**Игра 2048**

Версия Unity 2021.1.16f1

**1. Описание игры**

Суть игры: игрок перемещает стрелочками ячейки, на которых указаны значения: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048. При соединении рядом стоящих ячеек с одинаковыми значениями они образуют одну совместную. После появляется ещё одна ячейка на пустой клетке. Игровое поле состоит из 4х4 пустых клеток.

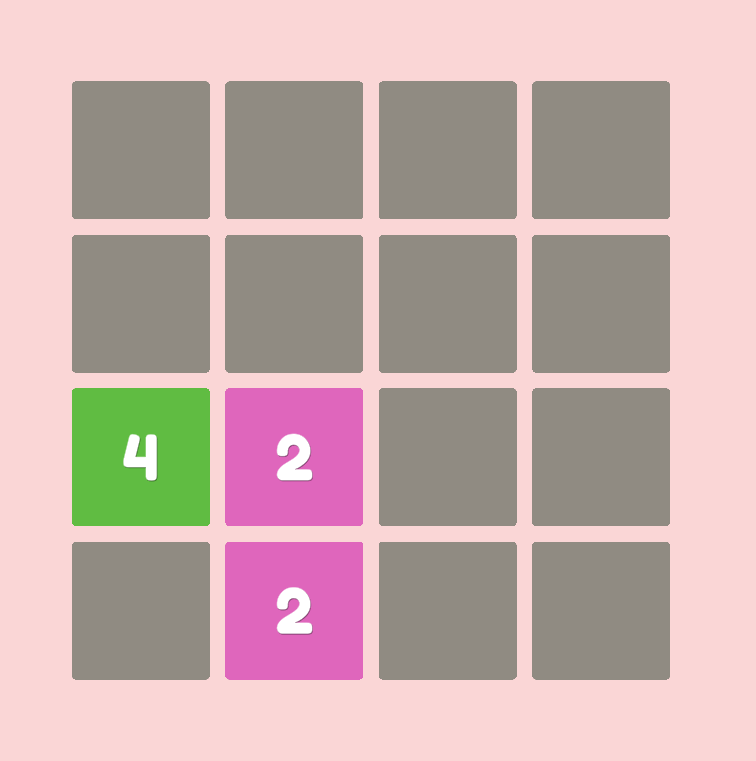
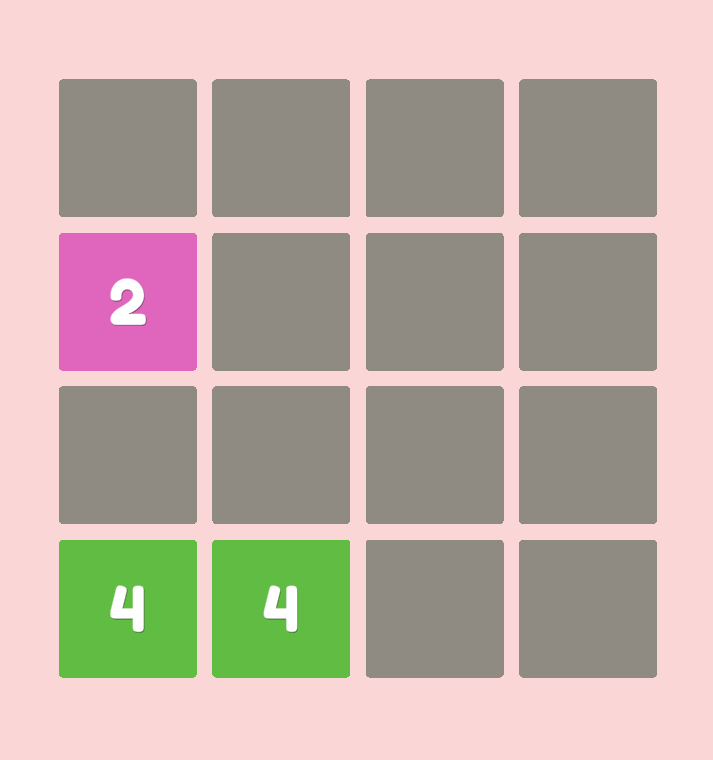
 

Рисунок 1.1 – Игровой процесс соединения двух ячеек 2 и 2 в значение 4.

**2. Экраны**

В игре присутствуют 2 экрана

* Главное меню
* Игровой экран

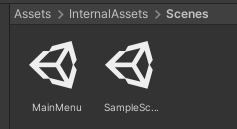


Рисунок 2.1 – Сцены игры

**2.1 Главное меню**



Рисунок 2.1.1 – Экран главного меню

На сцене главного меню представлены игровые объекты:

* Main Camera – главная камера сцены
* UI – игровой интерфейс пользователя
  + Background – задний фон сцены
  + Name game – название игры 2048
* Button play – кнопка перехода на сцену SampleScene (игровой экран)
* EventSystem – отслеживает нажатия по UI

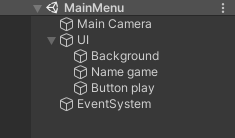


Рисунок 2.1.2 – Игровые объекты на сцене MainMenu

На сцене MainMenu игровой объект UI содержит скрипт UIController.

*\*Подробнее о скриптах описано ниже*

Переход на следующую сцену осуществляется через Button Play, который ссылается на публичный метод LoadScene в классе UIController.

**3.1 Игровой экран**

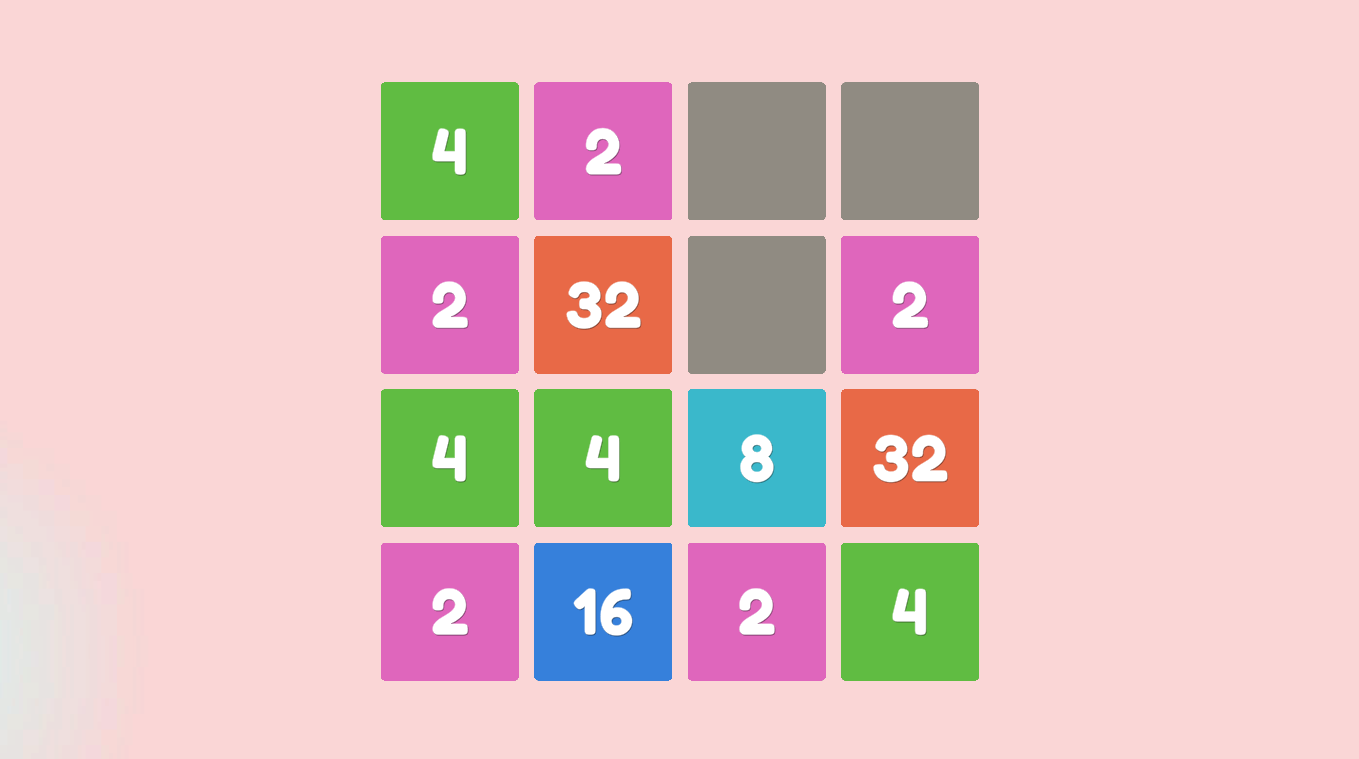


Рисунок 3.1.1 – Игровой процесс

В игровом экране до начала игры присутствуют игровые объекты:

* Main Camera – главная камера, которая отображает игровой процесс
  + Background – задний фон игры
* GameManager – пустой игровой объект. Содержит скрипт игровой логики. Управляет игрой.
* UI – пользовательский интерфейс
  + Win Screen – экран победы
    - Header – заголовок с текстом “Победа”
    - Info text - информация для игрока
    - Button Yes - согласится с информацией для игрока (Перезапустить игру)
    - Button No - Button No – отказать на информацию для игрока (Возврат в главное меню)
  + Lose Screen – экран поражения
    - Header – заголовок с текстом “Поражение”
    - Info text – информация для игрока
    - Button Yes – согласится с информацией для игрока (Перезапустить игру)
    - Button No – отказать на информацию для игрока (Возврат в главное меню)
* EventSystem - отслеживает нажатия по UI

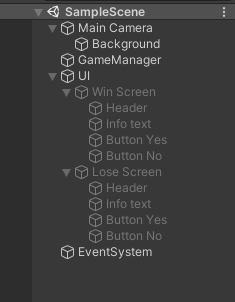


Рисунок 3.1.2 – Игровые объекты на сцене



Рисунок 3.1.3 – Экран победы

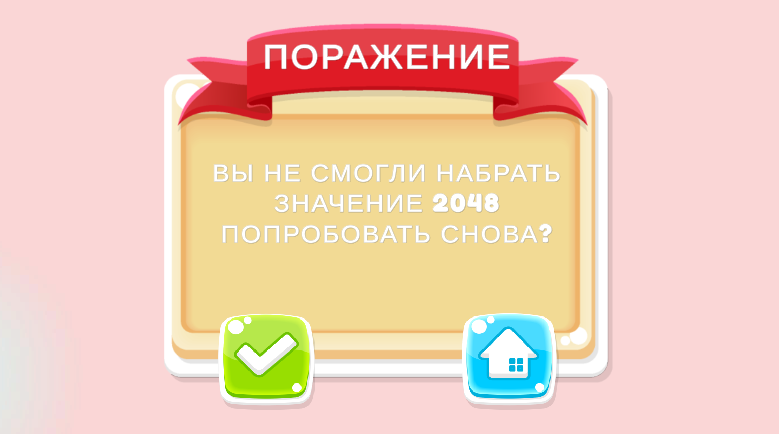


Рисунок 3.1.4 – Экран поражения

**3. Классы**

В проекте присутствуют классы:

* UIController – содержит логику для UI кнопок.
* Block – описывает игровой объект ячейку
* Node – описывает игровой объект клетку (по которым перемещается ячейка)
* GameManager – игровой менеджер. Содержит основную логику игры

**3.1 Класс UIController**

Поля:

* отсутствуют

Методы:

* LoadScene – загрузка сцены
* RestartGame – рестарт текущей сцены

**3.1.1 Метод LoadScene**

Объявление:

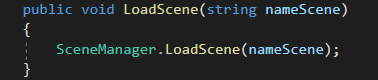
public void LoadScene(string nameScene)

Параметры:

|  |  |
| --- | --- |
| nameScene | Название сцены |

Описание:

Загружает сцену по имени. В параметр мы передаём название сцены. Класс SceneManager используется в пространстве имён *using UnityEngine.SceneManagement* у класса SceneManager есть метод LoadScene которому можно передать название сцены и он её загрузит



**3.1.2 Метод RestartGame**

Объявление:

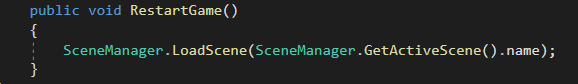
public void RestartGame()

Параметры:

Отсутствуют.

Описание:

Работает почти так же как и предыдущий метод LoadScene, только тут обращаемся к SceneManager, чтобы он вернул название активной сцены и получаем от активной сцены имя (GetActiveScene.name – получить активную сцену. имя)



**3.2 Класс Block**

Поля:

public int Value; - значение, которое отображается на ячейке

public Node Node; - поле, на котором находится ячейка

public Block MergingBlock; - ячейка, с которой будет соединятся эта ячейка для увеличения значения

public bool Merging; - возможность для соединения. (необходимо, чтобы 3 блока не соединялись в 1)

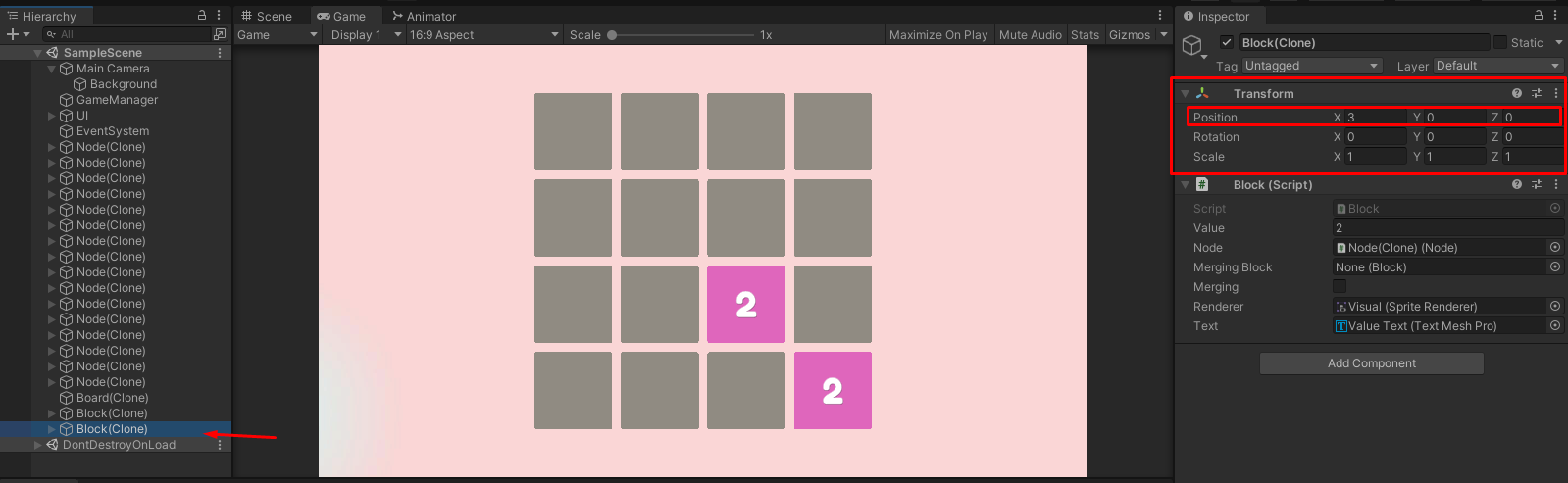
public Vector2 Pos => transform.position; - позиция ячейки в координатах X, Y

[SerializeField] private SpriteRenderer \_renderer; - ссылка на компонент SpriteRenderer

[SerializeField] private TextMeshPro \_text; - текст, который отображает значение ячейки

*\*transform.position*

*– это обращение к компоненту Transform, к позиции игрового объекта, на котором висит этот скрипт (класс).*



*\*\* SpriteRenderer – отображает графический объект. При помощи него можно изменить цвет для ячейки*

Методы:

* Init – инициализация ячейки. (выставление начальных значений)
* SetBlock – изменить ссылку на поле на которой стоит ячейка
* MergeBlock – объединить ячейки (устанавливает, что ячейки будут сейчас объединены)
* CanMegre – проверяет, может ли эта ячейка объединится с другой. Например, две ячейки 2 и 2, объединится могут в 4. А 4 и 2 не могут.

**3.2.1 Метод Init**

Объявление:

public void Init(BlockType type)

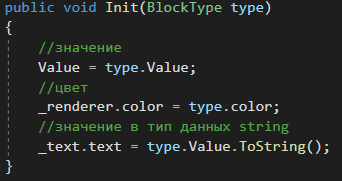
Параметры

|  |  |
| --- | --- |
| type | Тип блока, который содержит в себе значение и цвет |

Описание:

Инициализация начальных значений для ячейки:

Для данного блока присваивается значение, цвет и текст, который будет отображаться на блоке. Текст обычно соответствует значению.



**3.2.2 SetBlock**

Объявление:

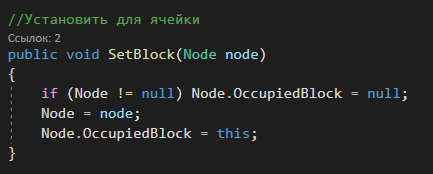
public void SetBlock(Node node)

Параметры:

|  |  |
| --- | --- |
| Node | Нода на которой будет установлена ячейка |

Описание:

Назначает ячейку на новую ноду, так же устанавливает для ячейки эту самую ноду.



**\*дальше я решил просто прописать комментарии в скриптах(классах)\***