

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное**

**учреждение Московской области**

**«Люберецкий техникум имени Героя Советского Союза,**

**лётчика-космонавта Ю.А. Гагарина»**

**КУРСОВАЯ РАБОТА**

**по МДК.01.01 «Разработка программных модулей »**

**по МДК.01.03 «Разработка мобильных приложений »**

специальность

09.02.07 «Информационные системы и программирование»

**Тема:** **Создание игрового процесса на Unity**

Группа 185

Выполнил/а/ студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ефимов Семен Сергеевич

(подпись) (ФИО полностью)

Руководитель Жирнова Юлия Витальевна

(подпись) (ФИО полностью)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Люберцы-2021 год

**Содержание**

[Введение 3](#_heading=h.gjdgxs)

[1.](#_heading=h.30j0zll) Теоретическая часть 4

[1.1.](#_heading=h.1fob9te) Актуальность (кому может понадобиться, зачем нужно) 4

[1.2.](#_heading=h.3znysh7) Теория (что необходимо знать, чтобы сделать приложение) 6

[1.3.](#_heading=h.2et92p0) Особенности IDE (в чем писали приложение) 9

[2.](#_heading=h.tyjcwt) Практическая часть 11

[2.1.](#_heading=h.3dy6vkm) Постановка задачи (Что нужно сделать) 11

[2.2.](#_heading=h.1t3h5sf) Структура приложения (и каких файлов состоит, зачем каждый файл) 13

[2.4.1.](#_heading=h.17dp8vu) [Ваше приложение](#_heading=h.1t3h5sf) (скриншоты результатов, что делает) [13](#_heading=h.17dp8vu)

[2.4.2.](#_heading=h.3rdcrjn)  14

[2.4.3.](#_heading=h.26in1rg)  15

[2.4.4.](#_heading=h.lnxbz9)  16

[2.4.5.](#_heading=h.35nkun2)  17

[Выводы 18](#_heading=h.1ksv4uv)

[Список литературы 19](#_heading=h.44sinio)

[Приложения 20](#_heading=h.2jxsxqh)

**Введение.**

Unity - Межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

На Unity написаны сотни игр, приложений и симуляций, Unity используется как крупными разработчиками (например, Blizzard), так и в создании инди-игр. Компьютерные игры на Unity охватывают множество платформ и жанров.

Цель данной курсовой работы: создать игровой процесс на Unity.

Для выполнения поставленной цели необходимо решить данные задачи:

1. Провести анализ основных инструментов разработки в индустрии игр.
2. Рассмотреть принципы работы различных игровых движков.
3. Выделить основные критерии для выбора игрового движка.
4. Разработать игру на выбранном игровом движке.

**1.1 Актуальность**

Разработчики игр – одни из самых востребованных IT-специалистов.

Индустрия игр по размерам сопоставима с киноиндустрией, а по темпам роста в последние годы значительно опережает ее. Игровой движок Unity является самым популярным инструментом для разработки игр всех типов. Сообщество разработчиков Unity чрезвычайно велико: активно занимаются разработкой на Unity3D более 1.5 млн человек. Unity позволяет создавать игры всех возможных типов:

* мобильные, десктопные и для консолей;
* 2D- и 3D-игры;
* классические, адаптированные для виртуальной и дополненной реальности.

Игровой движок Unity очень популярен среди разработчиков видеоигр, особенно среди начинающих, из-за простоты освоения и бесплатности. Несмотря на то, что движок ориентирован на трехмерные приложения, он поддерживает работу с двухмерной графикой. Однако не все компоненты редактора одинаково хорошо работают в 3D и 2D. В частности, встроенная система освещения не позволяет двухмерным объектам служить полноценным препятствием для света и отбрасывать тени на другие объекты.

Ко всем объектам прикреплен одинаковый материал с одинаковым шейдером. Если куб является полноценным препятствием для света, то четырехугольник отбрасывает тень только от тех источников, которые не лежат в одной плоскости с ним, а спрайт и вовсе игнорируется системой расчета теней. В трехмерном пространстве это обусловлено физическим расчетом освещаемых областей, так как четырехугольник, в отличие от куба, не имеет толщины, однако в двухмерном пространстве объемные фигуры представлены плоскими 2D объектами, поэтому необходимо обеспечить возможность взаимодействия со светом двухмерных объектов, идентичную взаимодействию трехмерных в 3D пространстве.

В последнее время Unity развивается стремительными темпами, привлекая все больше новых разработчиков. На базе этого движка были созданы такие популярные игры, как Ori and the Blind Forest (разработчик: Moon Studios), Cities: Skylines (разработчик: Colossal Order), Dungeon of the Endless (разработчик: Amplitude Studios), а также мобильная версия Hearthstone: Heroes of Warcraft (разработчик: Blizzard Entertainment). Кроме того, Unity используется для создания приложений, связанных с архитектурой, обучением, визуализацией данных, электронными книгами и многими другими отраслями. Так при съемках фильма Книга джунглей (2016, режиссёр: Джон Фавро) приложение на базе Unity Engine использовалось для визуализации в режиме реального времени перемещения по сцене и работы с освещением.

Примеры инструментов для работы с 2D освещением существуют, но, как правило, они либо определяют форму объекта через специальные компоненты Unity - коллайдеры, которые служат для обработки столкновений, и не всегда их применение возможно у данного объекта, либо, создавая области тени, отбрасываемой каким-либо объектом, не освещают сам объект.

* 1. **Теория**

Написание сценариев (скриптов) осуществляется на наиболее популярных языках программирования — C# и JavaScript.  
Таким образом, Unity3d является актуальной платформой, с помощью которой вы можете создавать свои собственные приложения и экспортировать их на различные устройства, будь то мобильный телефон или приставка Nintendo Wii.  
Для того чтобы создать свою игру, вам, как минимум, нужно владеть одним из доступных (на Unity) языков программирования: C#, JavaScript.

 Unity используется производительный язык программирования C#. Благодаря C# – кроссплатформенной реализации .NET, – Unity-проекты могут собираться и работать одинаково стабильно на разных операционных системах.

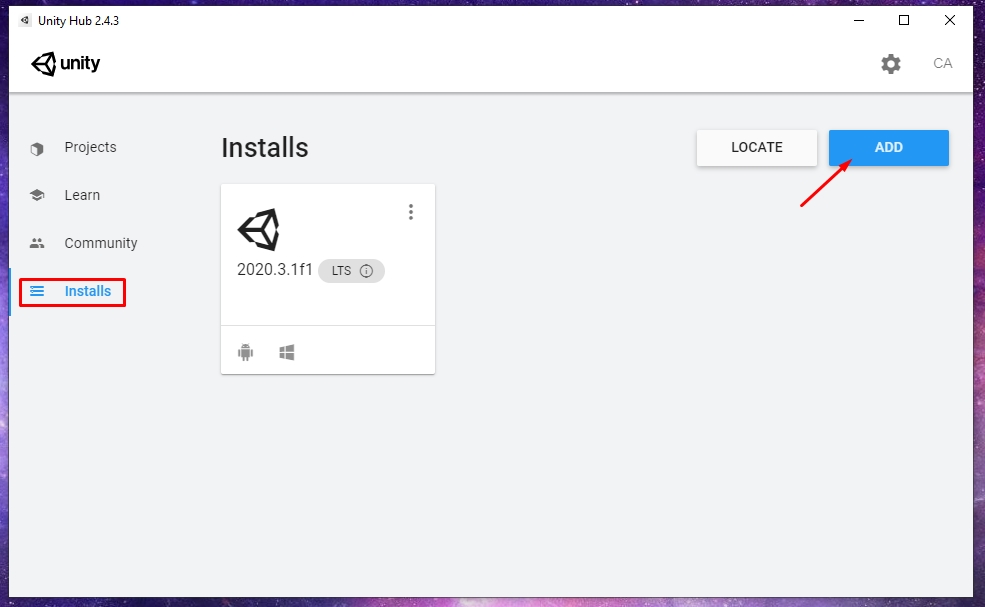
Общее для всех программистов требование – знание [алгоритмов и структур данных](https://proglib.io/p/data-structure-algorithms/). Эти знания не устареют, какой бы язык программирования вы не использовали и для какой бы операционной системы вы разрабатывали.

Кроме языка программирования и алгоритмов, обязательны для изучения [шаблоны проектирования](https://proglib.io/p/structural-patterns) – используемые программистами продуманные архитектурные решения. Нужно хорошо понимать, как и в какой ситуации использовать тот или иной шаблон.

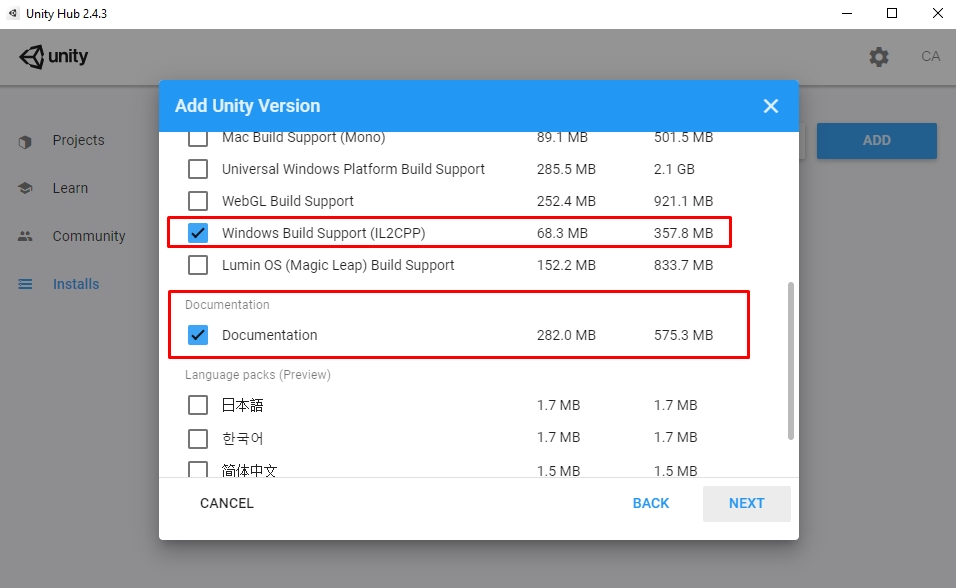
Большинство обучающих и справочных материалов написаны на английском языке, а потому мы советуем овладеть языком, как минимум на уровне Intermediate. Знание английского языка – не специфическое для Unity-программистов, а стандартное для IT требование.

1. Чтобы установить Unity нужно перейти на официальный сайт Unity, выбрать бесплатную лицензию.

* Зарегистрировать аккаунт.
* Скачать и установить Unity Hub. Запустить его.
* Перейти в раздел Installs, нажать ADD.



* Выбрать нужную версию, модули, языки и запустить установку.



1. Чтобы превратить проект в исполняемый файл нужно:

* выбираем пункт меню File — Build Setting
* нажимаем кнопку Build

## Редактор Unity3D

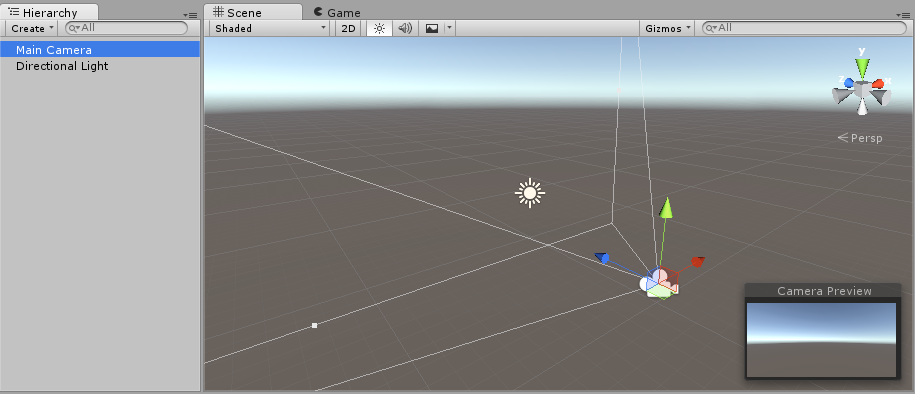
Большую часть времени Unity-разработчики проводят в редакторе, так что его нужно изучить досконально. Кроме того, программные библиотеки Unity требуют дополнительного изучения – Unity-проект существенно отличается от обычного C#-приложения. Чтобы стать профессиональным Unity-разработчиком, нужно изучить следующее:

* базовые манипуляции со сценой, объектами, assets и prefabs;
* векторы, проекции и математика для 3D графики;
* создание пользовательского интерфейса в игре;
* работа с материалами;
* модули физики для 2D и 3D;
* система анимации Mecanim;
* системы частиц;
* работа с аудио;
* свет и постобработка;
* работа с разными графическими пайплайнами;
* работа с искусственным интеллектом.

Для многих, это может казаться слишком сложным и непонятным. Но для разработки приложений достаточно знать основные понятия в самом игровом движке Unity.

1. Сцены
2. Игровые объекты
3. Источники света
4. Камеры
5. **Сцены.**

Сцены основаны на ваших играх. Они имеют право на создание главного меню, ограниченных уровней и для других целей. Каждый раз можно считать файл сцены результативным игровым улучшением. В каждом конкретном случае можно добавить ссылку на окружение, заграждение, декорации, по мельчайшим деталям дизайна и самой игры.



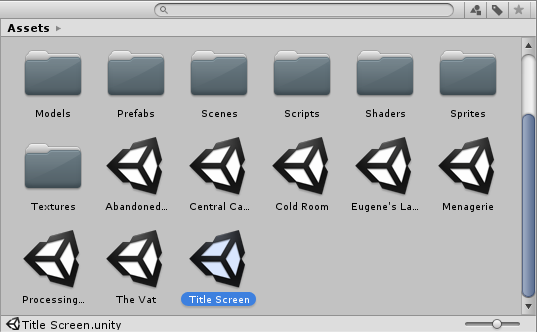
Новая пустая сцена, с дефолтными 3D объектами - камерой и направленным светом.

Когда вы создаете новый проект Unity, в представлении сцены будет отображаться новая сцена. Это безымянная и несохраненная сцена. Сцена будет пустой, за исключением объектов по умолчанию — либо ортогональной камеры, либо перспективной камеры и направленного света, в зависимости от того, в каком режиме вы начали проект: 2D или 3D.

Создание сцены

Чтобы сохранить сцену, над которой вы сейчас работаете, выберите в меню «Файл» - «Сохранить сцену» или нажмите Ctrl + S.

Сцены сохраняются как активы в папке Assets вашего проекта. Поэтому они появляются в окне проекта, как и любой другой актив.



Некоторые активы сцены, сохраненные и видимые в виде проекта

Создание сцены

Чтобы открыть сцену, чтобы начать или продолжить работу с ней, дважды щелкните актив сцены в окне проекта.

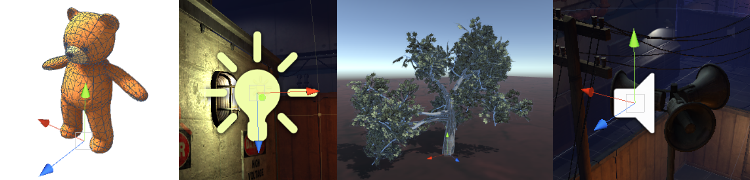
Если ваша текущая сцена содержит несохраненные изменения, вам будет предложено сохранить или отменить изменения.

1. **Игровые объекты**

Игровые объекты (GameObjects) — это самые важные объекты в Unity. Очень важно понимать, что такое GameObject и как его использовать.

## Что такое GameObject?

Каждый объект в игре — это GameObject. Однако, GameObject’ы ничего не делают сами по себе. Они требуют специальной настройки, прежде чем стать персонажами, предметами окружения или специальными эффектами. Если каждый объект — это GameObject, то как мы сможем различить интерактивные объекты и статичную комнату? Что отличает GameObject’ы друг от друга?

( Четыре разных игровых объекта, анимированный персонаж, свет, дерево и источник звука.)

GameObjects являются контейнерами. Они могут содержать различные части, необходимые для создания персонажа, света, дерева, звука или чего-то еще, что вы хотели бы построить. Итак, чтобы по-настоящему понять GameObjects, вы должны понять эти части, которые называются **компонентами.**

GameObject’ы являются контейнерами. Пустая коробка, которая может содержать внутри различные элементы, такие как остров с запечёнными тенями или физически корректный автомобиль. Чтобы действительно понять GameObject’ы, нужно понять их составляющие, который называются компонентами (Components). В зависимости от того, какой вы хотите создать объект, вы будете добавлять различные комбинации компонентов к GameObject’у. Представляйте GameObject как пустой кулинарный горшок, а компоненты как различные ингредиенты, которые составят ваш рецепт игрового процесса. Вы можете также создать ваши собственные компоненты, используя скрипты.

**С) Источники света.**

Источники света являются неотъемлемой частью каждой сцены. В то время как меши и текстуры определяют форму и внешний вид сцены, источники света определяют цвет и атмосферу вашего 3D окружения. Вы, вероятно, будете работать более чем с одним источником света в каждой сцене. Организация их одновременной работы требует небольшой практики, но результат может быть потрясающим.

Компонент Light имеет много различных свойств, которые можно менять в инспекторе (Inspector):



Просто меняя свойство Color (цвет) источника света, вы можете придать сцене совсем другое настроение.

К примеру:

Яркие, солнечные источники света

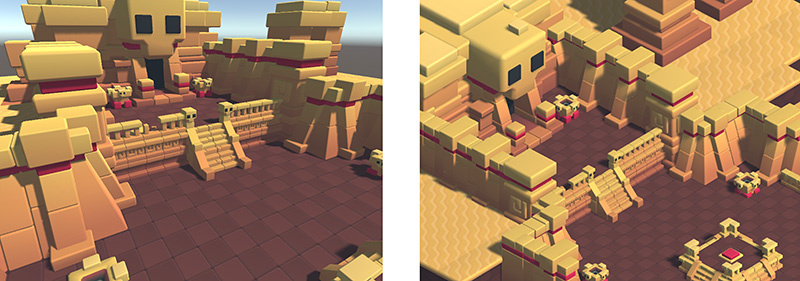
.

Жуткие ночные источники света. 

**D) Камеры.**

Сцена Unity создается путем размещения и перемещения объектов в трехмерном пространстве. Поскольку экран зрителя двумерный, должен быть способ захватить изображение и «сгладить» его для отображения. Это достигается с помощью Cameras .

Камера — это объект, определяющий вид в пространстве сцены. Положение объекта определяет точку обзора, а оси вперед (Z) и вверх (Y) объекта определяют направление взгляда и верхнюю часть экрана соответственно. Компонент камеры также определяет размер и форму области, попадающей в поле зрения. При настройке этих параметров камера может отображать на экране то, что она «видит» в данный момент. Когда объект камеры перемещается и вращается, отображаемый вид также будет перемещаться и вращаться соответственно.

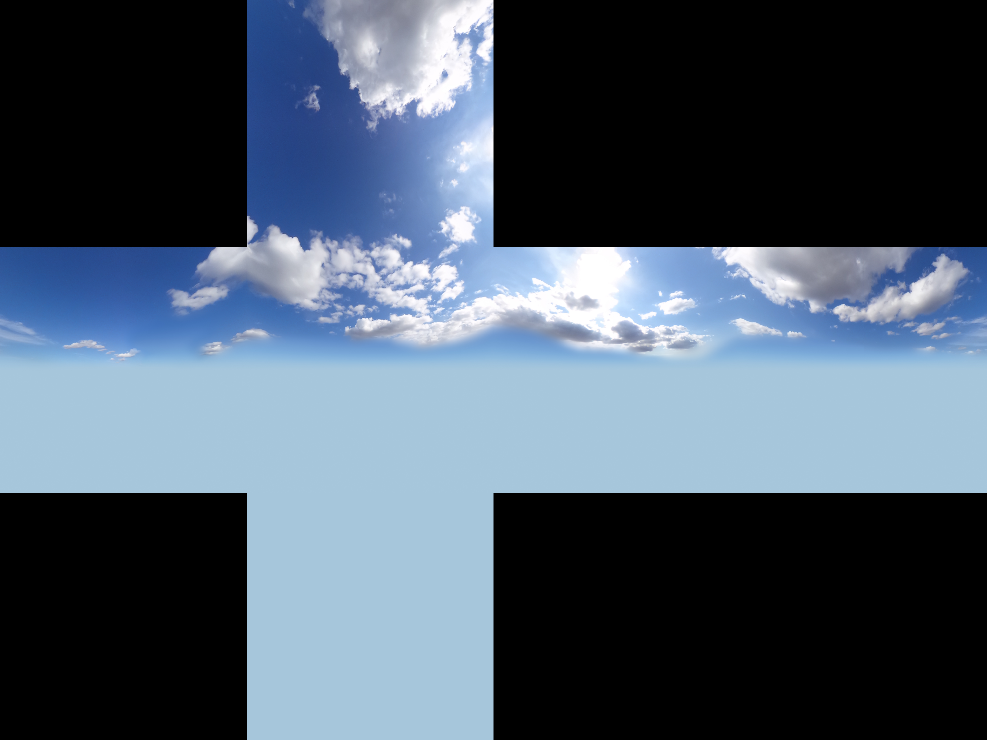


Та же сцена, показанная в режиме перспективы (слева) и орфографическом режиме (справа).

Камера в реальном мире или даже человеческий глаз видят мир таким образом, что объекты кажутся меньше, чем дальше они находятся от точки обзора. Этот известный эффект перспективы широко используется в искусстве и компьютерной графике и важен для создания реалистичной сцены.

Камера не может оставить фон совершенно неопределенным, поэтому она должна чем-то заполнить пустое пространство. Самый простой вариант — очистить фон до однородного цвета перед рендерингом сцены поверх него. Вы можете установить этот цвет с помощью свойства фона камеры либо из инспектора, либо из скрипта. Более сложный подход, который хорошо работает с наружными сценами, заключается в использовании Skybox .

Как следует из названия, скайбокс ведет себя как «коробка», заполненная изображениями неба. Камера эффективно размещена в центре этой коробки и может видеть небо со всех сторон. При вращении камера видит другую область неба, но она никогда не смещается от центра (поэтому камера не может приблизиться к небу). Скайбокс рендерится за всеми объектами сцены, поэтому он представляет собой вид на бесконечное расстояние. Чаще всего используется для представления неба в стандартной сцене на открытом воздухе, но на самом деле коробка полностью окружает камеру, даже снизу. Это означает, что вы можете использовать скайбокс для представления частей сцены (например, холмистой равнины, простирающейся за горизонт) или кругового обзора сцены в космосе или под водой.



**1.3 Особенности IDE.**

Unity – больше, чем движок, это среда для разработки компьютерных игр, в которой объединены различные программные средства, используемые при создании ПО. При этом, благодаря удобству использования, Unity делает создание игр максимально простым и комфортным и позволяет разработчикам игр охватить как можно большее количество игровых платформ и операционных систем.

Unity3D дает возможность разрабатывать игры, не требуя для этого каких-то особых знаний. Здесь используется компонентно-ориентированный подход, в рамках которого разработчик создает объекты (например, главного героя) и к ним добавляет различные компоненты (например, визуальное отображение персонажа и способы управления им). Благодаря удобному и функциональному графическому редактору движок позволяет рисовать карты и расставлять объекты в реальном времени и сразу же тестировать получившийся результат.

Так же преимущество движка в наличие огромной библиотеки ассетов и плагинов, с помощью которых можно значительно ускорить процесс разработки игры. Их можно импортировать и экспортировать, добавлять в игру целые заготовки – уровни, врагов, поведения ИИ и так далее. Многие ассеты доступны бесплатно, другие предлагаются за небольшую сумму, и при желании можно создавать собственный контент, публиковать его в Unity Asset Store и получать от этого прибыль.

Unity 3D поддерживает огромного количества платформ, технологий, API. Созданные на движке игры можно легко портировать между ОС Windows, Linux, OS X, Android, iOS, на консоли семейств PlayStation, Xbox, Nintendo, на VR- и AR-устройства. Unity поддерживает DirectX и OpenGL.

Физика твердых тел, ragdoll и тканей, система Level of Detail, коллизии между объектами, сложные анимации – все это можно реализовать силами движка.

При всех своих достоинствах, движок имеет и свои недостатки. Так, если команда захочет разработать что-нибудь сложнее простого кликера или платформера, то ей придется искать хорошего программиста на C#, который напишет скрипты и компоненты, внедрит их в игру и заставит работать.

Из этого вытекает другая проблема движка Unity – медлительность. Создание масштабных, сложных сцен с множеством компонентов может негативно повлиять на производительность игры, в результате чего разработчикам придется потратить дополнительное время и ресурсы на оптимизацию, а возможно – и удаление некоторых элементов из проекта.

Кроме того, приложения, созданные на Unity, много весят: даже самая простая пиксельная игра может занимать несколько сотен мегабайт на ПК. Да, для жесткого диска компьютеров это небольшой объем, но, если проект разрабатывается и для мобильных платформ, следует задуматься об оптимизации его размера.

|  |  |  |  |
| --- | --- | --- | --- |
| Особенности | Unity | Construct 2 | Game Editor |
| Простой | + | + | + |
| Бесплатный | + | + | + |
| 2D/3D | + | - | - |
| Добавление медиафайлов | + | + | + |
| Поддержка мультиплеера | + | - | - |
| Совместный режим работы | + | - | + |
| Настройка поведения объектов | + | + | + |
| Предпросмотр игрового процесса | + | + | + |

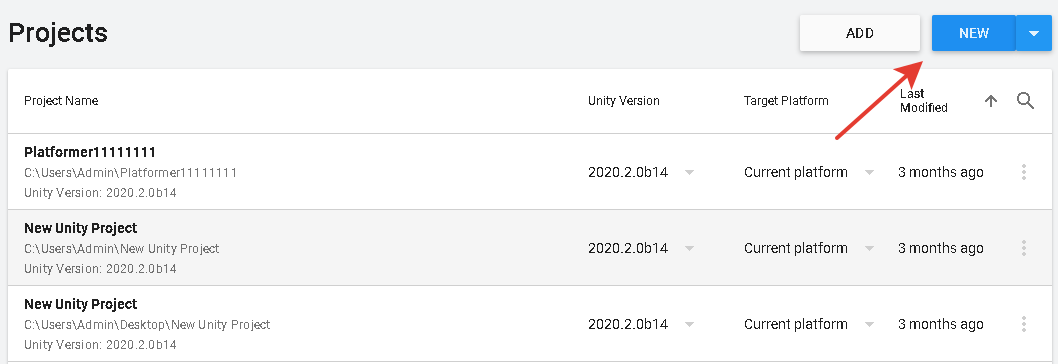
При всех своих достоинствах, движок имеет и свои недостатки. Так, если команда захочет разработать что-нибудь сложнее простого кликера или платформера, то ей придется искать хорошего программиста на C#, который напишет скрипты и компоненты, внедрит их в игру и заставит работать.

Главная проблема движка Unity – медлительность. Создание масштабных, сложных сцен с множеством компонентов может негативно повлиять на производительность игры, в результате чего разработчикам придется потратить дополнительное время и ресурсы на оптимизацию, а возможно – и удаление некоторых элементов из проекта.

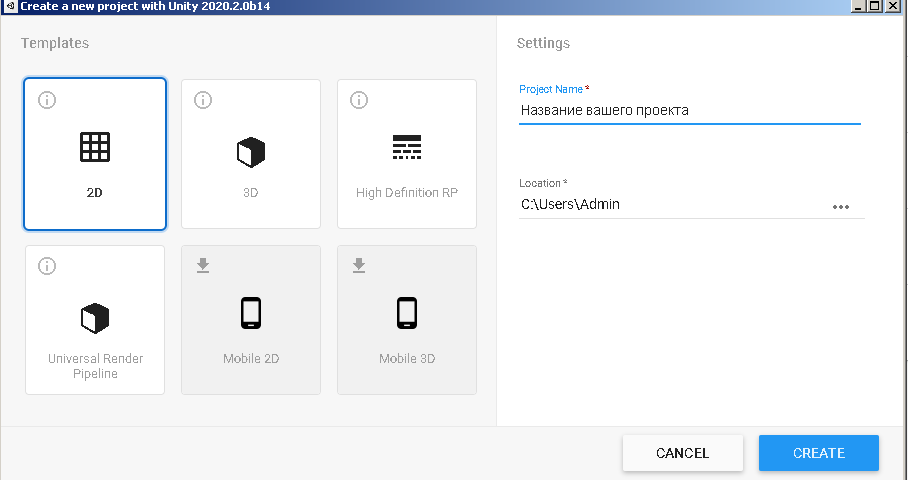
**Практическая часть.**

Будет продемонстрировано создание игрового процесса на игровом движке Unity, на примере платформера.

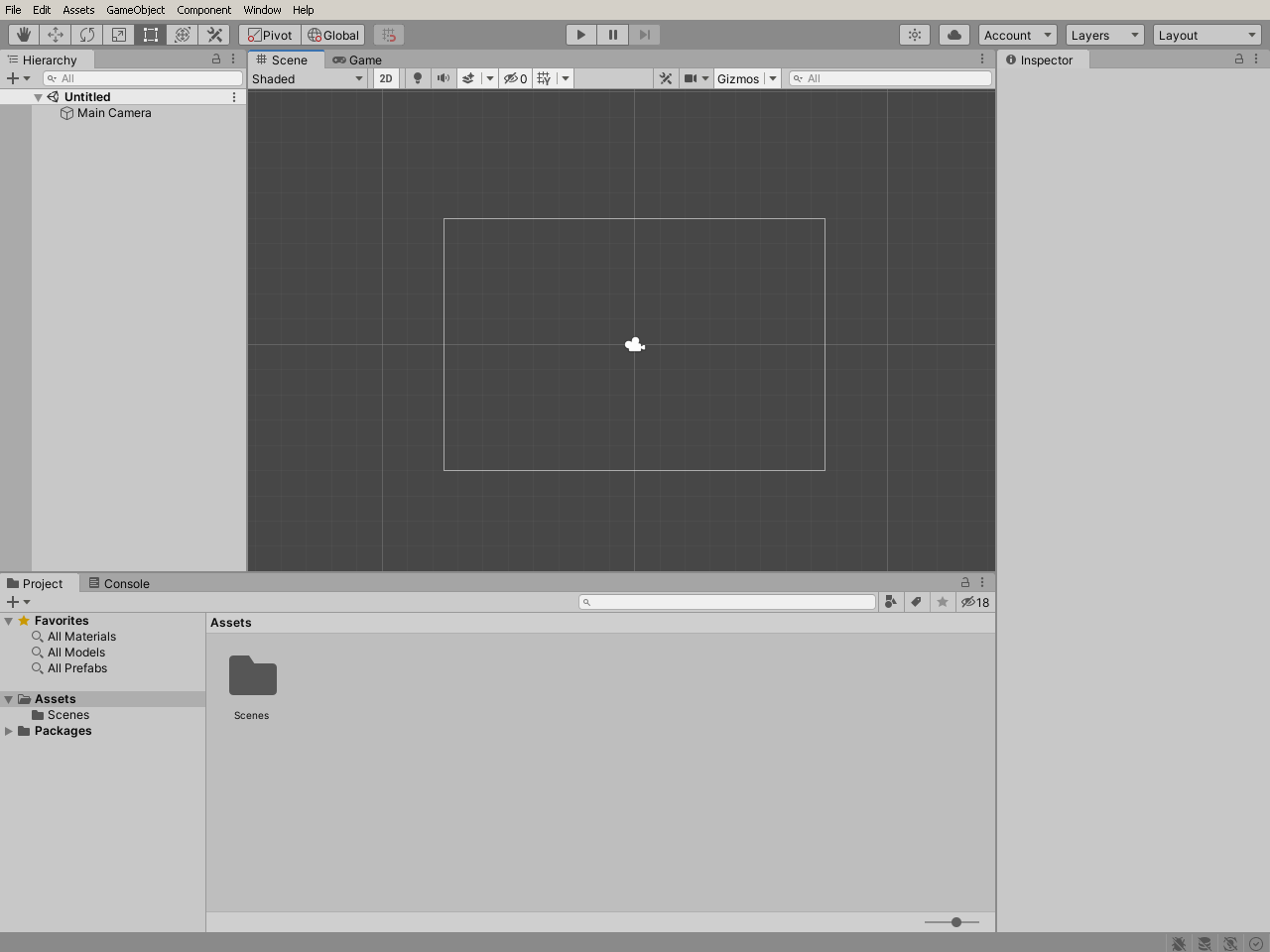
Откройте Unity Hub и начать новый проект. Назовите проект как хотите, но не забудьте выбрать 2D для настройки проекта.



Обязательно используйте 2D-шаблон.



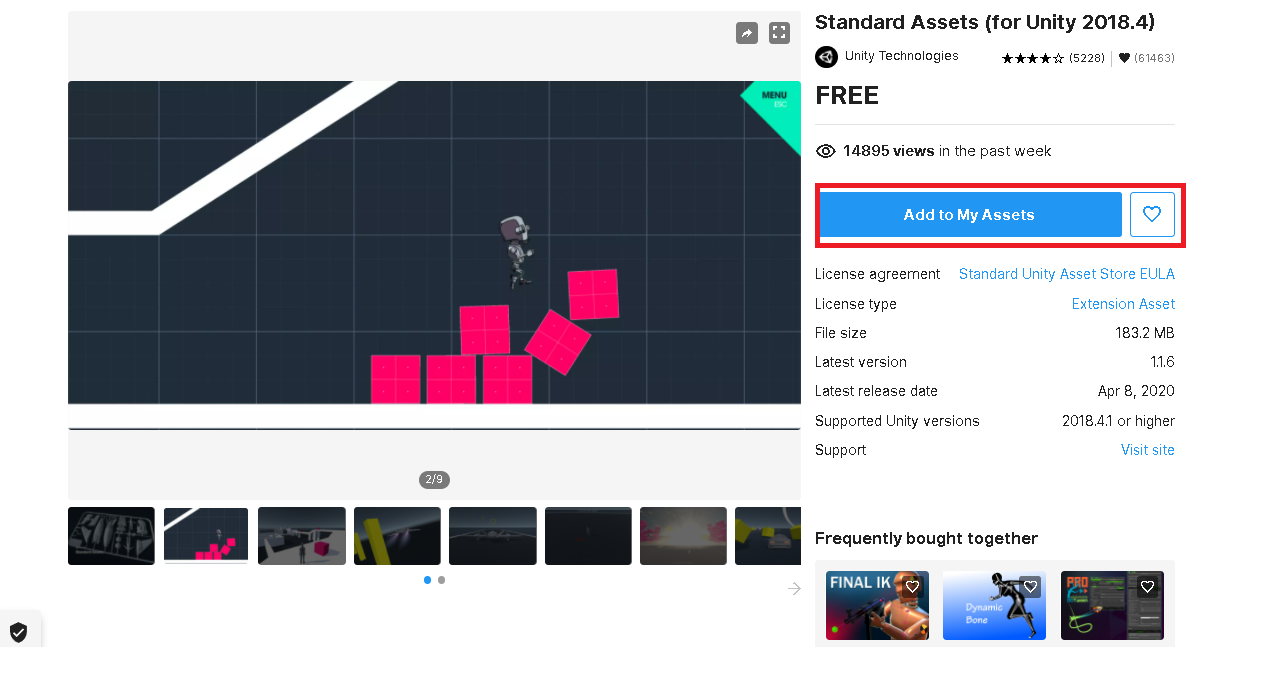
Теперь вы должны посмотреть на редактор Unity. Чтобы перемещаться по 2D-пространству в окне редактора, используйте среднюю кнопку мыши для панорамирования и колесо прокрутки для увеличения. Камера в центре поля показывает то, что увидит игрок во время игры.



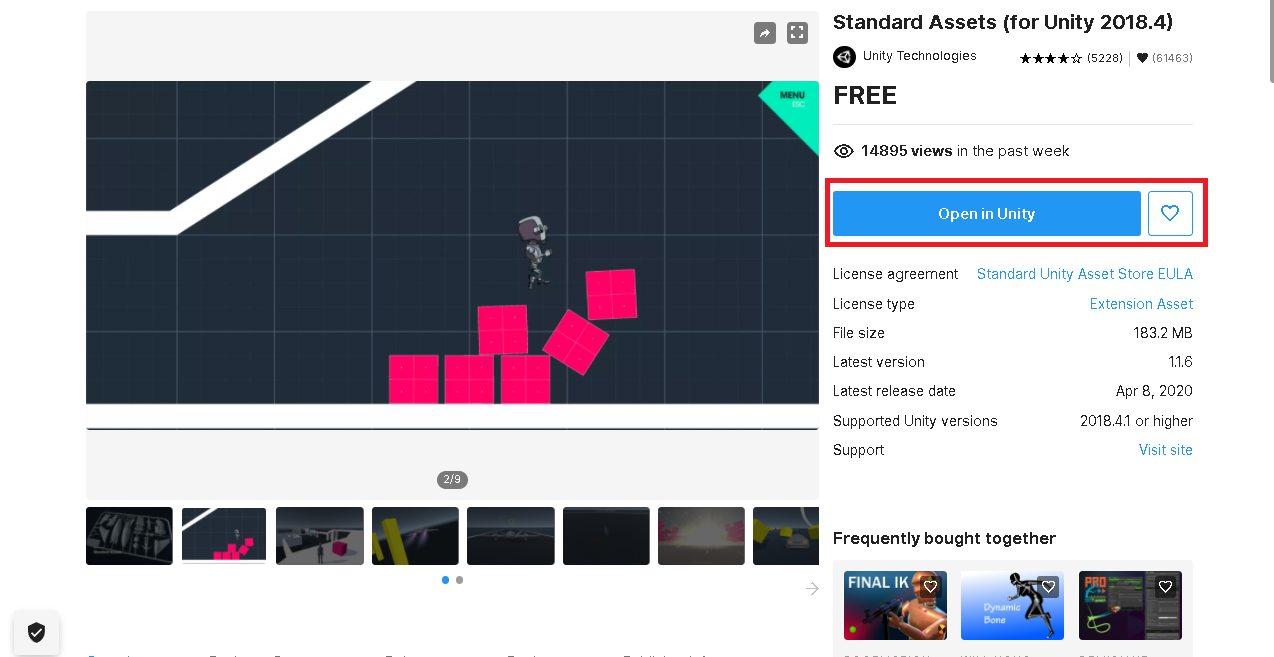
В Unity есть невероятная функция под названием Unity Asset Store. Asset Store полон готовых фрагментов игры, таких как модели, текстуры, утилиты или программы AI. Они доступны для продажи, а некоторые даже предлагаются бесплатно. Из-за этого легко создать игру, не создавая слишком много ресурсов самостоятельно.

Переходим на сайт <https://assetstore.unity.com/> и выбираем нужные нам файлы для разработки. В моём случае это будет «Standard Assets».

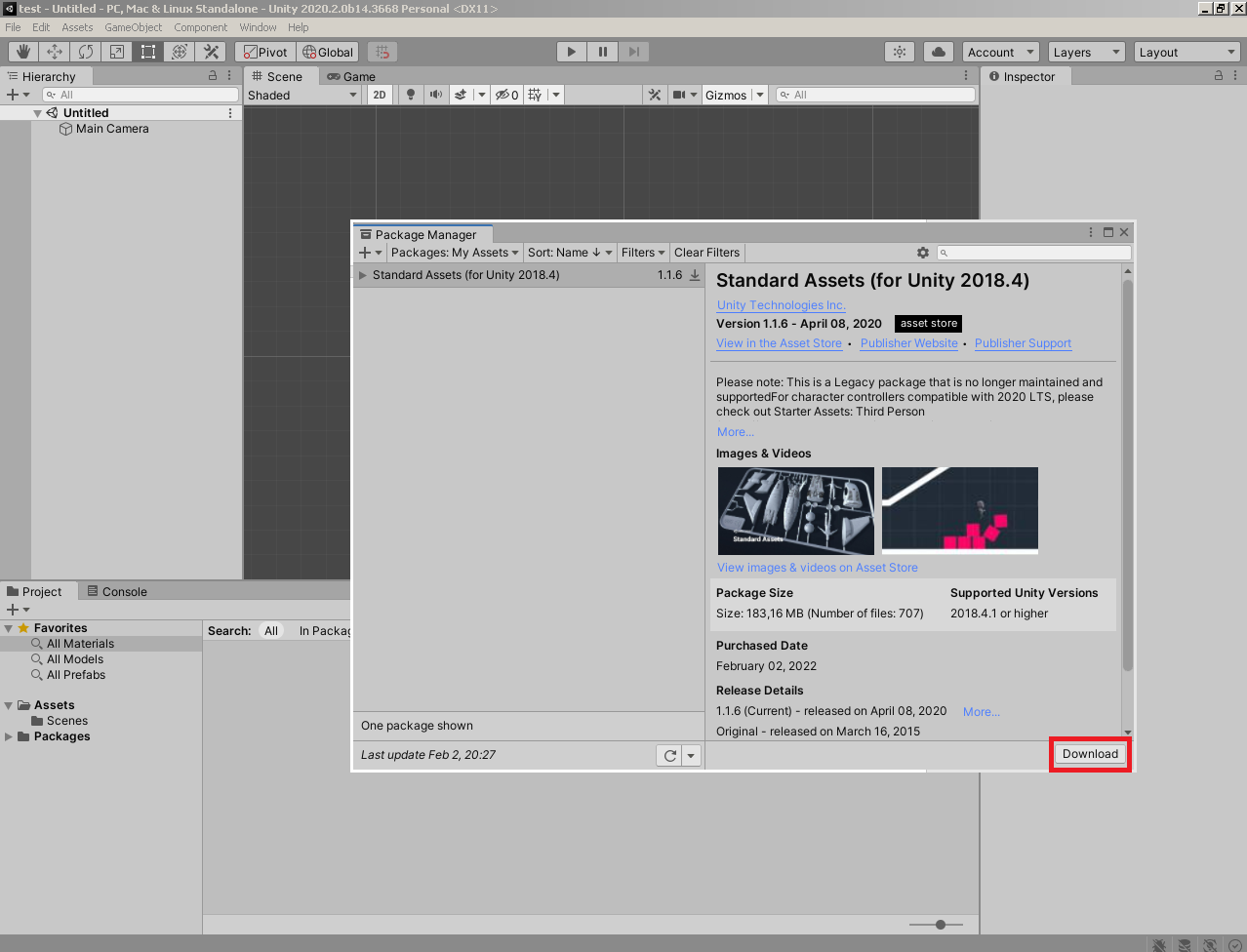
Нажимаем кнопку «Add to My Assets»



Потом нажимаем «Open in Unity».



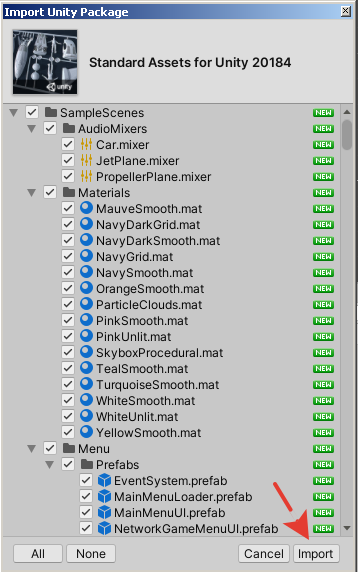
У нас откроется меню скаченного файла.. Нажимаем на Download, чтобы скачать данные файлы.

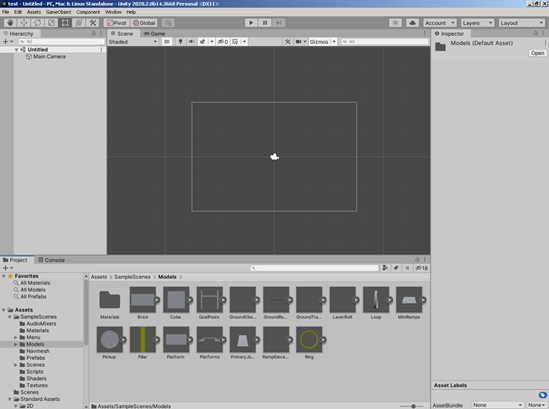


Потом кнопку «Import».



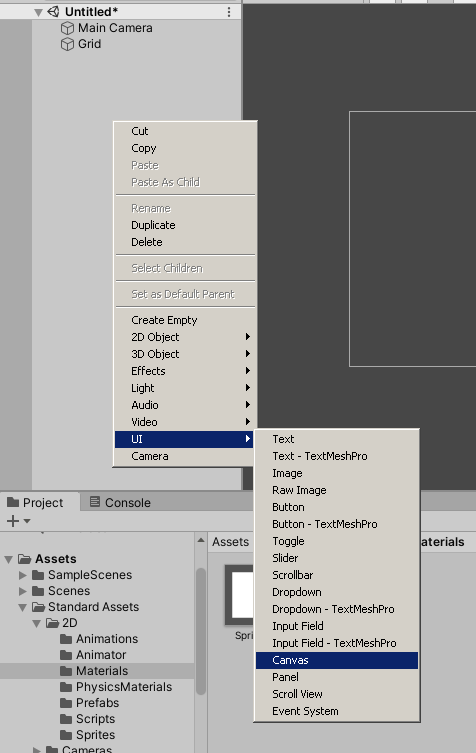
И снова кнопку «Import».



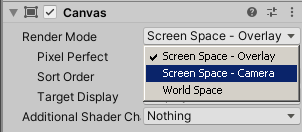
Теперь у нас есть много разных инструментов для создании игры. 

Можем начать разработку игры. Для начала сделаем фон самой игры.

Открываем меню UI/Canvas , переименовываем на Background.

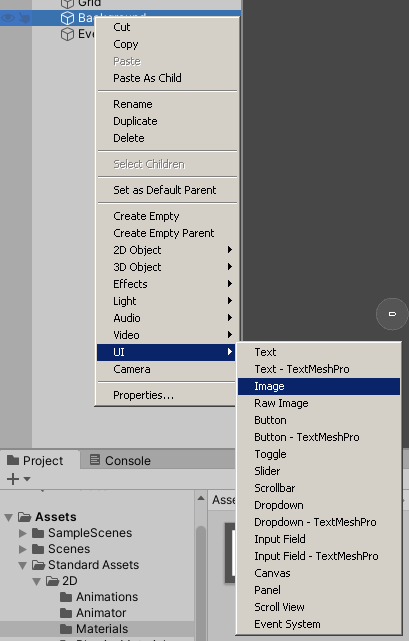


В настройках Render Mode меняем на Screen Space – Camera.

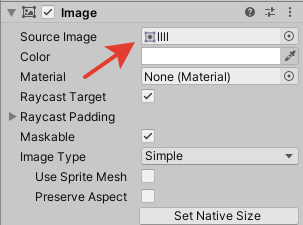


Перетаскиваем Main Camera с меню слева в Render Camera

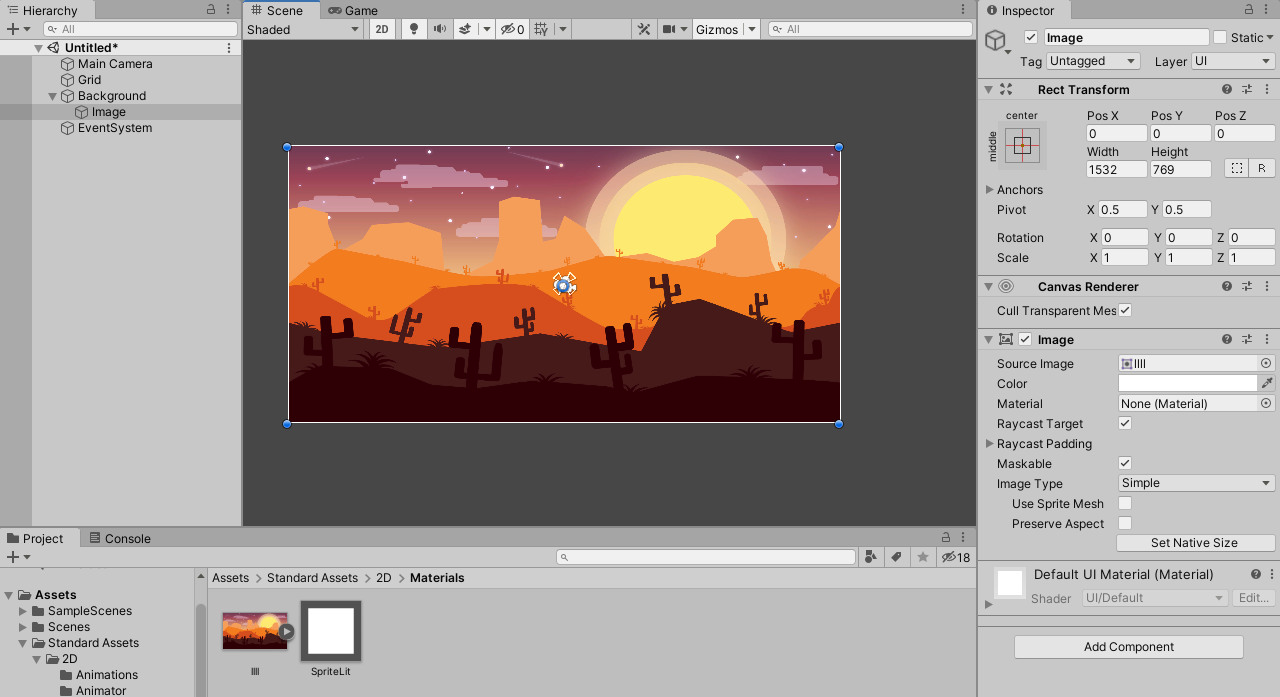


Дальше в BackGround нажимаем UI – Image 

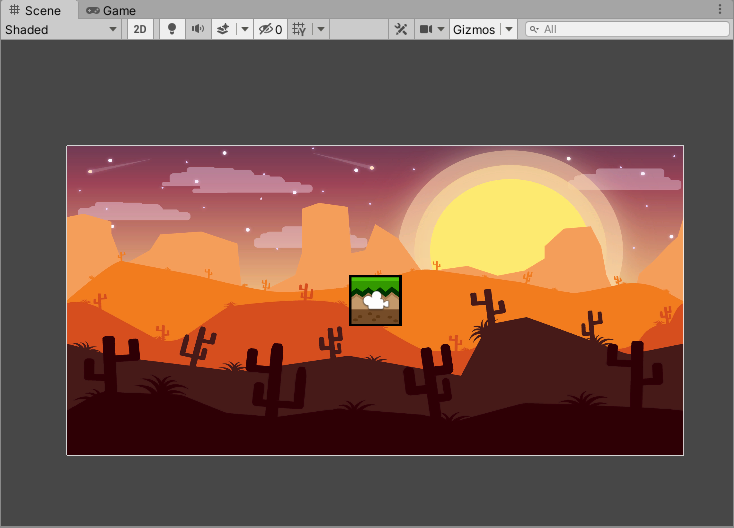
Дальше добавляем любую картинку в «Sourse Image» , предварительно добавив её в Assent – Material



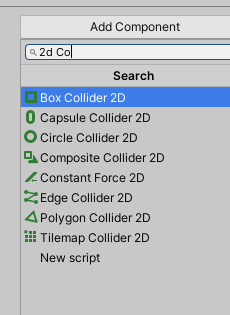
Должно получиться так.



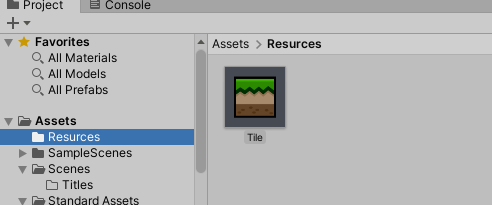
Дальше кликаем ПКМ по меню слева и создаём пустой обьект и в нем создаём квадратный Sprite.В Новом Sprite меняем изображение на картинку , также добавив её в Assent – Material.



В самом кубике в меню справа нажимаем на Add Component и добавляем компонент Box Collider 2D. Это нужно, чтобы кубик считался обьектом, до которого можно докаснуться, а не просто картинкой.

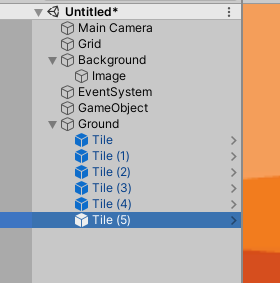


Дальше в Assets создаём папку Resurces и добаляем туда наш кубик.

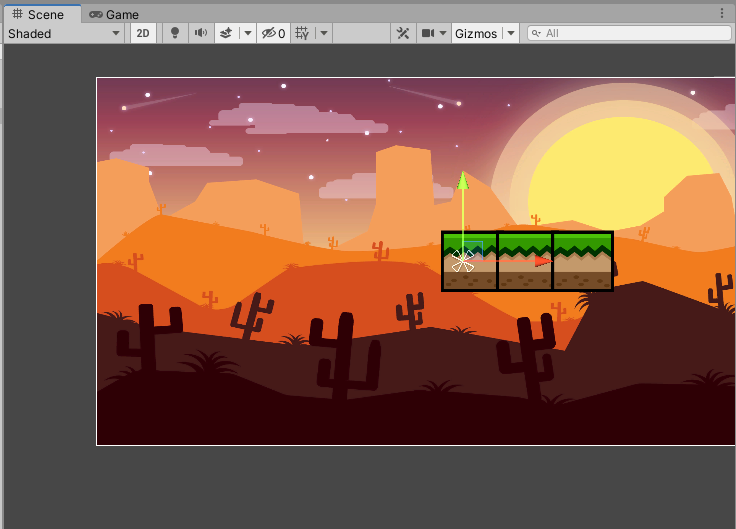


Теперь этот кубик можно использовать для построения того, чего нам нужно.

Создаём папку Ground и перемещаем наш кубик туда и клонируем его сочитанием клавиш Ctrl+D.

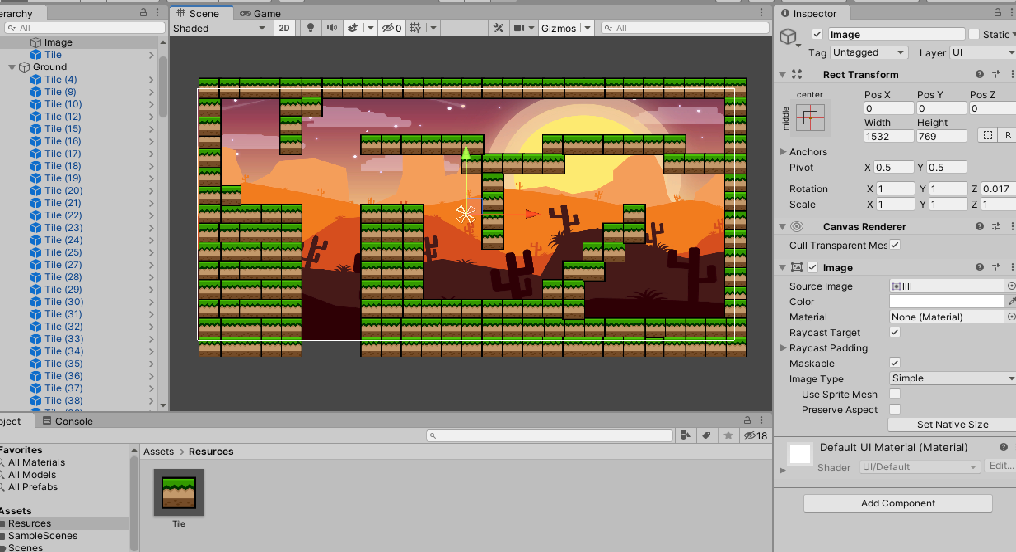


Перетаскиваем с помощь клавиши Ctrl, чтобы было ровно.



Кубик получился большим, поэтому меняем значения Scale в меньшую сторону. В данном случае все числа «1» я заменил на «0,5»

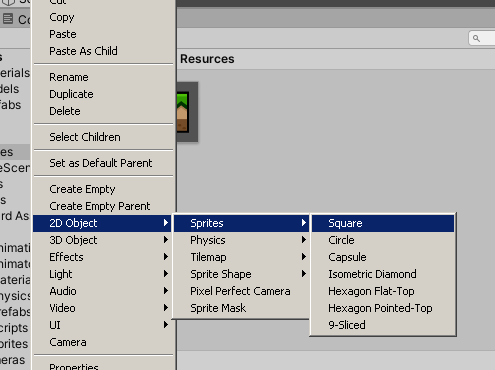
Начинаем делать локацию.

У меня получился вот такой уровень.

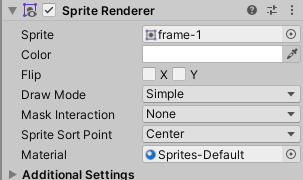
Теперь нужно добавить самого персонажа, который сможет ходить по локации.

Создаём в папке Ground пустой обьект и назовём Hero.

В обьекте создаём новый спрайт.



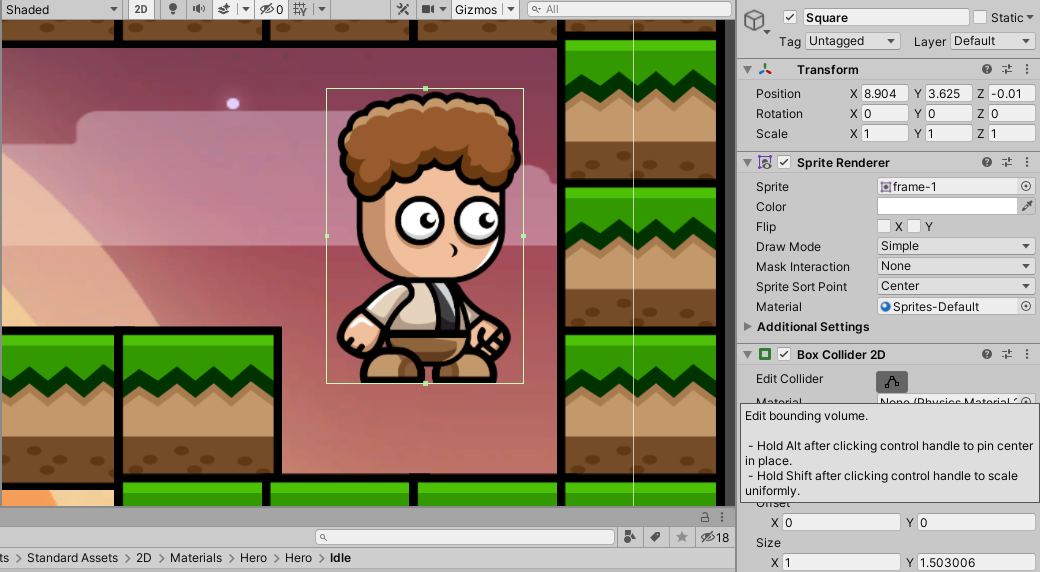
Скачиваем картинки человечка для игр в разных положениях и кидаем в Materials.

Загружаем в Sprite картинку человечка. 

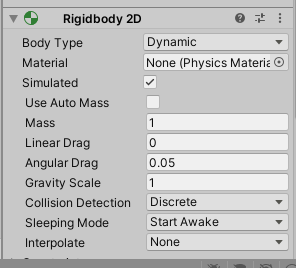
Перетаскиваем его на нужно место. Вмоём случае он будет стоять в верхнем левом углу моей локации.



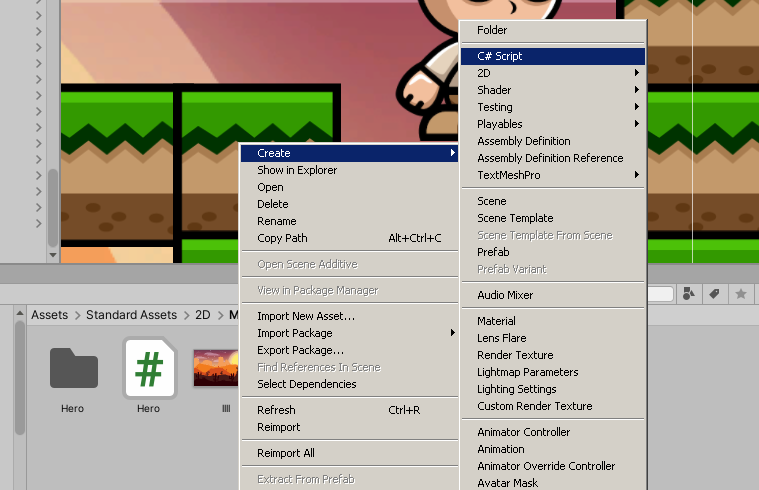
Добавляем компенент Box Collider 2D, как мы делали раньше.

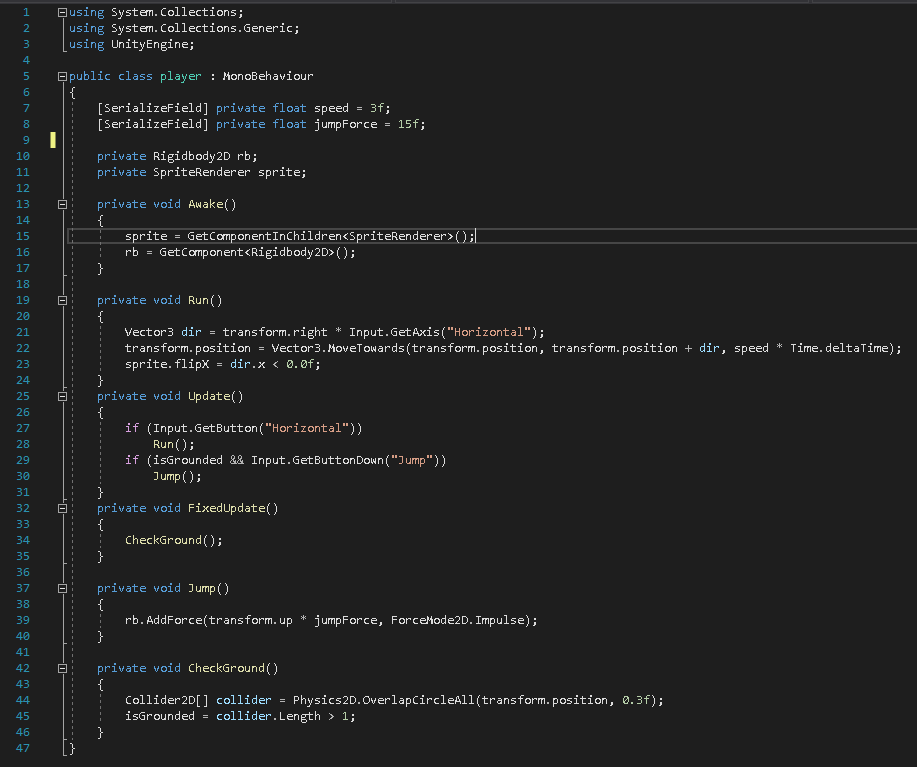
Нажимаем на Edit Collider и меняем так, чтобы человечек был замкнуть в этом квадрате.

Добавляем компонент RigidBody 2D, чтобы потом дать управление на этот обьект, тоесть на самого человечка.



Создаём С#скрипт.и называем его Hero.



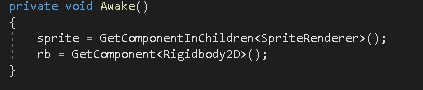
В скрипте на C# мы будем давать герою управление.

Вот такой код у нас получился.



Первой командой мы обознаачаем скорость бега человечка.

Второй командой – сила, с который человек будет прыгать.



В методе Awake мы подключаем компоненты, которые мы дали герою.

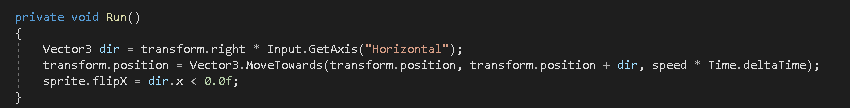
Обязательно то, что в первом случае мы используем команду

GetComponentInChildren, а во втором GetComponent.

Это из-за того, что RigidBody 2D находится на «главном» объекте, а SpriteRenderer находится на самом спрайте.

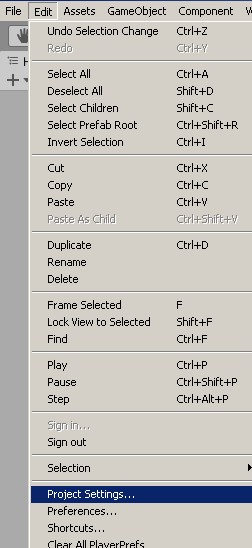


Настроим сам бег нашего героя.

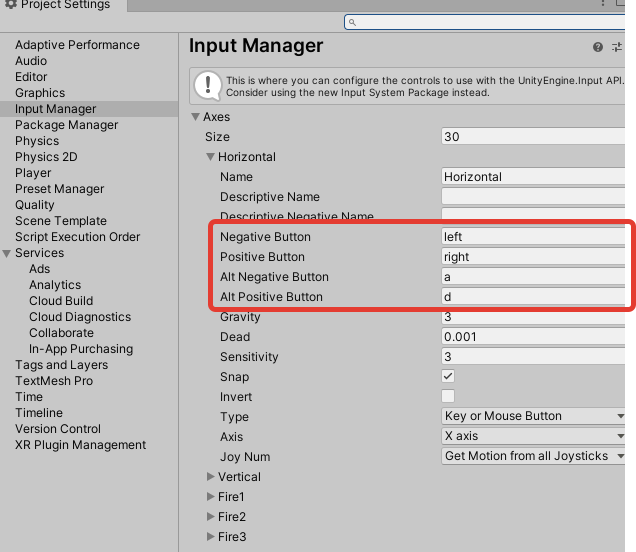


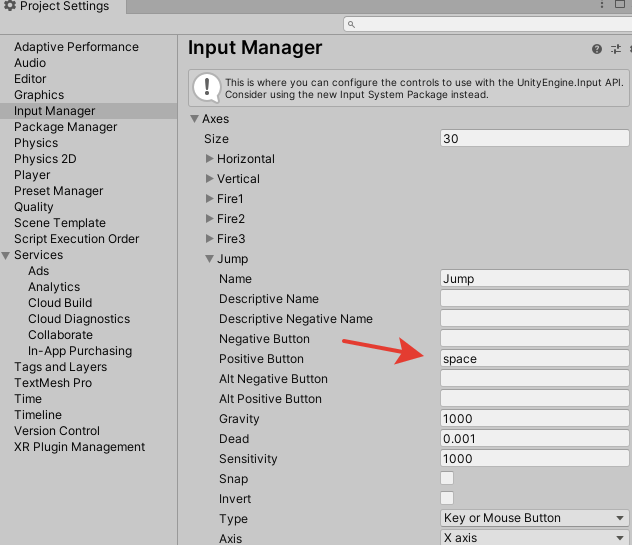
Первой командой мы даём управление персонажа, назначая кнопки передвижения. Это делается в самом Unity.

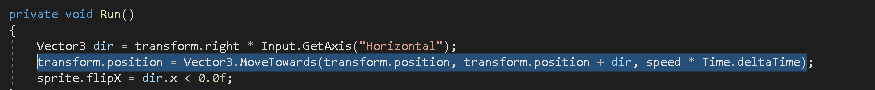
Делается это во вкладке Edit-Project Settings-Input Manager-Axes



Во вкладке Horisontal мы смотрим движение по горизонтальной оси, тоесть влево и вправо.



Так же заходим в Axes-Jump, чтобы настроить прыжок. 

Этой командой мы задаём движение. Этими параметрами мы указываем:

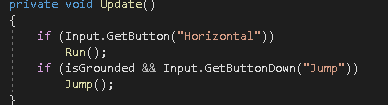
«Transform.position» –текущее местоположение;

«Transform.position + dir» - место, куда нужно переместиться;

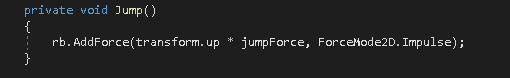
«Speed \* Time.deltaTime» - скорость.

Метод Update, в нём мы отслеживаем нажатия на кнопки.

Ставим условия, что если нажата клавиша движения по горизонтали, то запускается метод Run, тоесть – персонаж начнёт двигаться.

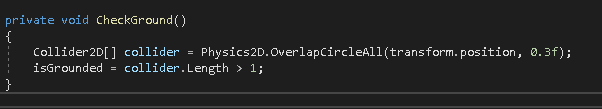


Дальше мы добавляем метод Jump, где указываем направление вверх и умножаем на силу прыжка.



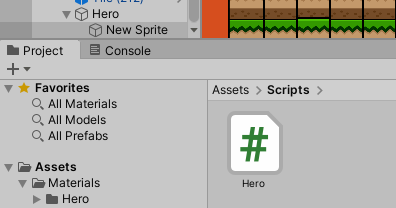
Теперь наш персонаж может бегать и прыгать.

Но есть проблема в том, что мы дали герою возможность прыгает, и он модет прыгнуть несколько раз, поэтому надо написать послднюю команду.



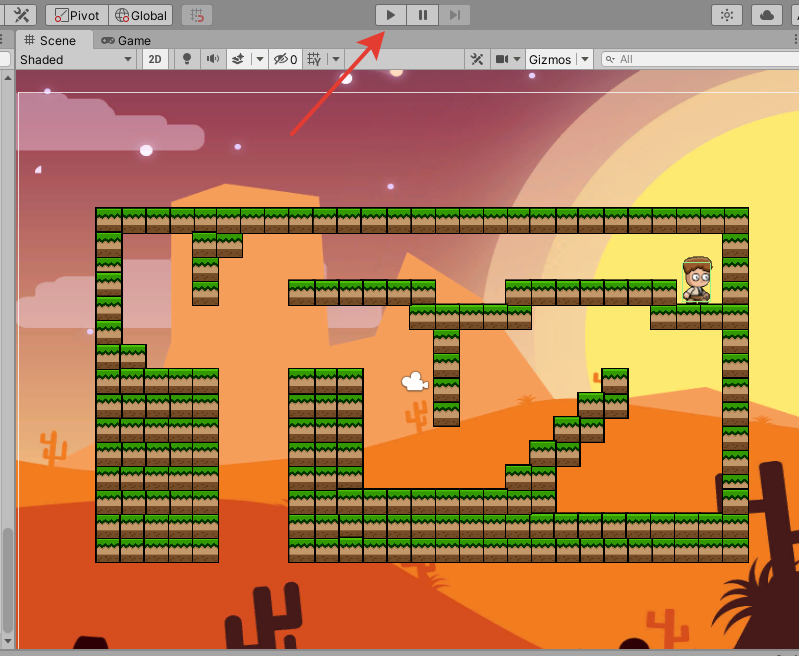
Тут мы поставили условия, что герой может прыгать, если под ногами есть земля.

Скрипт сохраняем и прикрепляем к герою.



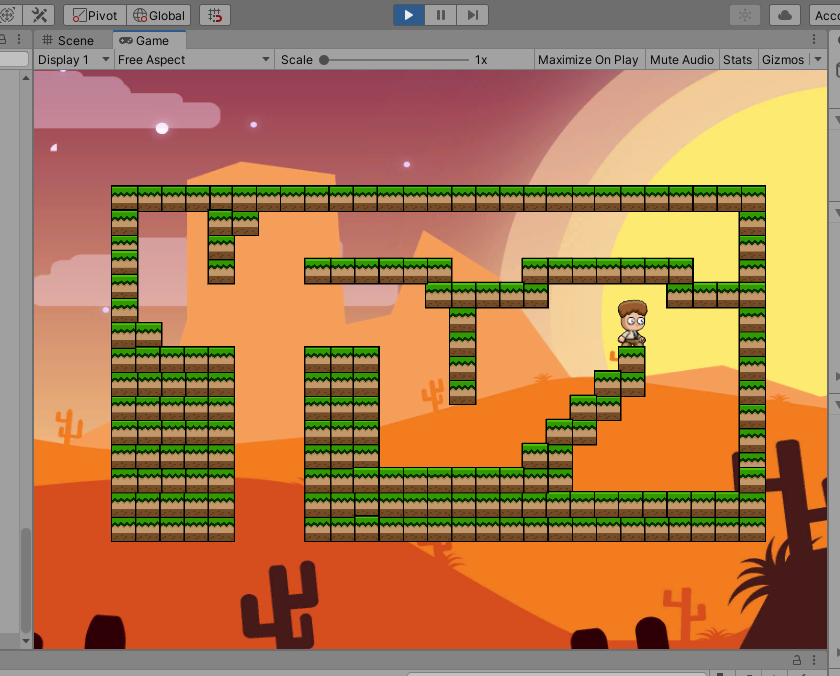
Теперь нужно проверить, всё ли правильно работает.

Нажимаем на кнопку Play сверху Unity и проверяем игру.



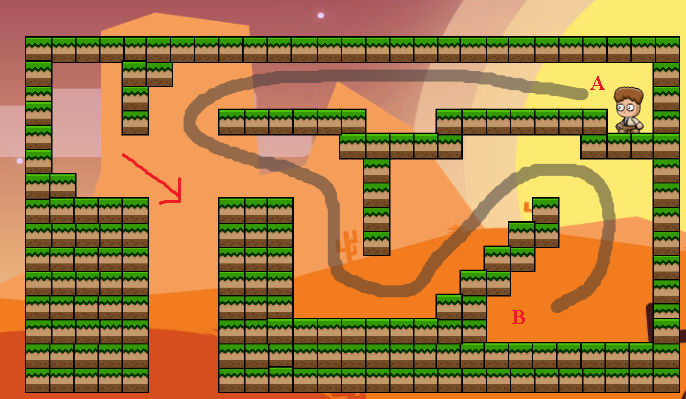
Теперь герой может спокойно передвигаться по карте.

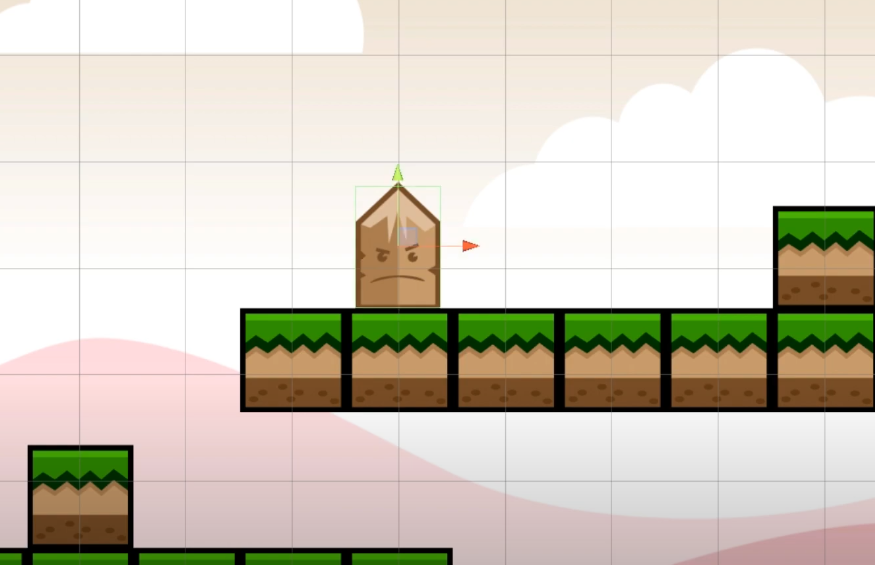




Можно много что придумать. Несколько таких локаций, переходя от точки

А, к точке B с ловушками или пропастями, добавить монстров, который будут передвигаться по локации, не давая вам пройти по ней. Докасаясь ловушек, ваш герой будет вернут на точку А, и заставит вас проходить маршрут заново. Сделать красивую графику, анимации, звук, музыку и многое другое.









**Вывод.**

Целью данной курсовой работы было создать игровой процесс на Unity.

Для выполнения поставленной цели необходимо было решить данные задачи:

1. Провести анализ основных инструментов разработки в индустрии игр.
2. Рассмотреть принципы работы различных игровых движков.
3. Выделить основные критерии для выбора игрового движка.
4. Разработать игру на выбранном игровом движке.