МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФГБОУ ВО

Московский авиационный институт

(национальный исследовательский университет)

Кафедра 304

«Вычислительные машины, комплексы, системы и сети»

Пояснительная записка к курсовому проекту

по дисциплине:

«Конструирование ПО»

Тема работы:

«Метание в цель (“миномётный обстрел”) - игра»

Выполнил студент группы М3О-311Б-21:

Давыдов А.П.

Принял преподаватель:

Гагарин А.П.

Москва 2023

Оглавление

Введение. Уточненные требования к программе....3
Область применения....3
Требования к программе....3
Проект программы....4
Основные конструктивные решения....8
Тестирование программы....9
Заключение....9
Приложение 1. Инструкция по применению программы....11
Приложение 2. Текст программы на исходном языке....14

Введение. Уточненные требования к программе

Unity – это очень универсальный двигатель для, по заявлению разработчиков, создания игр. Тем не менее, на Unity вполне возможно сделать любое приложение из-за широчайшего инструментария.

Unity предоставляет от себя работу с физикой, контроль коллизий, удобные окна для взаимодействия с объектами C# и так называемыми GameObject-ами, являющими собой объект в рабочем пространстве, называемом «Сценой».

Область применения

Unity предназначен для создания игровых приложений любого жанра и совершенно различной сложности.

Требования к программе

В любом игровом приложении с технической точки зрения самое главное — это оптимизация проекта. То, сколько у пользователя будет FPS (Кадров в секунду) напрямую влияет на качество.

В рамках проекта следовало сделать систему поиска лучшей позиции стрельбы, логику перемещения персонажей а так же предугадывание полёта снаряда по баллистической траектории.

Проект программы

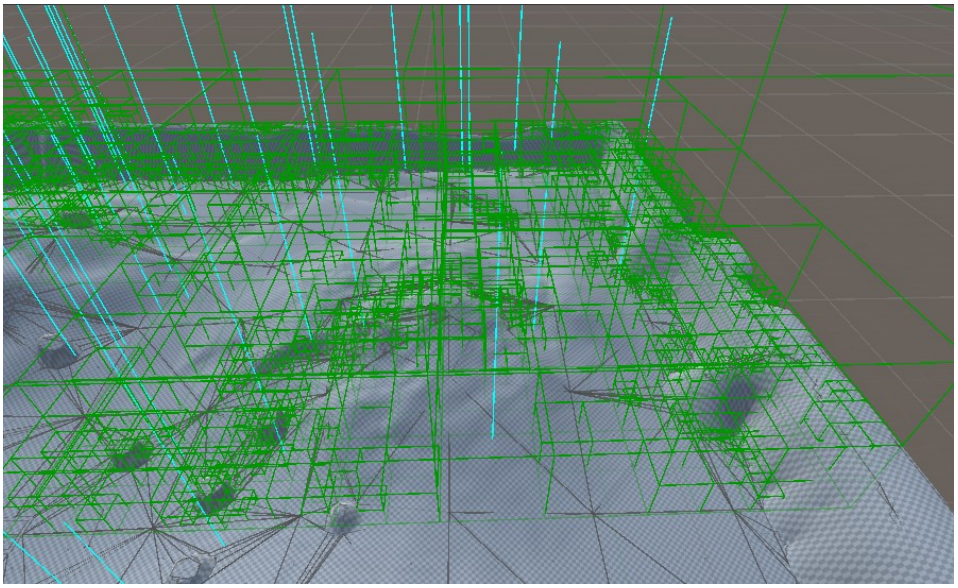
Unity работает в паре с C#, на одном уровне. По этой причине проект состоит из следующих основных элементов:

1) Объекты C#, которые также можно использовать в Unity
2) GameObject – объект Unity, являющийся наследником C#-объекта. Всё, что есть на сцене — это GameObject, он всего создаётся в паре с компонентом Transform.
3) Component — Объект Unity, который можно добавлять к GameObject'у
4) Scene — это совокупность всех GameObject'ов, расставленных в пространстве.

Следующие элементы — это созданные в рамках курсовой работы модификации Unity

5) Мортира представляет собой класс-компонент, при проверке траектории вычисляющий точки предполагаемого полёта одну за другой. Далее от каждой точки до каждой следующей запускается проверка коллизий линией (`Physics.Raycast()`). Если в результате хотя бы одной проверки обнаруживается коллизия — проверяемая клетка отбрасывается. О клетках — далее.
6) Структура данных OctTree – Дерево трёхмерного поиска, используемого для поиска клеток для пользователя мортиры.

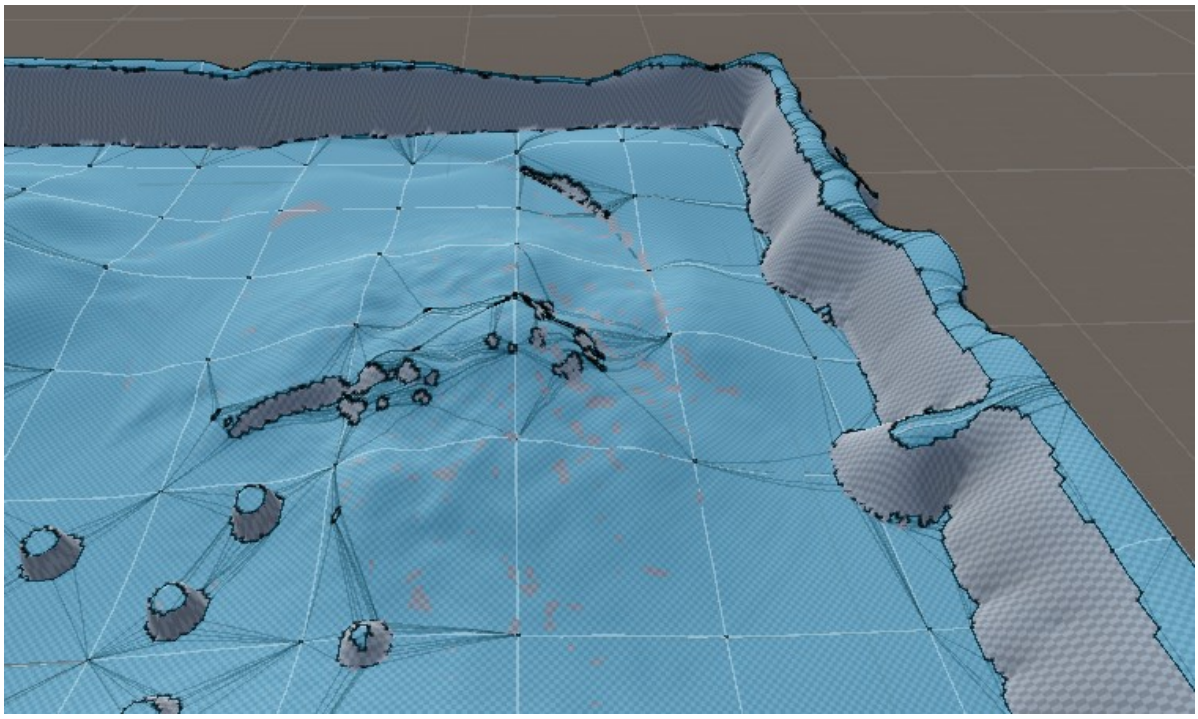
   Представляет собой куб, рекурсивно разбитый на 8 частей. В проекте выглядит вот так:



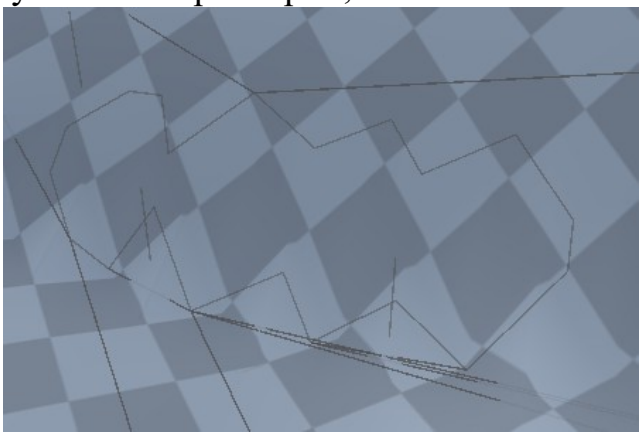
   Поиск начинается с самого большого куба, и далее рекурсивно

продолжается в наследниках до тех пор, пока наследник не окажется Листом Дерева.

7) NavMeshCalculations – Компонент, контролирующий клетки для стрелков, а так же производящий необходимые вычисления поиска наиболее оптимальных маршрутов и позиций для стрельбы.
   Он использует описанный выше OctTree.
   Принцип построения (Инициализации) выглядит следующим образом:
   1) Сначала инициализируемый поверхность движения Агентов Unity



   2) Далее эта поверхность разбивается на треугольники. Следуя указанным размерам, слишком маленькие — объединяются

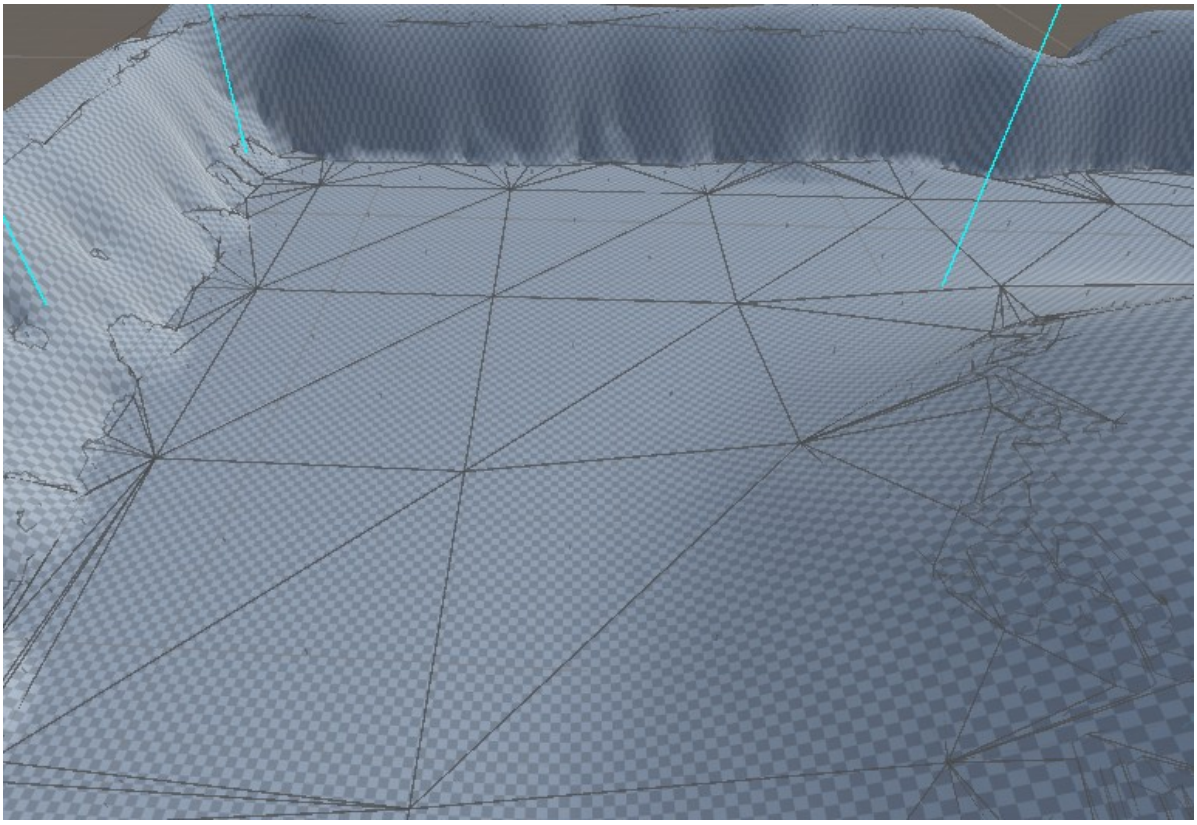


   А слишком большие — разбиваются.

   3) После этого треугольники разбиваются в Кластеры. Они нужны для

целей, выходящих за рамки курсовой работы.

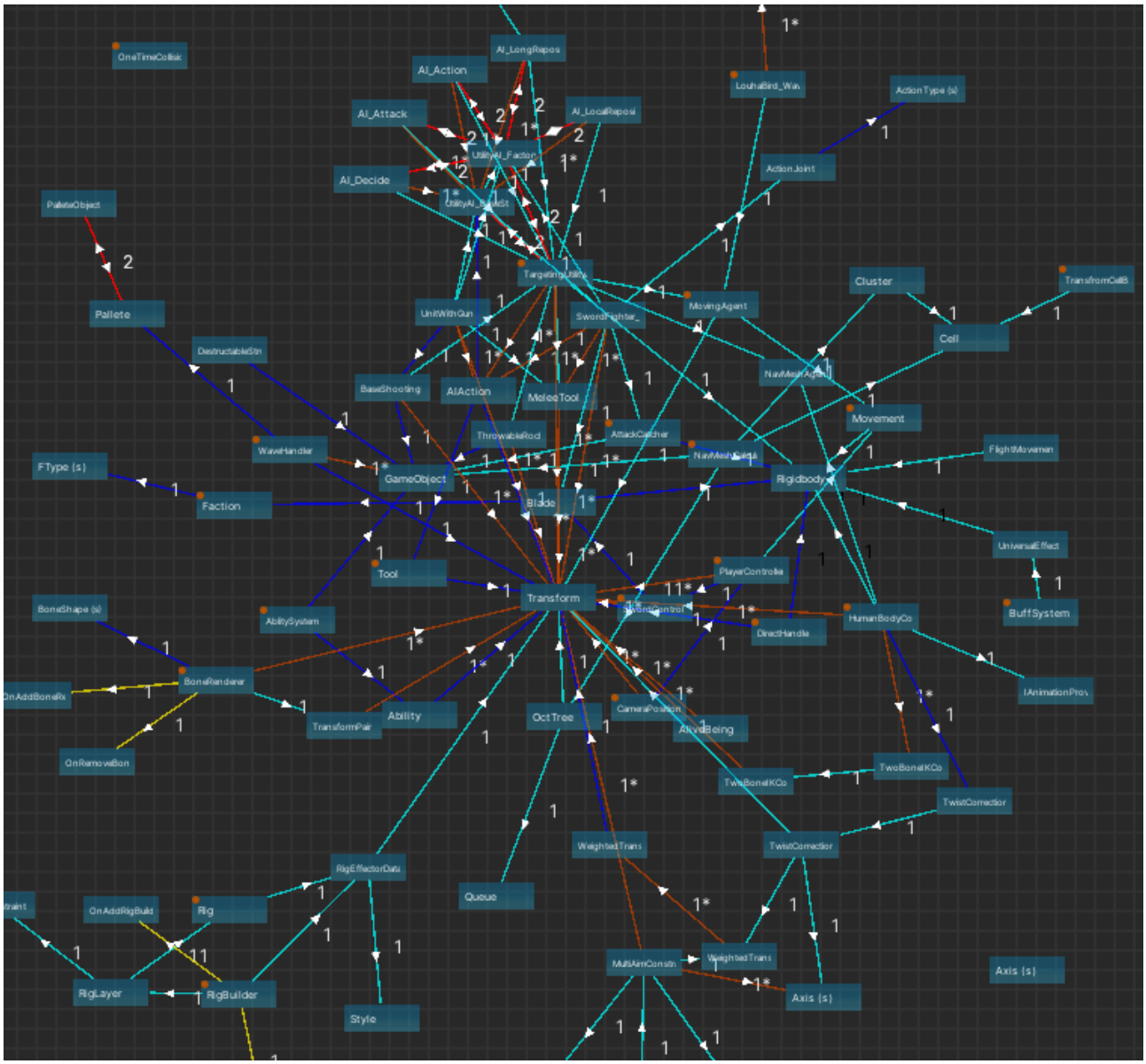
4) В конечном итоге получается сетка следующего вида:



В центре каждого треугольника есть уходящая вверх линия — это обозначение центра.

OctTree позволяет найти ближайшую клетку к цели. Далее от этой клетке по алгоритму Дийкстры (Через движение по соседям) начинаются описанные ранее проверки траектории. Если проверка не срабатывает — то соседи этой клетки не будут добавляться в стек дальнейшего поиска.

Проект целиком (Включая не входящие в курсовую работу элементы) выглядит следующим образом в виде карты функциональной зависимостей объектов:



Структура проекта в рамках курсовой работы:

BaseShooting.cs → ThrowableRocks.cs – Проверка траекторий, часть функционала описана в родителе BaseShooting.

TargetingUtilityAI.cs – Контроль персонажа, управляющего мортирой.

NavMeshCalculations.cs

PlayerController.cs – Контроль вашего персонажа.

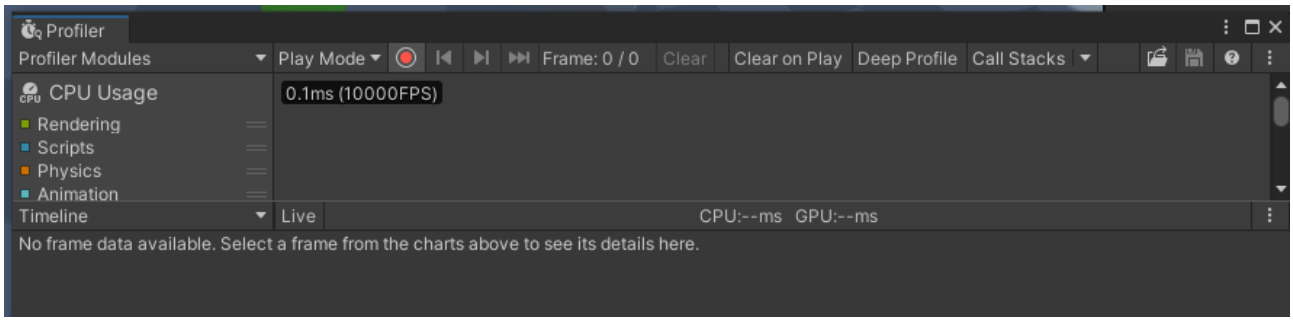
Bullet.cs – Снаряд, выпущенный из мортиры. Наносит урон при коллизии.

Основные конструктивные решения

1. Использованы паттерны проектирования и программирования:
    1. Singleton для менеджеров объектов
    2. Builder для создания объектов
    3. Factory и Abstract Factory
2. Переработка физики Unity для перемещения персонажей, а точнее изменение системы трения и переработка движения по покатым поверхностям
3. Структура Данных OctTree для поиска объектов в трёхмерном пространстве.
4. NavMeshCalculations – это глобальный Singleton, отвечающий за изменение и работу над встроенной в Unity системой поиска пути NavMesh. Конкретно в Unity создаётся сетка для перемещения всех Агентов, а этот класс позволяет сделать эту сетку удобной для выбора точек стрельбы, а так же реализации моих собственных специализированных систем поиска поиска пути, таком как Путь Стрельбы (Выбирается клетка как можно дальше от цели, но как можно ближе к Персонажу-Мортирщику)
5. Throwable Rocks – Это класс-компонент, вычисляющий траекторию стрельбы из всех потенциально подходящих клеток, что предоставил NavMeshCalculations. Отвечает также за саму стрельбу, запуская контролируемый физикой Unity шар-снаряд. Является наследником класса-компонента BaseShooting, в котором прописана проверка коллизий и логика кода достижимости.

Тестирование программы

Для тестирования производительности программы достаточно встроенного в Unity окна Profiler, показывающую нагрузку всех функций за один кадр.



Он отображает количество вызовов, нагрузку на ЦП и количество выделенной памяти.

Для тестирования работоспособности действовать приходится по следующему алгоритму:

- Запустить проект после перезагрузки всех файлов C# и их обновления
- Обнаружить GameObject с отредактированным компонентном
- Убедиться, что более нет никаких других воздействий.
- Выполнить взаимодействие с этим объектом, что включает в себя:
  - Воздействие на него физикой
  - Воздействие на него коллизией
  - Воздействие на него другими компонентами.

Заключение

В рамках курсовой работы была изучена работа физики Unity и создана система предугадывания полёта снаряда относительно его начальной скорости запуска.

Кроме того, была изучена и имплементирована структура данных OctTree, широко применяемая не только для определения клетки, на которой находится использующий мортиру персонаж, но и для нахождения лучшей позицией стрельбы по игроку.

При изучения тем были приобретены следующие навыки:

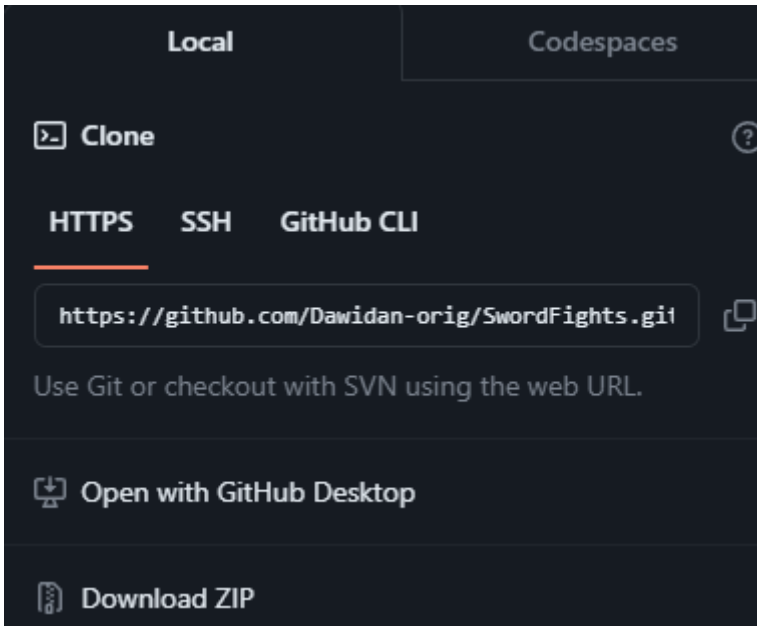
- Физика Unity
- Знание алгоритмов

    - Поиска пути
    - Оптимизации
    - Структур данных
- Паттерны проектирования при разработке игр
- Работа с интерфейсами данных Unity (UnityEditor)
- Визуализация работы алгоритма

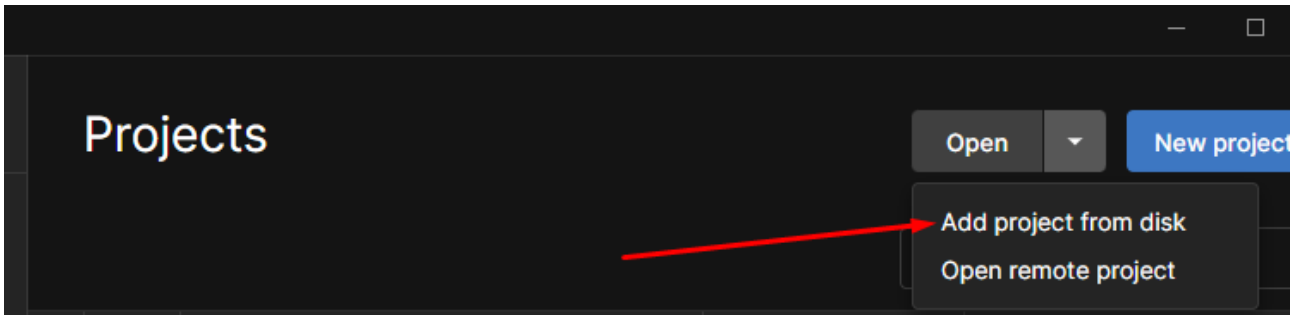
Исходя из полученных знаний и навыков, можно сделать вывод что Unity – это высокоуниверсальная и очень гибкая система разработки приложений, при должном управлении которой можно создавать совершенно любые компьютерные приложения, начиная от систем менеджмента до миров виртуальной реальности.

Приложение 1. Инструкция по применению программы

Чтобы запустить приложение в режиме разработчика (И иметь возможность видеть визуализацию траектории движения), достаточно клонировать проект в нужную папку используя GitHub:

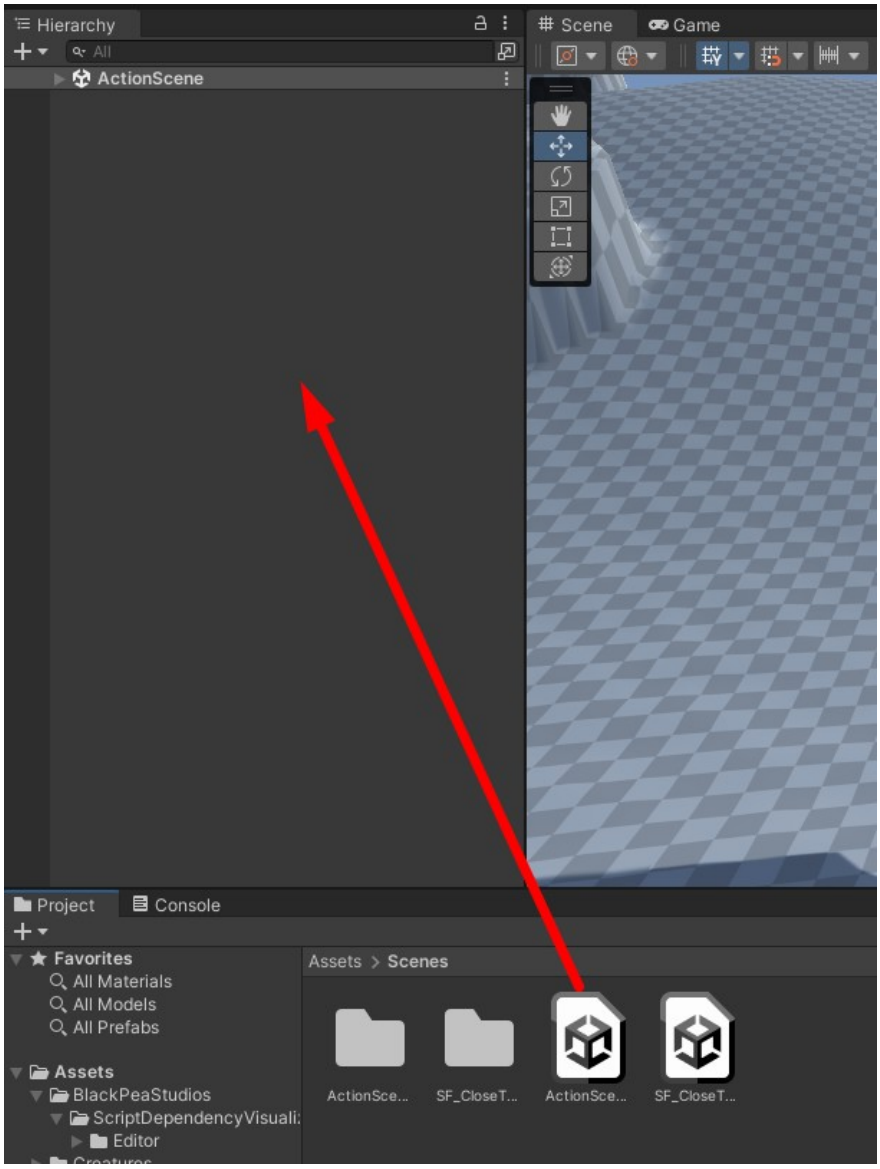


Затем найти эту папку в Unity Hub и открыть проект:



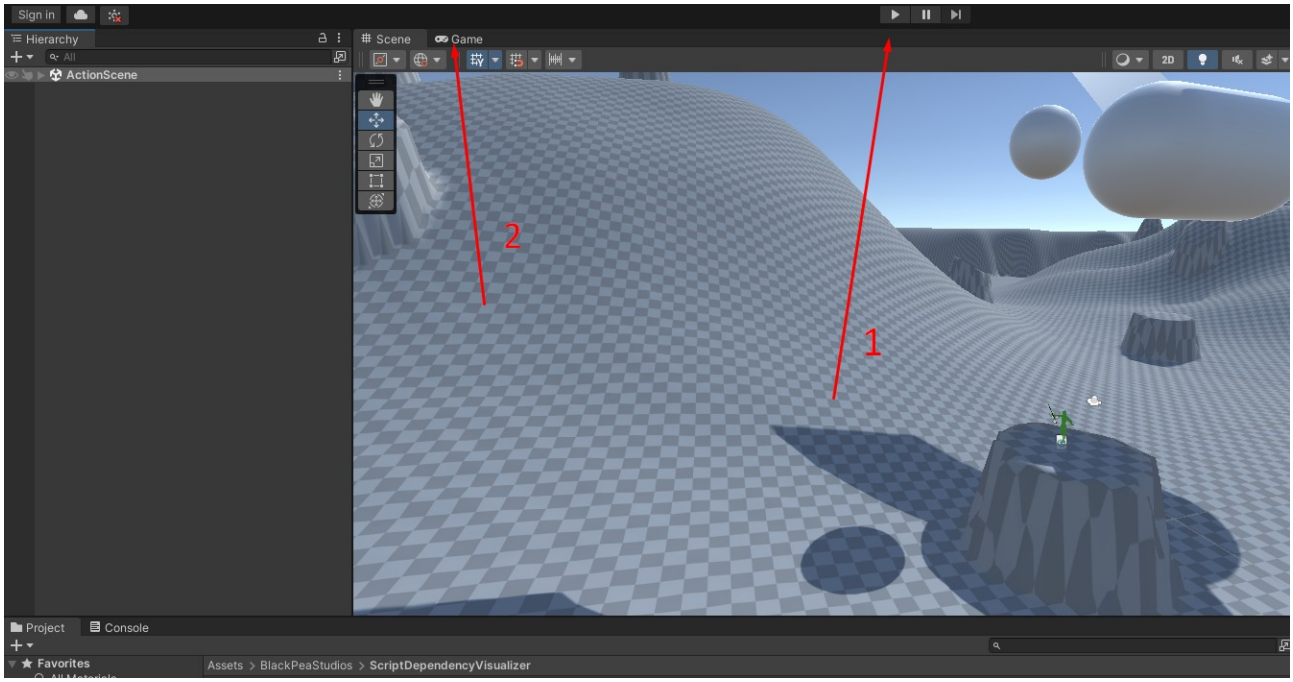
После чего Unity сделает всё сам и откроет проект.

Как только проект откроется, следует убедиться, что сцена присутствует:



Для этого внизу, во вкладке Project Надо найти папку Assets, оттуда перейти в папку Scenes и перенести из этой папки сцену ActionScene.

Дальше нужно нажать кнопку Play (1) и затем перейти во вкладку Game.

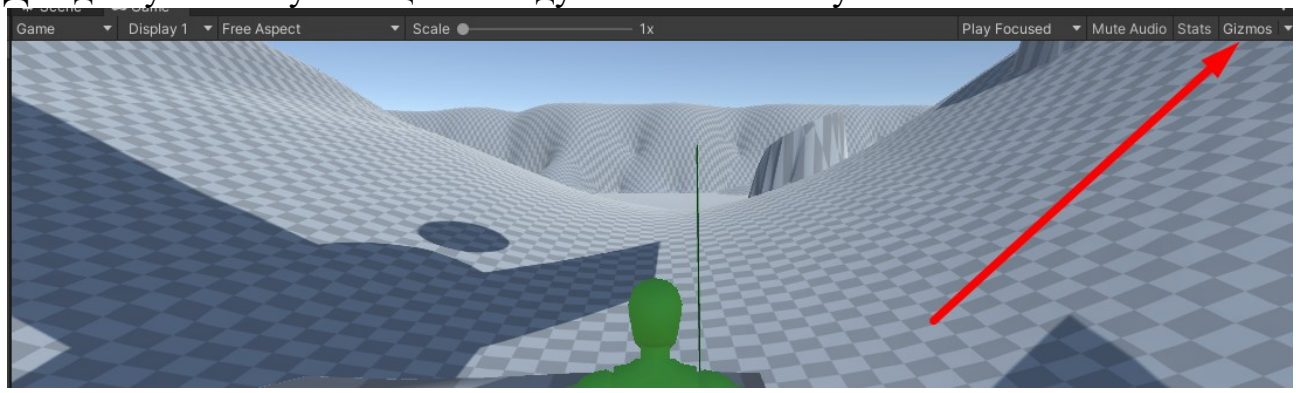


Если приложение вдруг остановилось, и появилось сообщение об ошибке внизу, достаточно нажать кнопку возобновления:

Это происходит из-за того, что данные используется ещё до запуска игры, а после нажатия кнопки эти данные уничтожаются и пересоздаются, так работает Unity. По этой причине в самый первый кадр, когда данные ещё не успели уничтожится, но всё ещё используются вызывается NullReferenceException.

Эта ошибка исправима отключением режима разработки, но тогда визуализации перестанут работать. По этой причине гораздо проще использовать «Возобновление»

Для доступа к визуализациям следует нажать кнопку Gizmos



Управление вашим персонажем происходит с использованием кнопок WASD - Вперёд, Влево, Назад, Вправо соответственно.

Приложение 2. Текст программы на исходном языке

Весь проект с курсовой работой представлен по ссылке:
https://github.com/Dawidan-orig/SwordFights/tree/KPO