Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

ХТИ – филиал СФУ

Кафедра прикладной информатики, математики

и естественно-научных дисциплин

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Преподаватель И. В. Янченко

Студент гр. 51-1, зачетная книжка № 562154195 В. А. Иванов

Абакан, 2022

Содержание

[Введение 3](#_Toc109061971)

[Основная часть 4](#_Toc109061972)

[1. Общие правила безопасности при использовании компьютерной техники. 4](#_Toc109061973)

[2. Средство разработки Unity и язык программирования C#. 5](#_Toc109061974)

[3. Разработка игры. 6](#_Toc109061975)

[3.1 Создание пустого проекта. 6](#_Toc109061976)

[3.2 Интерфейс Unity. 6](#_Toc109061977)

[3.3 Создание сетки 7](#_Toc109061978)

[3.4 Управление свайпом. 10](#_Toc109061979)

[3.5 Настройка камеры. 12](#_Toc109061980)

[3.6 Движение кисти. 12](#_Toc109061981)

[3.7 Соединение линий. 13](#_Toc109061982)

[3.8 Создание уровня на основе данный. 13](#_Toc109061983)

[3.9 Создание решения уровня. 13](#_Toc109061984)

[3.10 Статус игры. 13](#_Toc109061985)

[3.11 Игровой интерфейс. 13](#_Toc109061986)

[Заключение 14](#_Toc109061987)

[Список использованных источников 15](#_Toc109061988)

[Приложения 16](#_Toc109061989)

# Введение

Мобильная игра – это видеоигра, в которую обычно играют на мобильном телефоне. Термин также относится ко всем играм, в которые играют на любом портативном устройстве.

В 1997 году, Nokia запустил игру Snake, которая была предустановлена на большинстве мобильных устройств, выпускаемых Nokia, с тех пор стала одной из самых популярных игр и установлена более чем на 350 миллионах устройств по всему миру.

Сегодня мобильные игры обычно загружаются из магазина приложений, но в некоторых случаях также предварительно загружаются на портативные устройства с помощью OEM или оператором мобильной связи при покупке.

Подавляющее большинство игр написано в жанре «аркада» и «головоломка». Изредка встречаются представители других жанров. Основным ограничением для «разнообразия жанров» является устройство ввода. Например, для удобной игры в 3D-шутер желательна возможность одновременного наведения на цель и стрельбы – использование двух клавиш одновременно.

Цель работы – разработать мобильную игру жанра головоломка средствами среды Unity.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Познакомиться с межплатформенной средой разработки игр Unity.
2. Изучить основы языка программирования C#.
3. Подготовить для игры все необходимые спрайты/объекты.
4. Создать игру.

# Основная часть

## Общие правила безопасности при использовании компьютерной техники.

Применяя технические средства в школе, необходимо строго руководствоваться санитарно-гигиеническими нормами и правилами безопасности. Современные технические устройства, как правило, сложная техника, требующая соблюдения определенных инструкций. Каждое покупаемое техническое устройство должно иметь инструкцию на русском языке.

Существует очень много схожих между собой технических устройств. Выбор устройства, оптимально подходящего для решения конкретной задачи, обычно очень сложное дело. Надо научиться правильно использовать консультации специалистов.

Вся компьютерная техника питается электротоком напряжением 220 В, которое опасно для жизни человека. Поэтому все лица, допущенные к работе с компьютерной техникой, должны пройти инструктаж по технике безопасности и соблюдать следующие правила:

* Работать только на исправной компьютерной технике.
* Знать блок-схему используемой компьютерной техники и правила её эксплуатации, порядок включения, выключения и заземления аппарата.
* Перед включением общего электропитания проверить исходное положение всех выключателей, розеток и вилок и выключить их.
* Запретить разборку аппарата компьютерной техники учащимися.
* При работе с компьютерной техникой пользоваться только внешними элементами управления.
* В случае замыкания (появления искр, запаха гари) - отключить электропитание.
* Замену деталей электроаппаратуры и ее ремонт проводить при выключенных источниках питания.
* Запрещается определять наличие напряжения путем прикосновения руками к токоведущим деталям аппаратуры.
* Нельзя менять и ставить предохранители на электроаппаратуру, находящуюся под напряжением.
* Соединительные провода автотрансформатора с сетью и усилителя с громкоговорителем не должны находиться па пути выхода зрителей.
* Запрещается использовать воду и пенные огнетушители для тушения загоревшейся электроаппаратуры, так как эти средства являются проводниками тока и, следовательно, могут привести к короткому замыканию и поражению током человека, производящего тушение.
* Во избежание ожогов нельзя прикасаться к проекционным и радиолампам в течение 10 мин после их выключения.
* Не разрешается касаться деталей аппаратуры во время её работы.
* Нельзя включать в сеть аппараты со снятыми фальшпанелями, задними крышками. Это открывает доступ к деталям, находящимся под высоким напряжением, достигающим в телевизорах и дисплеях ЭВМ величины до 12000-25000В. Снятие надолго крышек с аппаратов приводит их к загрязнению, вызывающему нарушение нормальной работы электрических частей устройств.
* Нельзя пользоваться аппаратами, у которых не работает вентилятор, ибо это может привести к перегоранию или более серьезным неисправностям.
* При замене проекционной лампы аппарат следует отключить от сети и подождать, пока лампа остынет.
* Устанавливать новую лампу можно только специальным пинцетом, чтобы не оставлять отпечатков пальцев на колбе, что может вызвать разрушение колбы и преждевременный выход ее из строя.
* В диапроекторах, снабженных пультами дистанционного управления, используют диапозитивы только в пластмассовых рамках.
* В разных странах приняты разные стандарты на напряжение в сети и форму розетки. В нашей стране в качестве стандарта принято напряжение 220 В частотой 50 Гц. Перед подключением к розетке нового электрического прибора необходимо проверить, на какое напряжение он рассчитан.

## Средство разработки Unity и язык программирования C#.

**Unity** – современный кросс-платформенный движок для создания игр и приложений, разработанный Unity Technologies. С помощью данного движка можно разрабатывать не только приложения для компьютеров, но и для мобильных устройств, игровых приставок и других девайсов.

Поговорим немного о характеристиках движка. Во-первых, стоит отметить то, что в среду разработки Unity интегрирован игровой движок, иными словами, вы можете протестировать свою игру не выходя из редактора. Во-вторых, Unity поддерживает импорт огромного количества различных форматов, что позволяет разработчику игру конструировать сами модели в более удобном приложении, а Unity использовать по прямому назначению –разработки продукта. В-третьих, написание сценариев(скриптов) осуществляется на популярном языке программирования –C#.

На Unity написаны тысячи игр, приложений и симуляций, он используется как крупными разработчиками, так и в создании инди-игр. Игры на Unity охватывают множество платформ и жанров.

Таким образом, Unity является актуальной платформой, с помощью которой вы можете создавать свои собственные приложения и экспортировать их на различные устройства.

**C#** (C Sharp) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. C# относится к семье языков с С-подобным синтаксисом.

## Разработка игры.

### Создание пустого проекта.

Для начала нам необходимо создать новый проект в Unity Hub выбрав нужное для разработки нашей игры ядро, в моём случае это «3D». После этого нужно указать имя проекта и его расположение.

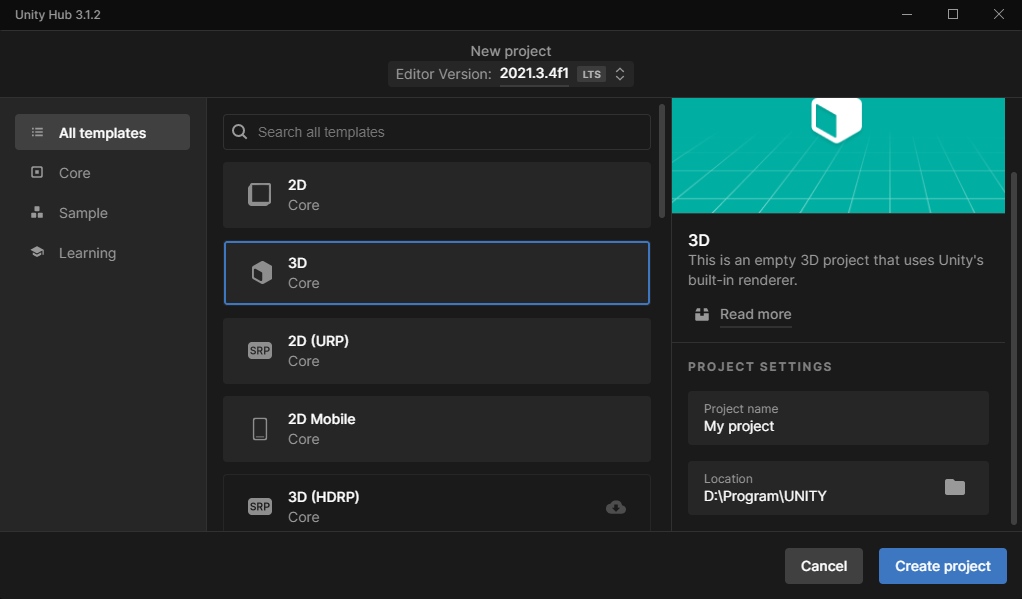


Рисунок 1 – Создание нового проекта

### Интерфейс Unity.

После создания проекта нужно ознакомиться с интерфейсом, который увидим перед собой.

Расположение окон по умолчанию обеспечивает практический доступ к наиболее распространенным окнам. Если вы еще не знакомы с различными окнами в Unity, вы можете определить, по имени на вкладке. Наиболее распространенные и полезные окна показаны ниже:

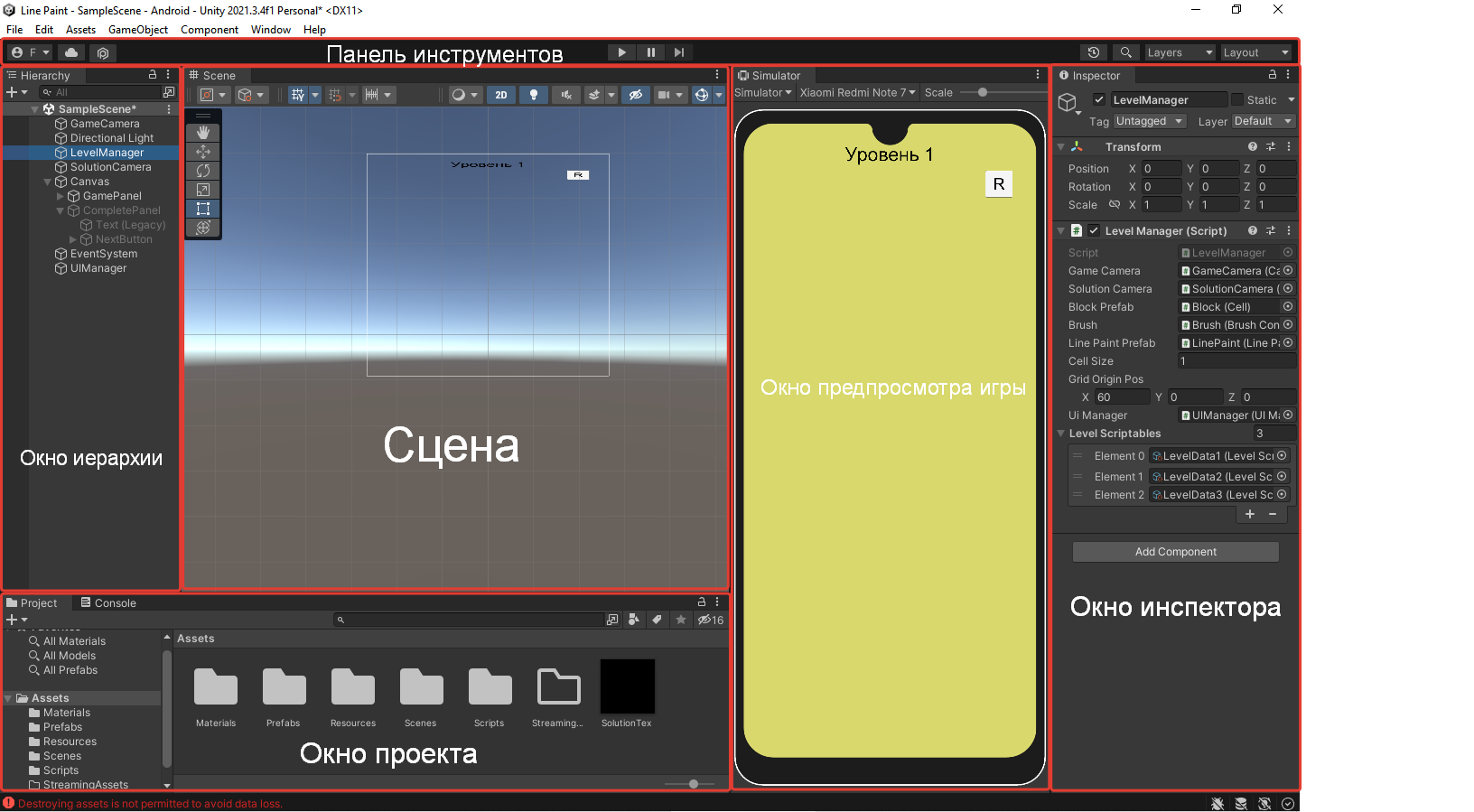


Рисунок 2 – Интерфейс Unity

**Окно проекта** отображает библиотеку активов, доступных для использования в вашем проекте. Когда активы импортируются в проект, они появляются здесь.

**Сцена** позволяет визуально перемещаться по сцене и редактировать ее. Вид сцены может отображать трехмерную или двухмерную перспективу, в зависимости от типа проекта, над которым вы работаете.

**Окно иерархии** представляет собой иерархическое текстовое представление каждого объекта в сцене. У каждого элемента сцены есть запись в иерархии, поэтому два окна неразрывно связаны. Иерархия раскрывает структуру того, как объекты связаны друг с другом.

**Окно инспектора** позволяет просматривать и редактировать все свойства выбранного в данный момент объекта. Поскольку разные типы объектов имеют разные наборы свойств, макет и содержимое окна инспектора будут различаться.

**Окно предпросмотра** позволяет увидеть и проверить игру без компиляции под нужную платформу.

**Панель инструментов** обеспечивает доступ к наиболее важным рабочим функциям. В центре находятся элементы управления воспроизведением, паузой и шагом. Кнопки справа открывают доступ к меню видимости слоя и, наконец, меню компоновки редактора (которое предоставляет некоторые альтернативные компоновки для окон редактора и позволяет сохранять собственные пользовательские компоновки).

### Создание сетки

Нам нужно создать скрипт «Grid», который будет обрабатывать сетку. В нём мы создаём поля для ширины и длины сетки, а также для размера её клеток.



Рисунок 3 – Поля размеров сетки и размера клеток

Дальше мы создаём приватный массив «gridArray» для хранения клеток и массив «GridArray» для обращения к массиву «gridArray» из другого скрипта.



Рисунок 4 – Массивы

Также мы создаём метод «Initialize», в который передаём ширину и длину сетки и размер клеток. Этот метод отвечает задаёт размеры массива сетки.

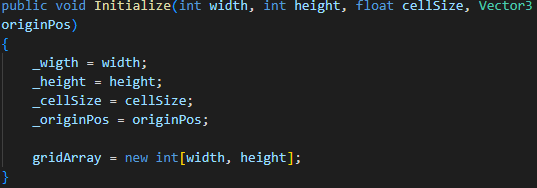


Рисунок 5 – метод «Initialize»

Дальше мы создаём новый скрипт «LevelManager» и в нём создаём ссылку на наш прошлый скрипт и те же три поля, что и в прошлом скрипте.

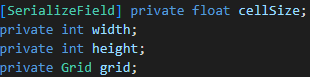


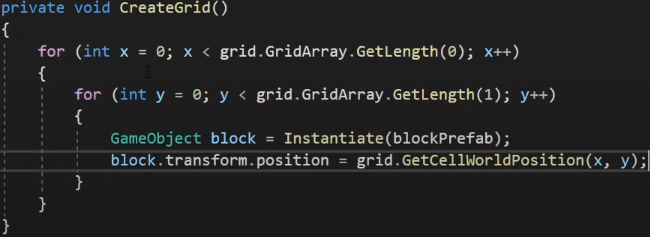
Рисунок 6 – Размеры массива и ссылка на скрипт

В методе «Start», который создаётся автоматически при создании скрипта, с помощью метода «Initialize» мы передаёт размеры массива в скрипт «Grid».



Рисунок 7 – Передача размеров в другой скрипт

Теперь создаём метод, отвечающий за создание нашей сетки и её позицию.

Рисунок 8 – Метод «CreateGrid»

В скрипте «Grid» мы создадим ещё один метод, который будет находить для нас позицию сетки.

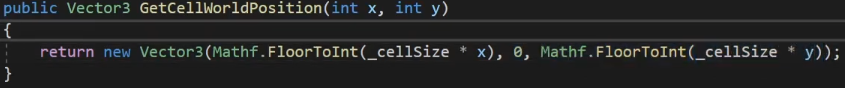


Рисунок 9 – Метод «GetCellWorldPosition»

В иерархии нашего проекта создадим пустой объект (Create Empty) и назовём его «LevelManager», а также создадим куб (3D Object → Cube) с названием «Block», который и будет клеткой нашей сетки.

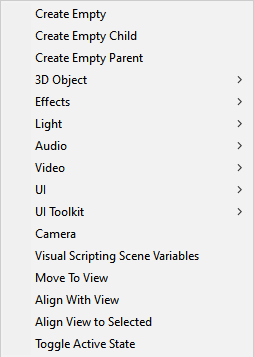


Рисунок 10 – Окно создания объектов

Выбираем наш пустой объект и в инспекторе в нужные полу вводим значения, в поле «Block Prefab» переносим ранее созданный куб.

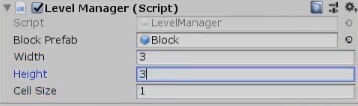


Рисунок 11 – Поля данных в инспекторе

### Управление свайпом.

Создаём скрипт «SwipeController». В нём создаём перечисление возможных свайпов и начальную и конечную координату свайпа.

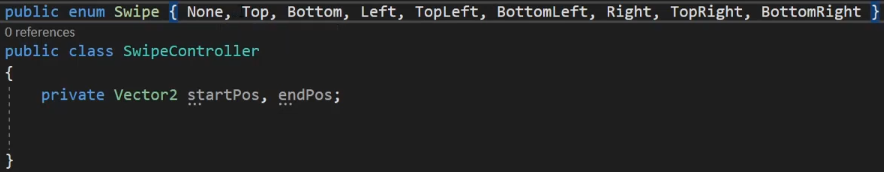


Рисунок 12 – Направления свайпа и его координаты

Создаём метод, в котором будет определятся направление, вектор текущего свайпа и его угол между «x» и «y».

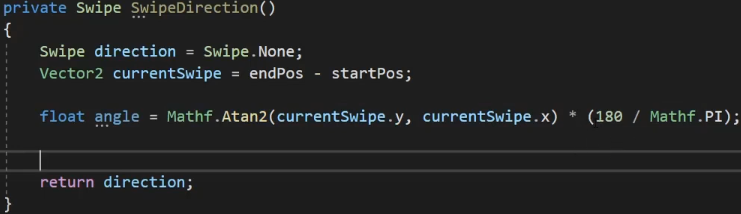


Рисунок 13 – Метод «SwipeDirection»

Добавляем проверку, которая определяем направление нашего свайпа с помощью угла.

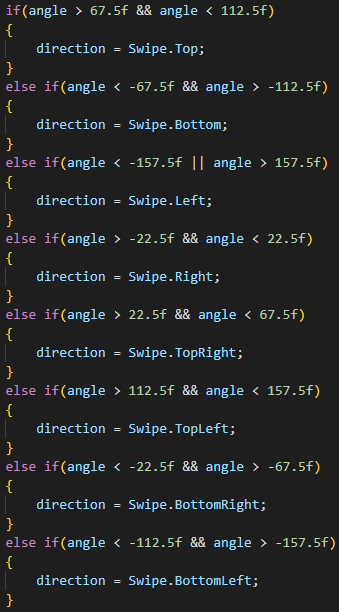
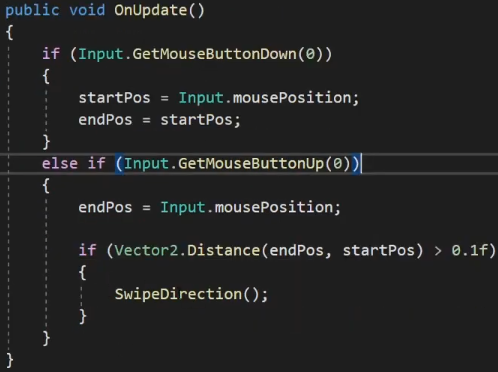


Рисунок 14 – Проверка направления свайпа

Создаём ещё один метод, в котором определяем, сделали мы клик или свайп. Если был сделан свайп, то мы вызываем метод «SwipeDirection».



Русинок 15 – Метод «OnUpdate»

В скрипте «LevelManager» создаём метод, который будет вызывать другой метод «OnUpdate».

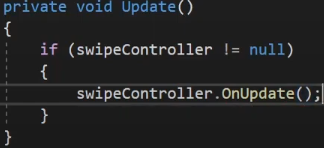


Рисунок 16 – Метод «Update»

### Настройка камеры.

Создаём скрипт «CameraZoom», а в нём метод, который будет определять, на каком расстоянии камера должна находиться от сетки, а также устанавливаем её позицию. Расстояние зависит от высоты и ширины сетки.

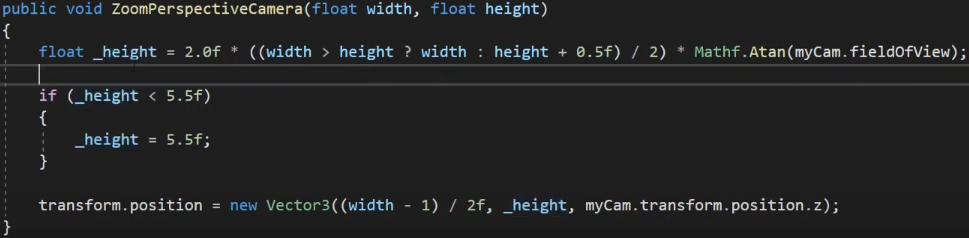


Рисунок 17 – Метод «ZoomPerspectiveCamera»

В скрипте «LevelManager» создаем ссылку на скрипт «CameraZoom», а также в методе «Start» вызываем метод «ZoomPerspectiveCamera».



Рисунок 18 – Вызов метода «ZoomPerspectiveCamera»

### Движение кисти.

Создадим скрипт, который будет содержать координаты кисти.

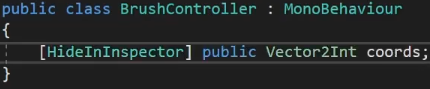


Рисунок 19 – Скрипт «BrushController»

Дальше в иерархии создадим пустой объект с именем «Brush» и добавим на него нам скрипт и цилиндр.

Для цилиндра добавим материал с нужным нам цветом.



Рисунок 20 – Материал кисти

Также цилиндру нужно указать параметры как позиция и размер.

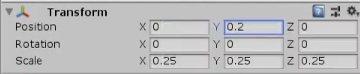


Рисунок 21 – Параметры цилиндра

В скрипте «LevelManager» добавим ссылку на нашу кисть.



Рисунок 22 – Ссылка на кисть

В методе «Start» напишем функцию для создания кисти на нашей сетке, а также присвоем ей начальные координаты.



Рисунок 23 – Создание кисти и присвоение ей начальных координат

В скрипте «Grid» создадим метод, в котором новой переменной будет присваиваться координаты, в зависимости от нашего свайпа.

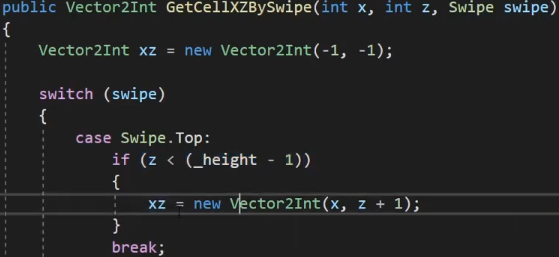


Рисунок 24 – Метод «GetCellXZBySwipe»

В скрипте «LevelManager» создадим метод, который будет вызывать метод «GetCellXZBySwipe» и его результат присваивать новой переменной.

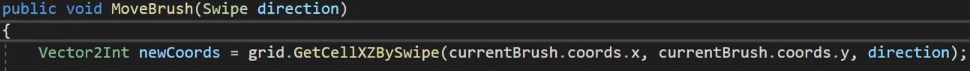


Рисунок 25 – Метод «MoveBrush»

### Соединение линий.

Создадим скрипт «Connection», а в нём переменные для начальных и конечных координат.

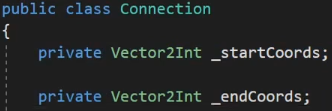


Рисунок 26 – Скрипт «Connection» и координаты

Также создадим список, в который будут добавляться все созданные соединения.



Рисунок 27 – Список соединений

Создадим пустой объект, на который мы добавим новый скрипт «LinePaintScript», а также «Line Renderer».

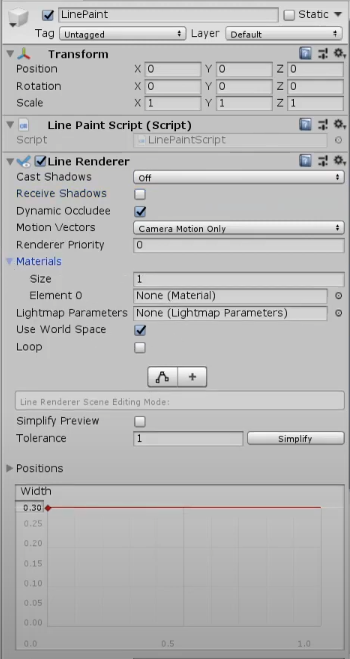


Рисунок 28 – Пустой объект со скриптом «LinePaintScript»

В скрипте «LinePaintScript» мы создадим ссылку на «Line Renderer» (рис. 29), а также метод для создания соединений, которые будут показывать в какой направлении мы двигает кисть (рис.30).



Рисунок 29 – Ссылка на «Line Renderer»

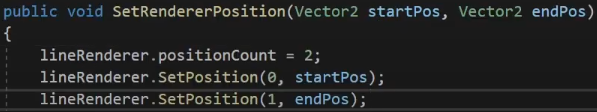


Рисунок 30 – Метод «SetRendererPosition»

В методе «MoveBrush» мы создаём функцию для создания соединения в самой игре, а также присваиваем ей координаты.



Рисунок 31 – Создание соединения в самой игре

### Создание уровня на основе данный.

Теперь создадим скрипт, в котором будут записываться данные об уровне, которые мы должны будем повторить в игре.

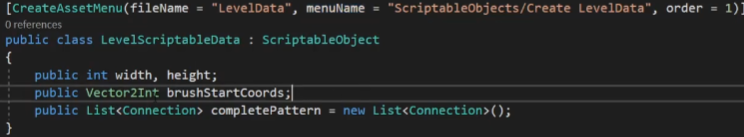


Рисунок 32 – Скрипт «LevelScriptableData»

В список «сompletePattern» мы будем записывать начальные и конечные координаты соединений, в «width» и «height» будем записывать размер нашего уровня, а в «brushStartCoords» – начальное положение кисти.

### Создание решения уровня.

Создадим вторую камеру, которая будем показывать нам решение уровня, которое мы должны повторить.

Создадим «Canvas» и в нём создадим «Panel». Дальше мы создадим картинку, которая будем показывать решение уровня.



Рисунок 33 – «Canvas»

Теперь создадим метод, который будет на картинке создавать решение, чтобы мы его могли видеть. Этот метод мы будем вызывать в методе «Start».

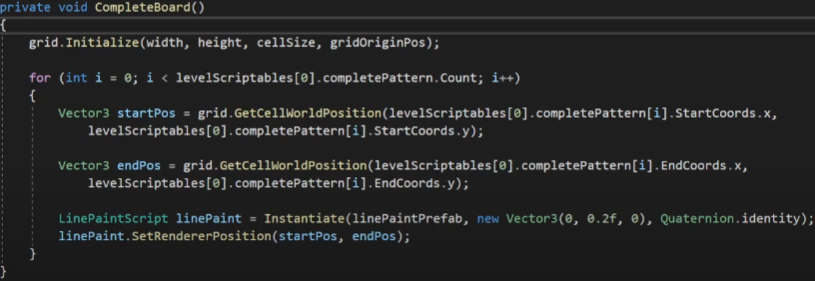


Рисунок 34 – Метод «CompleteBoard»

В скрипте «CameraZoom» создадим ещё один метод для определения расстояния, но уже для камеры решения.

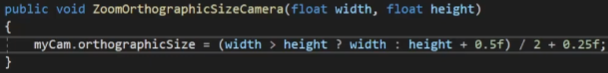


Рисунок 35 – Метод «ZoomOrthographicSizeCamera»

В методе «CompleteBoard» мы вызываем наш прошлый метод, а также передаём в него размеры решения.



Рисунок 36 – Вызов метода «ZoomOrthographicSizeCamera»

### Статус игры.

Создадим скрипт, в котором у нас будет перечисление игровых статусов, а также несколько переменных. Переменной «gameStatus» присвоим значение «Playing».

Создадим ещё один метод, который будет проверят завершён уровень или нет.

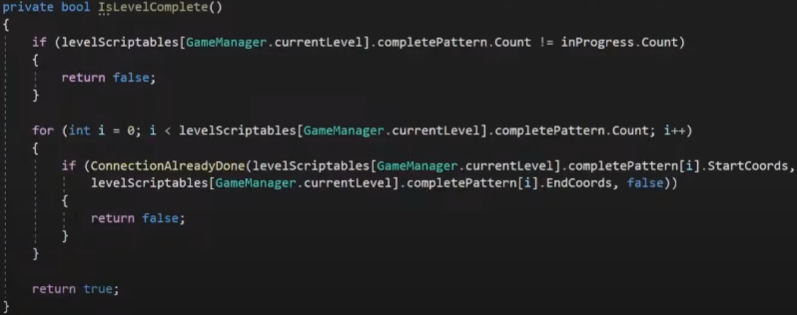


Рисунок 37 – Метод «IsLevelComplete»

В методе «MoveBrush» создадим проверки. И если наш уровень завершён, то мы меняем наш игровой статус на «Complete».

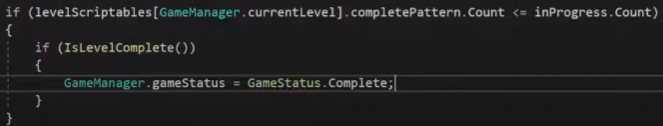


Рисунок 38 – Проверка для изменения статуса

В методе «Start» мы будем присваивать размерам сетки размеры решения.



Рисунок 39 – Новые размеры сетки

### Игровой интерфейс.

В ранее созданной игровой панели создадим текст и кнопку. Текст будет показывать нам текущий уровень, а кнопка будет повторять уровень.



Рисунок 40 – Текст уровня и кнопка повтора

В канвасе создадим ещё одну панель, в ней добавим текст, который будет показывать, что мы завершили уровень, а также кнопку, которая будет менять уровень на следующий.



Рисунок 41 – «CompletePanel»

В новом скрипте мы подключим библиотеки для работы с элементами пользовательского интерфейса и сценами.



Рисунок 42 –Подключение библиотек

А также добавим ссылки на все наши объекты интерфейса.

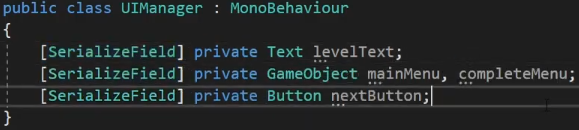


Рисунок 43 – Ссылки на объекты интерфейса

Создадим метод, который при завершении уровня будет нам показывать «CompletePanel».

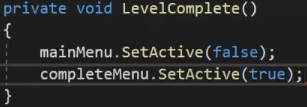


Рисунок 44 – Метод «LevelComplete»

Создадим ещё один метод, который будем запускать другую сцену при нажатии на ранее созданные кнопки.

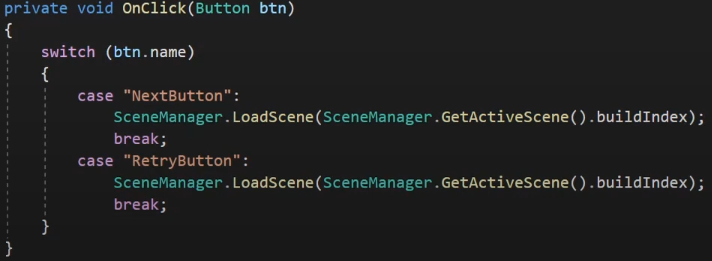


Рисунок 45 – Метод «OnClick»

В скрипте «LevelManager» создадим ссылку на наш скрипт для интерфейса.



Рисунок 46 – Ссылка на скрипт «uIManager»

В методе «MoveBrush» дополним проверку на завершение уровня. Теперь, если уровень завершён, у нас не только будем меняться игровой статус, а также будем увеличиваться уровень и вызываться метод «LevelComlete», если уровень будет последним, но после его завершения все уровни начнутся с самого начала.

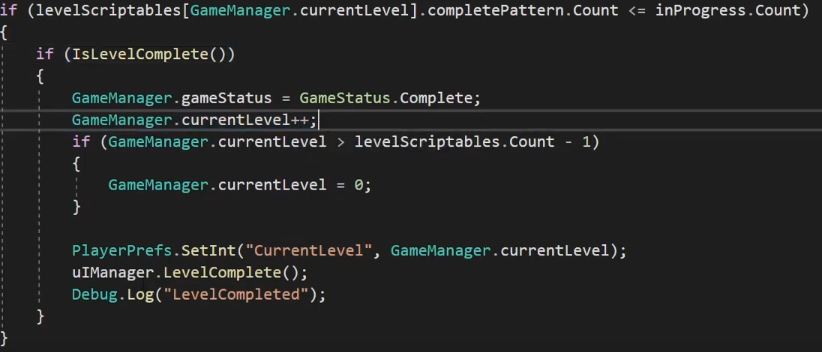


Рисунок 47 – Дополненный метод «MoveBrush»

На этом наша игра закончена и увидеть её геймплей можно на скриншоте ниже:

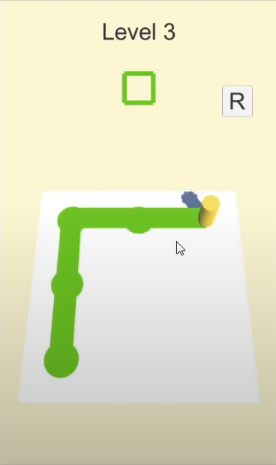


Рисунок 48 – Геймплей игры

# Заключение

В ходе разработки данного проекта было получено много полезного опыта и знаний в сфере разработки игр. За время выполнения проекта был изучен интерфейс Unity, а также основы языка программирования C#, на котором пишутся скрипты для игр на движке Unity.

В результате работы над данным проектом все цели и задачи были успешно выполнены, а это значит, что игра в жанре головоломки тоже была успешна создана.

Ссылка на игру: <https://github.com/TheRatKing1/SummerGame>

# Список использованных источников

1. С# уроки – <https://www.youtube.com/playlist?list=PLQOaTSbfxUtD6kMmAYc8Fooqya3pjLs1N>
2. Общие правила безопасности при использовании компьютерной техники – URL: <https://studopedia.ru/12_201478_obshchie-pravila-bezopasnosti-pri-ispolzovanii-kompyuternoy-tehniki.html>
3. Интерфейс Unity – <https://www.youtube.com/watch?v=68YzkGtwTWM>