**чу Поо «столичный БИЗНЕС КОЛлЕДЖ»**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Курсовая работа**

**По предмету**

**«Технология разработки ПО»**

**На тему**

**«Выбор среды разработки приложения»**

**Выполнила: студентка гр. 1304**

**Алейчик Виктория Леонидовна**

**Проверил: преподаватель  
Киселев Леонид Александрович**

**Москва 2021**

Содержание

[Постановка задачи 3](#_Toc90254805)

[Описание спрайтов 3](#_Toc90254806)

[Схема взаимодействий объектов 6](#_Toc90254807)

[Описание процесса разработки 7](#_Toc90254808)

[Разработка игрового уровня 7](#_Toc90254809)

[Объект «Grid» 7](#_Toc90254810)

[Объект «Player» 7](#_Toc90254811)

[Кристаллы и бонус «Вишенка». 12](#_Toc90254812)

[Объект CanvasPause 15](#_Toc90254813)

[Объект «PORTAL» 18](#_Toc90254814)

[Объект «CanvasFinish» 19](#_Toc90254815)

[Объект «PossumObj» 20](#_Toc90254816)

# Постановка задачи

В игре должны присутствовать следующие элементы: вид на игру сбоку; управление клавишами WASD; пауза(при нажатии кнопки ESC); запись количества собранных кристаллов игроком;количество жизней; враг; бонус «Вишня» который дает выше прыжок; выход(дверь в домике).

# Описание спрайтов

Одним из главных элементов игры является персонаж «Лисенок» (рис. 1); имеет 4 анимации: стоять, прыжок, бег и падение. Анимация «стоять» имеет 4 кадра спрайта (рис. 2), «прыжок» имеет 1кадр спрайт (рис. 3), «бег» имеет 6 кадров спрайтов (рис. 4), «падение» имеет 1 кадр спрайт (рис. 5).

|  |  |  |  |
| --- | --- | --- | --- |
| Рисунок 1. Персонаж «Лисенок» | | | |
|  |  |  |  |

Рисунок 2. анимация «Стоять»

|  |
| --- |
| Рисунок 3. анимация «Прыжок» |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Рисунок 4. анимация «Бег»

|  |
| --- |
| Рисунок 5. анимация «Падение» |

Враг «Опоссум» ходит по заданной траектории «туда-сюда» (рис. 6).

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Рисунок 6. Враг «Опоссум»

Спрайты кристаллов (рис. 7); жизни игрока (рис. 8); Бонус «Вишня» (рис. 9).

|  |  |  |
| --- | --- | --- |
| Рисунок 7. Кристалл | Рисунок 8. Жизнь игрока | Рисунок 9. Бонус «Вишня» |

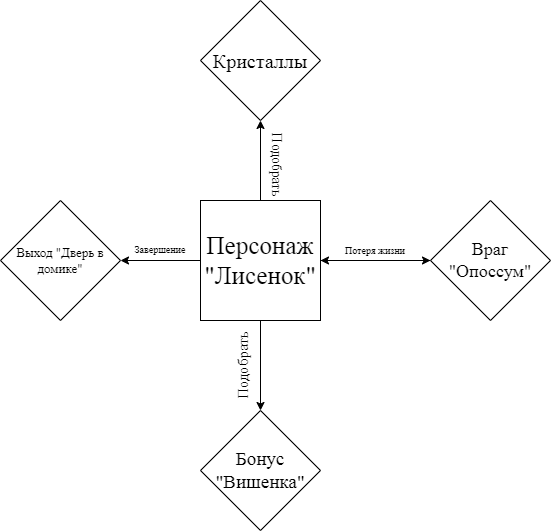
Сцена построена через «TileSet», так же есть отдельные декорации на заднем фоне(рис.10-12).

|  |
| --- |
| Рисунок 10. TileSet Рисунок 11. Декорации |
| Рисунок 12. Сцена с уровнем |

Выход (дверь в домике) (рис. 13)

|  |
| --- |
| Рисунок 13. Выход |

# Схема взаимодействий объектов



Персонаж «Лисенок» подбирает кристаллы и Бонус «Вишенка».

Персонаж «Лисенок» теряет жизнь с объектом «Опоссум» и наоборот.

Персонаж «Лисенок», касаясь «двери в домике» завершает уровень.

# Описание процесса разработки

## Разработка игрового уровня

### Объект «Grid»

Для начало игрового уровня был создан объект «Grid» с объектами «GroundAktiv» и «GroundPasiv». В «GroundAktiv» и «GroundPasiv» создаем компонент «TileMap», вместе с ним создается компонент «TileMap Renderer». В объекте «GroundAktiv» создаем компонент «TileMap Collider 2D», это нужно для того, чтоб персонаж не проваливался сквозь текстуры. Дальше открываем в верхней панели Window – 2D – TilePalette. В Tile Palette добавляем TileSet и рисуем уровень и дополняем все это декорациями (рис. 14).



Рисунок 14. Создание «Grid»

### Объект «Player»

Далее создаем объект «Player» с объектом «isGround», это будущий персонаж «Лисенок» (рис. 15). К объектам добавляем компонент «Capsule Collider 2D». К объекту «Player» добавляем еще «Sprite Renderer»и «Rigidbody 2D». И добавляем скрипты «Controller» и «ComplitedMenu» (рис. 16-19). В скрипте «Controller» описываем движение и взаимодействия с другими объектами персонажа «Лисенок». Создаются переменные (рис. 20). Метод Void Star остается пустым, в методеVoid HeartLoss создаем отнятие жизней персонажу (рис. 21). В методе private void MakeJump создаем прыжок персонажу, с помощью добавления силы. В методе private void PlayerMove создается передвижение персонажа с помощью скорости по осиY. В методеprivate void PlayerFlip создаем повороты по оси Y. В методе void Fixed Update вызываются методы PlayerMove и PlayerFlip. Метод PlayerMove вызывает движение, и если оно больше 0 и персонаж не повернут вправо или персонаж повернут вправо и меньше 0, то вызывается метод PlayerFlip(рис. 22). В методах private void OnTriggerEnter2D и OnTriggerExit2D описываем пересечение коллайдеров с другими объектами (рис. 23). В скрипте ComplitedMenu описываем завершение уровня. Метод Void Star остается пустым. В методе OnTriggerEnter2D описываем пересечение коллайдера и вызова объекта (рис. 24). В методах public void Next и public void Menu создаются переходы в другие сцены (рис. 25).



Рисунок 15. Объект «Player»

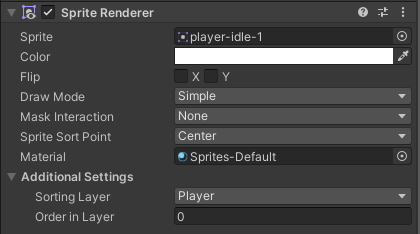


Рисунок 16. Компонент «Sprite Renderer»

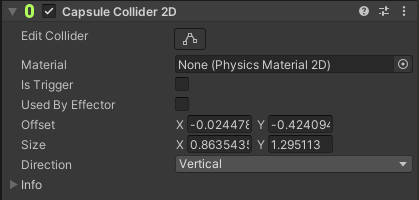


Рисунок 17.Компонент «Capsule Collider 2D»

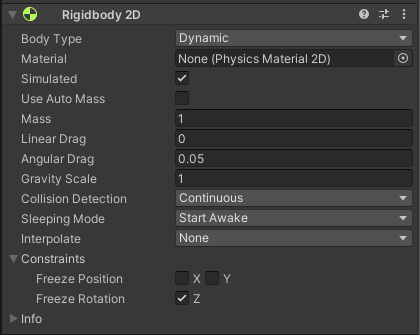


Рисунок 18. Компонент «Rigidbody 2D»

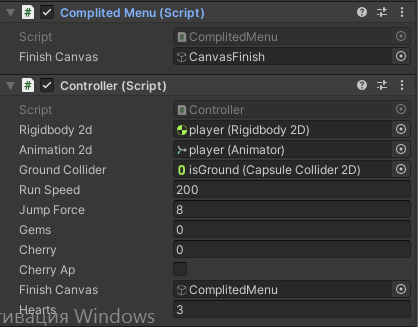


Рисунок 19.Скрипты «Controller» и «ComplitedMenu»

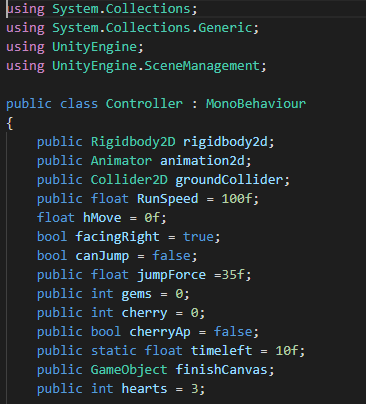


Рисунок 20. Создание переменных.

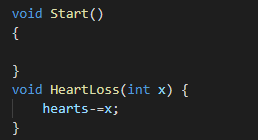


Рисунок 21. Start и HeartLoss



Рисунок 22. Движение персонажа

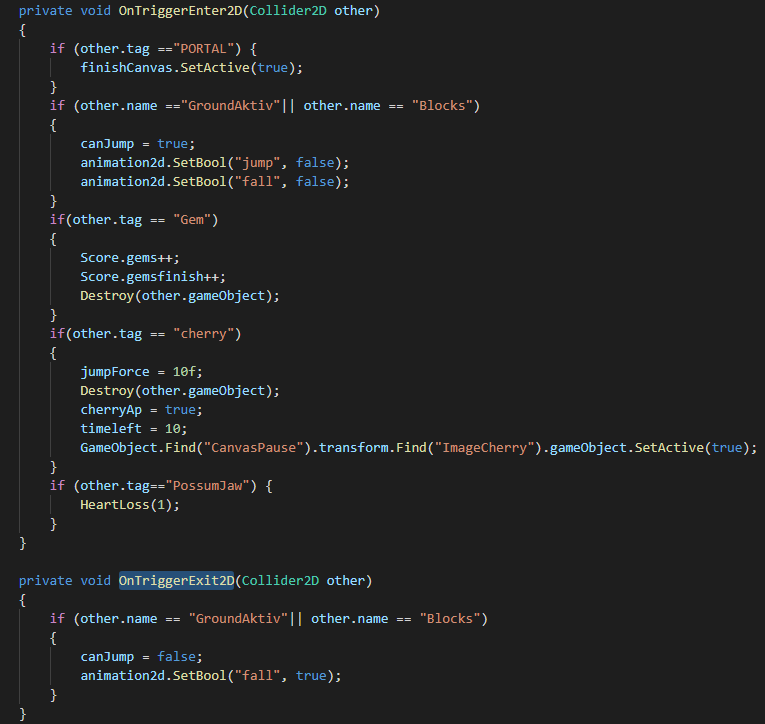


Рисунок 23. Методы OnTriggerEnter2D и OnTriggerExit2D

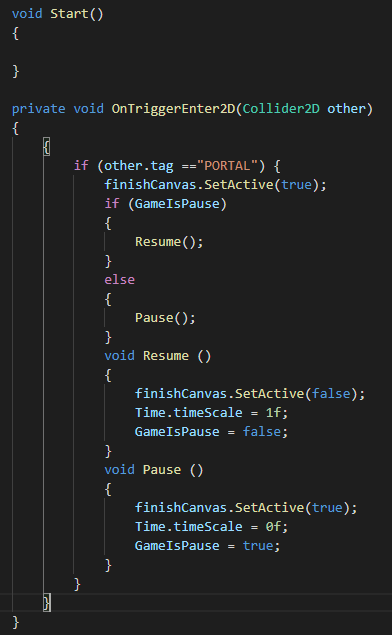


Рисунок 24. Методы Start и OnTriggerEnter2D

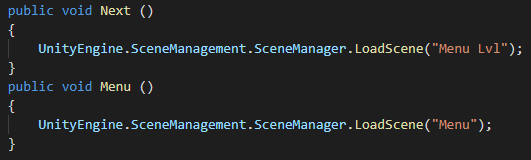


Рисунок 25. Методы Next и Menu

### Кристаллы и бонус «Вишенка».

Далее создаем кристаллы и бонус «Вишенка». Для этого создаем объект «Gems», там будут Prefab кристаллов и «Вишенки». Дальше создаем еще один объект под названием «ScoreGem», тут будет скрипт подсчета кристаллов (рис. 26). К Prefab кристаллов и «Вишенки» добавляем компоненты «Sprite Renderer» и «Circle Collider 2D» (рис. 27-28).

К Prefab «Вишенки» добавляем еще скрипт «CherryCont» (рис. 29).К объекту ScoreGem добавляем скрипт «Score» (рис. 30). В скрипте «CherryCont» записываем метод private void OnDestroy там будет счетчик, сколько вишенок собранно (рис. 31). В скрипте «Score» создаем нулевые переменные для кристаллов и «Вишенки». Далее в методе void Update высчитываем, сколько собрали кристаллов и «Вишенок» и выводим на экран (рис. 32).



Рисунок 26. Создание объекта «Gems»

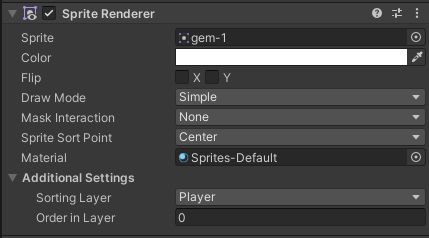


Рисунок 27. Компонент «Sprite Renderer»

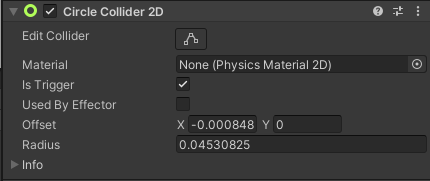


Рисунок 28. Компонент «Circle Collider 2D»



Рисунок 29. Скрипт «CherryCont»

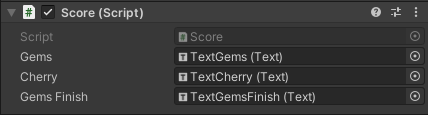


Рисунок 30. Скрипт «Score»

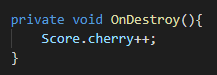


Рисунок 31. Метод OnDestroy

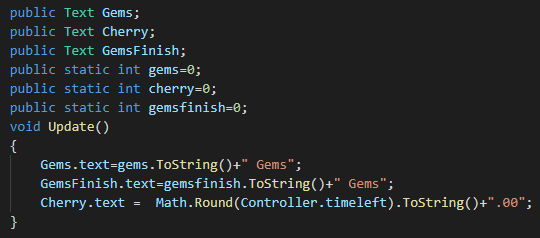


Рисунок 32. Переменные и метод Update

### Объект CanvasPause

Далее создаем паузу в игре и всплывающее окошко при нажатии кнопки «Esc». Создаем объект CanvasPause. В этом объекте создаем еще 4 объекта «imageGem», «imageCherry», «PauseMenu» и «Hp». В «imageGem» и «imageCherry» создаются еще объекты «TextGems» и «TextCherry». В объекте «PauseMenu» создается текст «Pause» и кнопки «Resume» и «Menu». В объекте «Hp» создаем еще 3 объекта «Hp1», «Hp2» и «Hp3» (рис. 33). Далее в объекте «CanvasPause» в компоненте «Canvas» в строчке «Render camera» подставляем камеру из сцены (рис. 34). И добавляем к нему скрипт «PauseMenu» (рис. 35). Дальше в объекте «Hp» добавляем скрипт «HPscript» (рис. 36). В скрипте PauseMenu в методе void Awake вызывается время равное 1 и игра не стоит на паузе (рис. 37). В методе void Update если нажать кнопку «Esc» вызывается пауза или если игра на паузе возвращает в ее в обратное состояние (рис. 38). Метод public void Menu вызывает сцену «Menu» (рис. 39). В скрипте HPscript создаем переменные (рис. 40). В методе void Update показывается количество жизни нашего персонажа, с помощью кейсов (рис. 41).

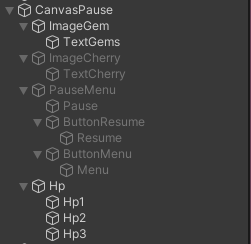


Рисунок 33. Создание объекта «CanvasPause»

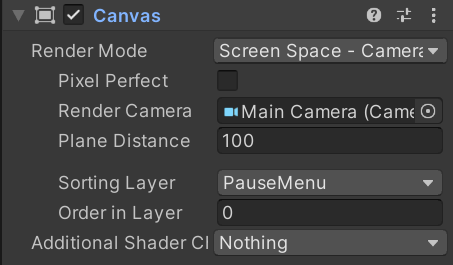


Рисунок 34. Компонент «Canvas»

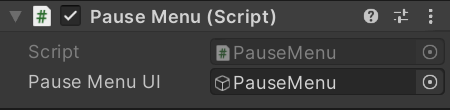


Рисунок 35. Скрипт «PauseMenu»

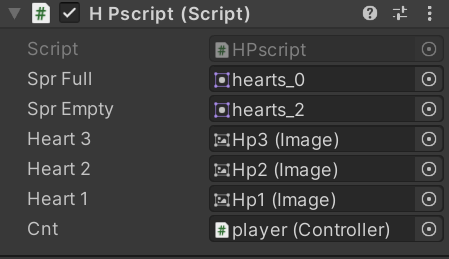


Рисунок 36. Скрипт «HPscript»

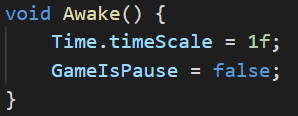


Рисунок 37. Метод Awake

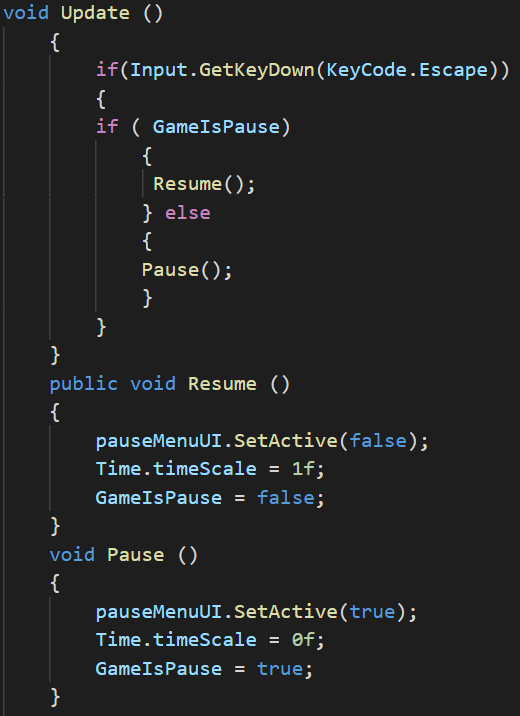


Рисунок 38. Метод Update

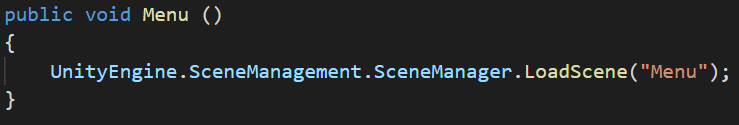


Рисунок 39. Метод Menu

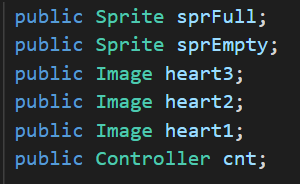


Рисунок 40. Создание переменных

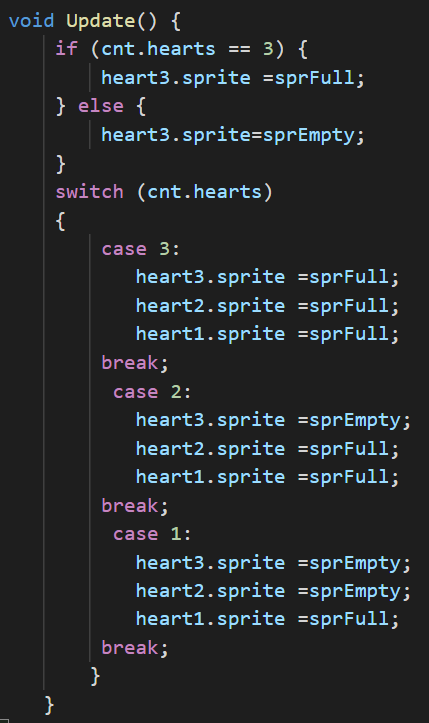


Рисунок 41. Метод Update

### Объект «PORTAL»

Далее создаем портал (дверь в домике) завершения уровня. Создаем объект «PORTAL» и даем ему компонент «Box Collider 2D» (рис. 42-43).



Рисунок 42. Объект «PORTAL»

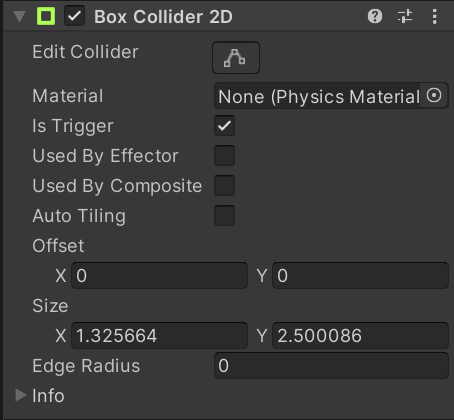


Рисунок 43. Компонент «Box Collider 2D»

### Объект «CanvasFinish»

Далее создаем выплывающее окно завершения уровня. Создаем объект «CanvasFinish», в нем создается объект «CanvasComplited», а в нем создаются уже объекты: текст «Complited», кнопки «ButtonNext» и «ButtonMenu», картинка с текстом для кристаллов «imageGem» и «TextGemsFinish» (рис. 44).

К объекту «CanvasFinish» добавляем уже написанный скрипт «ComplitedMenu» (рис. 45).

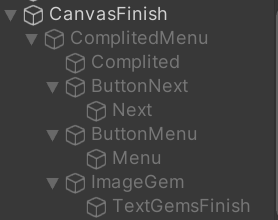


Рисунок 44. Объект «CanvasFinish»

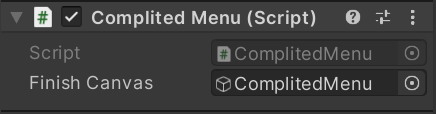


Рисунок 45. Скрипт ComplitedMenu

### Объект «PossumObj»

И последний объект это враг «Опоссум». Создаем объект «PossumObj», в нем создаем границы от и до куда может ходить Опоссум «PossumWall» и «PossumWall (1)».Так же создаем объект «opossum-1» с объектами «possumDie» и «possumJaw», и все делаем это Prefab (рис. 46). В объектах «PossumWall» и «PossumWall (1)» добавляем компонент «Box Collider 2D» (рис. 47). К объекту «opossum-1» добавляем компоненты «Rigidbody 2D» и «Capsule Collider 2D» и скрипт «OpossumControl» (рис. 48-49). К объекту «possumDie» добавляем компонент «Circle Collider 2D» и скрипт «possumDeath» (рис. 50). В скрипте «OpossumControl» в методе void Start создается объект «rg» с компонентом Rigidbody 2D. В методе void Update придаем силу объекту «rg» по оси Y (рис. 51).В методе private void OnTriggerEnter2D описываем пересечение коллайдера и поворот спрайта «Опоссума» (рис. 52).В методе public void PossumDestroyTotal вызывается убийство объекта (рис. 53). В скрипте «possumDeath» в методе private void OnTriggerEnter2D описывается, если коллайдер объекта «Player» пересекается с коллайдером обьекта «opossum-1», то вызывается метод PossumDestroy из

скрипта «OpossumControl» (рис. 54).

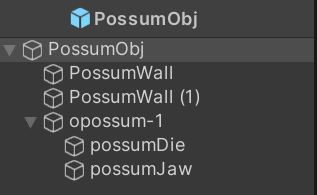


Рисунок 46. Prefab объекта «PossumObj»

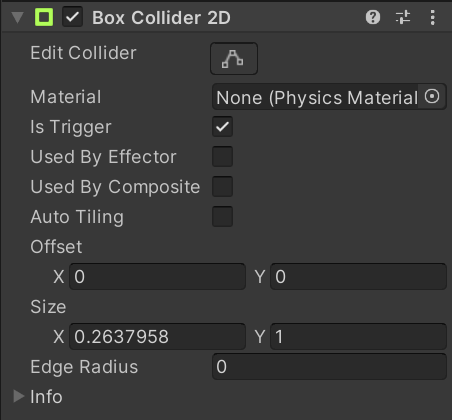


Рисунок 47. Компонент «Box Collider 2D»

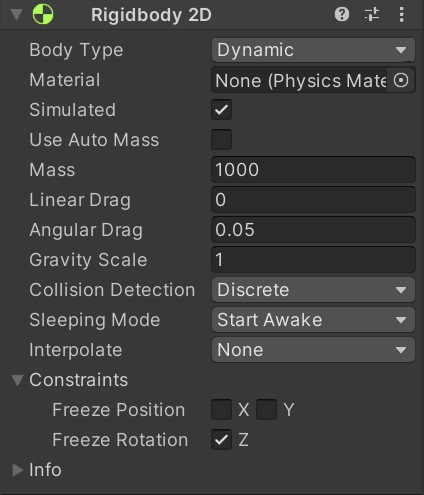


Рисунок 48. Компонент «Rigidbody 2D»

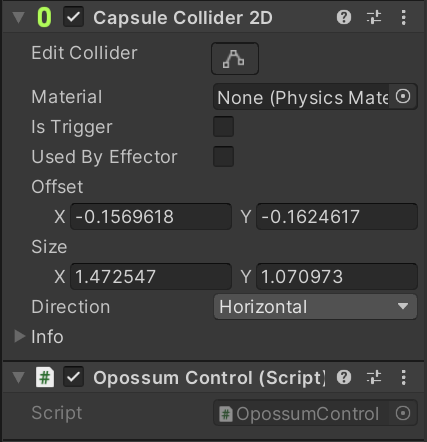


Рисунок 49. Компонент «Capsule Collider 2D» и скрипт «OpossumControl»

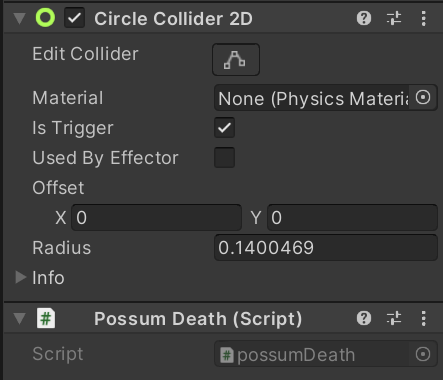


Рисунок 50. Компонент «Circle Collider 2D» и скрипт «possumDeath»

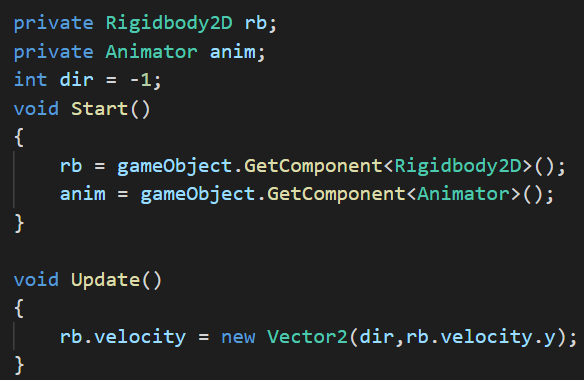


Рисунок 51. Методы Start и Update

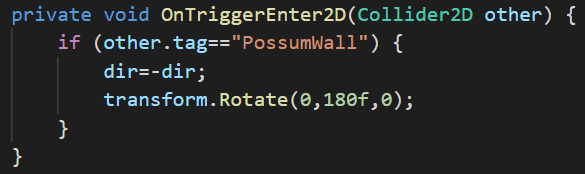


Рисунок 52. Метод OnTriggerEnter2D

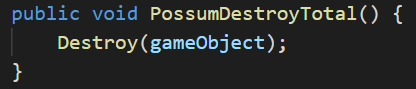


Рисунок 53. Метод PossumDestroyTotal

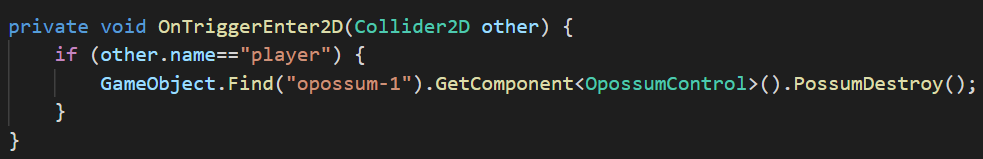


Рисунок 54. Метод OnTriggerEnter2D