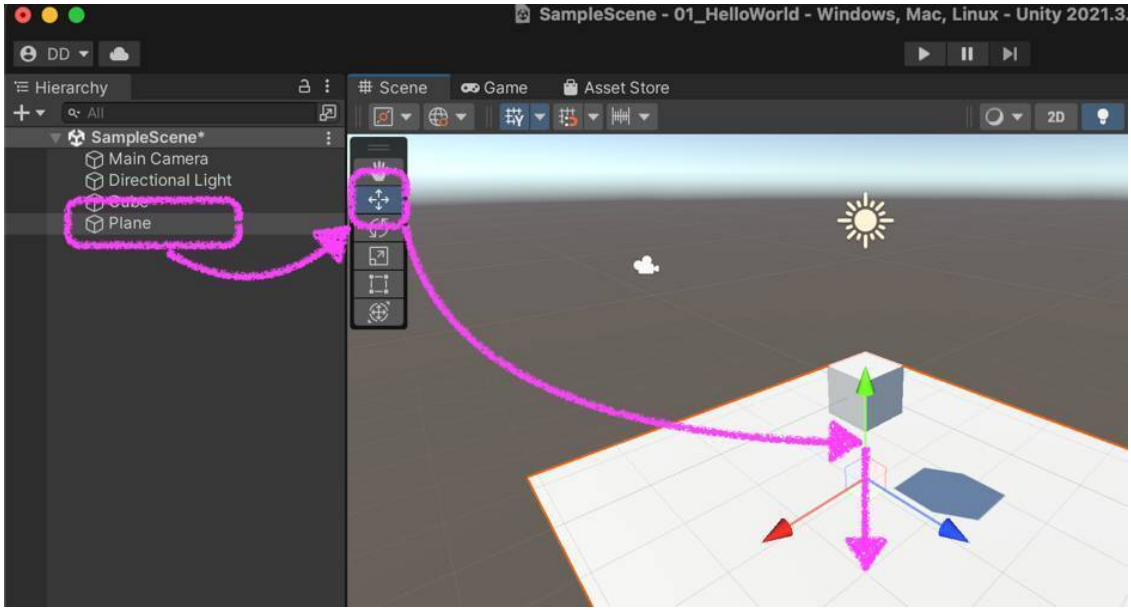


20. Компонент Rigidbody добавляет объекту свойства физики твердого тела, определенного в базовом движке Unity. Другими словами, если назначить этот компонент объекту, то он начнет вести себя в соответствии с законами механики: иметь массу, участвовать в упругих столкновениях, действовать на другие объекты с теми же свойствами и так далее. Запустите сцену еще раз (нажмите Run) и убедитесь, что теперь объект Cube начинает падать вниз.

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

22. После создания плоскости переместим ее немного ниже уровня объекта Cube. Для этого выделите объект Plane в окне Scene выберите инструмент перемещения Move Tool, клик-

нув на ось Y сдвиньте плоскость Plane ниже куба. Используя Move Tool вы можете двигать объекты на сцене:



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

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

HelloWorld.cs

Ничего не выбрано

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class HelloWorld : MonoBehaviour
6   {
7       // Start is called before the first frame update
8       void Start()
9       {
10          print("Hello World"); // НОВАЯ СТРОКА КОДА
11      }
12
13      // Update is called once per frame
14      void Update()
15      {
16          if (Input.GetKeyDown(KeyCode.Space))
17          {
18              Destroy(this.gameObject);
19          }
20      }
21  }
22
```

В листинг были добавлены следующие строки кода:

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

Теперь запустите сцену и проверьте, что она работает следующим образом:

– В окно Console выводится сообщение *“Hello World”*;

– Куб (Cube) начинает падать;

– Куб падает на плоскость Plane и останавливается;

– При нажатии на клавишу пробел объект Cube удаляется.

Выводы

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

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

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

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

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

Часть 2. Создание игрового прототипа

Введение

Обычно перед началом разработки следует определиться с основными требованиями, предъявляемыми к игре. Цель нашего практикума по разработке заключается в том, чтобы сделать игру и опубликовать ее на одном из онлайн-ресурсов (подробнее об этом см. Часть 7). Если говорить более обобщенно, то в списке требований, с которыми следует определиться при начале разработки, можно выделить следующие:

– платформа: WebGL (браузерная игра),
– ориентация: ландшафтная,
– длительность игровой сессии: 3–5 минут,
– система управления: легкое управление при помощи мыши и клавиатуры,
– однопользовательская игра.

В игре “Dragon Picker” центральным объектом в игре будет являться дракон, который периодически роняет драконье яйцо. Игровой процесс будет заключаться в том, чтобы ловить летящие вниз объекты. Позднее в игру могут быть добавлены различные виды объектов, одни из которых могут добавлять очки в игре или жизни, либо отнимать их. Для нас важно будет добавить один вид объектов, а разные виды других объектов вы сможете добавить самостоятельно по аналогии. В качестве основного референса можно указать игру Kaboom 1981 года, разработанную компанией Activision. При желании вы без труда найдете её браузерную версию на просторах сети Интернет:

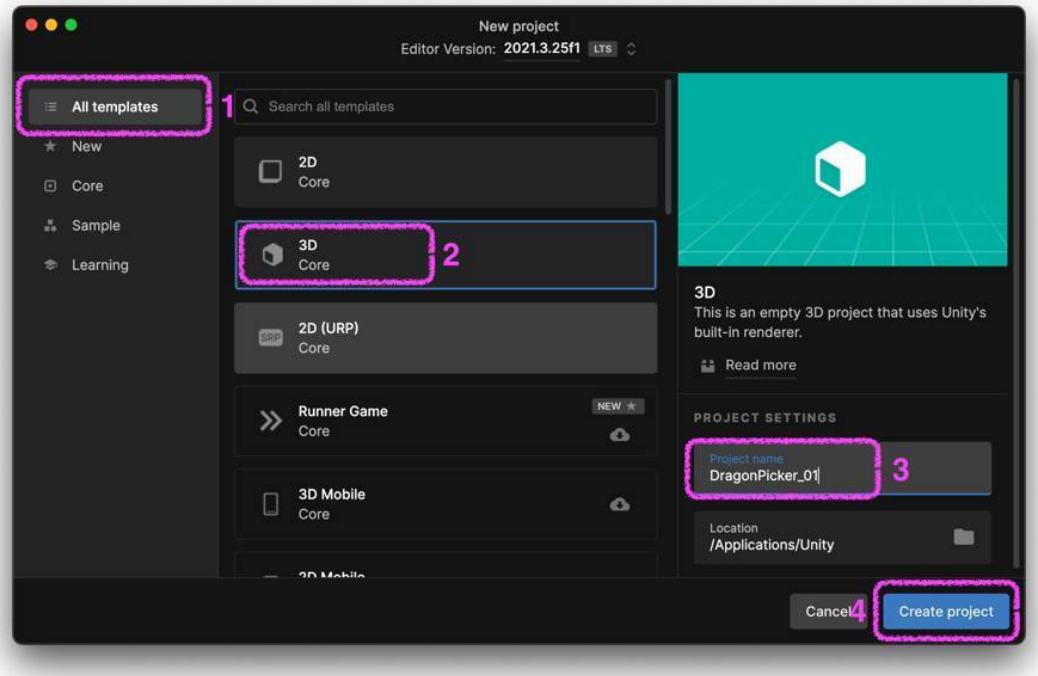
2.1 Создание проекта и первой сцены

Последовательность создания проекта и сцены внутри среды разработки не отличается для различных операционных систем.

1. Запустите Unity Hub, который был скачан и установлен в предыдущей главе;

2. Создайте новый проект, для этого нажмите New Project, как мы это делали ранее при создании первого приложения Hello World.

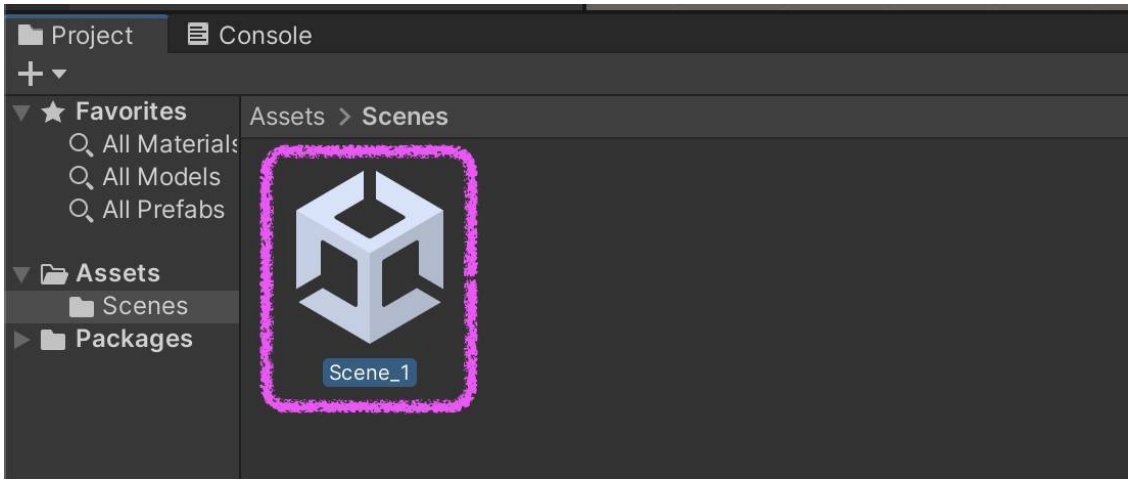
3. Выберите вид проекта All templates – 3D, в поле Project Name дайте имя проекту, например, DragonPicker, в поле Location укажите путь к папке с проектом. Напомним, что проект может находиться в любом месте на вашем компьютере.



4. Нажмите кнопку Create project. После этого откроется среда разработки Unity. Для первого запуска проекта может потребоваться значительное время, которое зависит от производительности вашего компьютера. Вид среды разработки после старта выглядит, как и ранее при создании нашего первого проекта Hello World. Также в первой главе была показана возможность изменения темы оформления на светлую, при желании вы можете использовать светлую тему оформления среды разработки.

5. В левой нижней части среды разработки внутри панели Project находится структура проекта. Внутри папки Assets хранятся исходные файлы проекта. В данный момент там существует только папка Scenes, в которой находится главная сцена с именем SampleScene.

6. Внутри сцен могут существовать отдельные объекты и персонажи, с которыми происходят различные действия во время игры. Давайте переименуем главную сцену. Это можно сделать также, как и в большинстве операционных систем при взаимодействии с папками. Кликните по SampleScene правой кнопкой мыши, выберите в выпадающем меню Rename и напишите новое имя Scene_1. После переименования среда разработки предложит перезагрузить сцену, соглашаемся, нажатием кнопки Reload. Теперь в проекте существует сцена с названием Scene_1:



7. Внутри сцены будут происходить основные действия с созданными игровыми объектами. Позже мы добавим и другие сцены, например сцену для стартового меню.

2.2 Импорт игровых объектов и персонажей

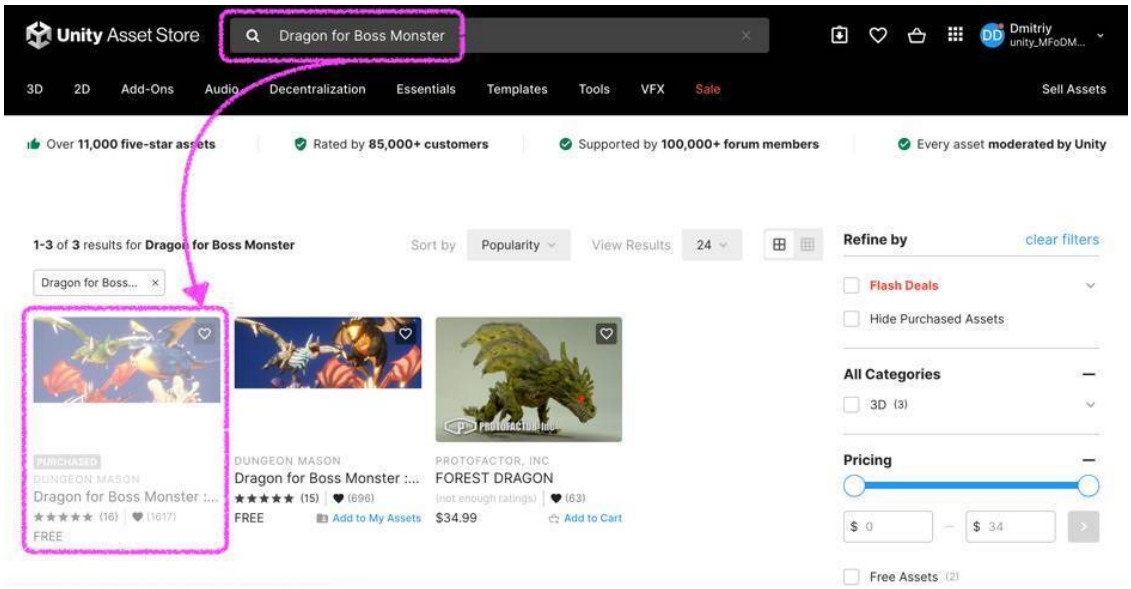
В качестве основного источника визуальных моделей и визуального оформления для своих игр на первом этапе вы можете использовать Unity Asset Store – это магазин, в котором можно приобрести (или скачать бесплатно) различные ресурсы для Unity. Для простоты понимания мы будем называть их ассетами. В Asset Store можно найти:

– 3D модели,
– звуковые эффекты/музыку,
– элементы пользовательского интерфейса,
– шейдеры/частицы,
– наборы спрайтов,
– готовые скрип-файлы и многое другое.

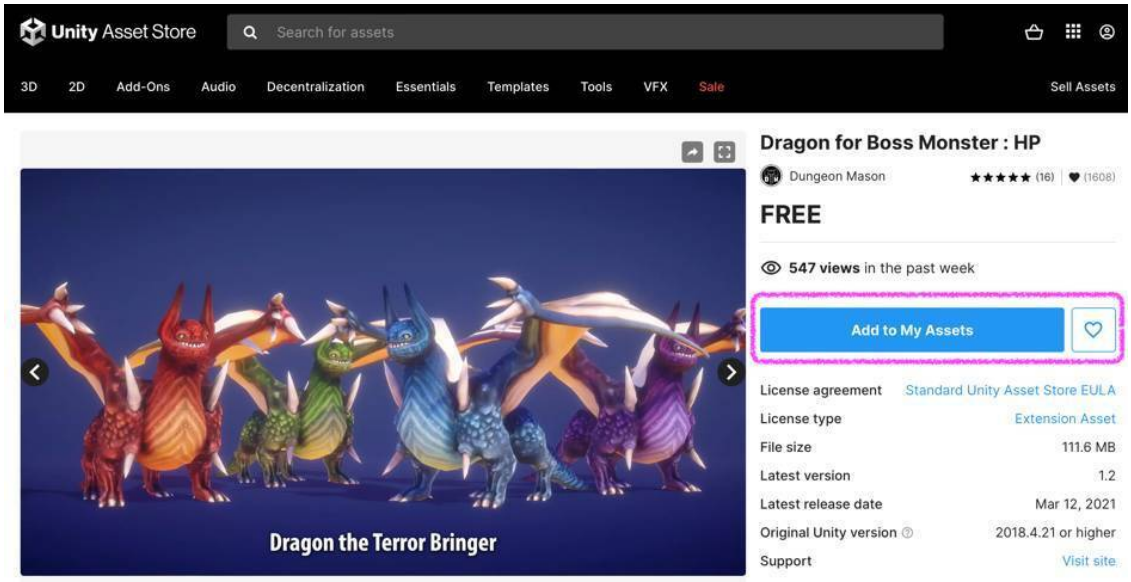
Основным источником персонажей в нашей игре станет набор моделей Dragon for Boss Monster, который распространяется бесплатно. Далее будет приведена подробная последовательность действия по добавлению этого ассета с игровыми персонажами – драконами в Unity.

1. Зайдите на сайт assetstore.unity.com и войдите под своей учетной записью (своим Unity ID). Для добавления ассета используется личный кабинет пользователя с тем же Unity ID, который указывался при регистрации и входе в Unity Hub.

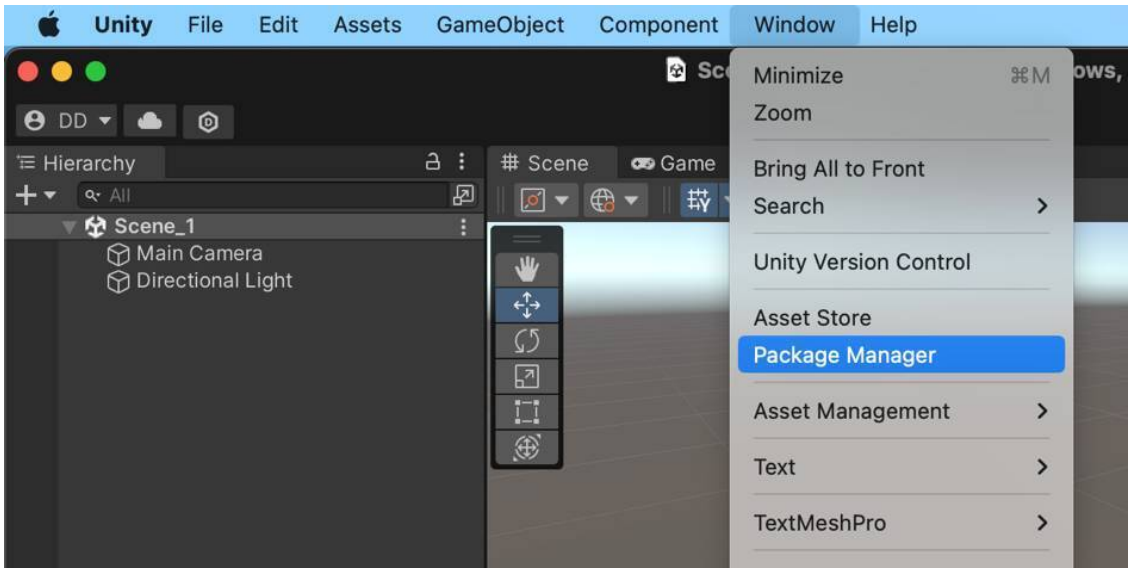
2. Используя строку поиска в верхней части сайта, найдите Dragon for Boss Monster:



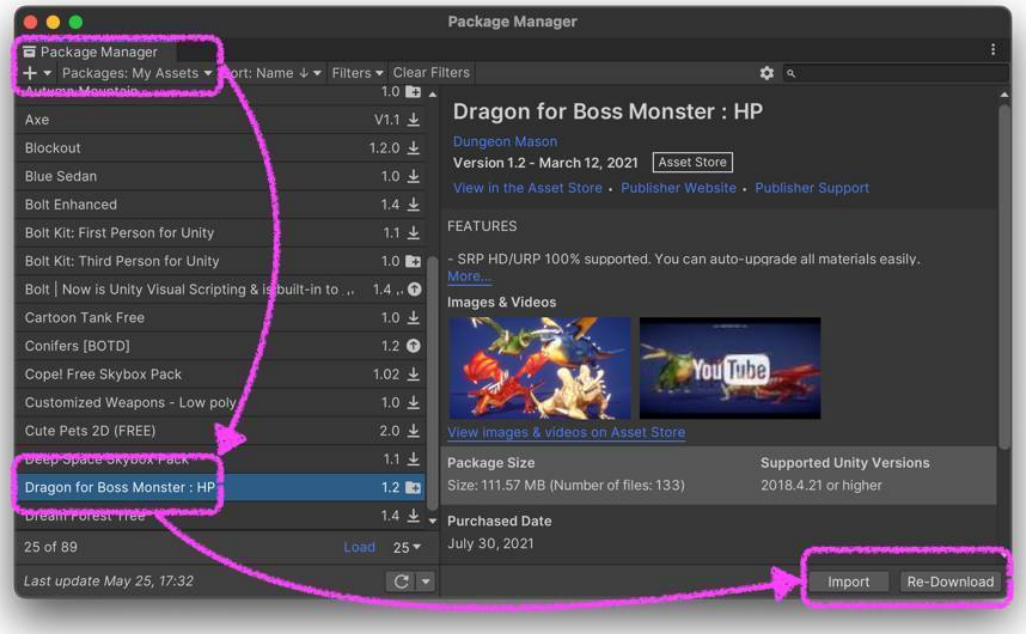
3. Откройте найденный ассет, кликнув по нему в браузере, и нажмите кнопку “Add to My Assets”. Следом на том же месте появится кнопка Open in Unity, которая позволяет открыть пакет в среде разработки (если этого не произошло автоматически, см. пункт ниже):



4. После добавления в коллекцию ассет будет привязан к вашему Unity ID. Пакет будет доступен в менеджере пакетов Unity. Вернитесь в проект Unity, на верхней панели инструментов выберите Window – Package Manager:



5. В менеджере пакетов доступно достаточно большое количество расширений для установки в Unity. Чтобы увидеть список с пакетами, добавленными вручную, выберите отображение пакетов, добавленных вами – Package Manager – Packages – My Assets. Среди установленных пакетов найдите только что добавленный Dragon for Boss Monster и скачайте его, нажав кнопку Download:



6. После завершения скачивания, рядом с кнопкой Download появляется кнопка Import, нажмите ее. Появится новое окно со списком импортируемых файлов, еще раз нажмите Import чтобы импортировать файлы из ассета в проект Unity.

7. На данном этапе мы нашли подходящий ассет-пак в магазине unity asset store, научились импортировать пакет в проект. На следующем этапе мы добавим скачанные модели на сцену.

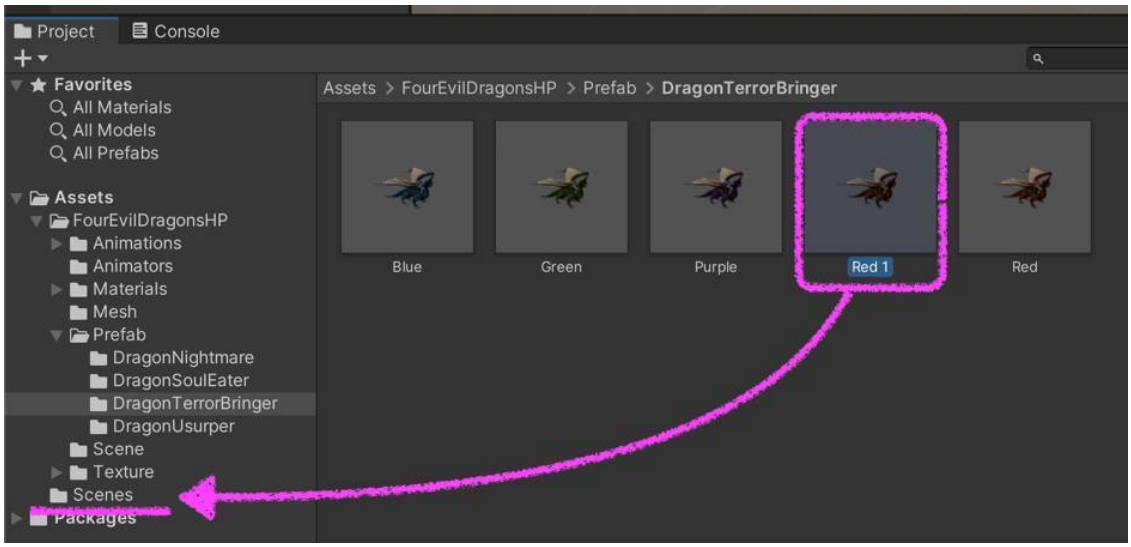
2.3 Добавление дракона с анимацией

Импортированный в проект ассет-пак содержит несколько видов драконов и большое количество анимаций. Все они разбиты и структурированы по папкам. Теперь наша задача заключается в том, чтобы выбрать некоторые 3D-модели, которые мы планируем использовать в проекте, и добавить на их на игровую сцену.

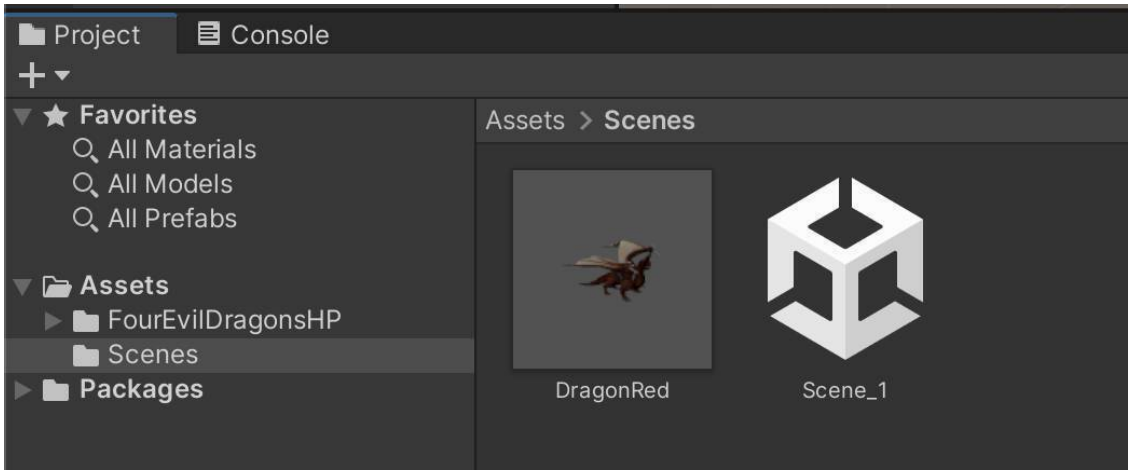
1. В окне Project откройте папку с префабами драконов. Путь к папке: Assets – FourEvilDragonsHP – Prefab – DragonTerrorBringer.

2. Создайте дубликат дракона Red, для этого выберите его (кликнув левой кнопкой мыши) и нажмите комбинацию клавиш Ctrl+D (или Command+D для MacOS). Автоматически будет создана префаб-копия с именем Red 1. Мы создаем копию модели, чтобы не использовать оригинальную модель из скачанного пакета. Возможно в будущем нам понадобятся оригинальные файлы из первоисточника, в этом случае правилами хорошего тона при разработке является «сохранность» исходных ресурсов.

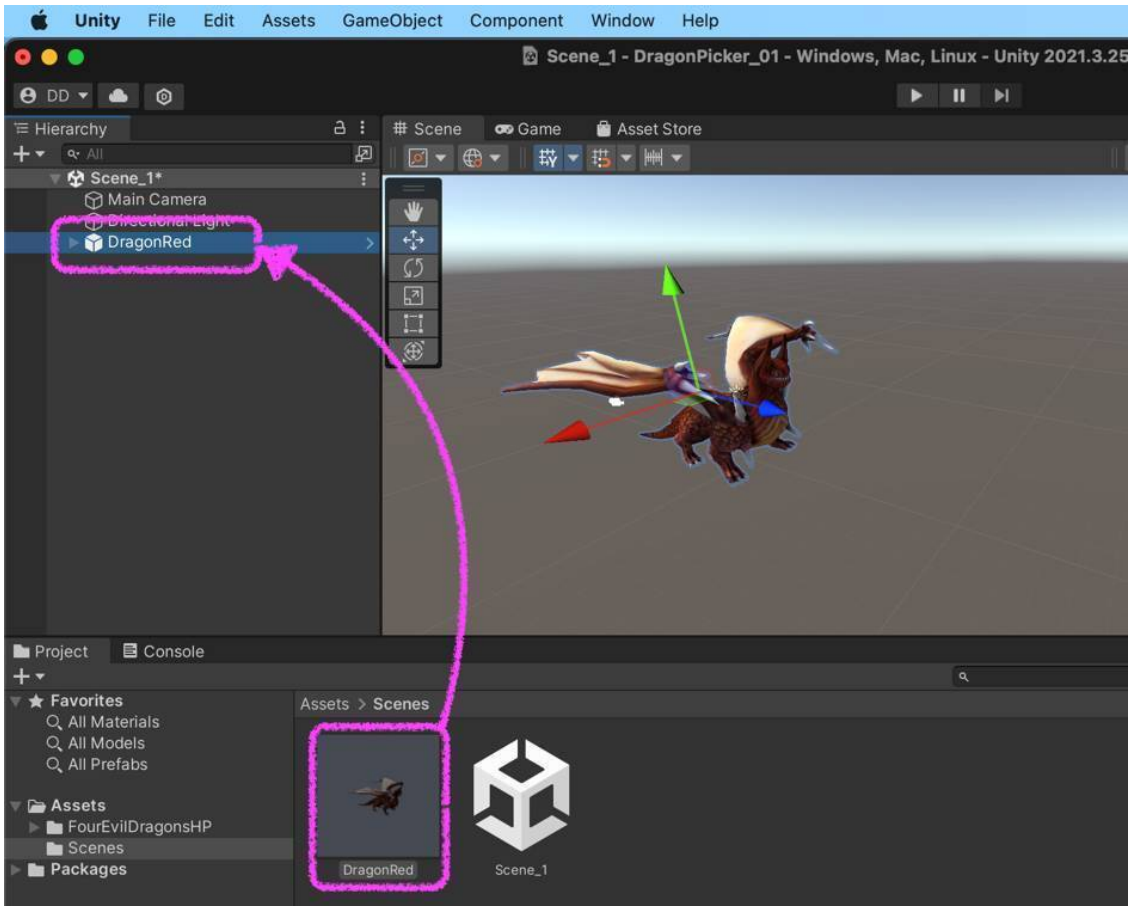
3. Перетащите префаб дракона с именем Red 1 в папку Scenes. Для этого наведите курсор мыши на дракона Red 1 и зажав левую кнопку мыши перетащите в папку Scenes:



4. Таким образом, в папке Scenes вашего проекта теперь должно находиться два файла: сцену и префаб с драконом Red 1. Переименуйте дракона Red 1 в DragonRed, для этого кликните левой кнопкой мыши по объекту, нажмите Rename и введите подходящее имя:

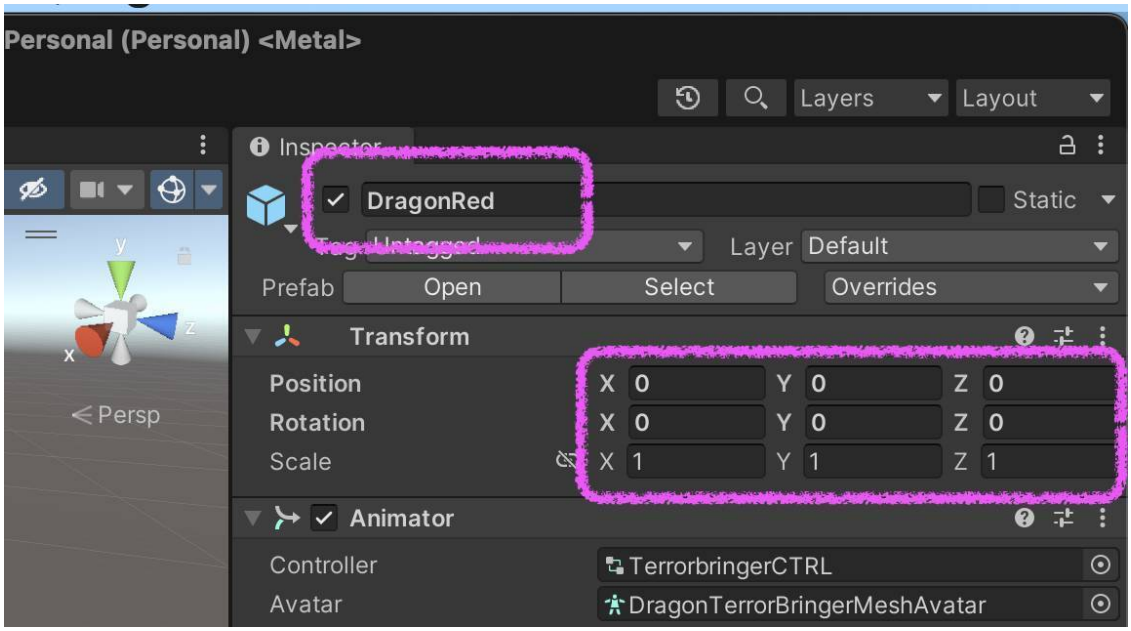


5. Теперь добавим персонажа DragonRed на игровую сцену. Для этого перетащите префаб из окна Project в окно Hierarchy:

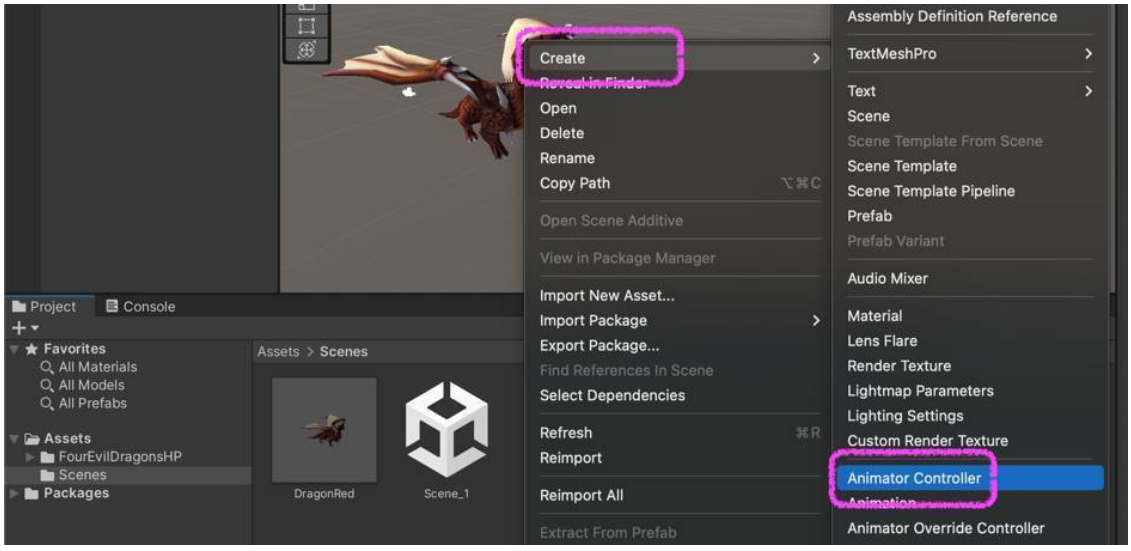


6. После этого персонаж DragonRed автоматически появится в окне Scene (в центральной части среды разработки Unity).

7. Как было указано ранее, координаты добавляемых на сцену объектов отображаются в окне Inspector (справа). Чтобы узнать координаты добавленного персонажа DragonRed, кликните по нему в окне Hierarchy, после этого в окне Inspector отобразятся его свойства. Нас интересуют параметры компонента Transform. Установите их значения: Rotation: 0, 0, 0; Position: 0, 0, 0; Scale: 1, 1, 1:

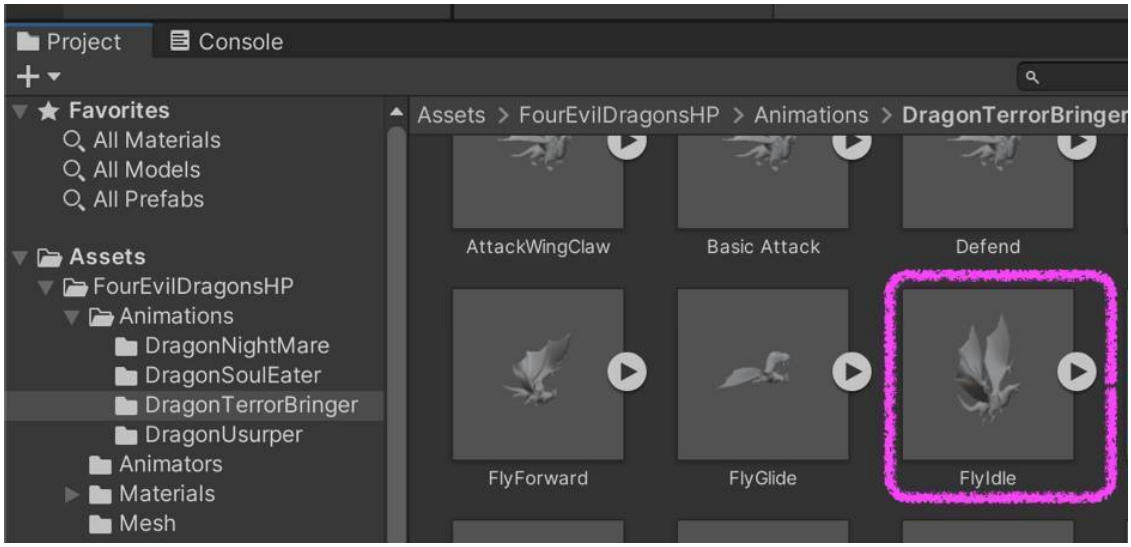


8. Теперь добавим дракону анимацию движения. Для этого нужно будет создать контроллер анимации. В окне Project, внутри папки Scenes (там, где находится префаб DragonRed) кликните правой кнопкой мыши и выберите из контекстного меню Create – Animator Controller. Дайте ему имя DragonRedController:



9. Контроллер нужен для того, чтобы строить “дерево анимации”, в котором описывается порядок, условия запуска и переключения анимации игровых объектов. Кликните дважды по созданному контроллеру DragonRedController чтобы открыть его.

10. Окно Animator выглядит пустым, так как контроллер был только что создан и в него не добавлено ни одной анимации. Давайте добавим подходящую анимацию парения дракона в воздухе, для этого найдите в скачанном Asset-паке анимацию FlyIdle. Она находится в папке Assets – FourEvilDragonHP – Animations – DragonTerrorBringer – FlyIdle:



11. Создайте дубликат анимации FlyIdle, как вы уже делали это ранее (комбинация клавиш Ctrl+D для Windows или Command+D для MacOS), переместите созданный дубликат анимации в папку Scenes, переименуйте файл анимации в FlyDragonRed.

12. Теперь перетащите FlyDragonRed в окно Animator. Автоматически будет создана связь Entry -> Fly Float, которая говорит контроллеру о том, что после запуска игры должна запускаться анимация полета:

Конец ознакомительного фрагмента.