# Слайд 1

Тема моей дипломной работы – **Создание игрового приложения под Android с использованием gameEngine**. Работу выполнил студент 4 курса 2 группы Суражев Кирилл. Научный руководитель – Барвенов Сергей Александрович

# Слайд 2

* Актуальность работы заключается в том, что, разрабатывая новую игру приходится использовать знания из многих разделов и программирования, и дизайна. Также игры можно монетизировать. Например, можно внедрить внутриигровые покупки, рекламу, разместить игру в магазине и т.д. Разработчики игр могут создавать свои миры, людей, флору и фауну, различных существ и связь всего этого друг с другом. И эти творения способны развлекать, развивать и обладают силой затрагивать человеческие сердца.

# Слайд 3

Целью данной дипломной работы разработать игру под Android с использованием gameEngine.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* Проанализировать и выбрать gameEngine для разработки игры
* Проанализировать и выбрать жанр разрабатываемой игры
* Проанализировать особенности выбранного жанра и классических представителей этого жанра.
* Провести исследования и дать характеристику целевой аудитории игр в выбранном жанре
* Разработать игру в выбранном жанре
* Провести тестирование игры

# Слайд 4

Теперь я расскажу про используемое программное обеспечение

Для создания всех спрайтов в игре была использована программа Adobe Photoshop 2020, а также скрипт Export Layers To Files для удобного превращения файлов в спрайты

# Слайд 5

Unity 2D представляет собой Кроссплатформенный игровой движок, способный рассчитать физические явления, имеет простой и удобный интерфейс, даёт возможность писать код для игры на C#, JavaScript и Boo и идеален для разработки 2D игр

# Слайд 6

Для каждого уровня второй игры (меню, первый и второй) была создана отдельная музыкальная тема (с помощью программы FL Studio 20)

В целом, процесс создания семплов в FL Studio 20 можно описать следующим образом: берутся разные простые семплы из других источников из Интернета (например, с сайта FreeSound) или создаются сами, потом эти семплы редактируются програмно, помещаются на разные дорожки и получившееся сливается в один семпл.

# Слайд 7

* Для работы с C# скриптами была использована программа Visual Studio 2019

# Слайд 8

Игровой движок (game engine) — программное обеспечение, предназначенное для разработки игр на различных платформах

В данной работе будет рассмотрено 3 наиболее распространенных игровых движка для разработки мобильных игр: Unity 2018, Unreal Engine 4, CryEngine V.

* Unity 2020

кроссплатформенный движок для разработки игр, разработанный компанией Unity Technologies [2]. Имеет огромное количество поддерживаемых платформ (ПК, приставки, мобильные ОС, VR платформы и др.). Можно отметить довольно низкие системные требования данного движка.

* CryEngine 5

Этот игровой движок делает большой упор на визуальную составляющую проекта, поэтому имеет большое количество функций и инструментов для достижения превосходного качества изображения. Кроме того, в движке реализовано немало инструментов для работы с физикой объектов.

* Unreal Engine 4

игровой движок, разработанный компанией Epic Games [4]. Обладает широким спектром поддерживаемых платформ, хотя и не настолько широким, как у Unity.

Unreal Engine также имеет широкую поддержку разработки мультиплеерных игр. Данный движок поставляется с масштабируемой и проверенной архитектурой клиент/сервер.

Данный движок подходит прежде всего для разработки трехмерных игр с упором на мультиплеер.

В заключение можно сказать, что все 3 рассмотренных движка являются неплохими инструментами для создания игр под Android инди-разработчиками, при этом каждый имеет свои отличительные черты, которые смогут помочь каждому разработчику выбрать именно тот, что лучше всего подойдет для реализации его проекта. В данной работе был выбран движок Unity, т.к. он проще остальных в освоении, а с точки зрения необходимых для дальнейшей работы встроенных в движок функций подходит идеально.

# Слайд 9

1.Action

Этот жанр означает использование виртуального пространства, в котором есть мишень и стрелок. Может быть человек, например, и какое—либо транспортное средство (танк, космический корабль, самолет). Пространство игры или двухмерное, или трехмерное. Основная цель «экшена» – поразить как можно большее количество мишеней. В качестве примера: Left 4 Dead, третий Doom, четвертый Devil May Cry.

2.MMORPG

Речь идет о многопользовательском игровом жанре, в котором предусмотрен отдельный виртуальный мир. В этом пространстве приходится уживаться большому количеству различных персонажей. Чаще всего, речь идет о мире фэнтези. Здесь, как известно, можно встретить кого угодно – фантастические персонажи. У каждого из них есть определенные способности, собственная цель, а также роль в игре. Игрок имеет возможность стать тем, кто ему больше по душе.

Сравнивая со стандартной RPG, геймплей MMORPG не исчезает никуда (не имеет значения, находится пользователь в игре или нет). В качестве примера: World Of Tanks, World of Warcraft, второй LineAge.

3.RPG

Категория ролевых игр может иметь как развлекательные, так и учебные разработки. Игра начинается в придуманном мире. В основе может использоваться сюжет художественного вида или от начала до конца придуманный разработчиками игры. В данной вымышленной реальности геймеру, как правило (но не всегда), приходится решать какую—нибудь определенную задачу. Во время игры необходимо достичь определенных навыков. В виртуальной реальности можно увидеть адекватную реакцию на различные действия, а также поступки. Для примера: второй Mass Effect, King’s Bounty, четвертый Fallout.

4.Аркады

Достаточно примитивный жанр игры, несмотря на это, очень популярной в наши дни. «Перекочевал» на компьютеры с различных игровых автоматов, а также консолей. Аркадными бывают гонки, файтинги и еще большое количество разновидностей игр. Суть обычной аркады основывается на том, что геймеру необходимо пройти определенный уровень за минимальный период времени. Во время игры приходится преодолевать различные препятствия, а также собирать бонусы. В качестве примера: Super Mario, Super Meat Boy, Pac—Man.

5.Головоломки

О смысле этой игры можно легко догадаться из названия. Геймеру нужно разобраться с какой—нибудь непростой задачей. К примеру, речь может идти о составлении предмета или выбора из большого количества вариантов одного правильного решения. Также возможны и другие вариации.

Известно, головоломки являются самым старым видом игр на Земле. В древнюю китайскую игру «Танграм» коротали время свыше 3000 лет назад. В качестве примера: Crayon Physics Deluxe, World of Goo.

6.Гонки

Смысл этого жанра понятен каждому геймеру. Игрок должен участвовать в различных гонках. Из возможных вариантов могут быть соревнования «Формулы 1», турниры стритрейсеров, а также демонстрация мастер—класса на более сложных транспортных средствах – космические корабли или самолеты. Гонки могут быть как трехмерными, так и двухмерными.

7.Приключения

Данный вид игр отличается наличием специального литературного сюжета. Именно он используется для создания геймплея. Герой, которым придется стать игроку, будет жить в данном мире. Во время игры он будет решать большое количество различных интеллектуальных задач, в том числе и головоломок.

Сравнивая с RPG, в данном случае геймеру не придется случайно взаимодействовать с другими персонажами игры – есть четкий сюжет, за рамки которого нельзя выходить. В качестве примера: Syberia, Fahrenheit, The Longest Journey.

8.Симуляторы

Речь идет о полноценной и очень точной имитации управления определенным объектом (может быть машина, танк или самолет). Сначала симуляторы создавались для военных солдат, чтобы обучать их управлять техническими устройствами. Однако прогресс стал причиной того, что обучающее программное обеспечение заняло свое отдельное место в индустрии игровых разработок.

Сравнивая с гонками, в симуляторах реализовано более точное управление и нет необходимости соревноваться. В качестве примера: Trainz, пятый Silent Hunter.

9.Спортивные

Данная категория игр представляет собой соревнования различных команд. В основном, речь идет о футбольном, баскетбольном, хоккейном или теннисном матчах. Геймерам придется взять шефство над командой: формировать ее состав и привести к победе. В качества примера: серия PES и серия Fifa.

10.Стратегии

Наверное, мы сейчас говорим о наиболее интеллектуальном виде игровых разработок. С самого начала геймер получает возможность создать собственный мир или, в крайнем случае, государство, где придется не только жить, а и управлять. Это не так легко и приятно, как можно подумать сначала. Необходимо детально продумать инфраструктуру, тонко организовать все планы, создать для жителей подходящие условия проживания, дать требуемое количество рабочих мест. В этом деле будут мешать жадные правители других государств – им так и хочется заполучить вашу территорию, поэтому ее нужно будет защищать.

В качестве примера: третий Warcraft, Age of Empires III.

# Слайд 10

Если обобщить, все аркады имеют несколько общих черт, по которым их можно легко узнать. Это, в частности:

* упрощенный, но интенсивный процесс;
* упрощенная графика;
* отсутствие явно прослеживаемого сюжета;
* бесконечность (в игре может быть огромное число уровней);
* возможность набрать очки за выполнение различных действий;
* небольшой промежуток времени, необходимый для их прохождения;
* наличие нескольких жизней, чтобы была возможность пройти один и тот же уровень несколько раз.

# Слайд 11

Примеры наиболее известных аркад сейчас на экране, это:

* **Jetpack Joyride**
* **Ski Safari**
* **Subway Surfers**

# Слайд 12

Также был проведён анализ целевой аудитории. При проведении опроса было опрошено 1000 человек, среди них 500 человек женского пола и 500 человек мужского пола.

[Оглашаются результаты]

# Слайд 13

Платформеры стали популярными в 1980-х годах. Тогда же были сформулированы основные принципы этих игр. Они сводятся к следующему:

* персонаж перемещается по уровням, причём по мере продвижения прохождение вперёд становится всё более сложным;
* у уровней, как правило, есть много секретов для завлечения игроков, любящих исследовать миры.
* время от времени персонаж может получать новые способности, позволяющие проходить туда, куда раньше было не попасть (также является элементом метроидвании)
* для взаимодействий с объектами достаточно к ним прикоснуться
* большое количество разнообразных и интересных с разных точек зрения игроков. Контакт с ними может привести к смерти или же к уменьшению здоровья. Игрок может избегать их, используя бег, прыжки или иные специальные возможности.
* появление ловушек и препятствий, которые нужно обходить или же преодолевать с помощью специальных сил;
* движущиеся платформы, падение с которых приводит к потере здоровья или жизни;
* возможность собирать деньги или другие ресурсы, иногда просто очки.

# Слайд 14

Примеры наиболее известных аркад сейчас на экране, это:

* **Super Meat Boy**
* **Hollow Knight**
* **Ori and The Blind Forest**

# Слайд 15

Также был проведён анализ целевой аудитории. При проведении опроса было опрошено 1000 человек, среди них 500 человек женского пола и 500 человек мужского пола.

[Оглашаются результаты]

# Слайд 16

Ниже будут описаны самые распространённые способы монетизации игр

1. **Разместить игру в Google Play или ином похожем сервисе и сделать её платной.** Это позволит получать прибыль при каждом скачивании, однако игра должна быть высокого качества, чтобы её действительно покупали. В рамках выполнения дипломной работы была произведена попытка разместить созданные игры как независимый разработчик, однако это требует разового взноса в 25$ (оформление подписки разработчика), что является неприемлемой тратой для студента, выполняющего эту работу.
2. **Реклама.** Является, пожалуй, самым известным и распространённым видом монетизации среди инди-разработчиков. Если обобщить – при определённых условиях выскакивает реклама, которую можно или нельзя пропустить. Реклама уместна во многих играх, однако, если её будет слишком много, игроки уйдут из игры. Для её внедрения и корректной также необходима подписка разработчика, также не представляется возможной реализация.
3. **Продажа внутриигровых предметов за реальные деньги (или же – донат).** Это могут быть как косметические предметы (например, скин на персонажа или оружие), так и сущности, дающие игрокам новые игровые возможности (например, новый играбельный персонаж с уникальными способностями, или какой-нибудь крайне полезный предмет). Очень популярная форма монетизации в играх, распространяющихся по модели Free To Play (условно-бесплатные игры). Однако, очень важно сделать платные предметы действительно желанными для игрока, не навязывать это и сделать необязательным для прохождения основного контента. Внедряется, как правило, опытными разработчиками в составе как минимум небольших студий.

# Слайд 17

* Задачи, которые были реализованы в игре:
* Реализовать главное меню с названием игры (Flappy Bird Simulator) и с кнопками Play (при её нажатии игра начинается) и Quit (при её нажатии происходит выход из игры)
* Реализовать игровой уровень, а именно:
  + Нижнюю границу уровня, пол (когда птичка касается её, она умирает)
  + Движение птицы (прыжок, гравитация, движение вправо)
  + Столкновение объектов друг с другом (трубы и пол не сталкиваются друг с другом, а птичка с трубами или полом – сталкивается)
  + Генерация новых препятствий и удаления старых, а также увеличение сложности по мере продвижения по уровню
  + Появление новых облаков и удаление старых
  + Смерть птички (когда она сталкивается с трубой или полом, она умирает, и появляется меню с надписью Game Over, рекордом и текущим результатом, а также кнопки Retry (при нажатии на неё уровень перезапускается) и Main Menu (при нажатии на неё происходит возврат на главное меню)

# Слайд 18

* Системные требования:

Игра запустится на любом современном компьютере или Android-е

* Интерфейс пользователя

В главном меню игры есть два пункта: Play и Quit. Play позволяет начать игру, Quit – выйти из неё. После нажатия на кнопку Start игра сразу же начинается. В игре вы можете нажать Space, чтобы прыгнуть, и Esc, чтобы выйти в главное меню. Также в игре есть интерфейс, показывающий результаты и текущий рекорд (он сохраняется при выходе из игры)

Также в игре есть интерфейсное меню, появляющееся после смерти персонажа с вариантами:

1)Retry (перезапуск уровня)

2)Main menu (вернуться на главное меню)

# Слайд 19

На этом слайде мы видим саму игру и её основные элементы

[описываю элементы]

# Слайд 20

Для того, чтобы создать игровой проект, необходимо: находясь в главном окне Unity, создать новый проект, назвать его Flappy Bird (название игры), из Templates выбрать 2D, выбрать местоположение и нажать Create (см. Рисунок 2.1)

Далее был создан объект GameHandler и к нему был привязан скрипт GameHandler.cs (cм. Приложение А), отвечающий за отладочную загрузку в начале

Далее была выполнены импорт и настройка текстур объектов, использующихся в игре (см. Рисунок 2.2)

Также на GameScene был добавлен спрайт птицы (см. Рисунок 2.3)

Далее был создан скрипт GameAssets.cs (см. Приложение А) для хранения всех объектов для лёгкого доступа, привязанный к новому пустому объекту GameAssets

2.3 Управление птицей, её физика, анимация

Был создан и привязан к объекту Bird скрипт управления и физики птицы Bird.cs (см. Приложение 3), отвечающий за перемещение птицы, физику и логику столкновений (если птица сталкивается с полом или препятствием, то появляется интерфейс Game Over, с результатом и меню с вариантами: 1) Restart, 2) Return to the main menu:

Также был разработан скрипт SpriteAnimator.cs (см. Приложение А), отвечающий за анимацию птицы

2.4 Генерация уровня, сложность

Был создан объект Level в GameScene и к нему был привязан скрипт Level.cs (см. Приложение А), отвечающий за генерацию новых препятствий, облаков в поле зрения камеры, а также удаление тех, что находятся вне поля зрения камеры

# Слайд 21

2.5 Интерфейс в игре

В редакторе Unity в GameScene были созданы следующие объекты (см. рисунки 2.4, 2.5, 2.6)

На объект ScoreWindow был наложен скрипт ScoreWindow.cs (см. Приложение А), который отвечает за изменение числа очков вверху экрана, а если оно становится выше, чем HIGHSCORE, то HIGHSCORE меняется после смерти персонажа на это число.

Также был создан файл Score.cs (см. Приложение А), который содержит основную логику ScoreWindow.cs

# Слайд 22

2.6 Экран загрузки

Была создана сцена Loading (для того, чтобы было видно, когда игра грузится) (см. Рисунок 2.7)

Далее были разработаны скрипты Loader.cs (см. Приложение А) и LoaderUpdate.cs (см. Приложение А), отвечающие за логику экрана загрузки

Скрипт LoaderUpdate.cs был привязан к объекту LoaderUpdate.cs

2.7 Главное меню

Была создана сцена MainMenu (см. Рисунок 2.8)

К объекту MainMenuWindow был привязан скрипт MainMenuWindow.cs (см. Приложение А), отвечающий за логику главного меню (если нажимаешь Play, игра начинается, если Quit, происходит выход из игры)

2.8 Звук

Также был создан скрипт SoundManager.cs (см. Приложение А), который отвечает за весь звук в игре

2.9 Ожидание игрока

Для большего удобства был добавлен экран ожидания игрока и был разработан скрипт WaitingToStartWindow.cs (см. Приложение А) (время в игре как бы остановлено, пока игрок не нажмёт кнопку прыжка

# Слайд 23

Здесь мы можем видеть, как выглядит игра

# Слайд 24

А здесь мы можем видеть, как выглядит диаграмма классов

[тут разные классы, методы, между ними нет связей, т.к. нет иерархии наследования]

Далее я расскажу про другую игру, AboveTale.

# Слайд 25

Задачи:

* Реализовать главное меню с названием игры (Android) и с кнопками New Game (при её нажатии игра начинается) и Quit (при её нажатии происходит выход из игры)
* Реализовать первый уровень, а именно:
  + Нижнюю границу уровня, пол (персонаж может по ней ходить)
  + Движение персонажа (прыжок, гравитация, движение вправо, влево, анимации)
  + Задний фон, представляющий собой набор из нескольких слоёв, перемещающихся вследствие движений игрока (эффект параллакса)
  + Систему инвентаря, добавления и удаления его элементов, а также систему квестов, завязанных на этих механиках (будет специальный NPC, давая предметы которому, можно получить предмет для продвижения вперёд)
  + «Умную» камеру, которая следует за игроком и не выходит за пределы уровня
  + Смену сцен в пределах одного уровня
  + Вторая сцена должна содержать область для перехода на второй уровень
  + Систему диалогов с NPC как альтернативный способ пояснения задачи игроку.
* Реализовать второй уровень, а именно:
  + Небольшой домик изнутри
  + Область для возвращения на первый уровень

# Слайд 26

* 1.Ресурсы разработки

Игра была разработана полностью с нуля с использованием Unity, спрайты были нарисованы с помощью Adobe Photoshop.

* 2.Описание системных требований

Игра запустится на любом современном компьютере или Android-е

* 3. Описание пользовательского интерфейса

В главном меню игры есть два пункта: New Game и Quit. New Game позволяет начать игру, Quit – выйти из неё. После нажатия на кнопку New Game игра сразу же начинается. В игре вы можете использовать джойстик, чтобы двигаться влево-вправо, а также кнопку прыжка, чтобы, очевидно, прыгать, есть кнопка, позволяющая выйти в главное меню. Также в игре есть интерфейс (Кнопка «Рюкзак», при нажатии на которую появляются слоты Инвентаря, каждый и которых показывает предмет, содержащийся в нём, а также под каждым слотом есть кнопка «Выбросить» (см. Рисунок 3.1)

[если попросят показать – на слайд 27]

# Слайд 27

Здесь изображена сама игра и её основные элементы

# Слайд 28

Для создания всех спрайтов в игре была использована программа Adobe Photoshop 2020, а также скрипт Export Layers To Files для удобного превращения файлов в спрайты

Демонстрация части первого уровня (см. Рисунок 3.1):

Задний и передний фон, а также управляемый игроком персонаж, являются объектами, состоящими из нескольких отдельных спрайтов (данная особенность необходима для анимаций и эффекта параллакса).

Объекту Player (см. Рисунок 3.5) были добавлены: объект Rigidbody 2D для реализации гравитации, Capsule Collider 2D для хитбокса. Объекту Grass (см. Рисунок 3.6) также были добавлен хитбокс, чтобы управляемый персонаж не проваливался сквозь пол.

Далее был создан скрипт PlayerController.cs, отвечающий за движение игрока. Следует упомянуть, что у игрока есть возможность прыгать, если объект feetPos, являющийся частью игрока, соприкасается с поверхностью на слое Ground (эта настройка необходима, чтобы игрок прыгал только один раз и не прыгал в воздухе)

# Слайд 29

Для объекта Player были созданы следующие анимации: idle, jump, takeOf, jump, land, run (см. Рисунок 3.7)

В случае объекта Player вся суть анимаций состоит в том, что при их создании за несколько десятков кадров разные части тела игрока перемещаются или изменяются в размере. А благодаря инструменту Animator настраивается логика проигрывания этих анимаций (в каком порядке они проигрываются, при каких условиях, каково время выполнения и т.д.) (см. Рисунок 3.8)

Также для анимации был дополнен скрипт PlayerController.cs. Если обобщить суть, после изменения при прыжке и беге переменные isRunning и takeOf меняют своё значение, и это приводит к последовательной смене анимаций.

# Слайд 30

Далее при разработке возникла следующая проблема: персонаж двигается по локации с анимациями, но камера не следует за ним, а фон выглядит, как однородная статичная картинка, а так быть не должно. Чтобы решить эти проблемы, во-первых, было реализовано движение камеры. Для этого был использован встроенный в Unity модуль Cinemachine. Если обобщить суть, необходимо импортировать модуль, вставить в проект улучшенную камеру из этого модуля, объединить эту камеру с нашей оригинальной с помощью настроек объектов, сделать триггер-хитбокс для всей локации, чтобы камера не выходила за границы уровня, а также написать три скрипта: ParallaxBackground.cs (привязывается к игровому уровню или к его части, в которой находятся слои, которые мы будем перемещать), ParallaxCamera.cs (привязывается к камере), ParallaxLayer.cs (привязывается к перемещаемым слоям) (см. Рисунки 3.9, 3.10, 3.11)

Параллакс-эффект состоит в том, что некоторые слои перемещаются с движениями персонажа, чтобы в игре всё выглядело естественно

# Слайд 31

Одна из главных механик игры – это подбор предметов в инвентарь и использование их для взаимодействия с NPC и другими объектами вроде стен, чтобы продвигаться дальше.

Для этого был реализован инвентарь. Были нарисованы спрайты для объектов интерфейса, связанного с инвентарём. (см. Рисунки 3.12, 3.13, 3.14)

Также был создан объект Canvas, который и хранит в себе весь настоящий интерфейс

Также были реализованы 4 скрипта: Inventory (привязывается к игроку и отвечает за заполнение слота и за то, чтобы в одном слоте был один предмет), Pickup (привязывается к подбираемым предметам и отвечает за их подбор), Slot (привязывается к слотам в UI и отвечает за логику слота), Spawn (ни к чему не привязывается и хранит логику выбрасывания предмета)

Также следует отметить, что все объекты, которые может подбирать игрок, перемещены в специальную папку Prefabs, из которой они могут быть использованы на сцене в любое время со всеми свойствами, в том числе из кода.

# Слайд 32

Вторая важная механика в игре – это система диалогов с NPC (см. Рисунок 3.15)

Для этого, во-первых, были сделаны новые спрайты интерфейса, перенесены в сам интерфейс и заполнены надписями (см. Рисунок 3.16):

Далее были сделаны анимации для этих кнопок: BoxClose, BoxOpen, StartDialogueOpen, StartDialogClose. Если обобщить всю суть, если подойти к NPC, то появляется кнопка «Начать диалог», и появляется диалоговое окно с именем персонажа и тем, что он говорит, и при нажатии на кнопку «Далее» текст последовательно меняется на тот, что мы задали заранее в свойствах объекта StartDialog. (см. Рисунок 3.17)

Были реализованы четыре скрипта: Dialogue.cs (абстракция для DialogueTrigger), DialogueAnimator.cs (анимации диалогового окна, привязан к объекту NPC), DialogueManager.cs (см. Приложение B2) (начало диалога и смена предложений, привязан к объекту DialogueManager), DialogueTrigger.cs (привязывается к объекту StartDialogue и отвечает за появление диалогового окна)

# Слайд 33

В пунктах 3.2 и 3.3 было реализовано движение по нажатию WASD и кнопки Space, что не подходит для управления на Android. Поэтому на UI были реализованы следующие кнопки: (см. Рисунок 3.18)

Для создания джойстика c Asset Store был импортирован объект Joystick.

Также скрипт PlayerController.cs был отредактирован и был подключён к этим кнопкам.

# Слайд 34

Также в игре были реализованы:

1.новая локация, Town (см. Рисунок 3.19)

2. переход между этой и предыдущей локацией. Для этого были созданы объекты Edge1 и Edge2, а также скрипт FrameSwitch.cs (привязан к этим объектам). Если обобщить всю суть, то при касании определённого Edge происходит переход на соответствующую локацию)

3. переход между сценой, на которой находятся две вышеупомянутые локации, и сценой внутри дома. (см. Рисунок 3.20)

Это было реализовано с помощью скриптов LevelChanger.cs (привязан к объекту LevelChanger.cs и отвечает за смену уровней и затемнение/освещение), ButtonTrigger.cs (привязан к кнопке выхода/входа в дом) и объекта PlayerPosition (для сохранения позиции игрока при входе/выходе)

4. сцена для главного меню (см. Рисунок 3.21) c кнопками начала и выхода из неё, а также кнопка возврата на сцену:

# Слайд 35

Для каждого уровня (меню, первый и второй) была создана отдельная музыкальная тема (с помощью программы FL Studio 20) (см. Рисунок 3.22)

В целом, процесс создания семплов в FL Studio 20 можно описать следующим образом: берутся разные простые семплы из других источников из Интернета (например, с сайта FreeSound) или создаются сами, потом эти семплы редактируются програмно, помещаются на разные дорожки и получившееся сливается в один семпл.

# Слайд 36

Здесь изображена диаграмма классов

# Слайд 37

В настоящей дипломной работе были (-а):

* Проанализированы разные gameEngine и выбран конкретный для разработки игр
* Проанализированы разные жанры мобильных игр и выбраны конкретные для разрабатываемых игр
* Проанализированы особенности выбранных жанров и классических представителей этих жанров.
* Проведены исследования и даны характеристики целевой аудитории игр в выбранных жанрах
* Разработаны игры в выбранных жанрах
* Проведено тестирование разработанных игр