**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное**

**учреждение Московской области «Люберецкий техникум имени Героя**

**Советского Союза, летчика-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:** Разработка игры на Unity

Группа № 185

Курс 4

Выполнил/а/ студент                                   Шумейкова Татьяна Александровна

(подпись) (ФИО полностью)

Руководитель                                                       Жирнова Юлия Витальевна

(подпись) (ФИО полностью)

Оценка  \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дзержинский 2021

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc95057751)

[1. Изучение предметной области 4](#_Toc95057752)

[2. Постановка задачи 5](#_Toc95057753)

[3. Выбор программных средств разработки 6](#_Toc95057754)

[4. Разработка игры на Unity 8](#_Toc95057755)

[4.1. Создание проекта в Юнити 8](#_Toc95057756)

[Выводы 9](#_Toc95057757)

[Список литературы 10](#_Toc95057758)

Введение

Практика — это одна из важнейших составляющих профессиональной подготовки специалистов. Учебная практика направлена на то, чтобы перенести полученные знания и умения из области теории в область повседневной профессиональной деятельности, на развитие экономического сознания студентов, развитие аналитических способностей, а также на самостоятельное решение выполнения функций управления.

Целью учебной практики этого модуля является разработка проектной документации для программного модуля.

Для достижения данной цели требуется решить следующие задачи:

* Описание предметной области.
* Изучение программных средств.
* Проектирование ER-диаграммы.
* Проектирование диаграммы классов UML (class diagram).
* Проектирование диаграммы деятельности UML (activity diagram).

В ходе учебной практики планируется освоение следующего вида деятельности: «Осуществление интеграции программных модулей» (ПК):

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

1. Изучение предметной области

Аббревиатура UML расшифровывается как Unified Modeling Language, дословно переводится как «унифицированный язык моделирования». UML — это язык моделирования, который позволяет создавать структуры программных систем.

UML состоит из графических обозначений, диаграмм, которые помогают создать дизайн программных проектов. С помощью UML-диаграмм проектные группы коммуницируют между собой, составляют и проверяют архитектурный дизайн ПО.

Работа с UML-диаграммами — важная часть проекта, так как на этом этапе продумывается его структура. Проектирование помогает в дальнейшем не запутаться в коде, снизить количество ошибок и упростить работу.

UML имеет единый синтаксис, поэтому является международным языком. Диаграммы будут понятны любому человеку, знакомому с ним. Также стоит отметить, что UML используется для разработки широкого спектра программ от информационных систем масштаба предприятия до распределенных веб-приложений.

Основные цели дизайна UML:

* Проектирование. Благодаря UML разработчики получили возможность создавать модели различных процессов, анализировать, проектировать и внедрять программные системы, рисовать схемы приложений, по которым впоследствии пишется код.
* Обеспечение роста рынка объектно-ориентированных инструментов и развития отрасли.
* Создание UML таким образом, чтобы можно было работать с любым языком программирования и любому пользователю.
* Коммуникация внутри команды и при общении с заказчиком.

2. Постановка задачи

3. Выбор программных средств разработки

Компьютерное построение диаграмм — это процесс создания масштабируемых диаграмм на ПК, которые могут использоваться в различных приложениях, таких как макетирование, организационные диаграммы, временные графики, планы этажей и даже прототипы пользовательских программных интерфейсов. Диаграммы состоят из фигур, объектов и трафаретов, которые при правильном объединении могут помочь спроецировать много полезной информации.

Современное компьютерное построение диаграмм основано на использовании векторных форм вместо растровых. Растровые фигуры или растровые изображения плохо масштабируются, и их разрешение и качество ухудшаются с каждым редактированием. Векторные же фигуры являются математическим представлением фигур и, следовательно, хорошо масштабируются. На векторные фигуры редактирование не влияет, и их наиболее целесообразно использовать, когда требуются точные измерения.

Программный продукт Microsoft Visio — векторный графический редактор для Windows, предназначенный для быстрой и качественной разработки графических документов любой сложности. С помощью него можно создавать схемы, блок-схемы, планы, технические проекты, модели, диаграммы, различные чертежи и многое другое.

MS Visio является наиболее популярным программным обеспечением для создания диаграмм, которое позволяет создавать диаграммы, визуализировать данные и моделировать процессы в привычном интерфейсе. Visio поставляется с массивом шаблонов и встроенных форм для создания практически любых диаграмм. Visio также позволяет пользователям определять свои собственные формы и импортировать их в чертеж.

Visio предназначен больше для корпоративного класса, поскольку домашним пользователям редко приходится использовать расширенные функции построения диаграмм. Однако многие пользователи покупают стандартную версию Visio для лучшей визуализации простых диаграмм, таких как генеалогическое дерево или планы этажей.

MS Visio тесно интегрируется с другими продуктами Microsoft Office, такими как Word, Excel и Access. Данные могут быть напрямую импортированы из этих программ и преобразованы в значимые диаграммы, которые изменяются в реальном времени в соответствии с данными.

Более новые версии Visio, от Visio 2013 и выше, поддерживают формат .vsdx, который обеспечивает улучшенное сжатие и многие другие функции по сравнению со старым форматом .vsd, поддерживаемым Visio 2010 и более ранними версиями.

Visio, входящий в состав пакета Office 365, постоянно обновляется.

Имеет следующие полезные функции:

* Визуализатор данных.

Data Visualizer помогает автоматически создавать диаграммы процессов из данных Excel. Готовые или настраиваемые шаблоны Excel, содержащие информацию, которая помещается в диаграммы, можно импортировать непосредственно в Visio с помощью визуализатора данных и преобразовывать в диаграмму Visio. Любые изменения, внесенные в исходный файл Excel, также отражаются на диаграмме Visio.

* Визуализация структур базы данных из исходных данных.

Последняя версия Visio позволяет осуществлять реверс-инжиниринг баз данных для создания их визуального представления. Visio может подключаться к различным базам данных, таким как MySQL, SQL Server, Oracle и т. д., и может обновлять диаграммы синхронно с изменениями в базе данных.

* Шаблоны для отраслевых диаграмм.

Последние обновления Visio для Office 365 поставляются с множеством новых стандартизированных шаблонов, предназначенных для таких отраслей, как бизнес-процессы, разработка программного обеспечения, информационные технологии и образование. Эти шаблоны предоставляются как Microsoft, так и другими третьими лицами, расширяющими возможности Visio.

4. Разработка игры на Unity

4.1. Создание проекта в Юнити

**1. Создание проекта в Unity.**

Пишем сюда (Рис. 1)

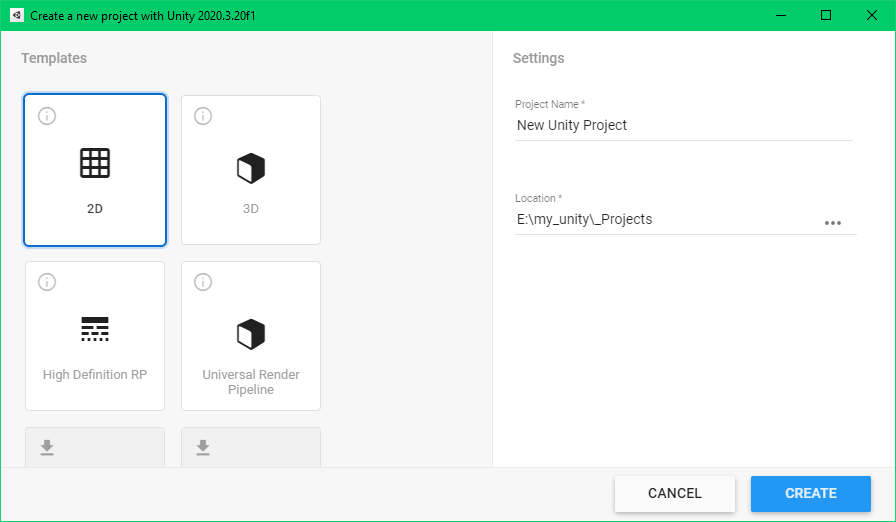


Рисунок 1. Название

**2. Создание игрока.**

Пишем сюда

Рисунок 2. Название

**3. Добавление …**

Пишем сюда

Рисунок 3. Название

**Создание игрока; возможность его передвижения в 4-х направлениях; анимация ходьбы.**

Игрока можно загрузить как готовое изображение. В нашем случае игрок был нарисован в онлайн-редакторе фотографий и конструкторе дизайна «Pixlr». Редактор можно найти по ссылке: https://pixlr.com/ru/e/

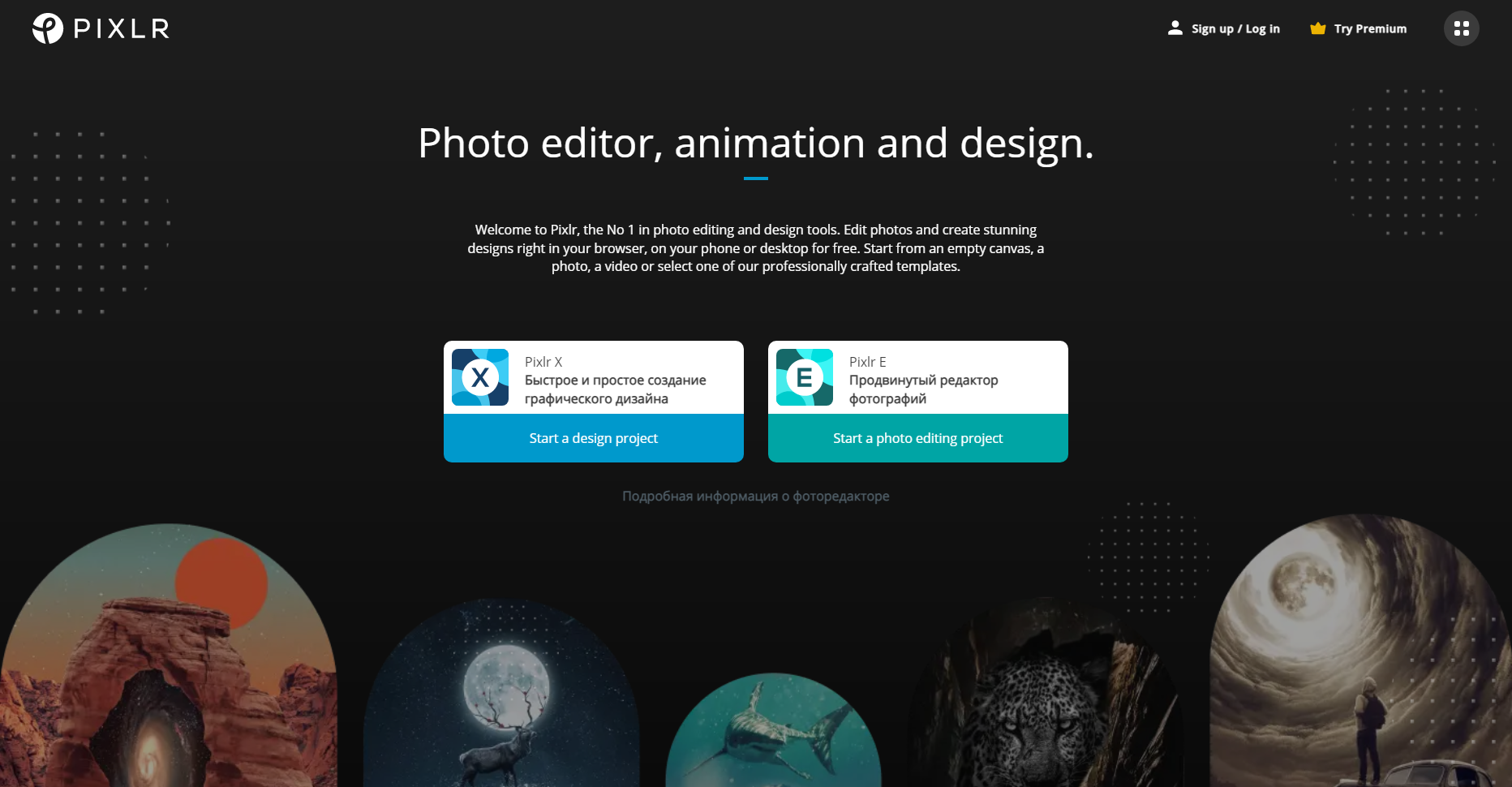


Рисунок 0. Название

Чтобы игрок имел возможность анимированно передвигаться в 4-х направлениях, следует иметь 4 ракурса его изображения (спереди, сзади, справа, слева).

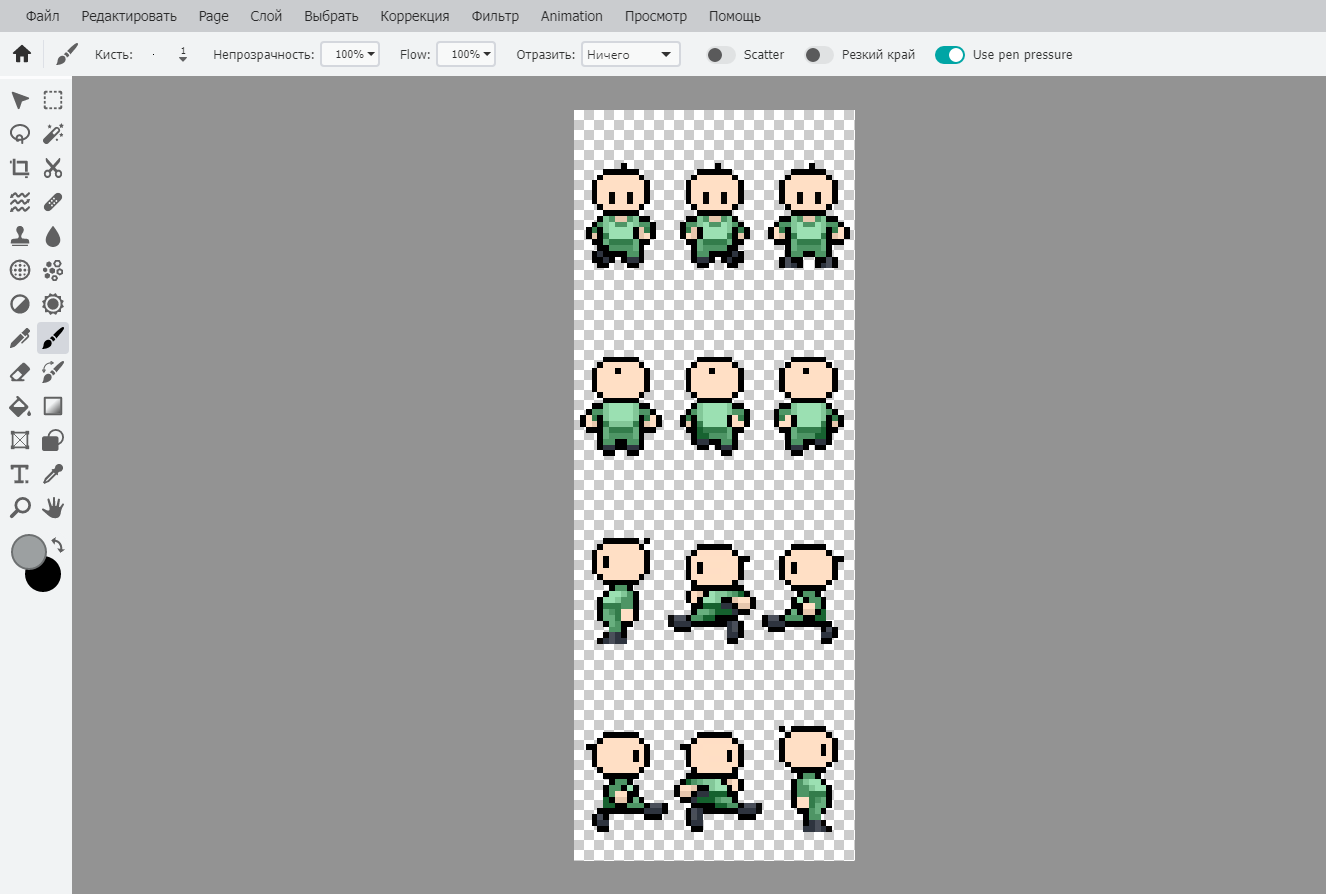


Рисунок 0. Раскадровка движений

Добавим в проект изображение игрока и разместим его на сцене.

Добавим игроку компонент Rigidbody.

Rigidbody — это основной компонент, подключающий физическое поведение для объектов.

Установим для свойства «Gravity Scale» значение 0, чтобы игрок мог свободно ходить по экрану и не падать вниз из-за действия гравитации.

Затем добавим ещё один компонент — Box Collider 2D. Этим элементом игрок будет сталкиваться с окружающими объектами. Изменим его размер под игрока (Рис. 0).

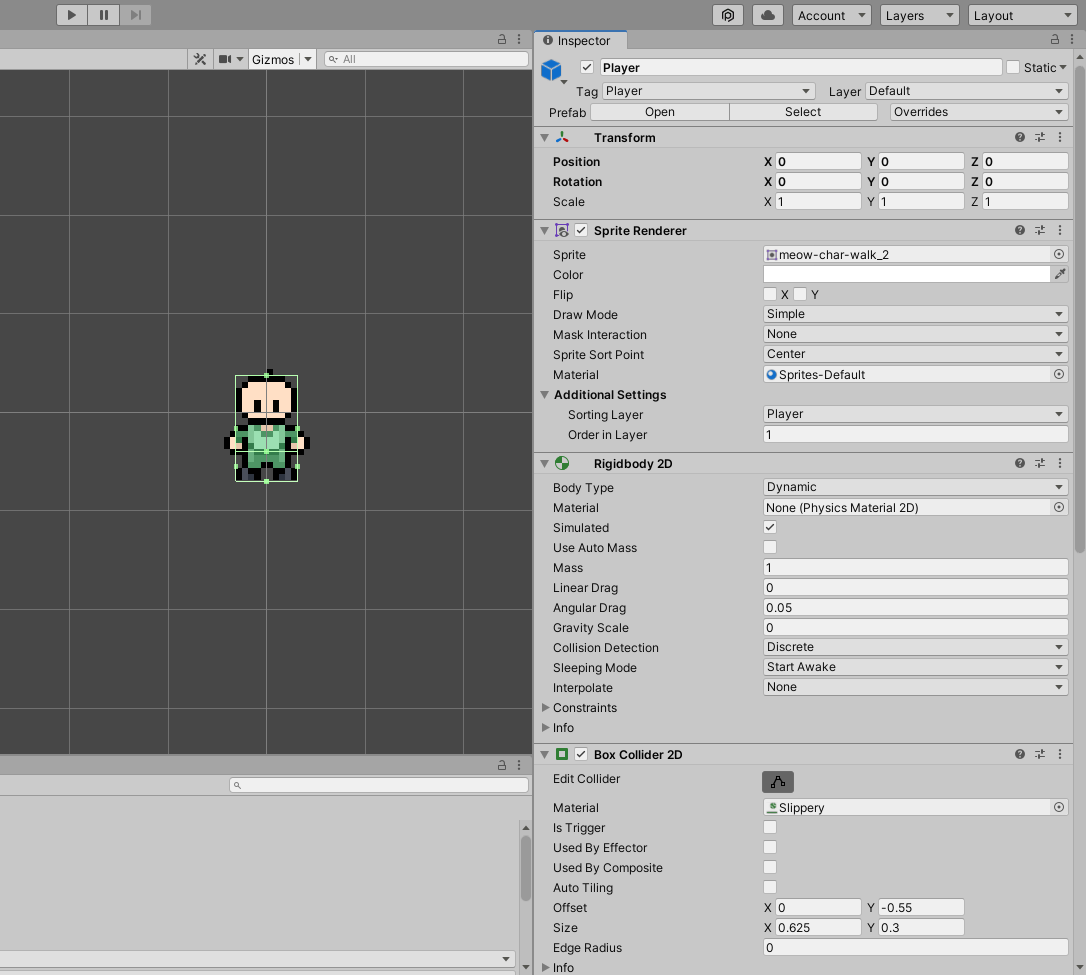


Рисунок 0. Название

Далее создадим скрипт C# и напишем в нем следующий код:

public float speed;

private Rigidbody2D myRigidbody;

private Vector3 change;

// Start is called before the first frame update

void Start()

{

myRigidbody = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update()

{

change = Vector3.zero;

change.x = Input.GetAxisRaw("Horizontal");

change.y = Input.GetAxisRaw("Vertical");

if (change != Vector3.zero)

{

MoveCharacter();

}

}

void MoveCharacter()

{

myRigidbody.MovePosition(transform.position + change \* speed \* Time.fixedDeltaTime);

}

Теперь мы можем перемещать игрока по экрану с помощью клавиш.

Реализуем анимацию ходьбы.

Для этого добавим игроку компонент Animator и создадим анимации для всех 4х направлений с использованием загруженных ранее спрайтов (Рис. 0-0).



Рисунок 0. Название



Рисунок 0. Название

Перейдем во вкладку «Animator» и отредактируем Дерево смешивания.

Деревья смешивания (Blend Trees) — используются для плавного смешивания нескольких анимаций путём объединения их всех в разной степени.

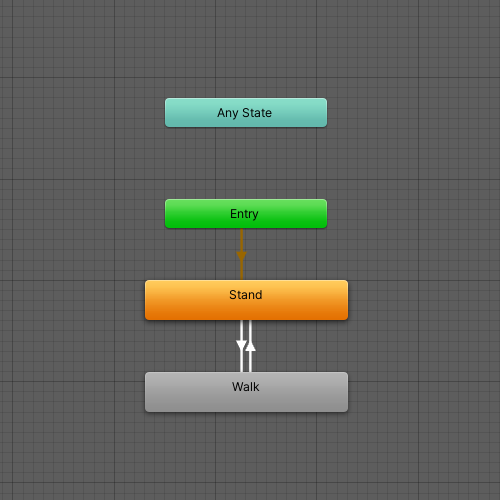


Рисунок 0. Название

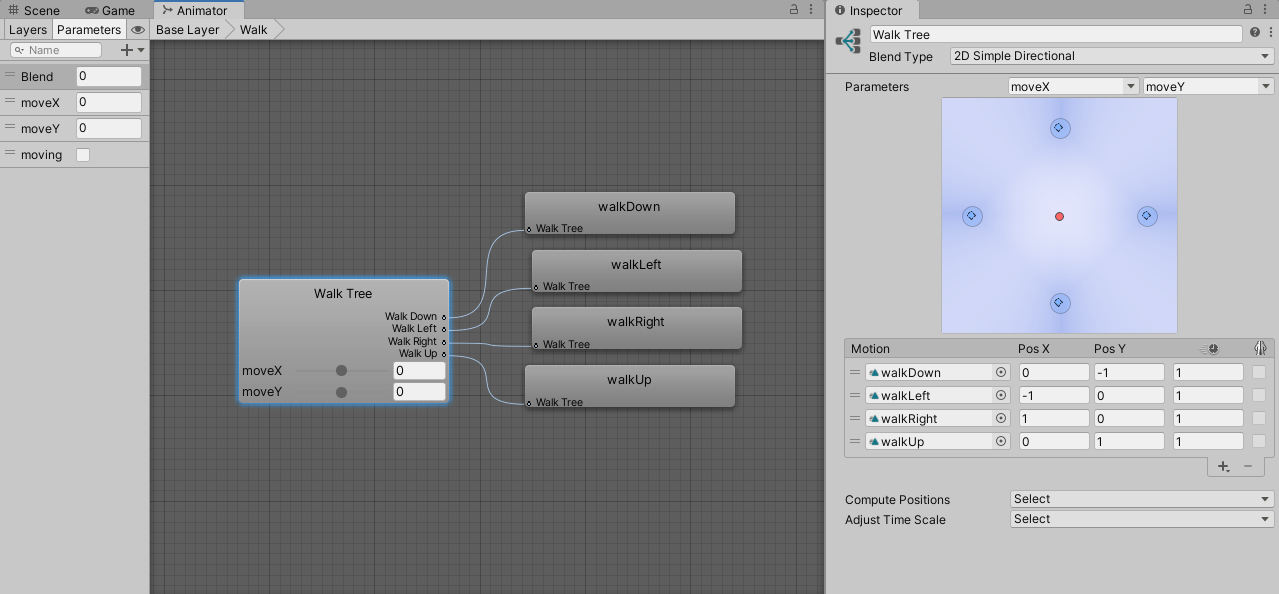


Рисунок 0. Название

Дополним наш скрипт:

public float speed;

private Rigidbody2D myRigidbody;

private Vector3 change;

// Start is called before the first frame update

void Start()

{

animator = GetComponent<Animator>();

myRigidbody = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update()

{

change = Vector3.zero;

change.x = Input.GetAxisRaw("Horizontal");

change.y = Input.GetAxisRaw("Vertical");

UpdateAnimationAndMove();

}

void UpdateAnimationAndMove()

{

if (change != Vector3.zero)

{

MoveCharacter();

animator.SetFloat("moveX", change.x);

animator.SetFloat("moveY", change.y);

animator.SetBool("moving", true);

}

else

animator.SetBool("moving", false);

}

}

void MoveCharacter()

{

myRigidbody.MovePosition(transform.position + change \* speed \* Time.fixedDeltaTime);

}

**Создание в игре плиточной карты (Tilemap).**

1. Добавим в игру лист спрайтов текстур для карты.

Спрайт (англ. sprite — «фея, эльф») — графический объект в компьютерной графике. Чаще всего — растровое изображение, которое можно отобразить на экране.

Лист спрайтов — мелкие графические элементы, упакованные в одно большое изображение.

Карта тайлов — это сетка тайлов, используемая для создания макета игры.

Тайловая карта (Tilemap) — поле игры, состоящее из небольших плиток (тайлов), представляющих собой сетку (tileset), которая состоит из спрайтов. Как правило, один спрайт = тайл, являющийся частью листа спрайтов (spritesheet).

Это набор графических элементов в сетке (вид травы, земли) из которых будет формироваться поле игры.

Перед созданием любого спрайта, нам нужна готовая текстура. Наиболее распространённый способ загрузки текстуры — это загрузка из файла изображения лежащего на диске.

2. создание палитры тайлов (Tile Palette); создание карты с помощью палитры. Описать тайлы

Тайл – это фрагмент 2D-изображения, например, спрайта или фона.

3. добавление коллизии (collision) для объектов, с которыми игрок может сталкиваться. Описать физичность

Коллизия (от англ. Collision — «столкновение») — соударение либо взаимное пересечение границ или плоскостей двух (или более) объектов.

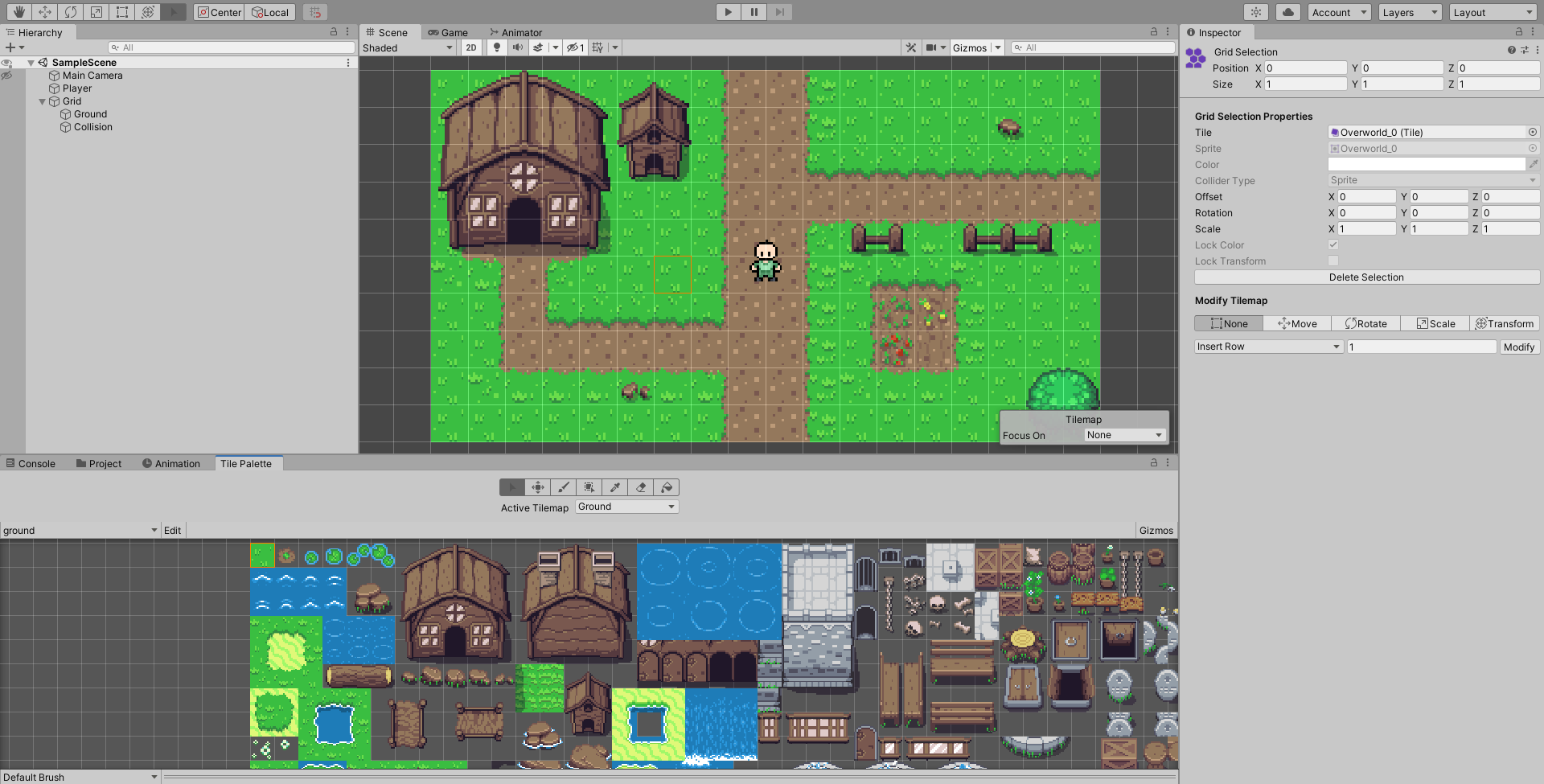


Рисунок 0. Название

**Надписи-пояснения**

Анимация атаки и хитбоксы.

**Объекты, которые можно сломать (горшки)**

Рисунок 0. Название

Для начала создадим анимацию разрушения горшка в редакторе анимаций Unity.

Затем поместим объект горшка на сцену.

Создадим новый скрипт C# и напишем следующий код:

public class Pot : MonoBehaviour

{

private Animator anim;

// Start is called before the first frame update

void Start()

{

anim = GetComponent<Animator>();

}

// Update is called once per frame

void Update()

{

}

public void Smash()

{

anim.SetBool("smash", true);

StartCoroutine(BreakCo());

}

IEnumerator BreakCo()

{

yield return new WaitForSeconds(0.3f);

this.gameObject.SetActive(false);

}

}

Возможность победить врага

Если его тыкать палкой, то он исчезнет

Создание новой сцены, переключение сцен (возможность войти в дом)



Рисунок 0. Название

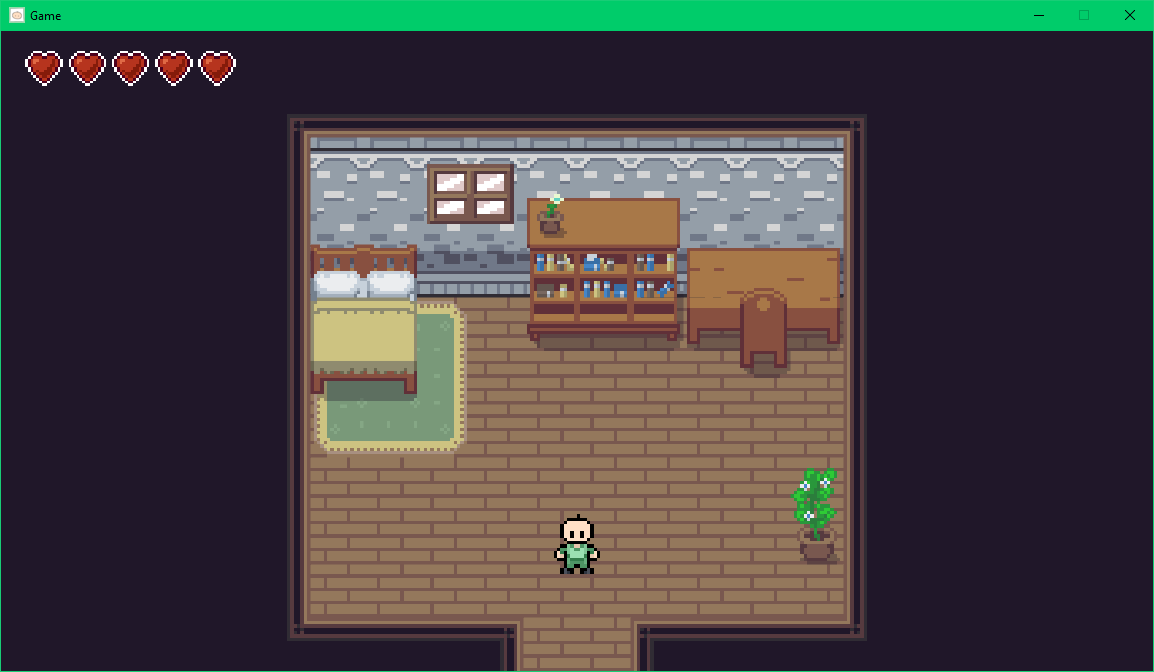


Рисунок 0. Название

Плавный переход между сценами

Некая заставка, которая проигрывается во время переключения сцен

Вопросительный знак над игроком



Рисунок 0. Вопросительный знак над игроком

Добавление сундука с предметом

Рисунок 0. Название

Добавление персонажа (пенёк-патрульный), передвигающегося по заданной дистанции от точки к точке



Рисунок 0. Название

Добавление новой локации (Пещера); дверь, открывающаяся с помощью ключа

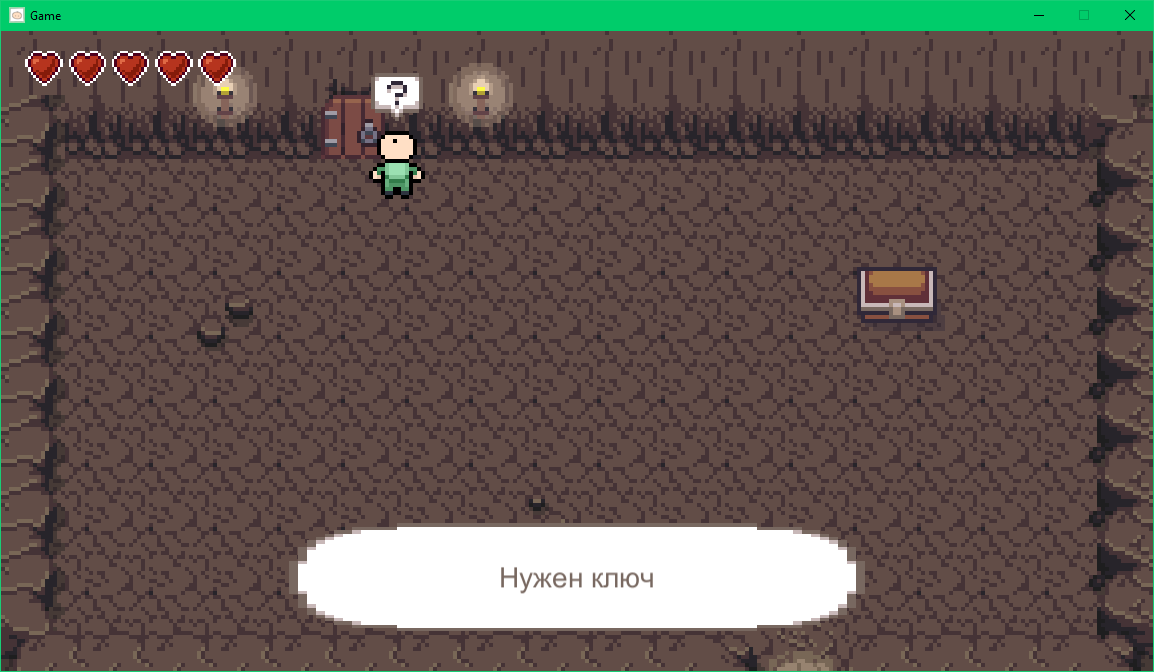


Рисунок 0. Для открытия двери нужен ключ

Добавление эффекта исчезновения врага

Рисунок 0. Эффект исчезновения — угасающий огонь

Добавление сердца (доп. жизнь), которое можно подобрать



Рисунок 0. Дополнительная жизнь — сердце



Рисунок 0. Восполнение жизни на шкале

Выводы

По окончании учебной практики были решены следующие задачи:

* Описана предметная область.
* Изучены программные средства.
* Спроектирована ER-диаграмма.
* Спроектирована диаграмма классов UML (class diagram).
* Спроектирована диаграмма деятельности UML (activity diagram).

Вид профессиональной деятельности был освоен, а также были освоены профессиональные компетенции:

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

Список литературы

1. GitHub репозиторий — URL https://github.com/MeoW-2020-2/Kursovaya
2. Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn — URL https://learn.unity.com/tutorials
3. Make a game like Legend of Zelda using Unity and C# - YouTube — URL https://www.youtube.com/playlist?list=PL4vbr3u7UKWp0iM1WIfRjCDTI03u43Zfu
4. Unity - Руководство: Руководство Unity — URL https://docs.unity3d.com/ru/530/Manual/UnityManual.html
5. Начало работы с Unity — загрузите и установите Unity Hub, а затем установите редактор — URL https://unity.com/ru/download