

Лабораторная работа №2: Пользовательский интерфейс.

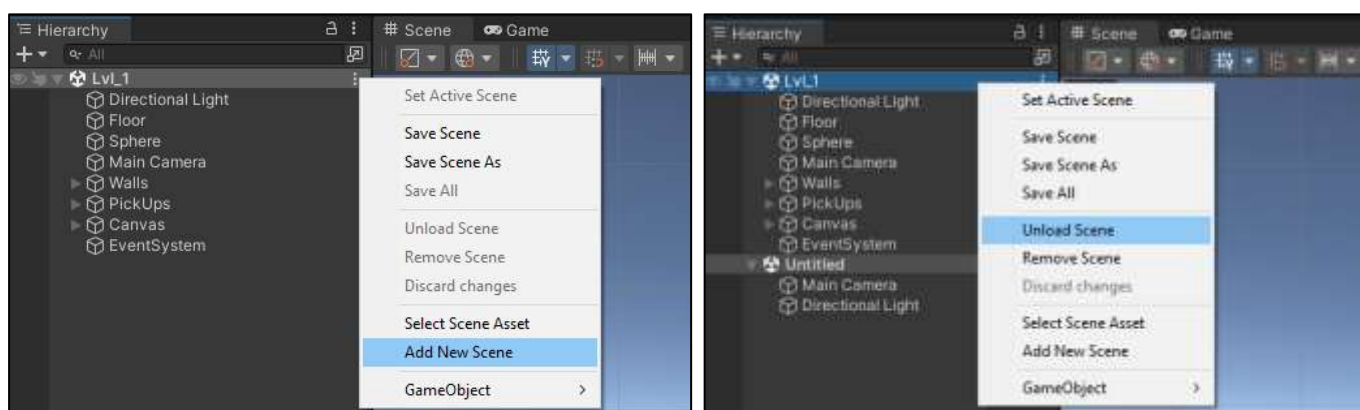
Целью работы является создание стартового меню, внутри игрового меню, загрузочного экрана и меню выбора уровней.

Оглавление

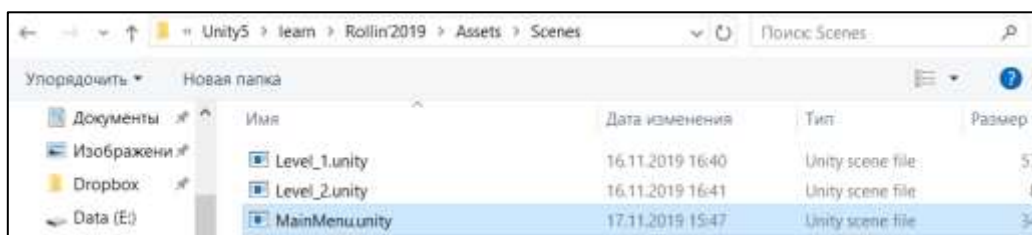
Стартовое меню.....	1
Панель помощи	9
Внутри игровое меню	10
Экран загрузки	12
Выбор уровня	15
Оптимизация	22
Задания:.....	23
Отчёт.	23
Приложение: обмен данными между скриптами.....	24

Стартовое меню

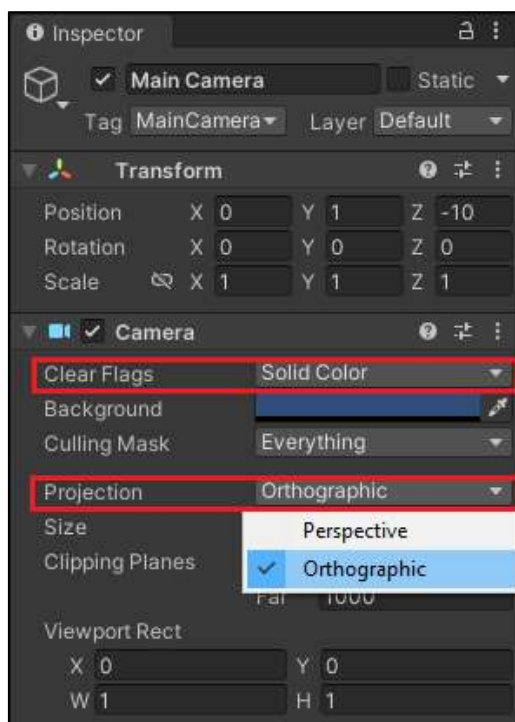
Для того, чтобы добавить стартовое меню, добавьте новую сцену, уже открытые можно выгрузить:



Сохраните новую сцену в папке Scenes:



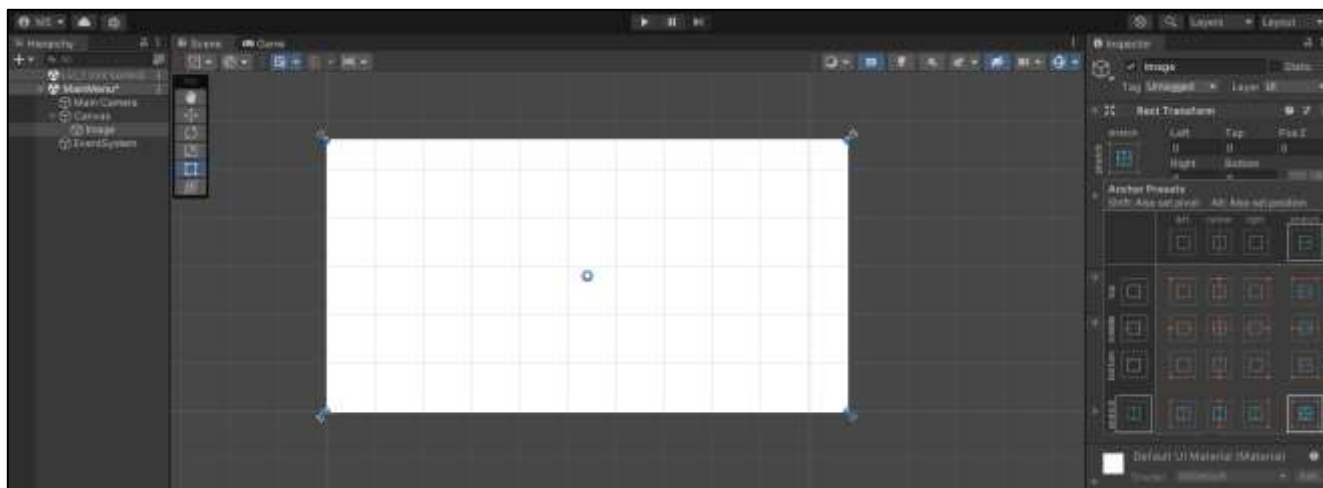
Если задний фон стартового меню планируется сделать двумерным, то имеет смысл удалить из сцены источник освещения и переключить камеру в режим ортогографической проекции:



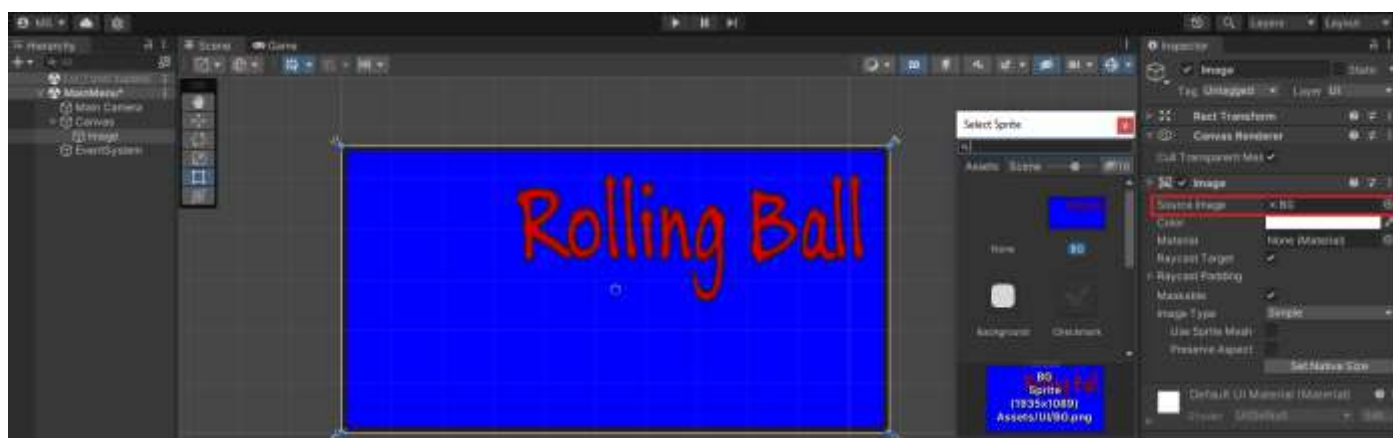
В качестве заднего фона можно использовать изображение размером с область экрана, которая, в нашем случае, равняется области объекта Canvas. Для этого, добавьте в проект изображение с задним фоном, сконвертируйте его в спрайт и нажмите Apply:



Добавьте объект Image на Canvas и укажите что объект должен быть закреплён за его края:

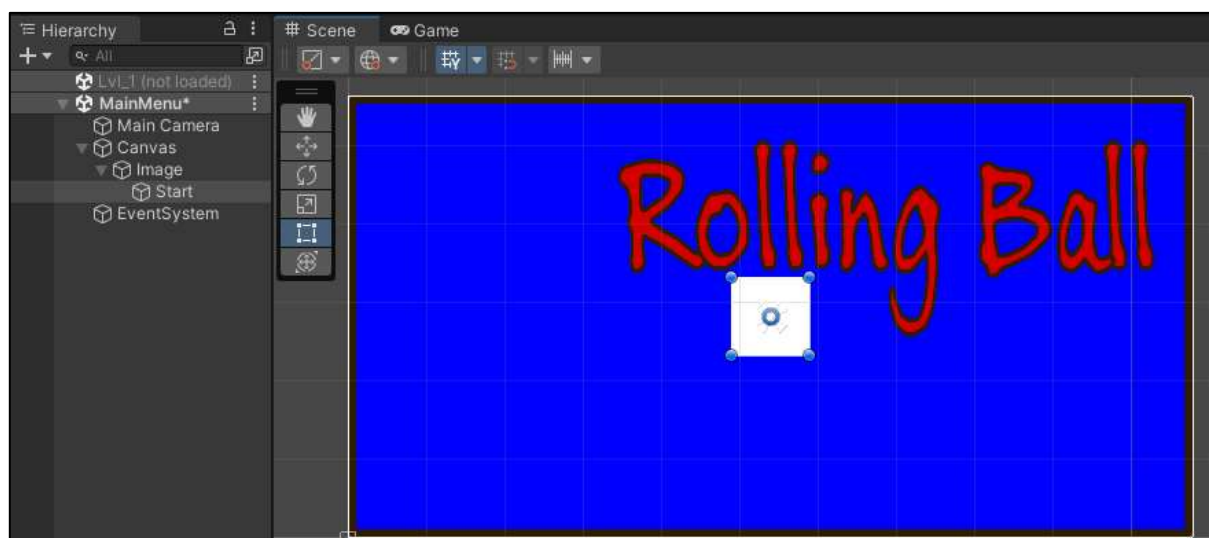


После чего, укажите в качестве исходного изображения спрайт с задним фоном:



Примечание: все последующие объекты интерфейса должны добавляться к этому объекту Image, иначе он может их перекрывать.

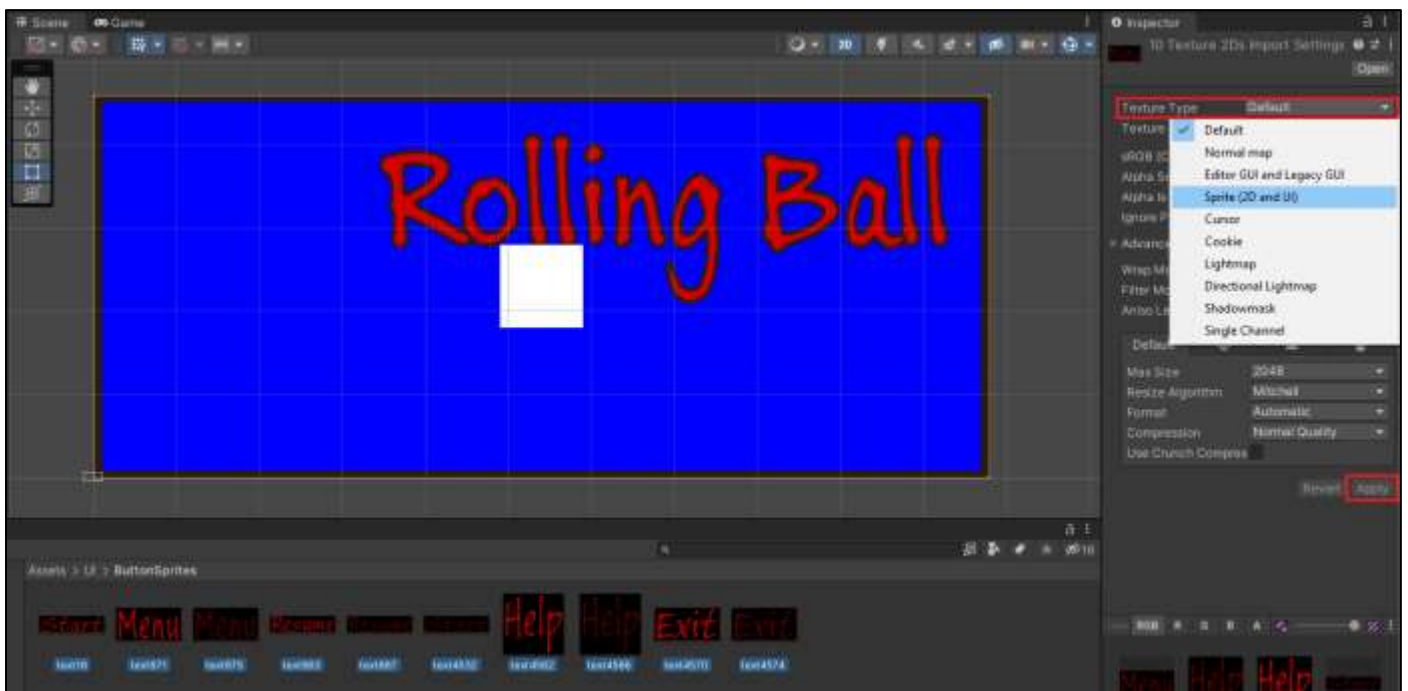
Для добавления кнопки в стартовое меню, добавьте в проект компонент Image и назовите его “Старт”:



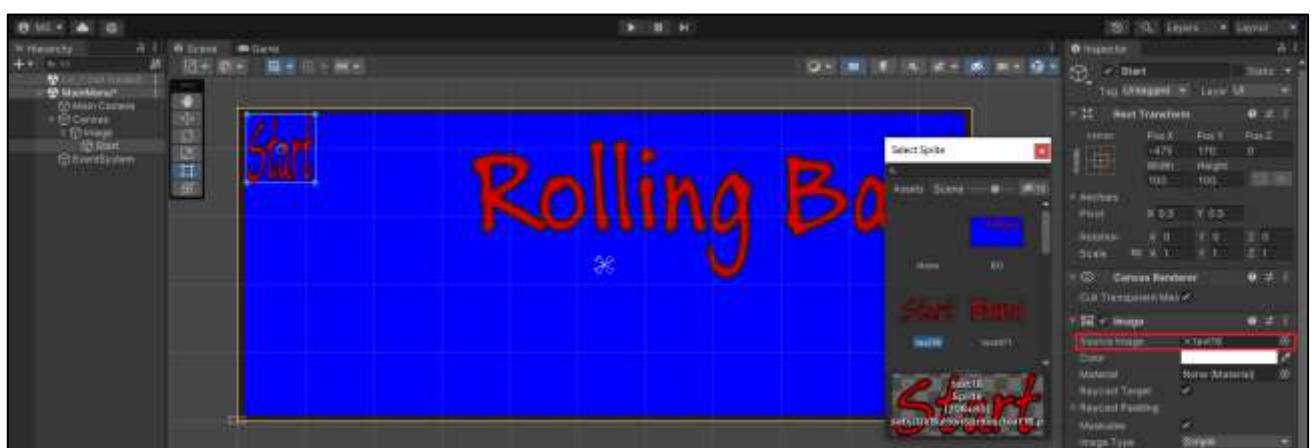
Создайте в разделе Project папку ButtonSprites и добавьте в неё изображения, отвечающие за кнопки:



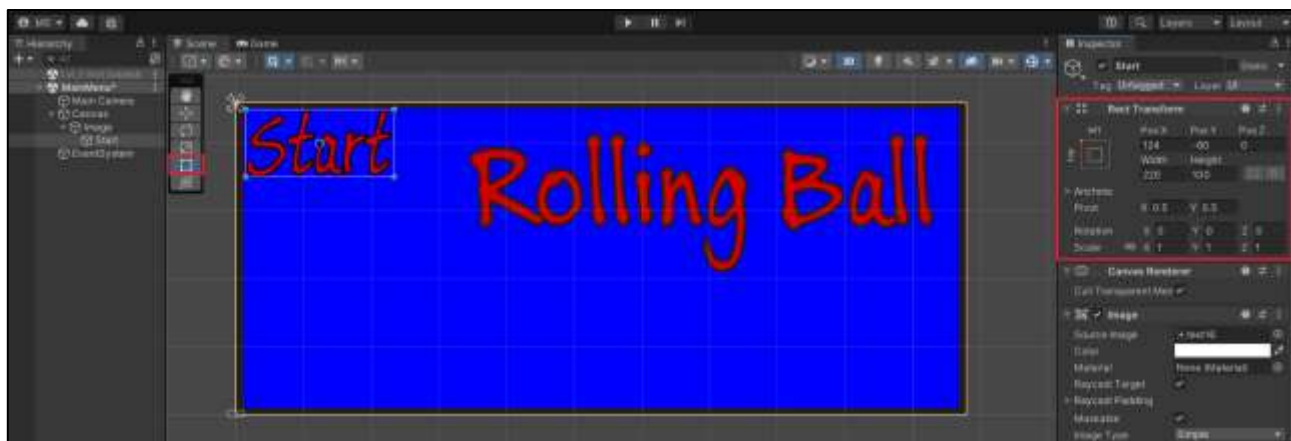
Выберите добавленные изображения (Shift + click) и сконвертируйте их в спрайты как показано на изображении ниже:



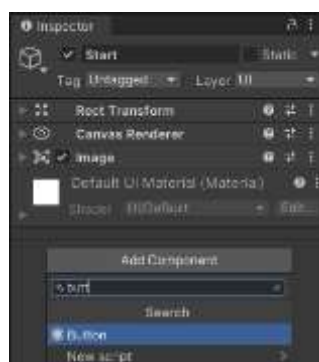
Для того, чтобы использовать полученные спрайты в качестве кнопок, выберите компонент Image на канвase и укажите желаемый спрайт в качестве источника:



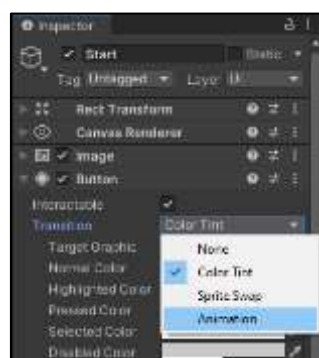
Задайте размер и позицию изображения используя инструмент Rect Tool:



Добавьте к изображению компонент Button, это позволит добавить ему функции кнопки:

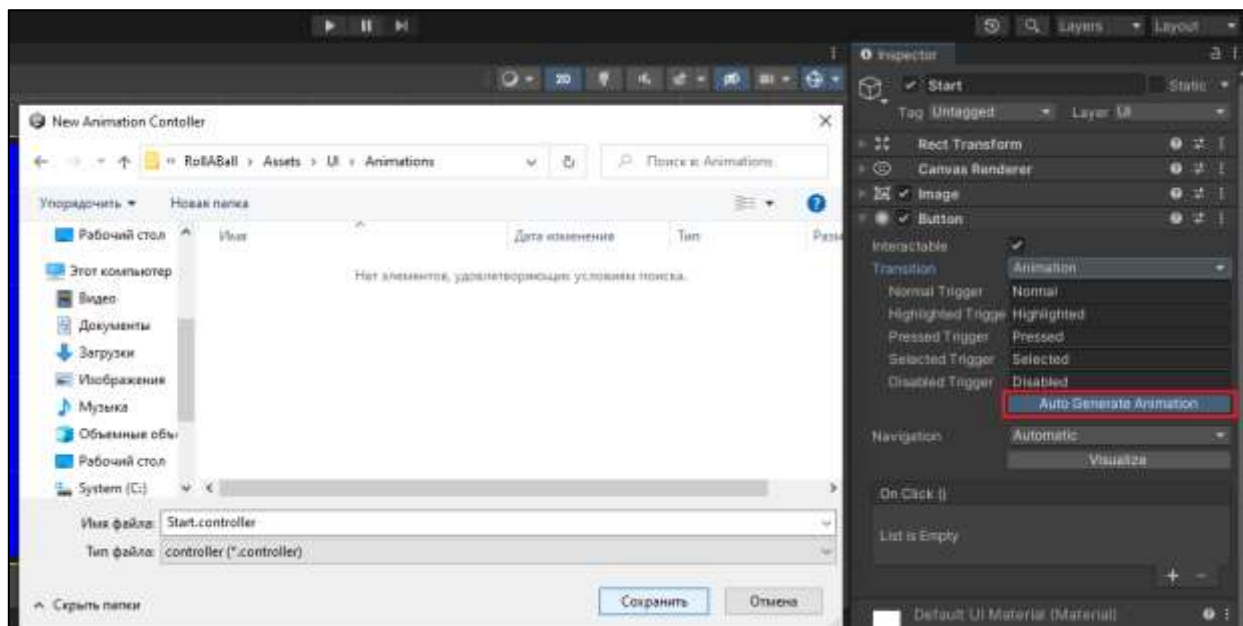


В качестве типа Transition выберите Animation, это позволит создавать анимации привязанные к событиям, связанным с работой кнопки:

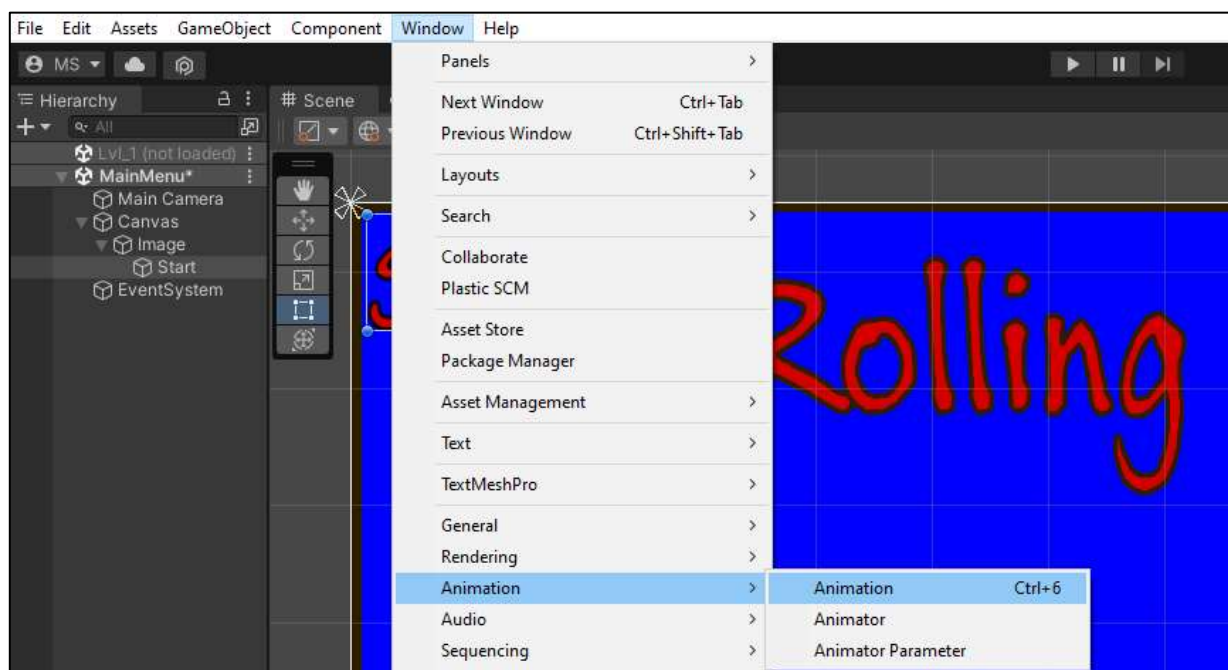


Другие два пункта это: изменение цвета и смена изображения, в ответ на возникновение события.

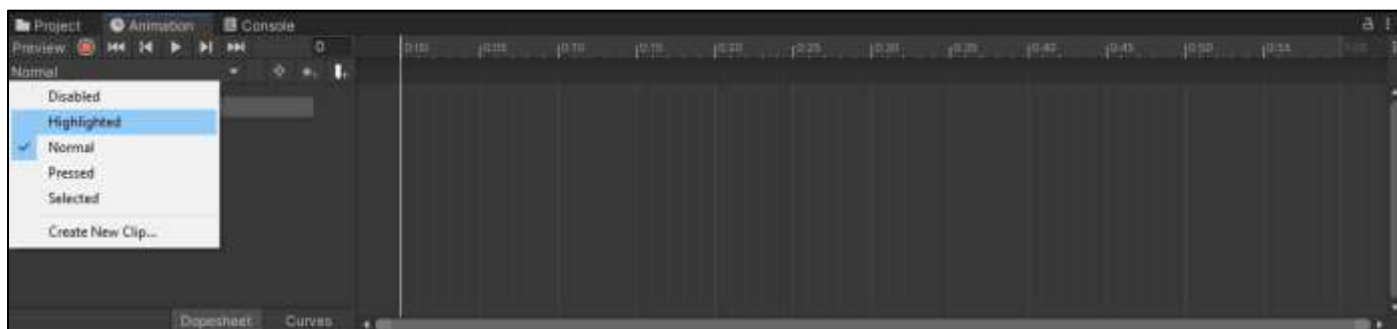
Для того, чтобы создать анимацию, привязанную к кнопке, нажмите Auto Generate Animation и сохраните анимацию в соответствующей директории:



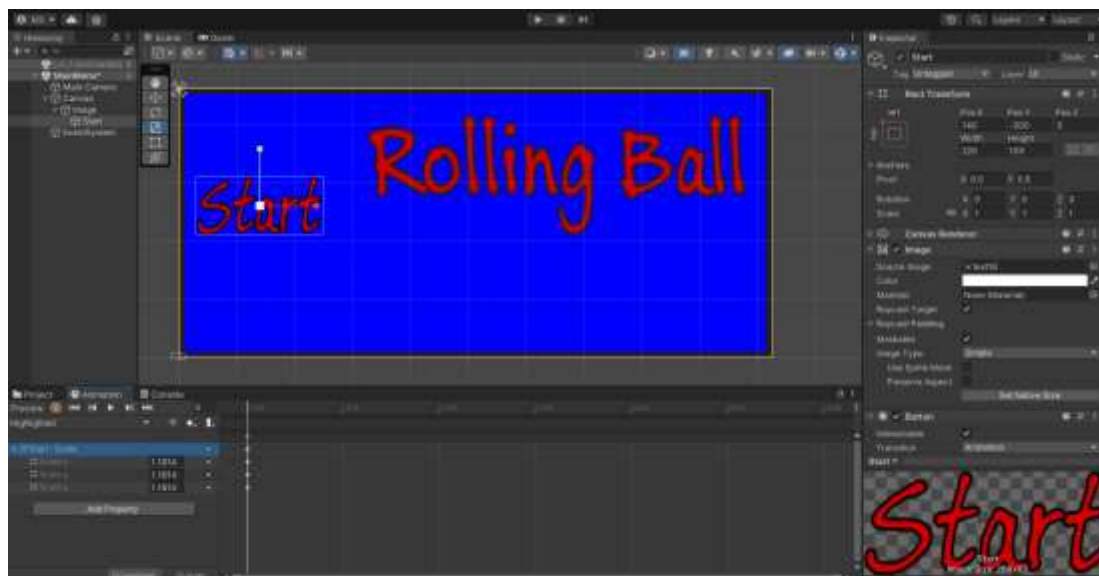
Если в интерфейсе отсутствует окно анимации, добавить его можно выбрав Window -> Animation -> Animation:



В окне анимации, выберите событие, для которого нужно создать анимацию. В данном случае, это подсветка:

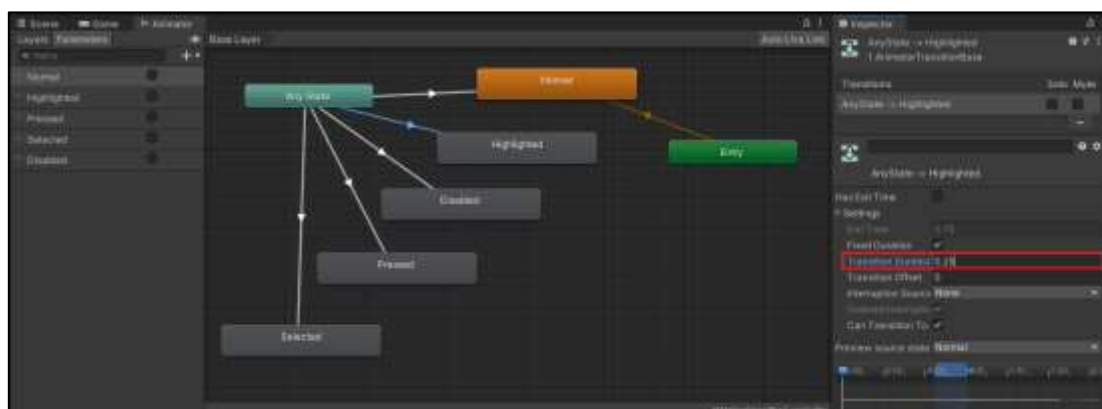


После чего, нажмите кнопку записи (красный круг) и совершите необходимые действия (например, изменение масштаба изображения):



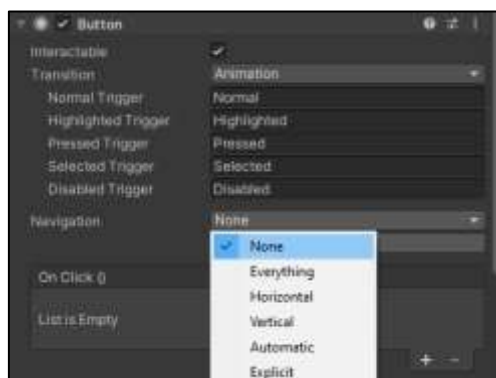
Помимо анимации изменения размеров, позиции и поворота изображения, можно задать, например, смену исходного спрайта. Это может быть использовано для реализации эффекта нажатия на кнопку.

Отрегулировать скорость перехода между состояниями изображения (обычное, подсвечена, нажата и т.д.), можно в разделе Animator. Для этого, выберите переход между состояниями и задайте необходимое значение в поле Transition Duration:



После завершения настройки кнопки “Старт”, добавьте ещё две, “Помощь” и “Выход”.

Примечание: по умолчанию, при нажатии на кнопку, она остаётся в состоянии Selected. Изменить это поведение можно выбрав в поле Navigation – None:



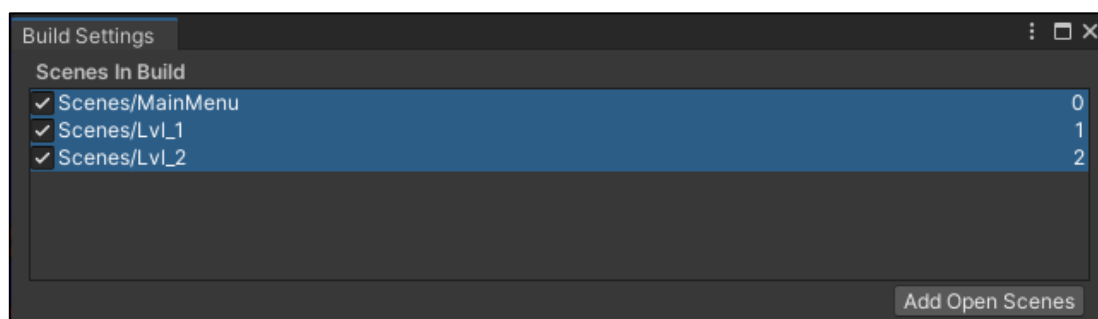
Подробнее о опциях навигации можно прочитать в официальной документации:

<https://docs.unity3d.com/2021.3/Documentation/Manual/script-SelectableNavigation.html>

В процессе работы приложения, можно переключаться только между сценами, добавленными в сборку. Что бы добавить сцены в сборку, откройте их в разделе Hierarchy выбрав в разделе Project и перетаскив:



После чего, перейдите в File -> Build Settings... и нажмите Add Open Scenes:



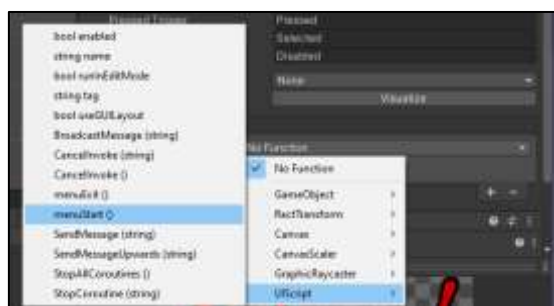
Обратите внимание, что при добавлении в сборку, всем сценам был присвоен индекс.

Добавьте к объекту Canvas новый скрипт. В этом скрипте будет описана реакция на нажатие кнопок:

```
1 using UnityEngine;
2 using UnityEngine.SceneManagement;
3
4 public class UIScript : MonoBehaviour
5 {
6     public void menuStart()
7     {
8         SceneManager.LoadScene(1); // загрузка сцены по её индексу
9     }
10
11     public void menuExit()
12     {
13         Application.Quit(); // выход из приложения (не работает в режиме проигрывания)
14     }
15 }
```

Примечание: вместо индекса сцены можно использовать её название.

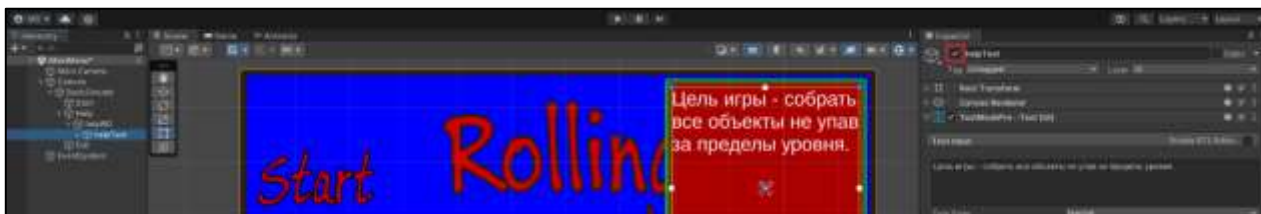
Назначьте кнопкам “Старт” и “Выход” соответствующие функции как показано ниже:



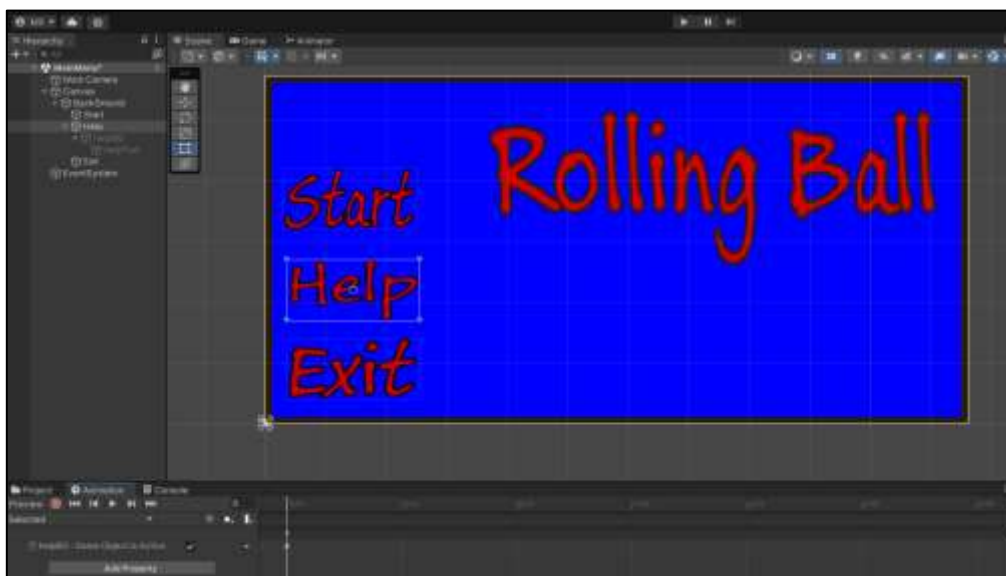
Если всё сделано правильно, то в режиме проигрывания, при нажатии кнопки “Старт” будет загружаться сцена с индексом 1. При нажатии кнопки “Выход”, в режиме проигрывания, ничего происходить не должно, однако в собранном проекте, нажатие приведёт к его закрытию.

Панель помощи

Панель помощи может быть реализована без использования скриптов. Для этого, добавьте к “Помощь” новое изображение, содержащее задний фон и границы, а также текст, содержащий подсказки:



После добавления и настройки, деактивируйте изображение. Выберите кнопку “Помощь”, перейдите к разделу анимации, выберите событие “Selected” и добавьте активацию изображения с подсказками к этому событию:



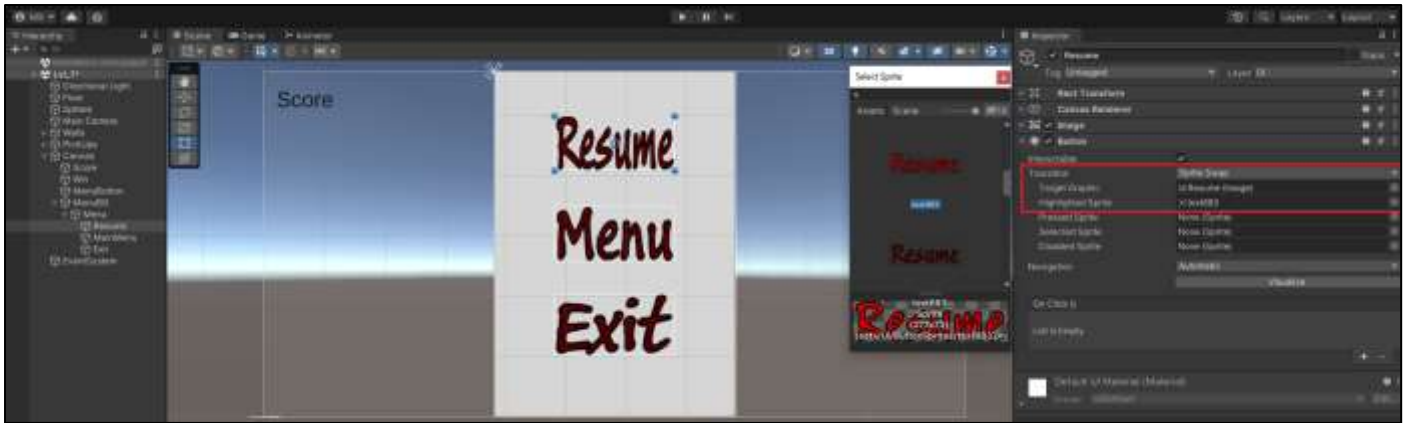
Если всё было сделано правильно, при нажатии на кнопку “Помощь”, будет появляться изображение с подсказкой.

Внутри игровое меню

Добавьте в игровую сцену кнопку “Меню”, по нажатию которой будет вызываться внутри игровое меню, затем, добавьте новую панель, которая будет являться фоном для меню, к этой панели добавьте ещё одну панель, которая будет являться формой меню, и на этой панели разместите кнопки “Продолжить”, “Выйти” и “Меню”:



В качестве обработчика Transitions, используйте Sprite Swap:



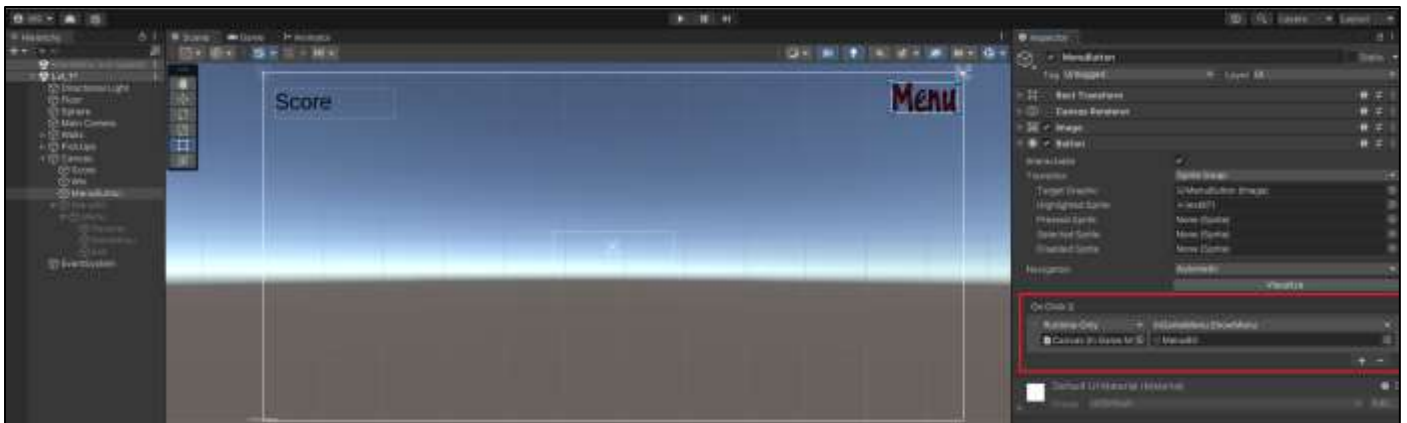
Создайте новый скрипт для объекта Canvas. Опишите в скрипте реакцию на нажатия кнопок:

```

1  using UnityEngine;
2  using UnityEngine.SceneManagement;
3
4  public class InGameMenu : MonoBehaviour
5  {
6      public void ShowMenu(GameObject menu) // показать внутриигровое меню, параметр - ссылка на панель с меню
7      {
8          menu.SetActive(true); // делает панель активной
9          Time.timeScale = 0f; // останавливает время
10     }
11
12     public void Resume(GameObject menu) // продолжение игры
13     {
14         menu.SetActive(false); // делает панель меню не активной, параметр - ссылка на панель меню
15         Time.timeScale = 1f; // запускает время с нормальной скоростью
16     }
17
18     public void MainMenu() // переход в главное меню
19     {
20         Time.timeScale = 1f; // запускает время с нормальной скоростью
21         SceneManager.LoadScene(0); // загрузка сцены главного меню
22     }
23
24     public void Exit() // выход из приложения
25     {
26         Time.timeScale = 1f; // запускает время с нормальной скоростью
27         Application.Quit(); // выход из приложения (не работает в режиме проигрывания)
28     }
29 }

```

Деактивируйте панель фона внутри игрового меню и назначьте кнопкам соответствующие обработчики:

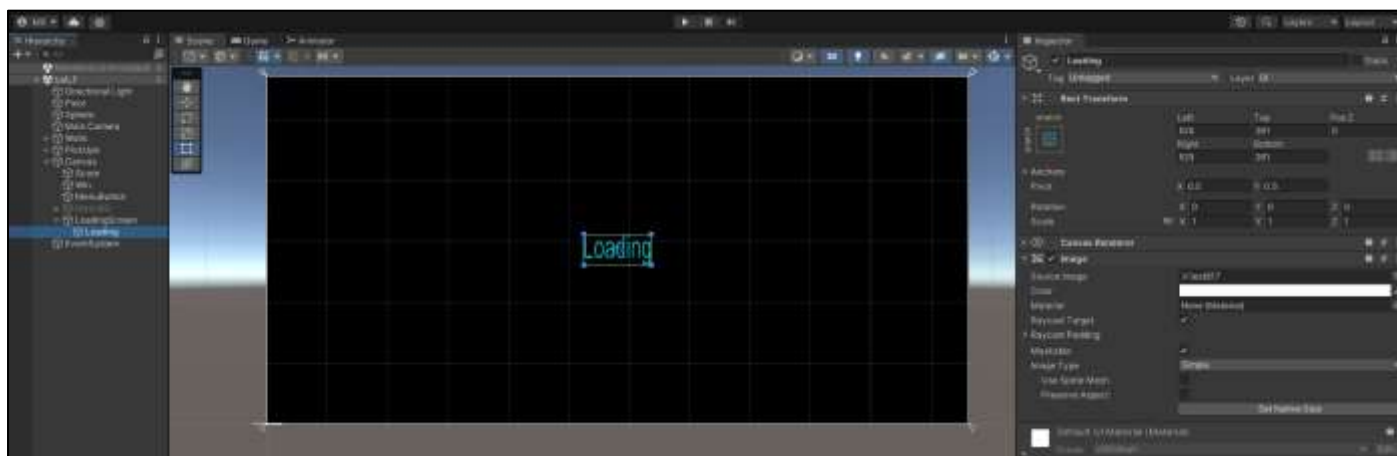


Обратите внимание, что в 2 обработчика необходимо передать ссылку на панель меню.

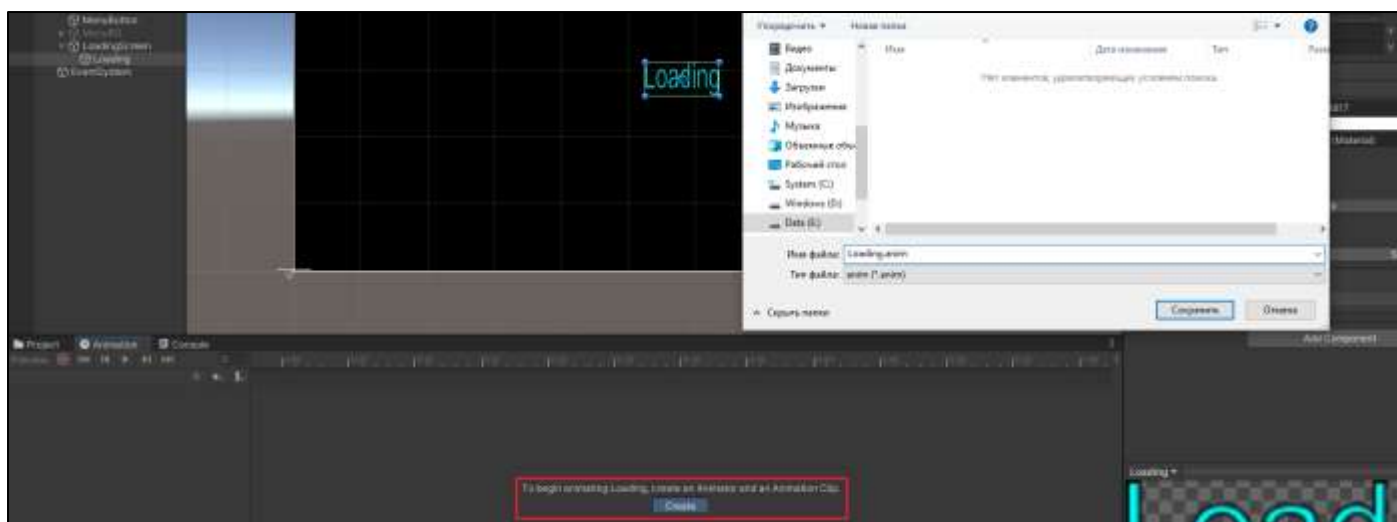
Если всё было сделано правильно, то при нажатии кнопки “Меню” на экране, игра должна останавливаться и должно появляться внутри игровое меню. При нажатии кнопки “Продолжить”, меню должно исчезать, а игра должна продолжаться, при нажатии “Меню”, должен происходить возврат в стартовое меню.

Экран загрузки

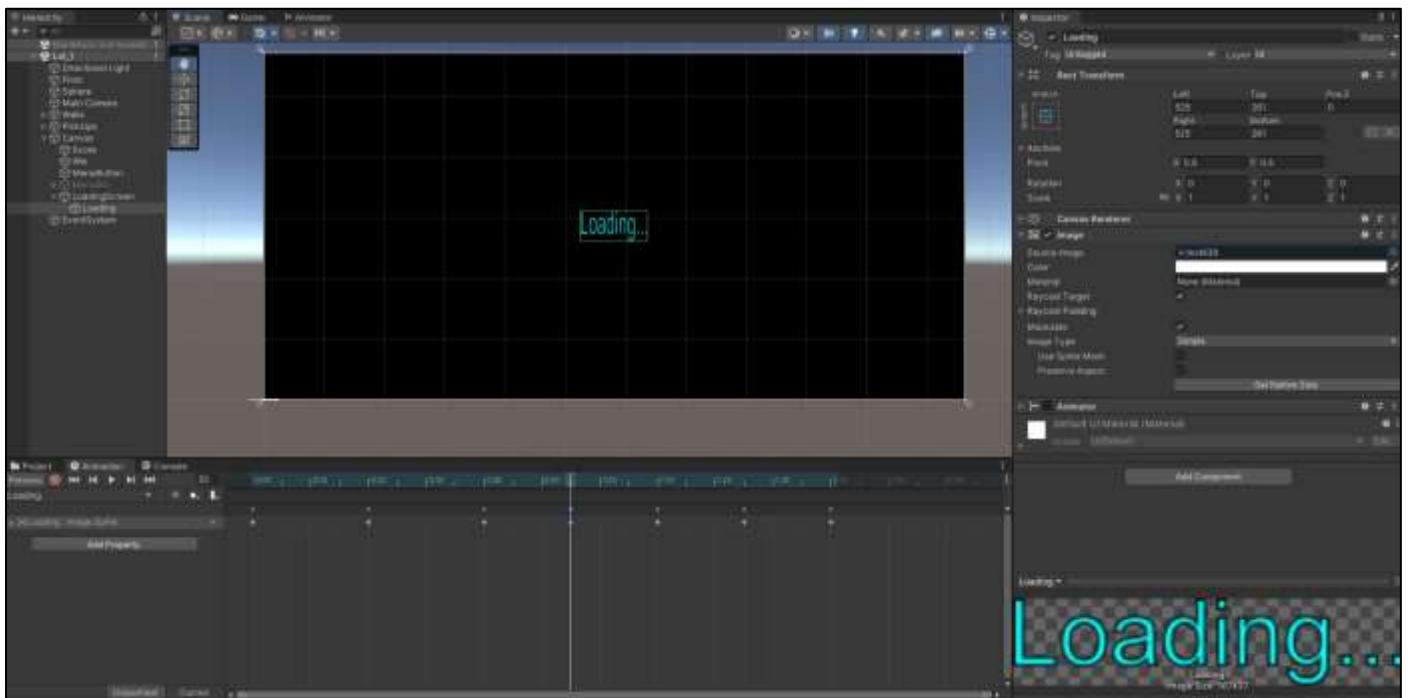
Что бы создать загрузочный экран, добавьте новое изображение “Экран загрузки” и изображение с надписью:



Для того, чтобы анимировать надпись, выберите соответствующее изображение, перейдите к разделу Animation и нажмите Create:



В качестве простой анимации, можно использовать смену изображений с течением времени. Для этого, включите запись анимации, выберите время и укажите требуемый кадр анимации:



Чтобы анимация корректно отображалась, загрузку следующей сцены нужно производить в фоновом режиме. Модифицируйте скрипт обработки нажатий на кнопки следующим образом:

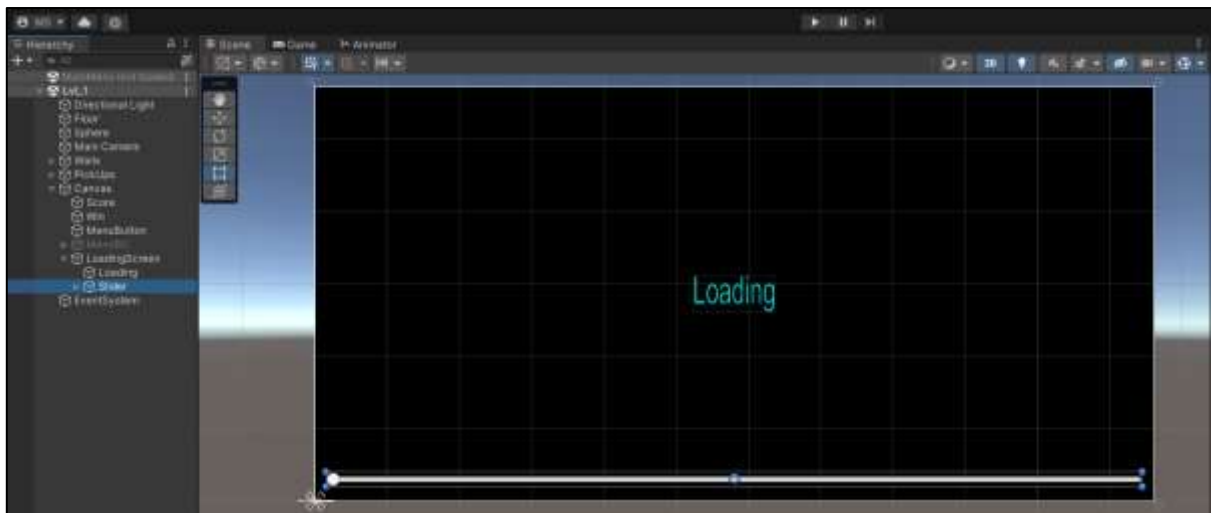
```

19 public void MainMenu(GameObject loadingScreen) // переход в главное меню, параметр - ссылка на окно загрузки
20 {
21     loadingScreen.SetActive(true); // делает активным загрузочный экран
22     Time.timeScale = 1f; // запускает время с нормальной скоростью
23     StartCoroutine(LoadLevelAsinc()); // запуск фонового процесса для загрузки сцены
24 }
25
26 private IEnumerator LoadLevelAsinc() // функция для загрузки новой сцены в фоновом режиме
27 {
28     AsyncOperation asyncLoad = SceneManager.LoadSceneAsync(0); // создание не синхронной операции
29
30     while (!asyncLoad.isDone) yield return null; // ожидание завершения загрузки
31 }

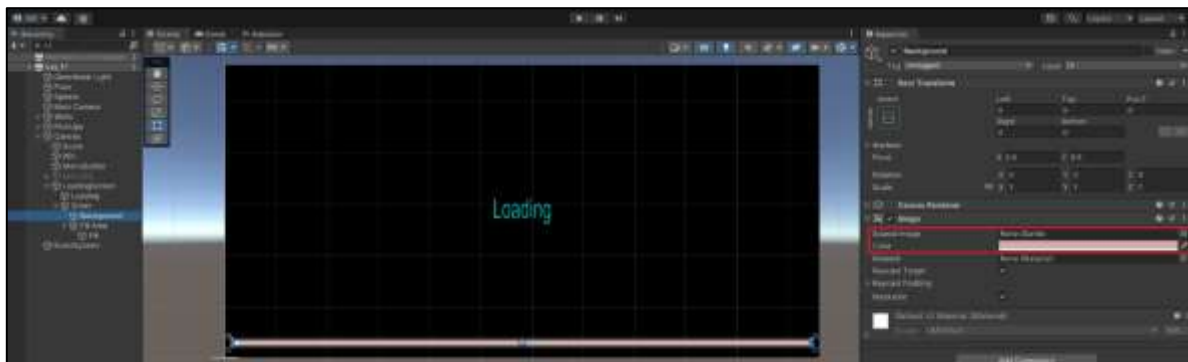
```

Для корректной работы скрипта, необходимо добавить `using System.Collections;`

Альтернативным способом отображения прогресса загрузки является Progress Bar. Для реализации Progress Bar добавьте к экрану загрузки компонент Slider:



После чего, удалите компонент Handle Slide Area, исходные изображения для заднего фона и полосы прокрутки и установите для них желаемые цвета:



Что бы использовать Slider в качестве прогресс бара, обновите скрипт обработки нажатия кнопок следующим образом:

```

1  using System.Collections;
2  using UnityEngine;
3  using UnityEngine.SceneManagement;
4  using UnityEngine.UI;
5
6  public class InGameMenu : MonoBehaviour
7  {
8      public Slider progressBar;
9
10     public void ShowMenu(GameObject menu) // показать внутриигровое меню, параметр - ссылка на панель с меню...
11
12     public void Resume(GameObject menu) // продолжение игры...
13
14     public void MainMenu(GameObject loadingScreen) // переход в главное меню, параметр - ссылка на окно загрузки...
15
16     private IEnumerator LoadLevelAsinc() // функция для загрузки новой сцены в фоновом режиме
17     {
18         AsyncOperation asyncLoad = SceneManager.LoadSceneAsync(0); // создание не синхронной операции
19
20         while (!asyncLoad.isDone) // ожидание завершения загрузки
21         {
22             progressBar.value = asyncLoad.progress; // отображение прогресса загрузки
23             yield return null;
24         }
25     }
26
27     public void Exit() // выход из приложения...
28 }

```

Примечание: в некоторых случаях, необходимо дожидаться реакции игрока перед окончательным переходом к новой сцене. Реализовать такое ожидание можно добавив компонент Text на экран загрузки, содержащий сообщение о завершении загрузки, и модифицировав скрипт обработки нажатия кнопок следующим образом:

```

6 public class InGameMenu : MonoBehaviour
7 {
8     public Slider progressBar; // ссылка на прогресс бар
9     public GameObject text; // ссылка на текст о необходимости нажать кнопку для продолжения
10
11     public void ShowMenu(GameObject menu) // показать интргровое меню, параметр - ссылка на панель с меню...
12
13     public void Resume(GameObject menu) // продолжение игры...
14
15     public void MainMenu(GameObject loadingScreen) // переход в главное меню, параметр - ссылка на окно загрузки...
16
17     private IEnumerator LoadLevelAsync() // функция для загрузки новой сцены в фоновом режиме
18     {
19         AsyncOperation asyncLoad = SceneManager.LoadSceneAsync(0); // создание не синхронной операции
20
21         asyncLoad.allowSceneActivation = false; // отключение активации сцены по завершении загрузки
22
23         while (!asyncLoad.isDone) // ожидание завершения загрузки
24         {
25             progressBar.value = asyncLoad.progress; // отображение прогресса загрузки
26
27             if (asyncLoad.progress >= 0.9f && !asyncLoad.allowSceneActivation) // если сцена загружена и не активирована
28             {
29                 text.SetActive(true); // показать текст о необходимости нажать кнопку
30                 progressBar.value = 1.0f; // отобразить полную загрузку
31
32                 if (Input.anyKeyDown) // если нажата любая кнопка
33                     asyncLoad.allowSceneActivation = true; // активировать загруженную сцену
34             }
35
36             yield return null;
37         }
38     }
39 }

```

Примечание: механизм запуска фоновых процессов может быть использован для создания ограниченных по времени эффектов. Например, если мы ходим изменить скорость игрока на 5 секунд:

```

void Event() // при возникновении какого-то события
{
    speed *= 2; // ускорение игрока в 2 раза
    StartCoroutine(BGProcess()); // запуск фонового процесса
}

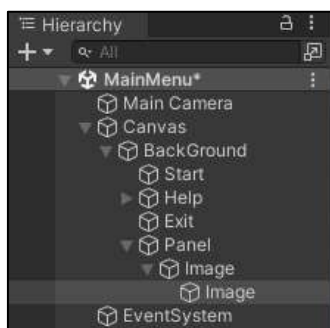
IEnumerator BGProcess() // фоновый процесс
{
    yield return new WaitForSeconds(5); // ожидает 5 секунд
    speed /= 2; // возврат прежней скорости через 5 секунд
}

```

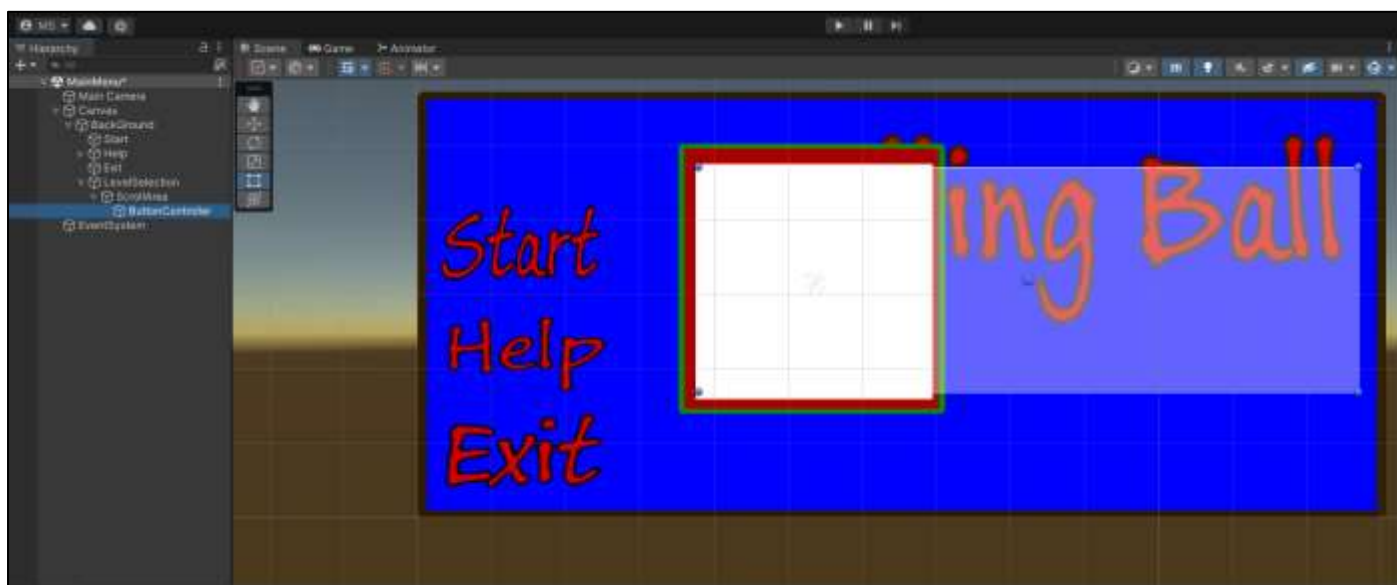
Важно: в ряде случаев, имеет смысл вынести механику загрузочного экрана в отдельную сцену. Это позволит уменьшить объём используемой приложением памяти и лучше структурировать проект.

Выбор уровня

Для создания панели выбора уровня, добавьте на холст панель и два изображения:

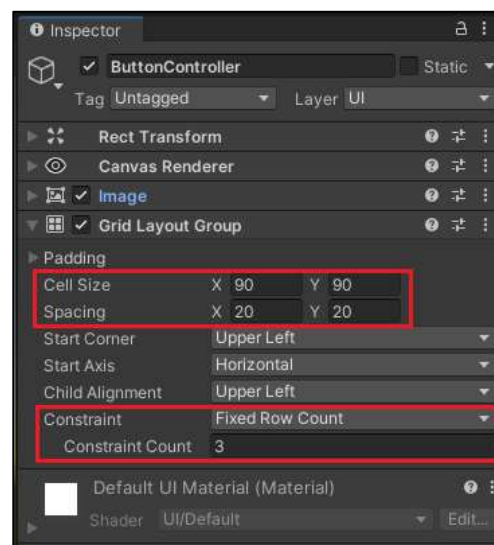
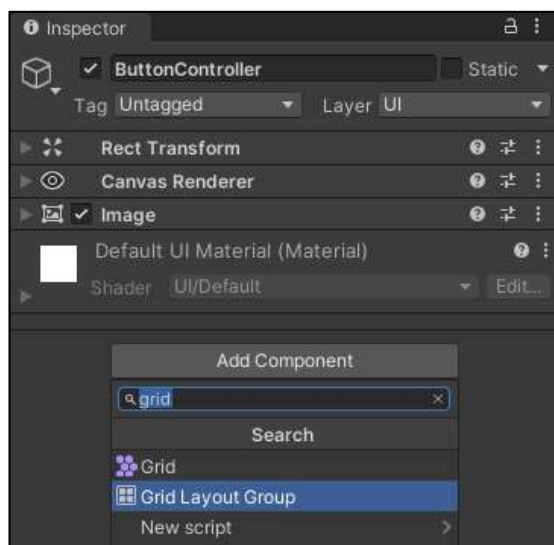


Переименуйте их в “Выбор уровня”, “Область прокрутки” и “Хранилище кнопок”:



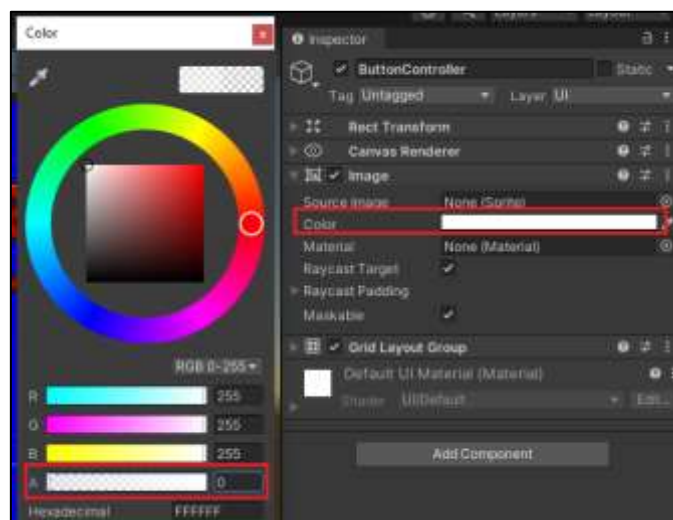
Разместите панель там, где вы хотите отображать меню выбора уровня. Область прокрутки должна располагаться внутри панели меню выбора уровня. Видимая часть хранилища кнопок должна совпадать с областью прокрутки.

Перед добавлением кнопок, добавьте объекту “Хранилище кнопок” компонент Grid Layout Group, это позволит добавлять кнопки в виде ячеек сетки, с заданными размерами и отступами:

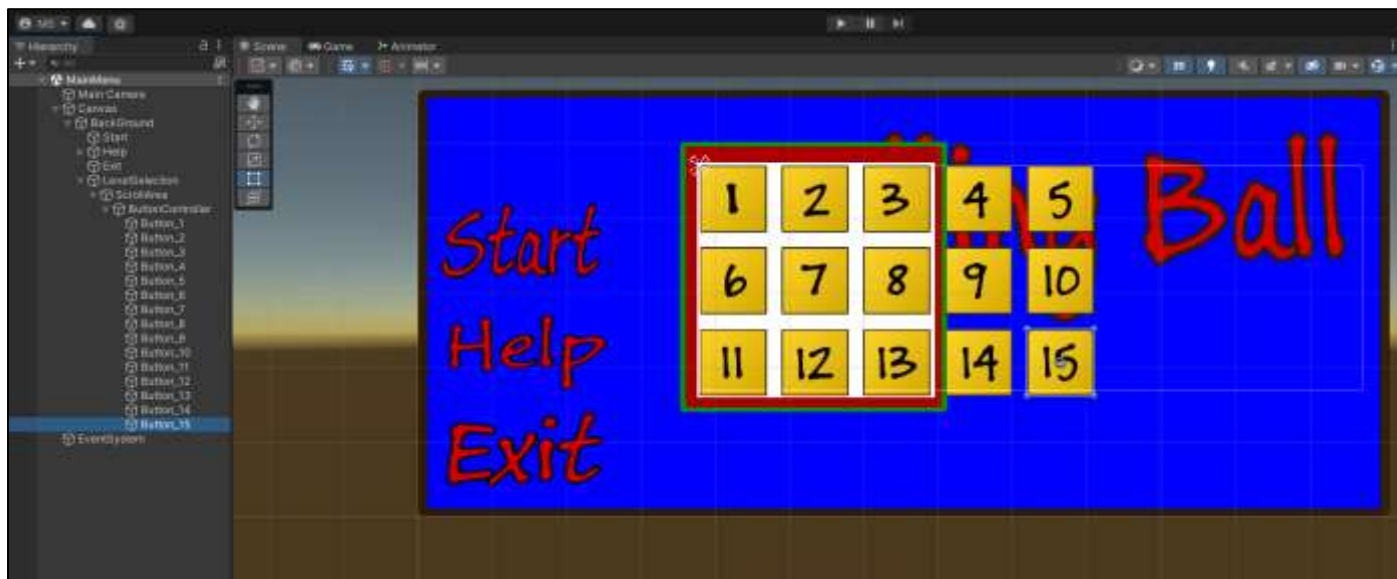


Примечание: желательно что бы порядок следования кнопок совпадал с порядком следования уровней. Для смены порядка размещения кнопок в сетке, используйте параметры Start Corner и Start Axis.

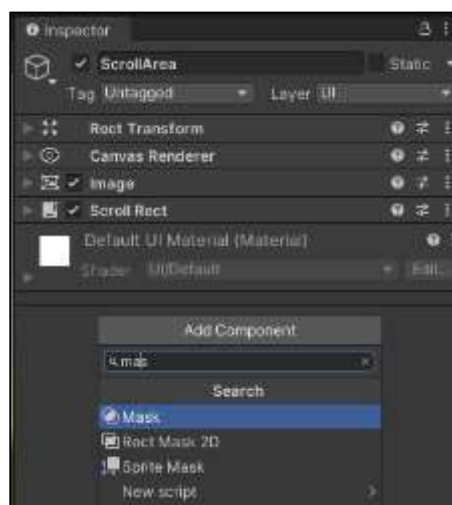
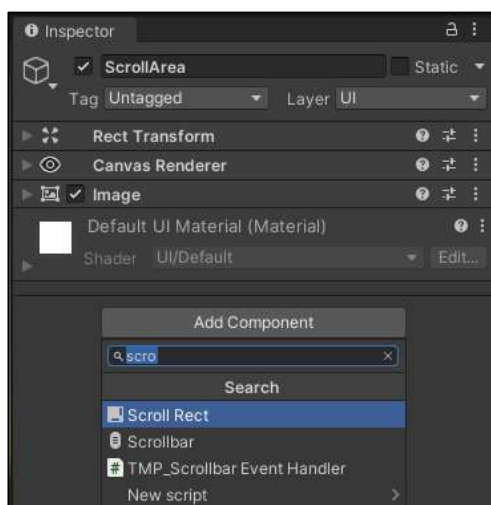
Отключить отображение заднего фона контейнера кнопок, установив значение прозрачности его цвета в 0:



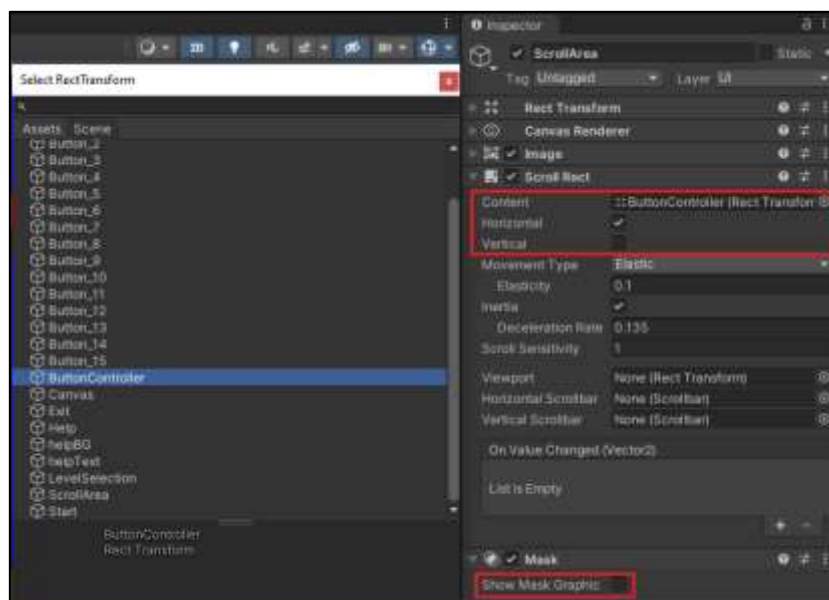
Добавьте в объект “Хранилище кнопок” изображение и добавьте ему компонент Button. Скопируйте его несколько раз, используя сочетание клавиш Ctrl + D. Назначьте изображения с номерами уровней:



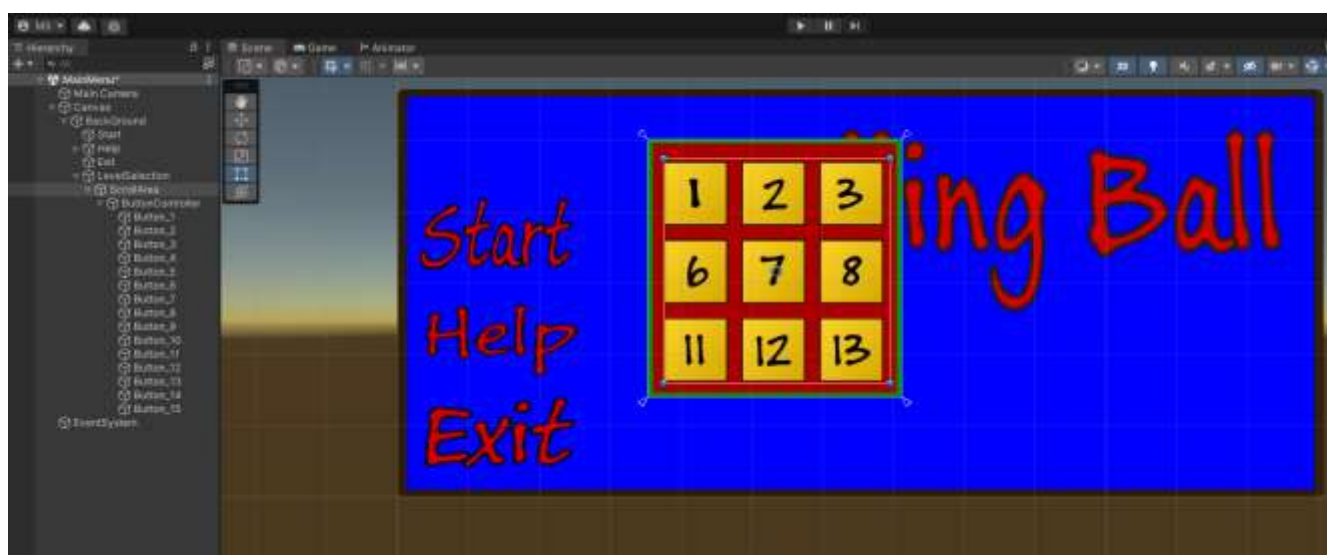
По сути, меню выбора готово, осталось добавить возможность прокрутки. Перейдите к “Области Прокрутки” и добавьте два компонента, Scroll Rect и Mask:



Назначьте хранилище кнопок в качестве контента Scroll Rect и снимите флаги Vertical и Show Mask Graphic:

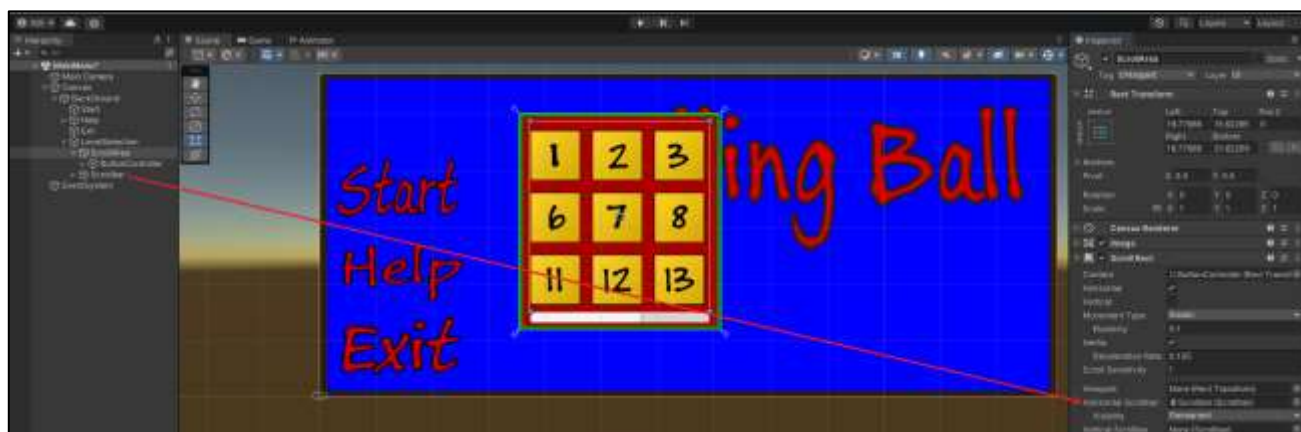


Результат должен выглядеть подобным образом:



Если всё было сделано правильно, в режиме проигрывания, вы сможете перетаскивать кнопки справа налево.

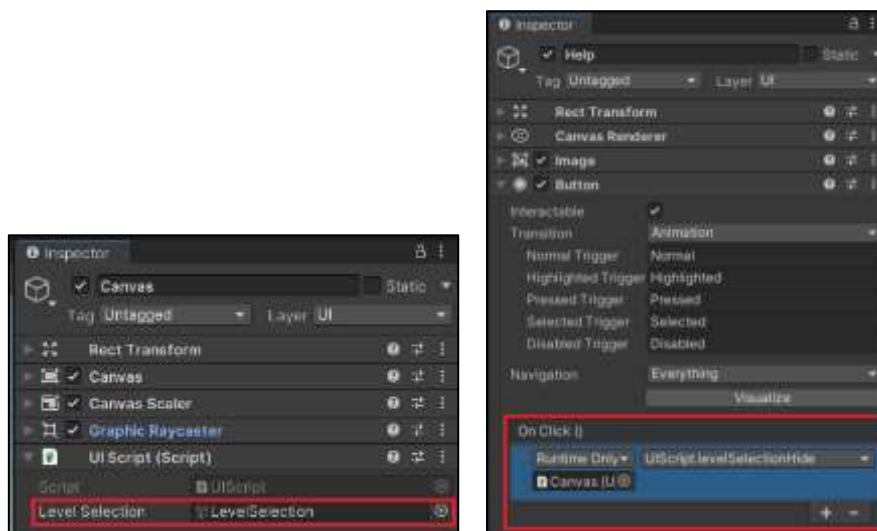
Примечание: Последним действием описания визуальной части меню выбора уровня, будет добавление полосы прокрутки. Для этого, добавьте к панели “Выбор уровня” компонент ScrollBar, а затем, передайте его в качестве параметра в ScrollRect области прокрутки:



Для того, чтобы при нажатии кнопки “Старт” появлялось меню выбора уровня, а при клике в любое другое место оно пропадало, модифицируйте скрипт, прикрепленный к Canvas следующим образом:

```
1 using UnityEngine;
2 using UnityEngine.UI;
3 using UnityEngine.EventSystems;
4
5 public class UIScript : MonoBehaviour
6 {
7     public GameObject levelSelection;
8
9     public void menuStart(Button Start)
10    {
11        EventSystem.current.GetComponent<EventSystem>().SetSelectedGameObject(null); // сброс выделения с кнопки "Старт"
12
13        if (levelSelection.activeSelf == true) // если меню выбора уровня на экране
14            levelSelection.SetActive(false); // скрыть меню выбора уровня
15        else
16            levelSelection.SetActive(true); // показать меню выбора уровня
17    }
18
19    public void levelSelectionHide() // закрытие стартового меню при нажатии на кнопку "Помощь"
20    {
21        levelSelection.SetActive(false); // скрыть меню выбора уровня
22    }
23
24    public void menuExit()
25    {
26        Application.Quit(); // выход из приложения (не работает в режиме проигрывания)
27    }
28 }
```

Не забудьте передать панель меню выбора в скрипт и назначить обработчик нажатия для кнопки “Помощь”:



После выполнения всех описанных действий, можно переходить к описанию скриптов разблокирования и переключения уровней. Скрипт разблокирования уровней будет выглядеть следующим образом:

```
using UnityEngine;
using UnityEngine.UI;

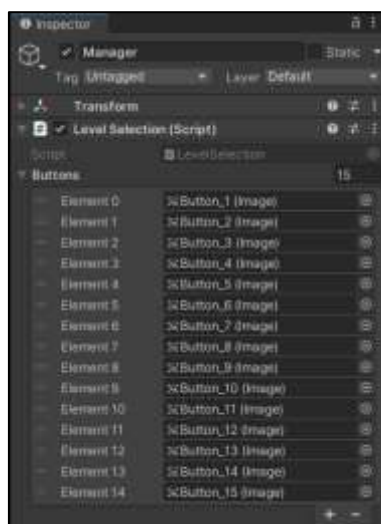
public class LevelSelection : MonoBehaviour
{
    public Image[] buttons; // массив кнопок выбора уровня

    private void Start()
    {
        int curLvl = PlayerPrefs.GetInt("curLvl", 1); // получение номера последнего открытого уровня

        for (int i = 0; i < buttons.Length; i++) // перебор всех кнопок
        {
            if (i + 1 > curLvl) // если номер кнопки больше чем номер открытого уровня
                buttons[i].GetComponent<Button>().interactable = false; // сделать кнопку не активной
        }
    }

    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.Delete)) // если нажата кнопка Delete
        {
            PlayerPrefs.DeleteAll(); // очистить прогресс игрока
        }
    }
}
```

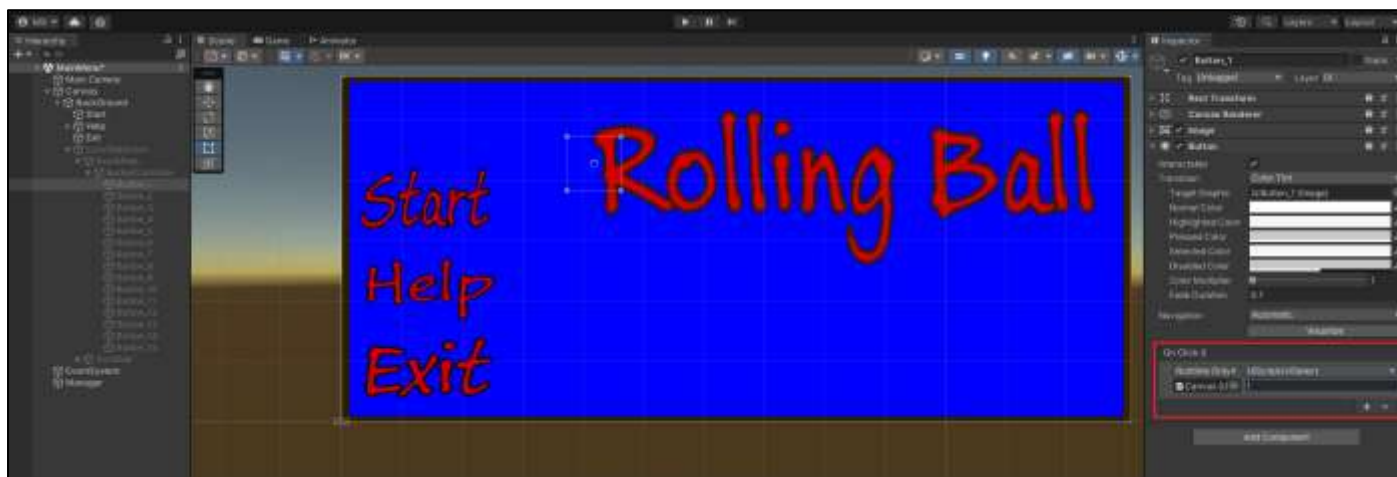
Создайте в сцене пустой объект, назначьте ему скрипт разблокирования, а затем, передайте в него кнопки:



Добавьте к скрипту обработки нажатия кнопок функцию загрузки выбранного уровня:

```
1 using UnityEngine;
2 using UnityEngine.UI;
3 using UnityEngine.EventSystems;
4 using UnityEngine.SceneManagement;
5 using System.Collections;
6
7 public class UIScript : MonoBehaviour
8 {
9     public GameObject levelSelection;
10    public Slider progressBar; // ссылка на прогресс бар
11    public GameObject text; // ссылка на текст о необходимости нажать кнопку для продолжения
12    public GameObject loadingScreen;
13
14    public void menuStart(Button Start) ...
15
16    public void levelSelectionHide() // закрытие стартового меню при нажатии на кнопку "Помощь" ...
17
18    public void menuExit() ...
19
20    public void lvlSelect(int lvl)
21    {
22        loadingScreen.SetActive(true); // делает активным загрузочный экран
23        Time.timeScale = 1f; // запускает время с нормальной скоростью
24        StartCoroutine(LoadLevelAsinc(lvl)); // запуск фонового процесса для загрузки сцены
25    }
26
27    private IEnumerator LoadLevelAsinc(int lvl) // функция для загрузки новой сцены в фоновом режиме ...
28    {
29    }
30 }
```

Назначьте функцию для кнопок выбора уровня, указав номер загружаемого при нажатии уровня в качестве параметра:



Модифицируйте скрипт, описывающий движение игрока по уровню, добавив номер уровня, к которому нужно перейти по завершению текущего:

```

1  using System.Collections;
2  using UnityEngine;
3  using UnityEngine.UI;
4  using UnityEngine.SceneManagement;
5
6
7  public class Movement : MonoBehaviour // класс, описывающий поведение сферы
8  {
9      public float speed; // переменная, определяющая силу воздействия на сферу
10     public int objects; // количество собираемых объектов на уровне
11     public int nextLVL = 0; // номер следующего уровня (по умолчанию, переход в стартовое меню)
12
13     public Slider progressBar; // ссылка на прогресс бар
14     public GameObject text; // ссылка на текст о необходимости нажать кнопку для продолжения
15     public GameObject loadingScreen;
16
17
18     private Rigidbody rb; // переменная для получения ссылки на физическую модель сферы
19     private int count; // счётчик собранных объектов
20     public Text countText; // ссылка на элемент интерфейса для подсчёта собранных объектов
21     public Text winText; // ссылка на текст с сообщением о победе

```

Модифицируйте условие победы так, чтобы при достижении победы начиналась загрузка следующего уровня, при этом, если был пройден последний открытый уровень, открывался следующий:

```

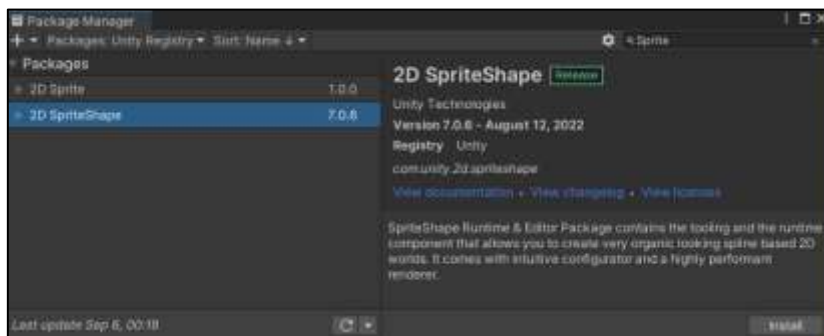
57 // инициализация элемента интерфейса для отображения
58 void SetCountText()
59 {
60     countText.text = "Count: " + count.ToString(); // обновление счётчика собранных объектов
61     if (count >= 1) // если все объекты собраны
62     {
63         winText.text = "You Win!"; // сообщение о победе
64
65         loadingScreen.SetActive(true); // делает активным загрузочный экран
66         Time.timeScale = 1f; // запускает время с нормальной скоростью
67         StartCoroutine(LoadLevelAsinc(nextLVL)); // запуск фонового процесса для загрузки сцены
68
69         if (nextLVL > PlayerPrefs.GetInt("curLVL")) // если мы переходим на уровень, на котором ещё не были
70             PlayerPrefs.SetInt("curLVL", nextLVL); // установить следующий уровень как последний открытый
71     }
72 }
73
74 private IEnumerator LoadLevelAsinc(int lvl) // функция для загрузки новой сцены в фоновом режиме

```

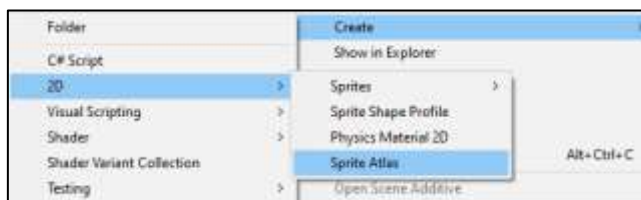
Оптимизация

В связи с особенностями реализации отрисовки двумерных изображений в Unity, использование множества небольших изображений может уменьшить производительность вашего приложения. Уменьшить накладные расходы на отрисовку отдельных изображений можно упаковав их в Sprite Atlas.

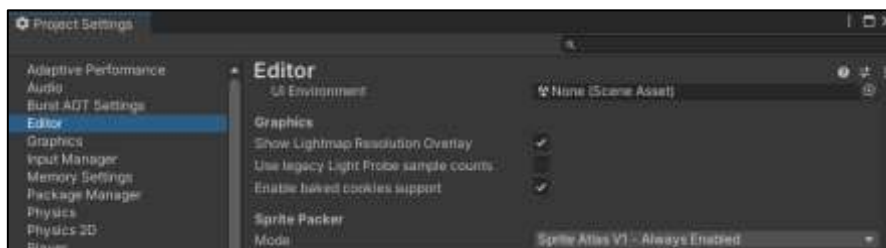
Что бы получить доступ к Sprite Atlas, установите пакет 2D SpriteShape:



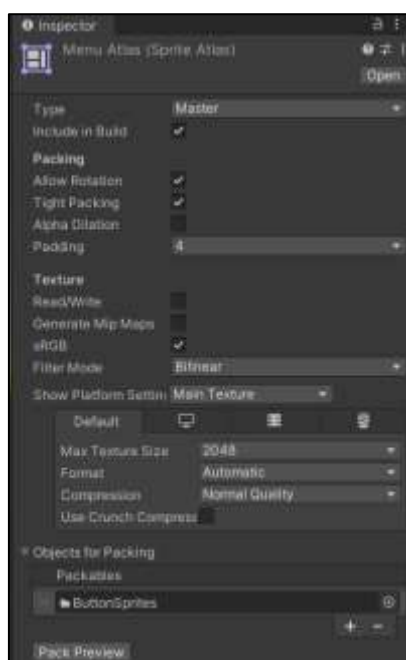
После чего, в области Project, создайте Sprite Atlas:



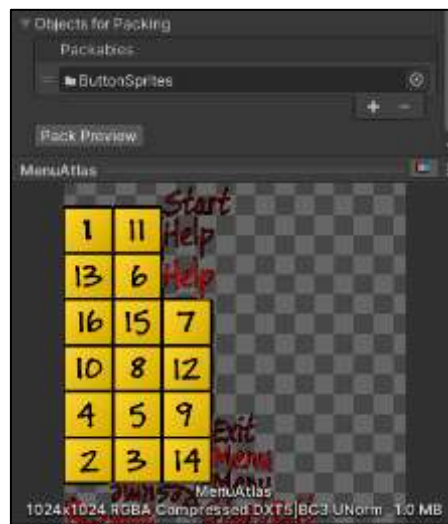
В настройках проекта, в разделе Editor -> Sprite Pecker, установите Mode – Always Enabled:



В самом атласе, в объектах для упаковки, укажите папку, содержащую спрайты кнопок:



После нажатия кнопки PackPreview будет сгенерирован атлас спрайтов подобного вида:



В случае, если при генерации атласа возникают накладки изображений или другие артефакты, попробуйте увеличить Padding, отключить Tight Packing и т.д.

Задания:

1. Реализуйте плавное появление панелей выбора уровня, помощи и внутри игрового меню, используя анимацию компонент и их настройки прозрачности.
2. Добавьте экраны победы и поражения перед загрузкой следующего, или перезапуском текущего уровня.
3. Вынесите экран загрузки уровня в отдельную сцену.
4. Реализуйте систему оценки прохождения уровней. (от 1 до 3 звёзд в зависимости от скорости прохождения уровня)

Отчёт.

В качестве отчёта вы должны предоставить документ, содержащий следующие пункты:

1. Цель и задачи лабораторной работы.
2. Краткое описание действий, которые вы выполняли для решения поставленных в лабораторной работе заданий с картинками, представляющими работу приложения.
3. Краткий вывод о проделанной вами работе
4. Ссылка на облачный диск/github с архивом в котором находится **написанные вами скрипты**.

Приложение: обмен данными между скриптами

Обмен данными между скриптами может быть реализован через класс, не привязанный к какому-либо объекту:

```
public class GlobalData // класс данных доступный из любого скрипта в проекте
{
    private static GlobalData instance = null; // ссылка на экземпляр класса

    public static GlobalData SharedInstance // метод доступа к классу
    {
        get
        {
            if (instance == null) // если объект класса не существует
            {
                instance = new GlobalData(); // создать объект
            }
            return instance; // вернуть ссылку на объект
        }
    }

    public float data; // поле хранящее данные
}
```

Тогда, доступ к полю data из других скриптов, будет осуществляться следующим образом:

```
float data = GlobalData.SharedInstance.data; // доступ к данным
```