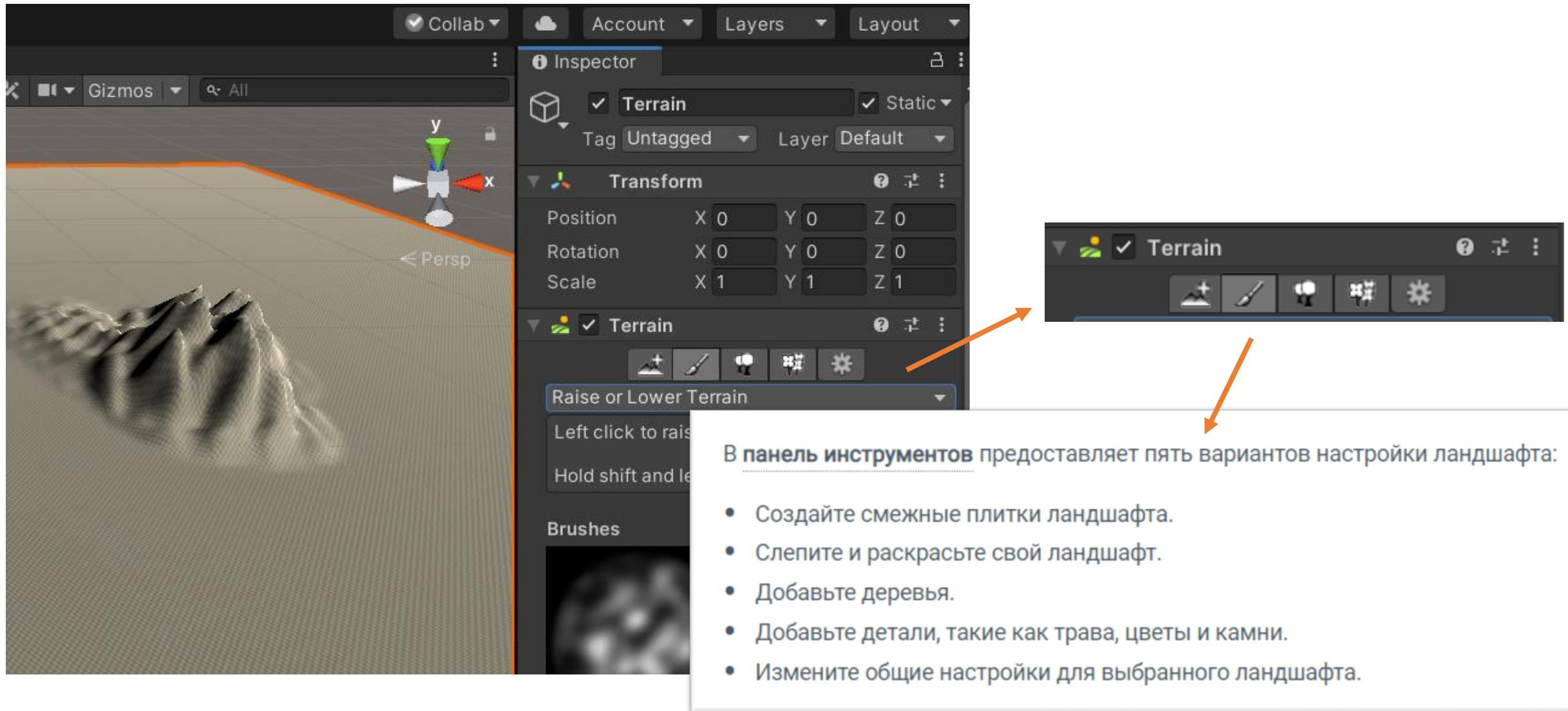


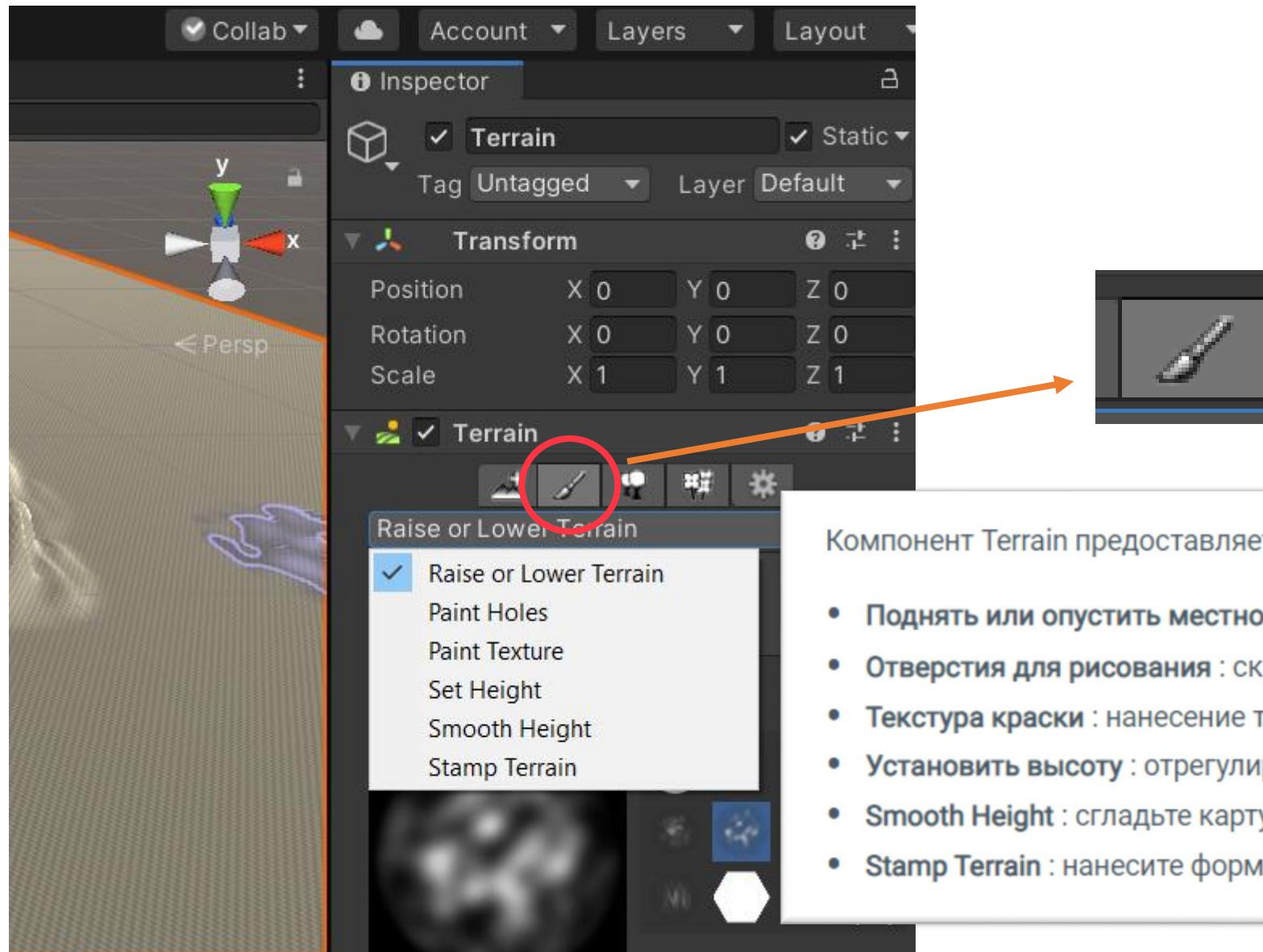
- Terrain
- Particle Systems
- Звуки
- Character Controller
- Пользовательский интерфейс
- Оптимизация проекта

# Создание и редактирование Terrain'ов

GameObject > Create Other > Terrain



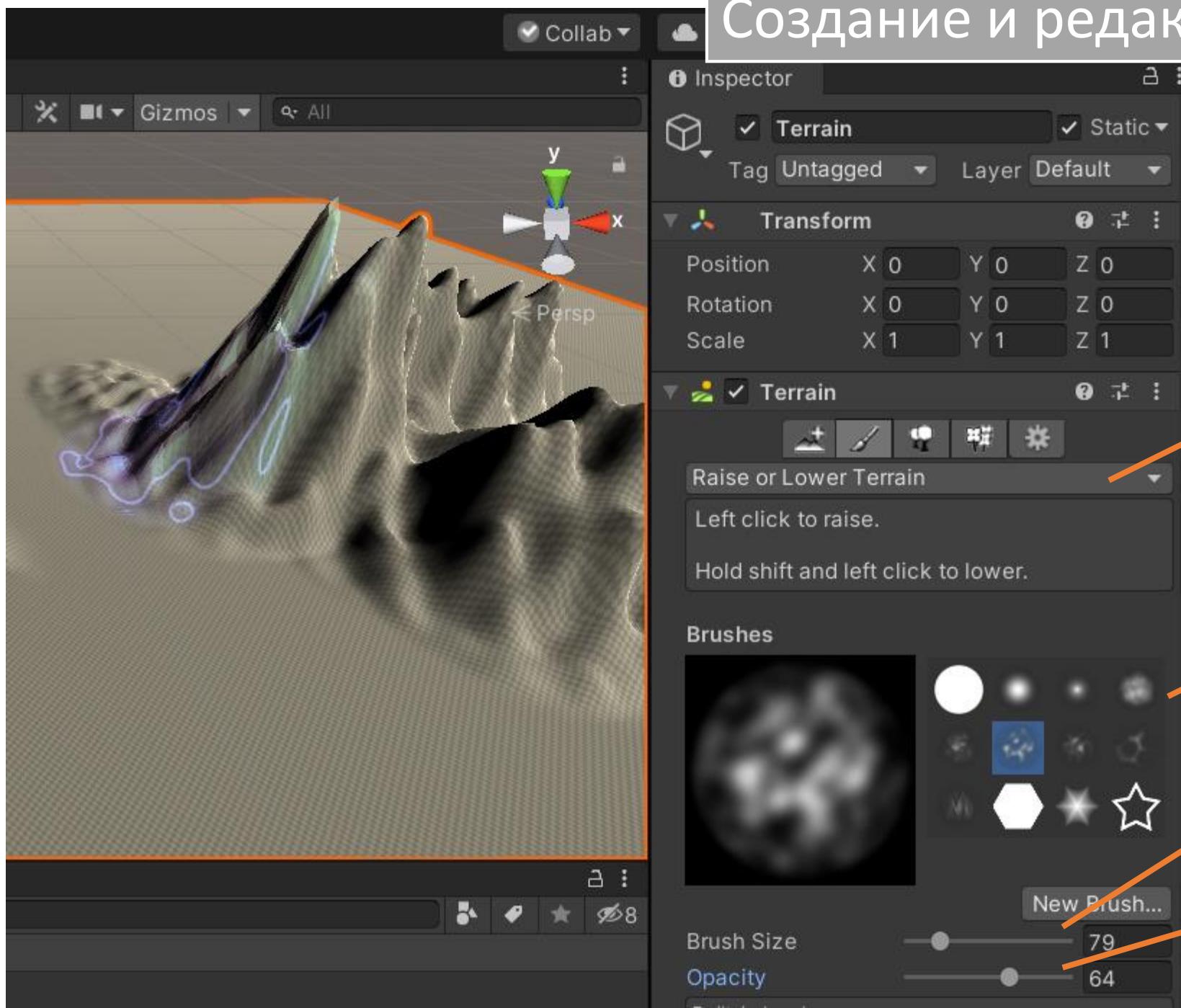
# Создание и редактирование Terrain'ов



Компонент Terrain предоставляет шесть различных инструментов:

- Поднять или опустить местность : нарисуйте карта высот кистью.
- Отверстия для рисования : скрыть части ландшафта.
- Текстура краски : нанесение текстуры поверхности.
- Установить высоту : отрегулируйте карту высот до определенного значения.
- Smooth Height : сгладьте карту высот, чтобы смягчить детали ландшафта.
- Stamp Terrain : нанесите форму кисти поверх текущей карты высот.

# Создание и редактирование Terrain'ов



Режим рисования карты высот

Форма кисти

Размер кисти

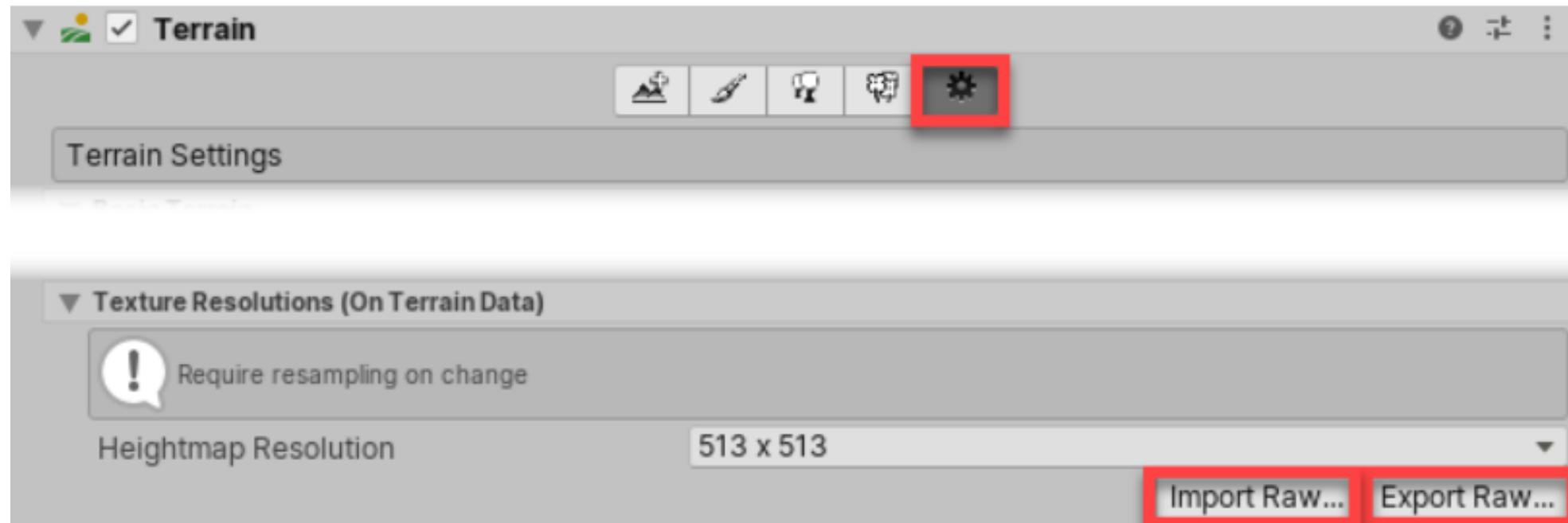
Сила кисти

## Импорт и экспорт карт высот

Инструменты ландшафта, влияющие на высоту используют текстуру в градациях серого, называемую **карты высот**. Можно импортировать и экспортировать карты высот в редактор Unity.

Можно использовать приложения 3D-моделирования, а затем импортировать ваш Terrain в Unity как карту высот. **Можно использовать онлайн-ресурсы для импорта реальных данных: <http://terrain.party/>**

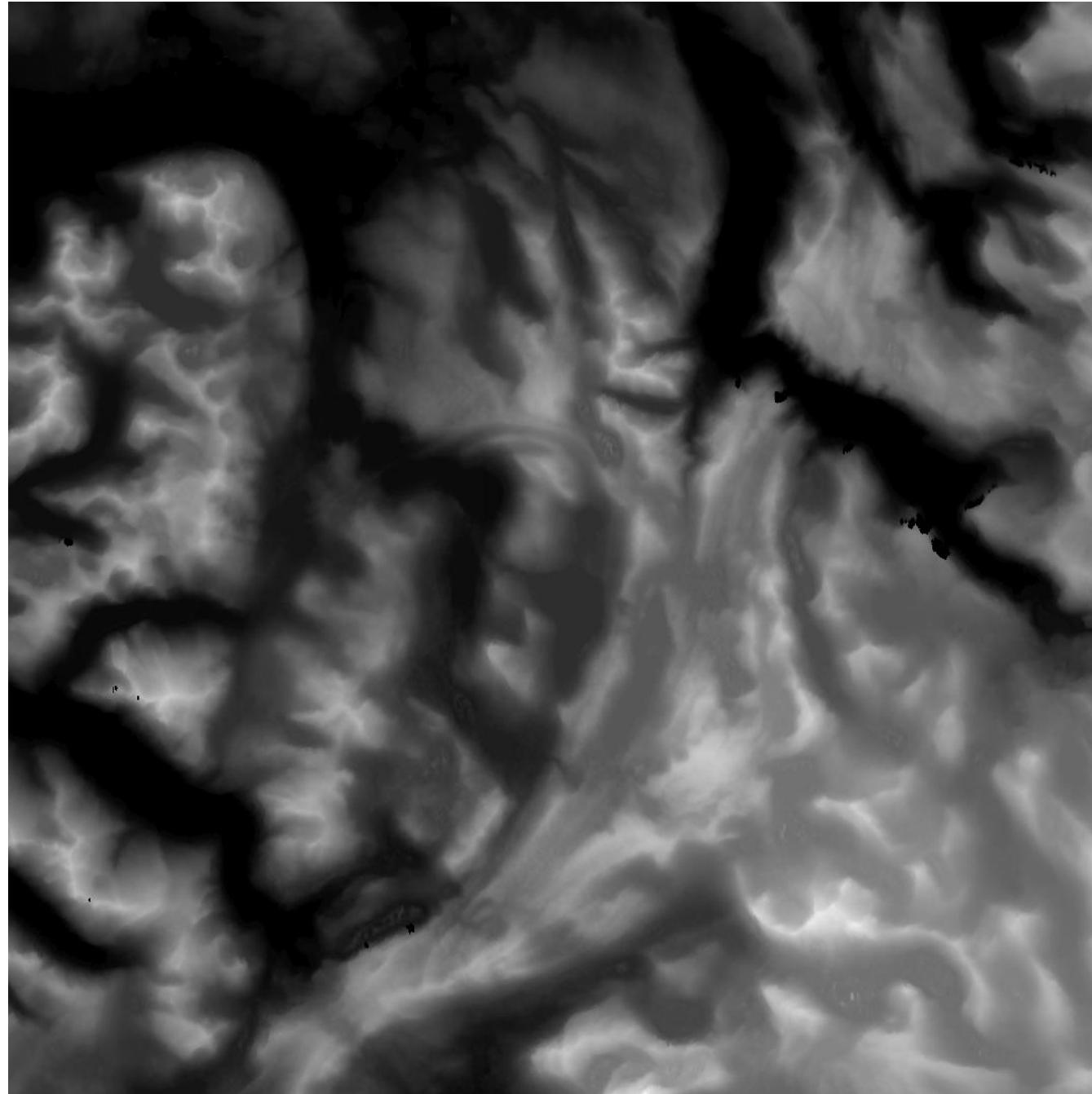
Хорошая практика - хранить карты высот в виде файлов RAW. В файле RAW используется 16-разрядный формат шкалы серого, совместимый с большинством редакторов изображений и альбомной ориентации. Редактор Unity позволяет импортировать и экспортировать файлы карты высот RAW для ландшафта.



Где то в  
Финляндии

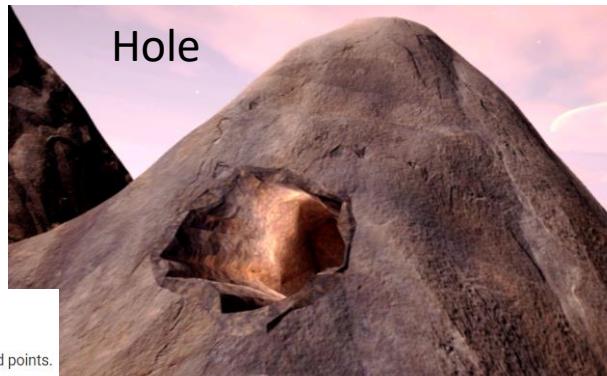
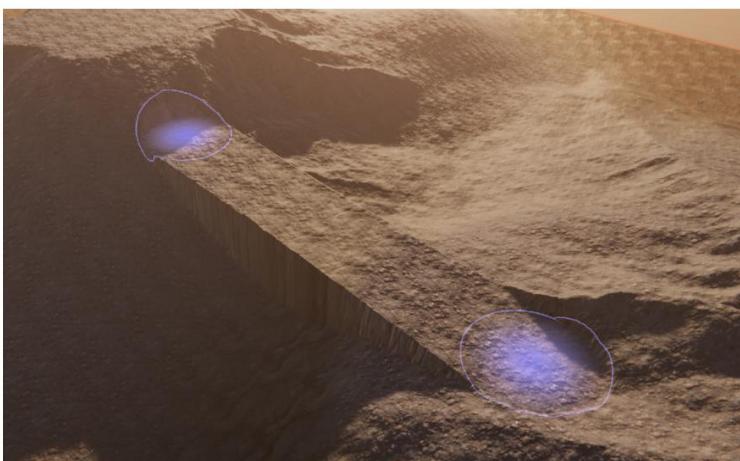
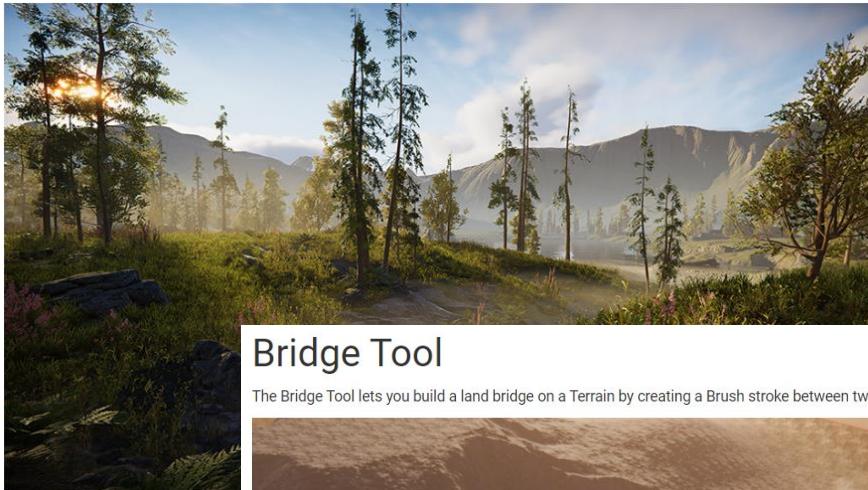
из

<http://terrain.party/>



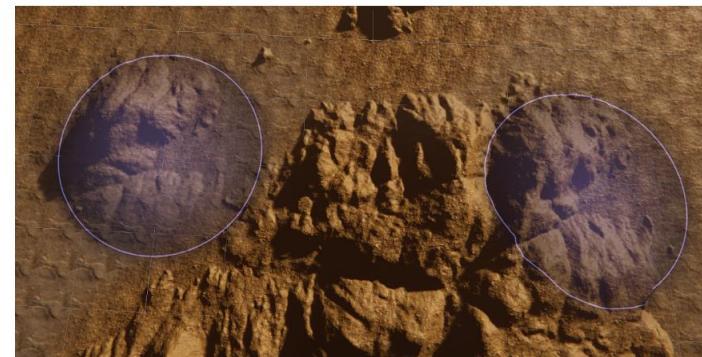
# Пакет Terrain tools

Для получения больших возможностей для создания окружающего мира можно воспользоваться пакетом Terrain tools, который загружается через диспетчер пакетов



Clone Tool

The Clone Tool duplicates Terrain from one region to another.



Terrace Tool

The Terrace tool transforms Terrain into a series of flat areas resembling steps.



<https://docs.unity3d.com/Packages/com.unity.terrain-tools@5.0/manual/index.html>

# Процедурная генерация ландшафта

## TerrainData

class in UnityEngine / Наследует от: [Object](#)

### Описание

Класс TerrainData содержит карты высот, позиции детализирован-

Компонент [Terrain](#) ссылается к данным ландшафта и визуализирует

### Переменные

[alphamapHeight](#) Высота альфа карты.

[alphamapLayers](#) Количество слоев альфа карты.

[alphamapResolution](#) Разрешение альфа карты.

[alphamapTextures](#) Текстуры альфа карты, используемы

[alphaWidth](#) Ширина альфа карты

```
public int width = 256;
public int height = 256;

void Start ()
{
    Terrain terrain = GetComponent<Terrain>();
    terrain.terrainData = GenerateTerrain();
}

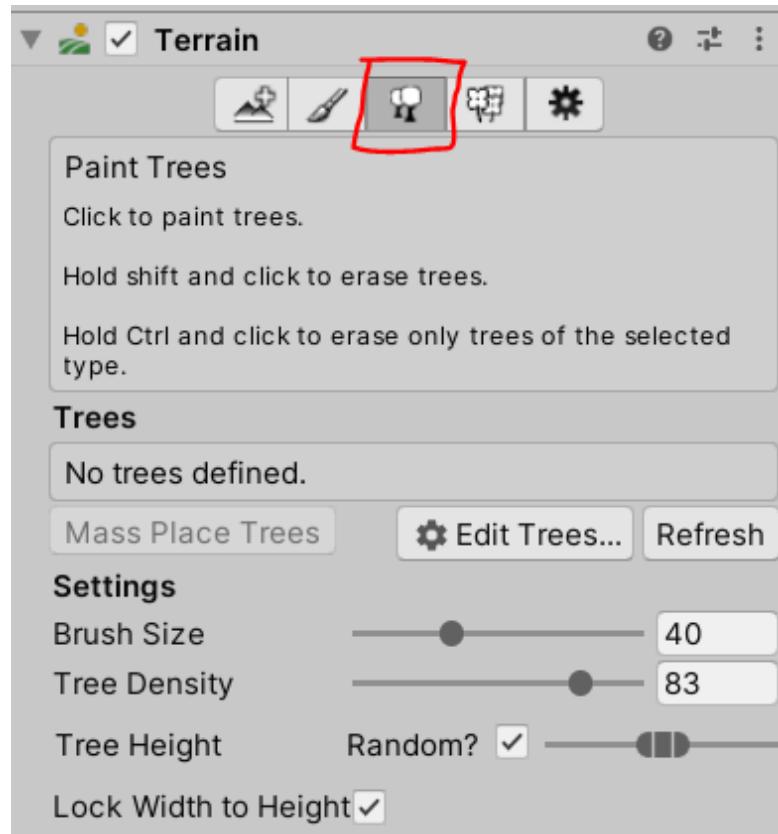
TerrainData GenerateTerrain ()
{
    |
}

float CalculateHeight (int x, int y)
{
    float xCoord = x / width * scale;
    float yCoord = y / height * scale;

    return Mathf.PerlinNoise(xCoord, yCoord);
}
```

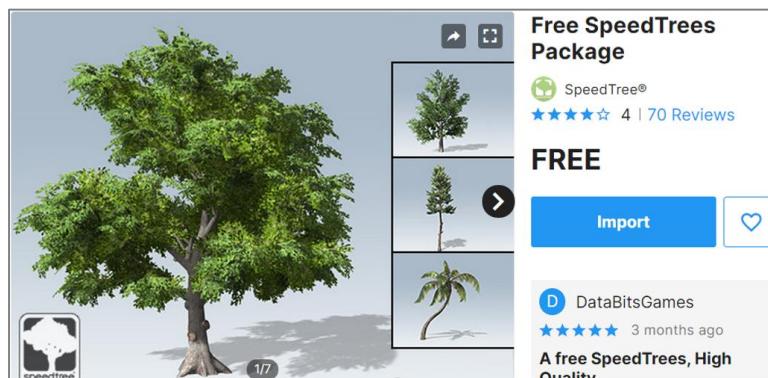
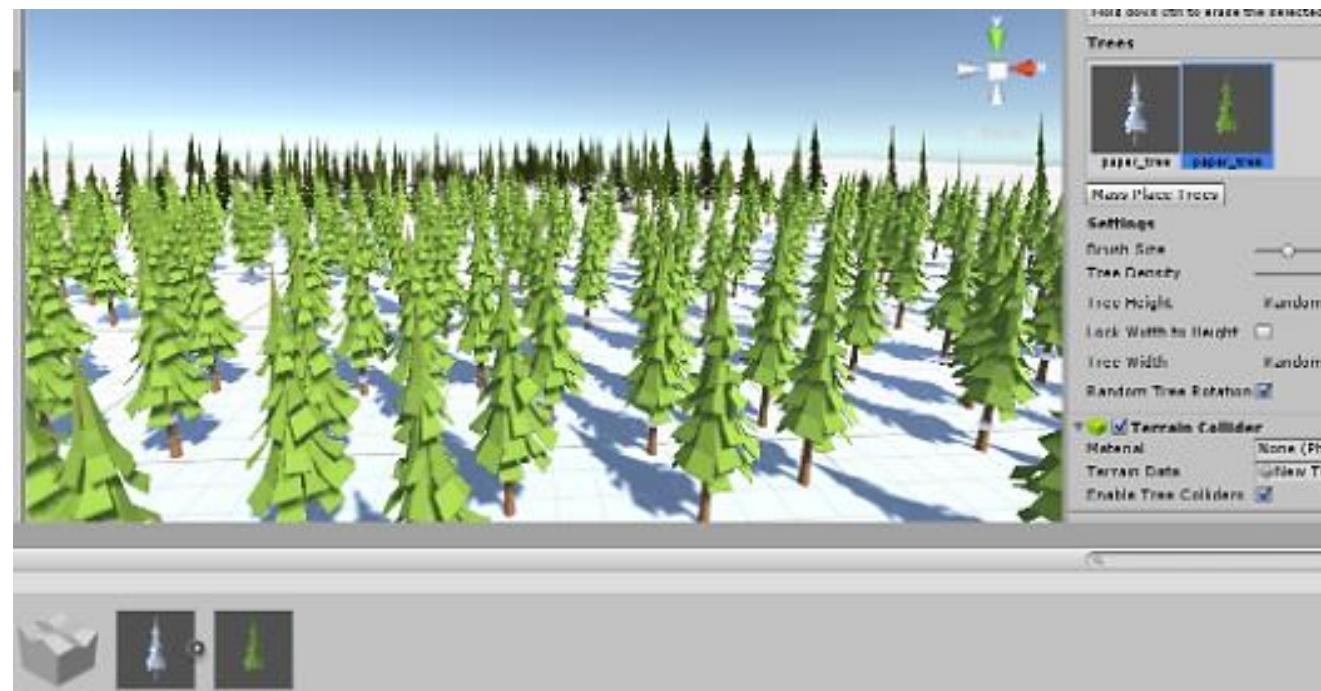
Кому интересно тема генерации ландшафтов, советую начать с изучения различных алгоритмы генерации ландшафтов, например здесь <https://habr.com/ru/post/111538/>

# Деревья



Инструмент **PaintTrees** позволяет наносить деревья на терраин группами, задавая в параметрах радиус области нанесение, плотность и высоту деревьев.

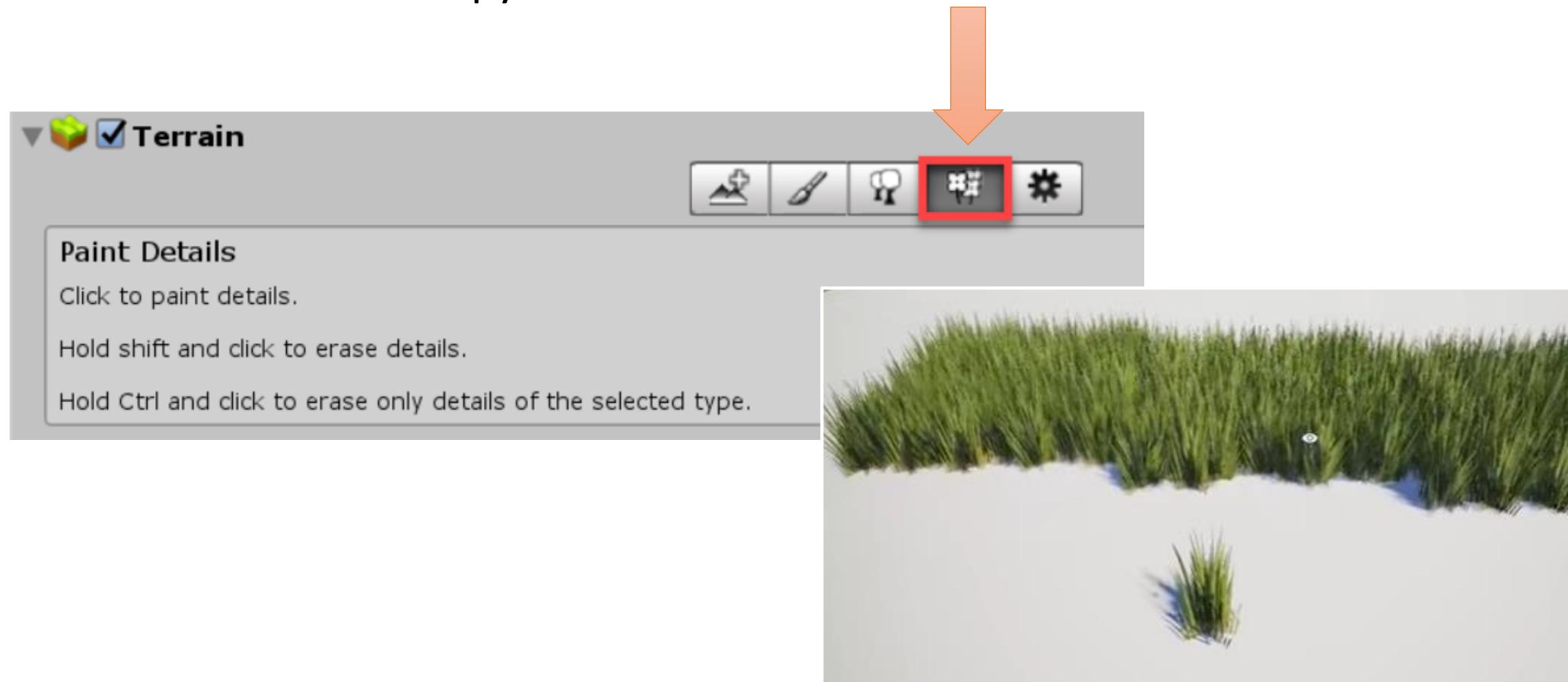
Изначально, у `terrain'`а не будет доступных деревьев, но если вы нажмёте на кнопку `Edit Trees` и в выпавшем меню выберите `Add Tree`, вы увидите окно, в котором можно выбрать ассет дерева из вашего проекта.



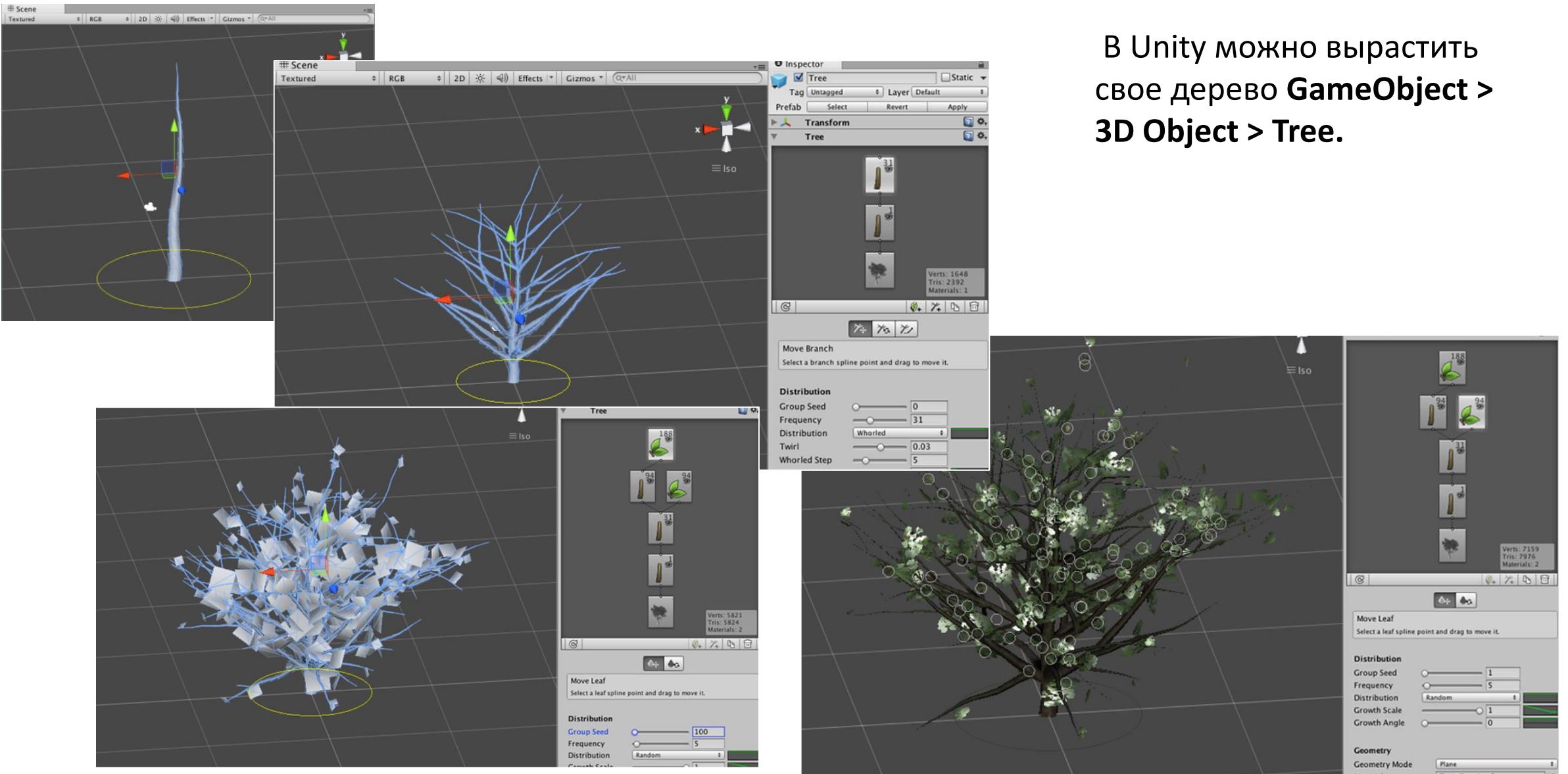
Можно импортировать ассеты деревьев из AssetStore.

**Трава и прочие мелкие объекты, например, камни**

Мелкие объекты тоже необходимо импортировать из ассетов и воспользоваться инструментом **PaintDetails**

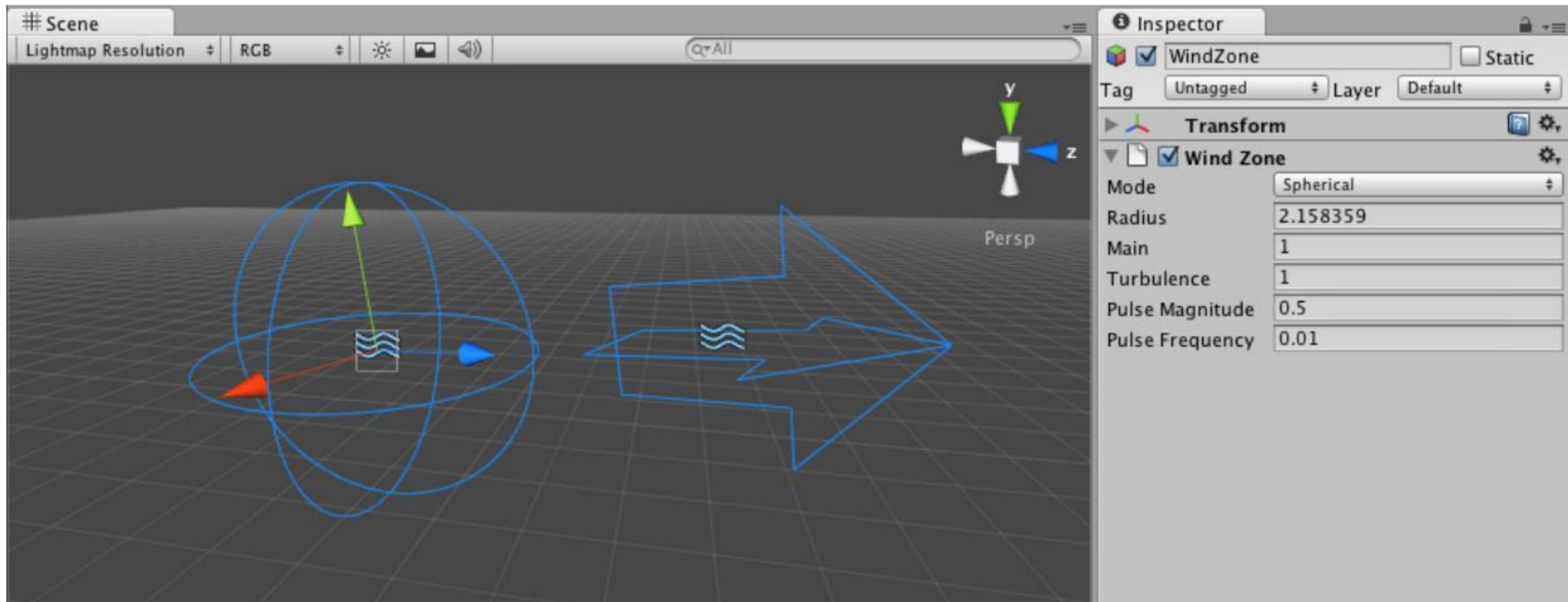


# Объект Tree (Дерево)



В Unity можно вырастить  
свое дерево **GameObject >**  
**3D Object > Tree.**

# Зоны ветра для деревьев



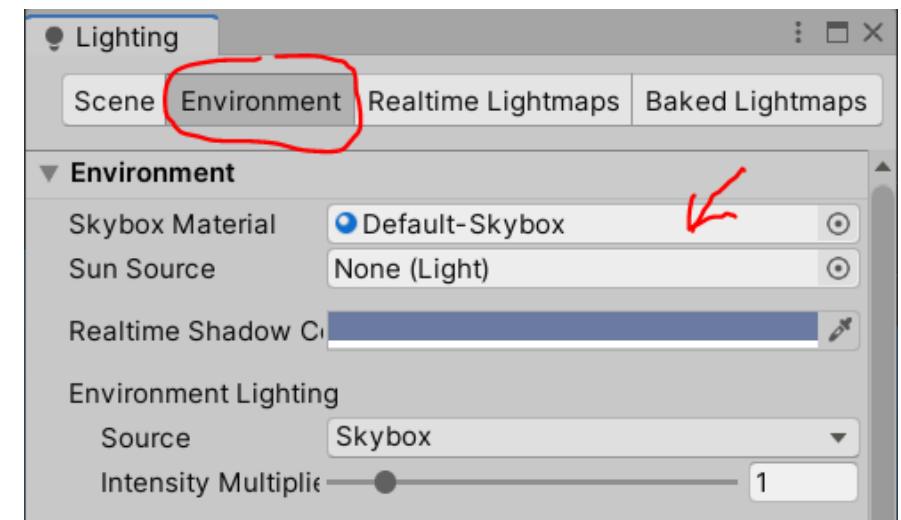
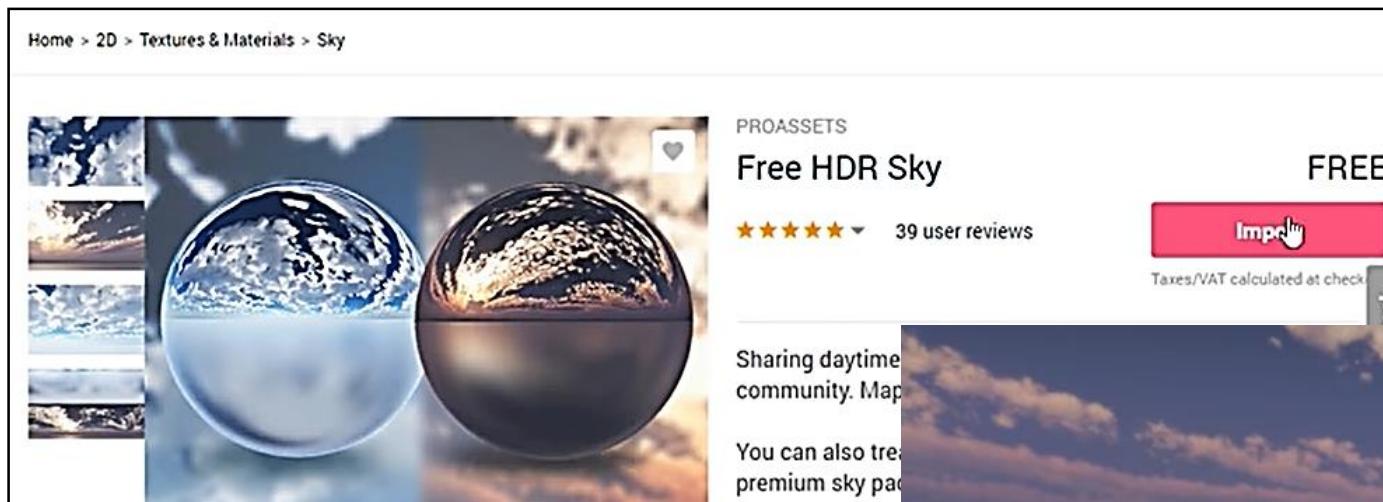
# Небо

Небо — это тип фона, который камера рисует до того, как отрисует кадр. Само небо может содержать что угодно, например облака, горы, здания и другие недоступные объекты, чтобы создать иллюзию отдаленного трехмерного окружения.

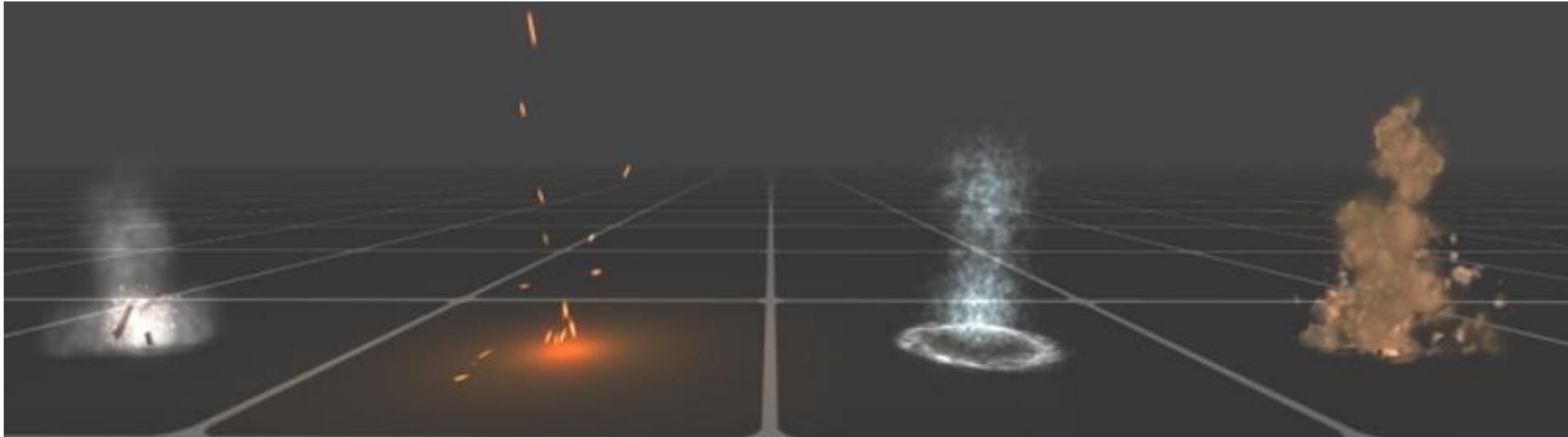
Установить желаемое небо можно скачав выбранный ассет и установив материал Skybox в окне

**Window → Rendering → Lighting.**

*Выберите любой ассет для неба в AssetStore*

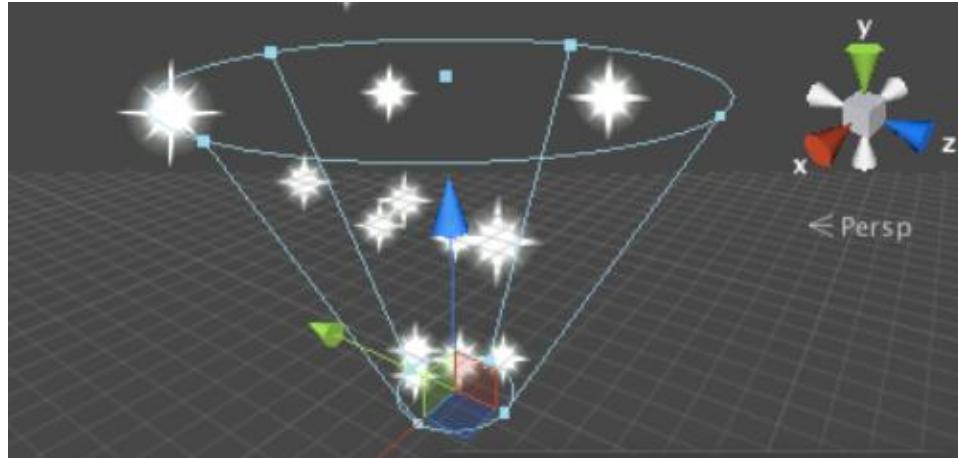


# Particle Systems (Система частиц)

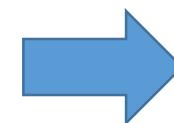


Компонент **Particle Systems** моделирует такие объекты как дым, облака, дождь, пламя путем генерации и анимации большого числа маленьких 2D – изображений.

# Particle Systems (Система частиц)



Particle System имеет много свойств, и для удобства Инспектор организует их в сворачиваемые секции, называемые «модулями».



Particle System

Open Editor...

Particle System

Emission

Shape

Velocity over Lifetime

Limit Velocity over Lifetime

Inherit Velocity

Force over Lifetime

Color over Lifetime

Color by Speed

Size over Lifetime

Size by Speed

Rotation over Lifetime

Rotation by Speed

External Forces

Noise

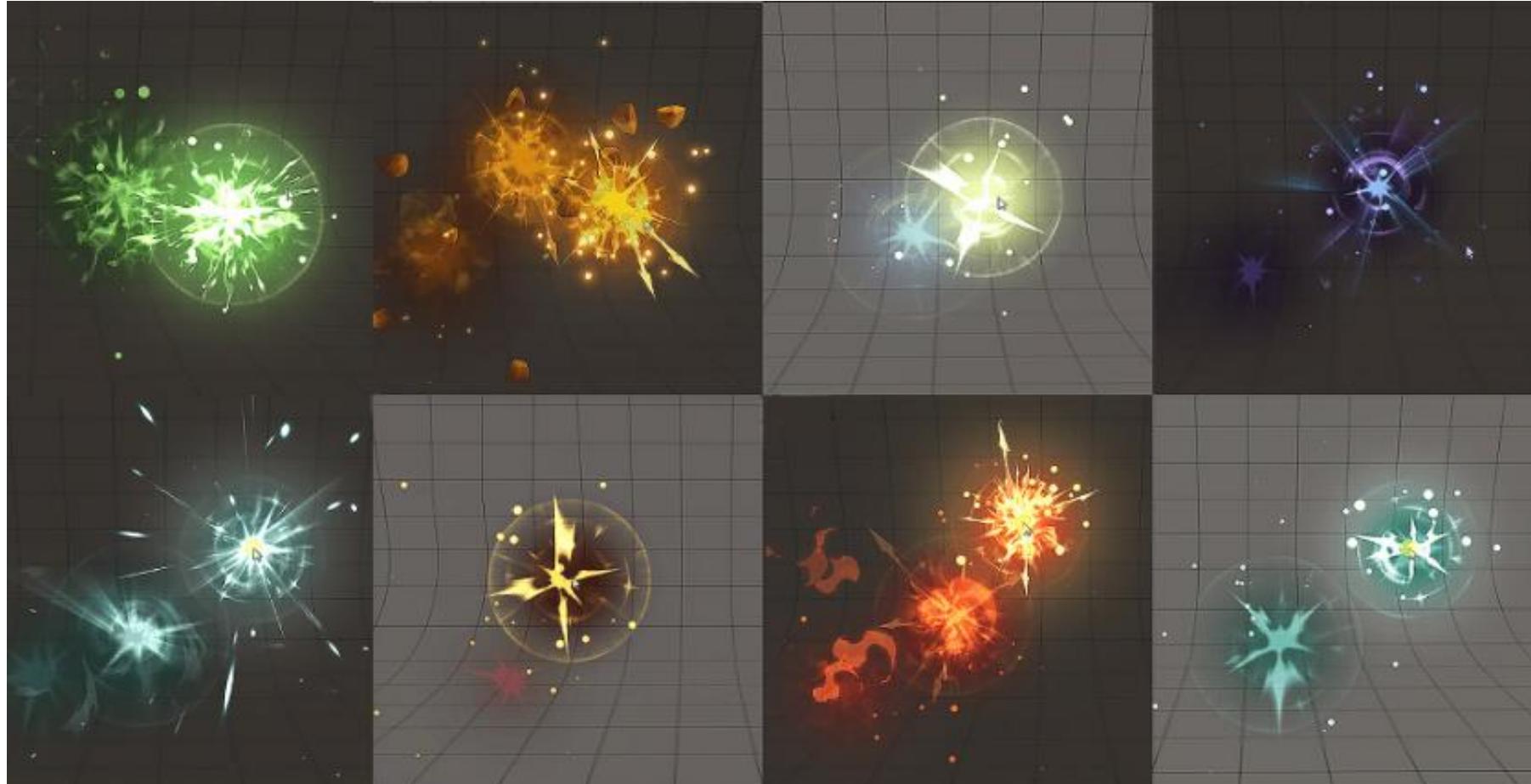
Collision

Triggers

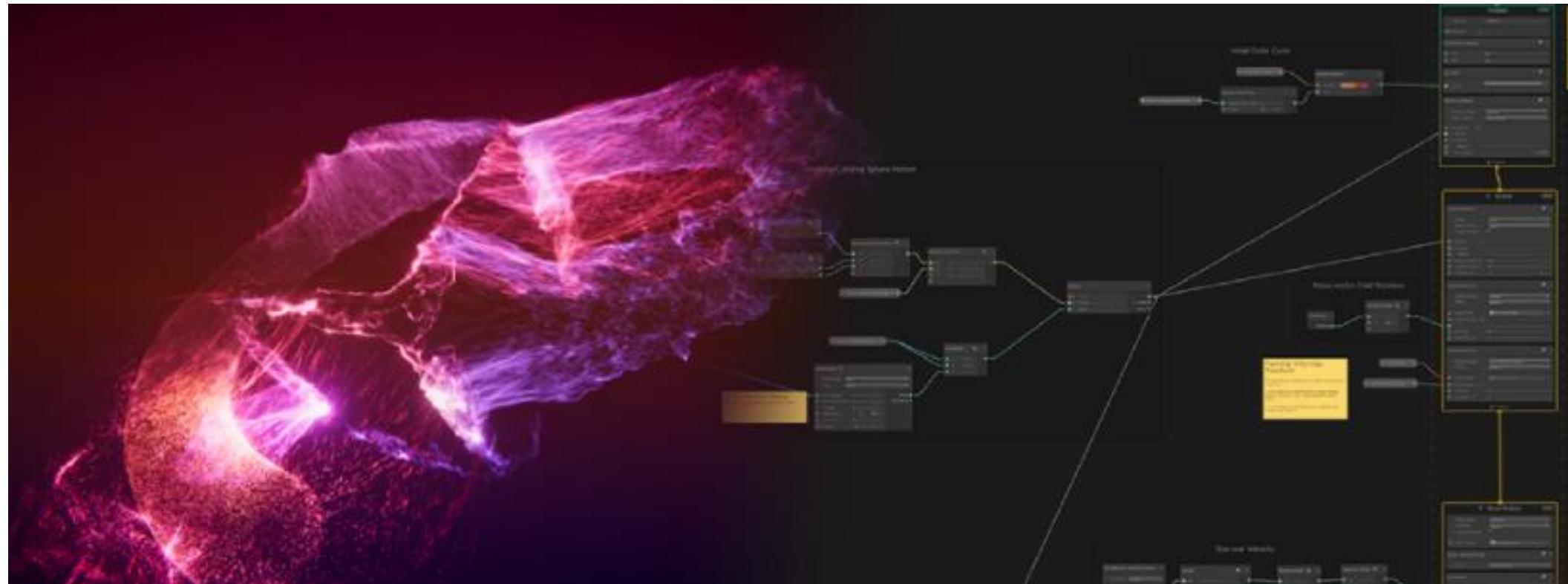
Sub Emitters

Texture Sheet Animation

Частицы представляют собой небольшие, простые изображения, которые отображаются и перемещаются в больших количествах системой частиц. Каждая частица представляет собой небольшую часть жидкой или аморфной сущности, и эффект всех частиц вместе создает впечатление полной сущности.



Visual Effect Graph - это пакет, который можно использовать для создания крупномасштабных визуальных эффектов для проекта Unity. График визуальных эффектов имитирует поведение частиц на графическом процессоре, что позволяет ему моделировать гораздо больше частиц, чем встроенная система частиц Particle Systems . Вся логика эффектов строится и храниться в виде графов.



Посмотреть можно здесь

<https://www.youtube.com/watch?v=Lkw2D-y2XmE>

# Звуки в Unity

---

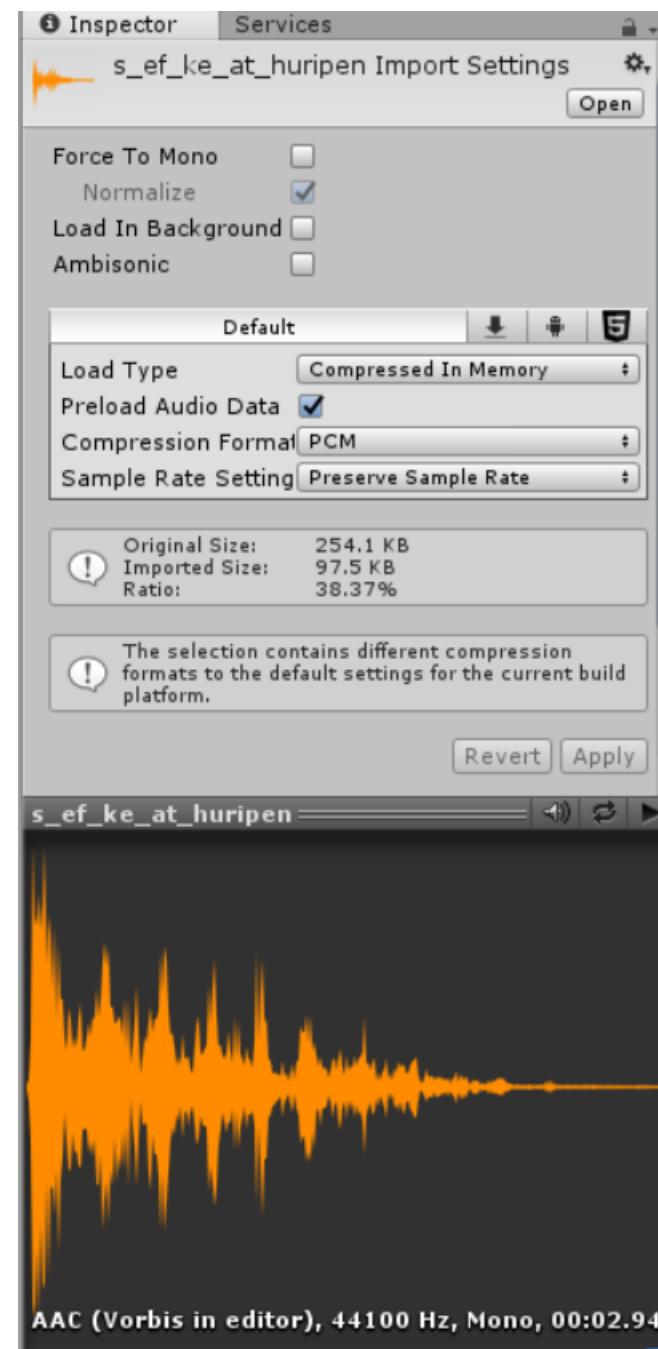
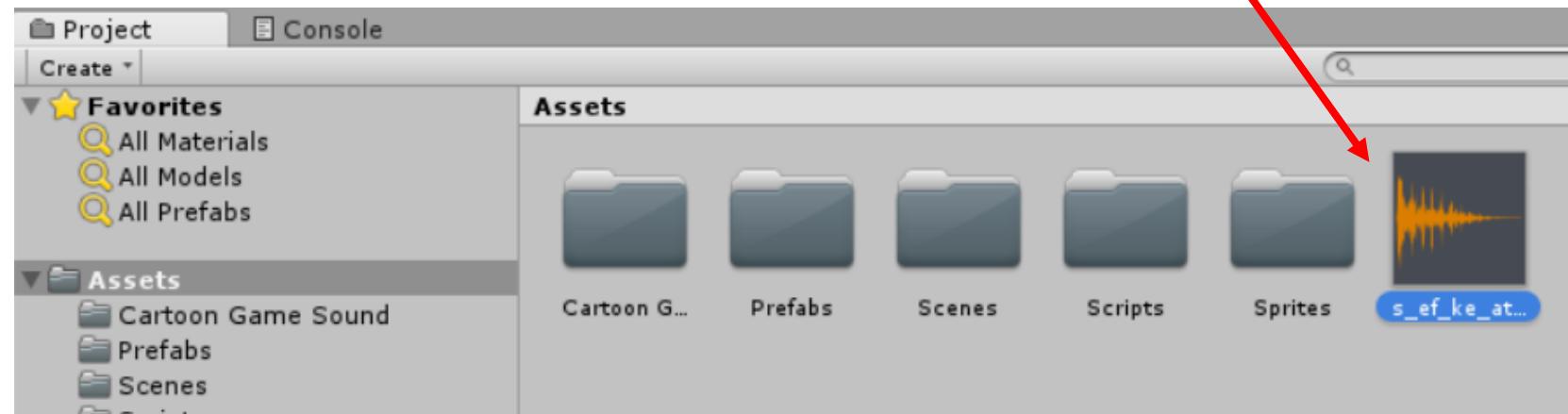
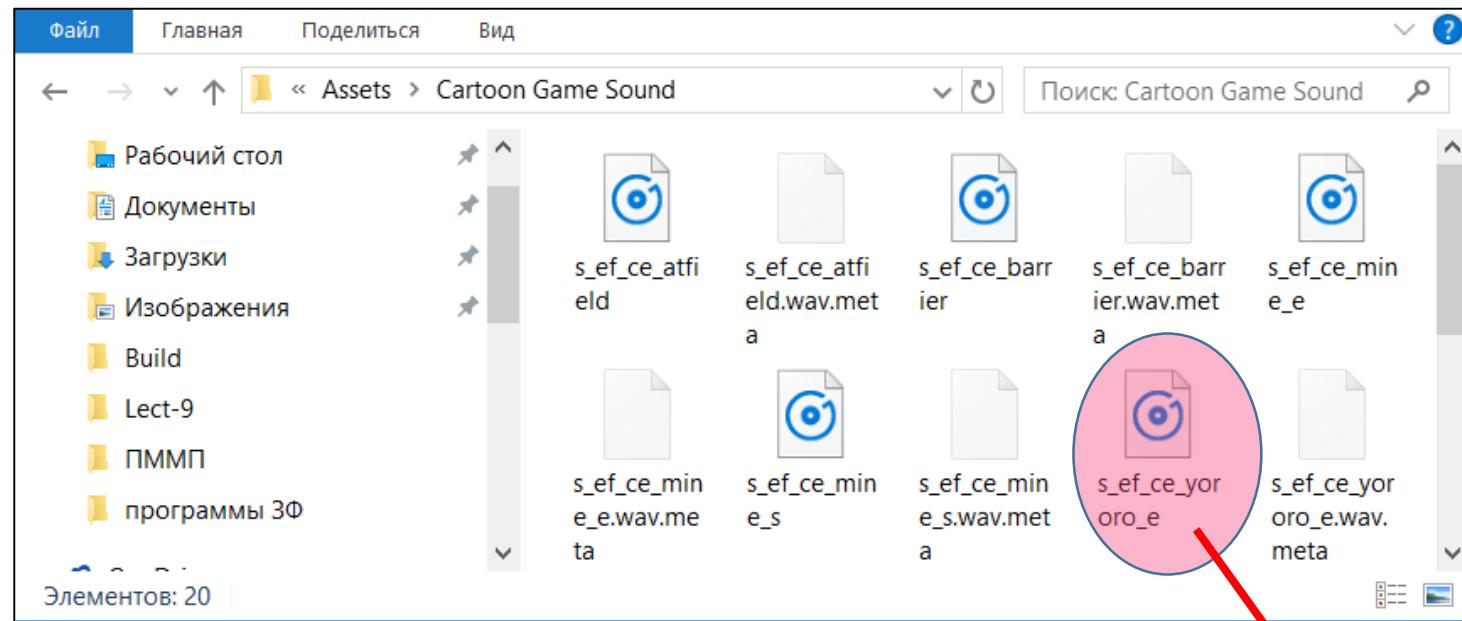
Аудиосистема Unity гибкая и мощная.

Она может импортировать большинство стандартных аудио форматов и **имеет сложные функции для воспроизведения звуков в 3D пространстве, с опциональными эффектами, такими как применение эха и фильтрации.**

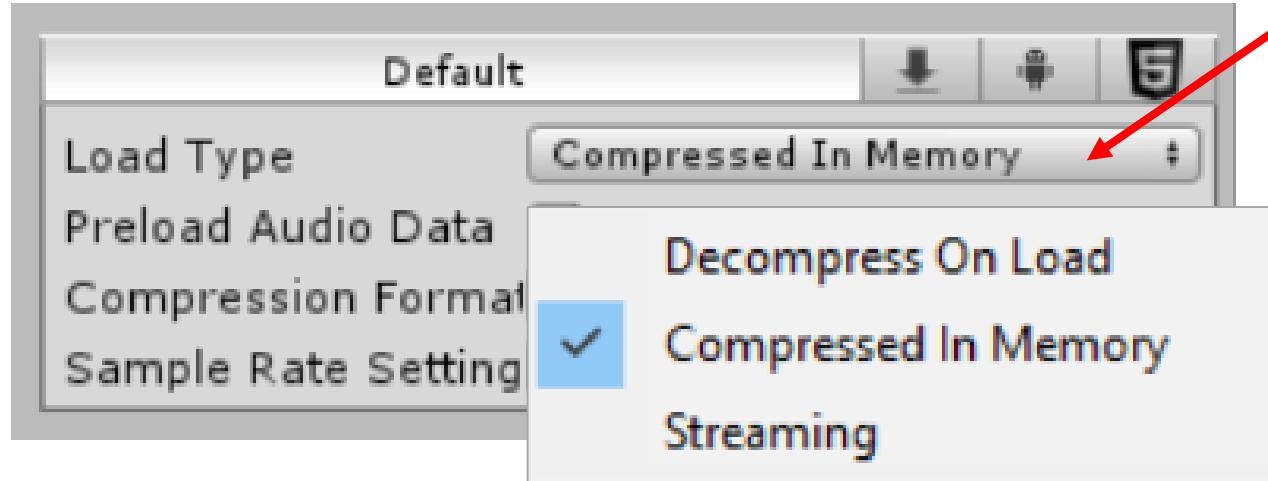
Unity также может записывать аудио из любого доступного микрофона на компьютере пользователя, для использования во время игры или для хранения и передачи.



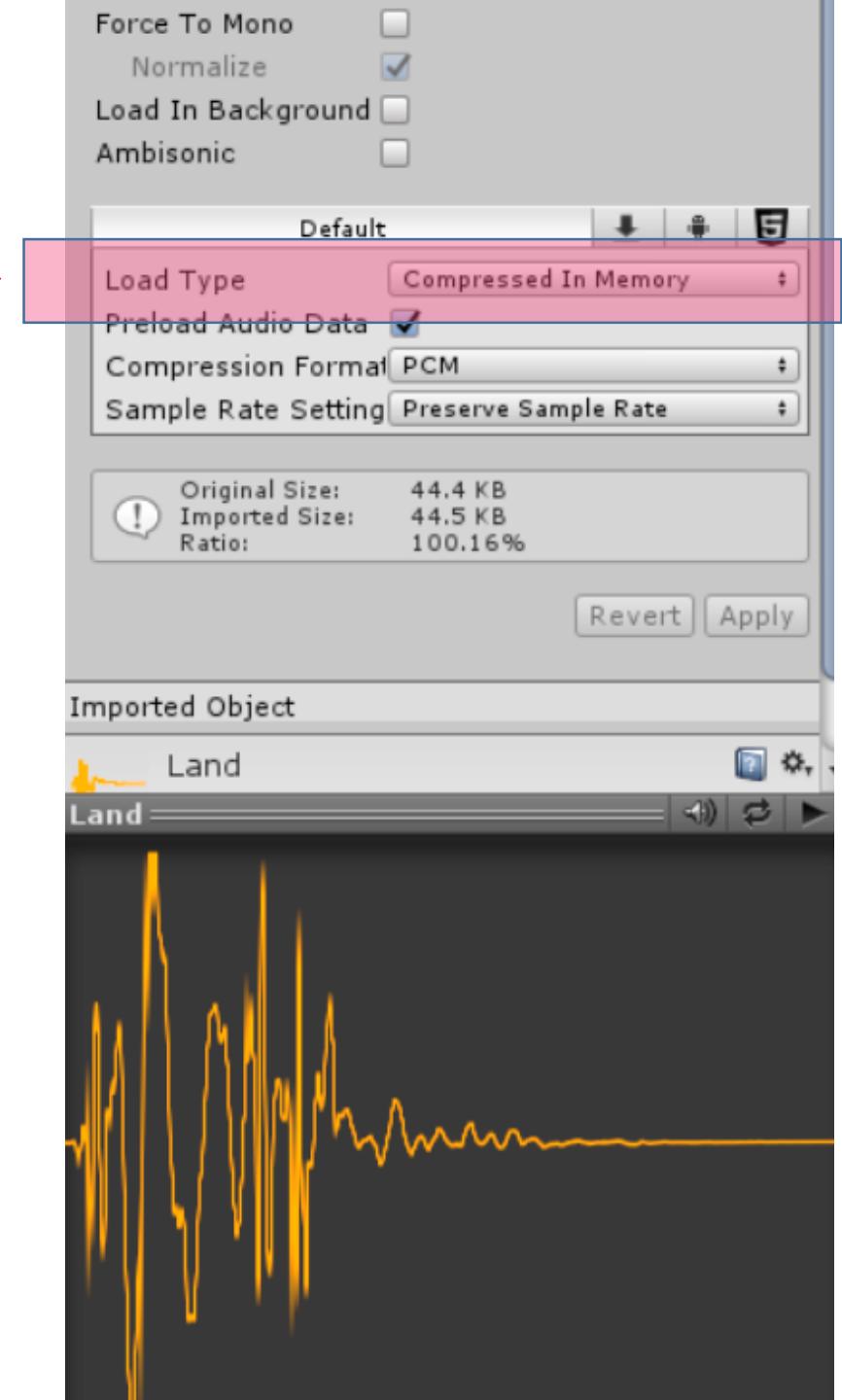
# Добавление звука



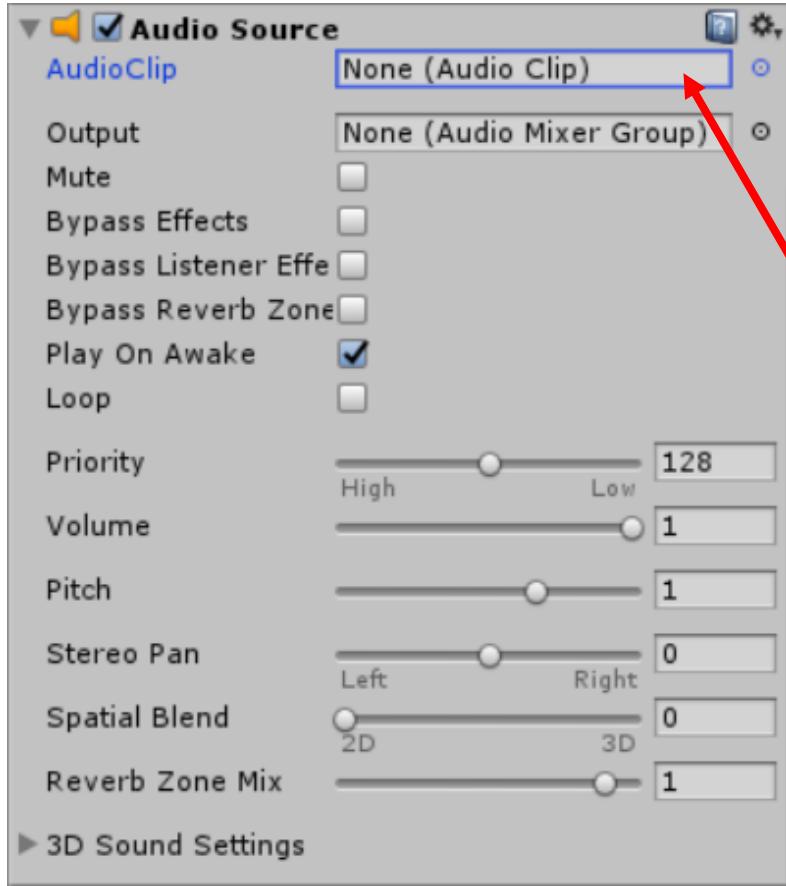
# Оптимизация звука



- Находится в оперативной памяти
- Тоже но в сжатом виде
- Проигрывается с жесткого диска



# Audio Source



# Audio Listener

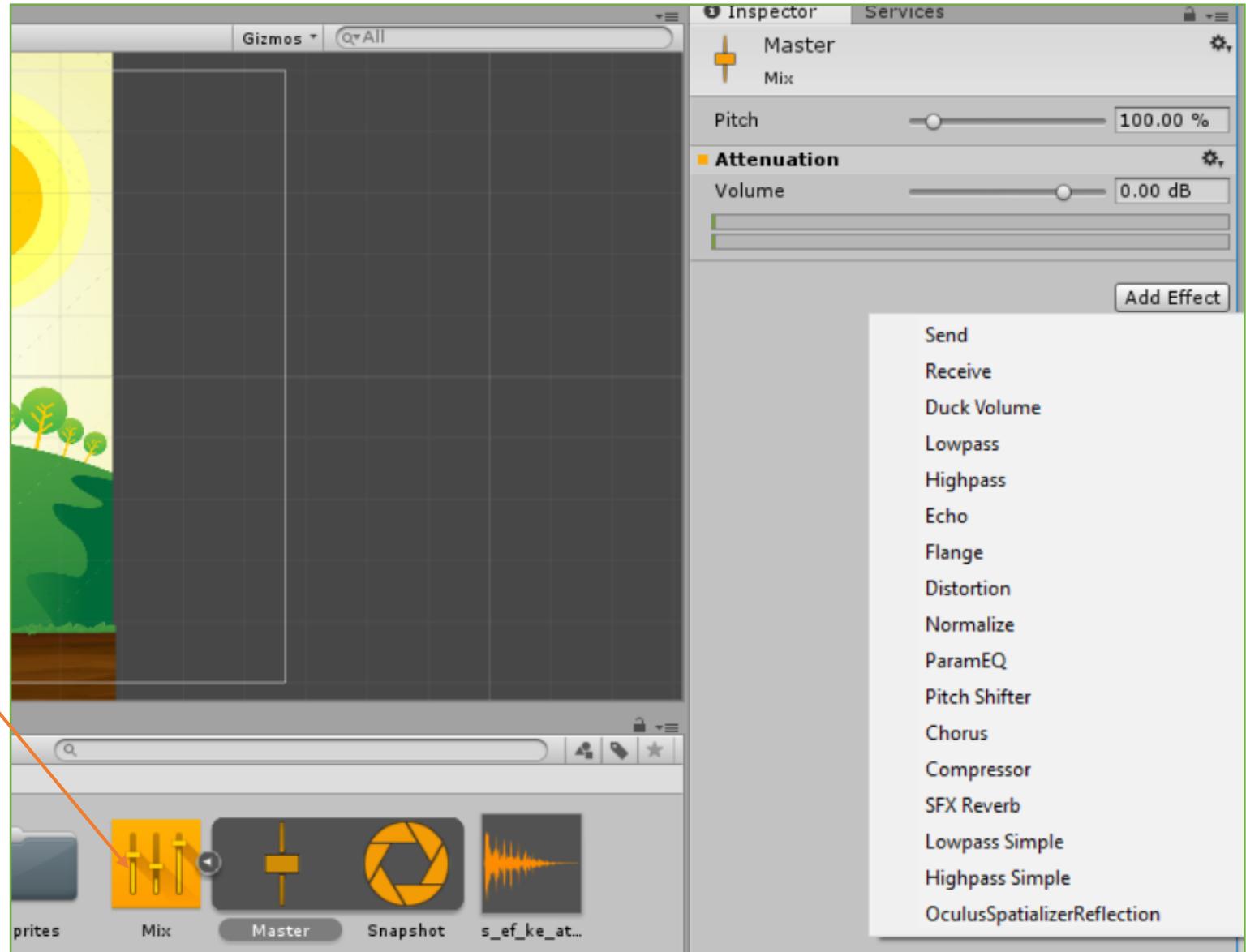


Audio Listener работает в связке с источниками звука (компонент Audio Source), позволяя создавать акустическое окружение в играх.

В каждой сцене может быть только один Audio Listener для корректной работы системы.

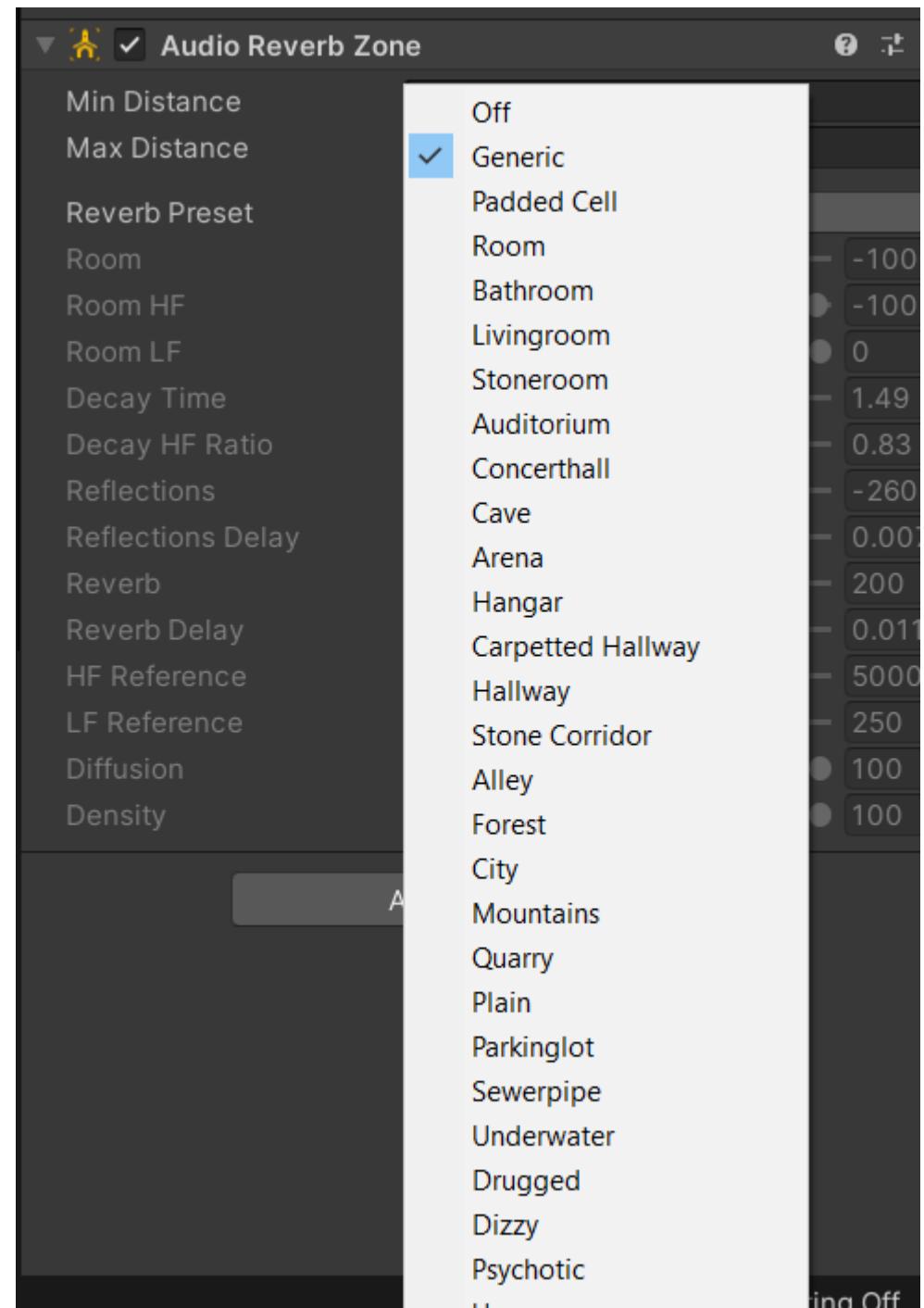
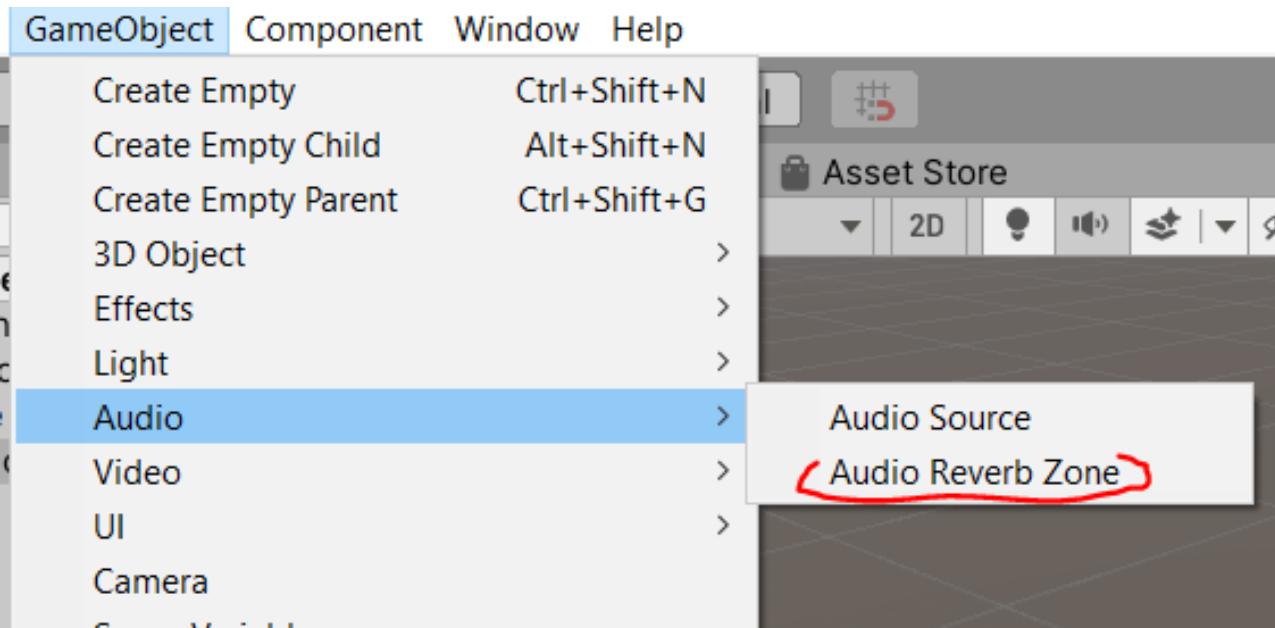
# Аудио эффекты

Можно создать свой  
**Микшер**



# Аудио эффекты

## Audio Reverb Zone



Атрибут **RequireComponent** автоматически добавляет необходимые компоненты в качестве зависимостей.

Когда вы добавляете скрипт, который использует RequireComponent в GameObject, необходимый компонент будет автоматически добавлен в GameObject.

```
using UnityEngine;

// PlayerScript requires the GameObject to have a Rigidbody component
[RequireComponent(typeof(Rigidbody))]
public class PlayerScript : MonoBehaviour
{
    Rigidbody rb;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

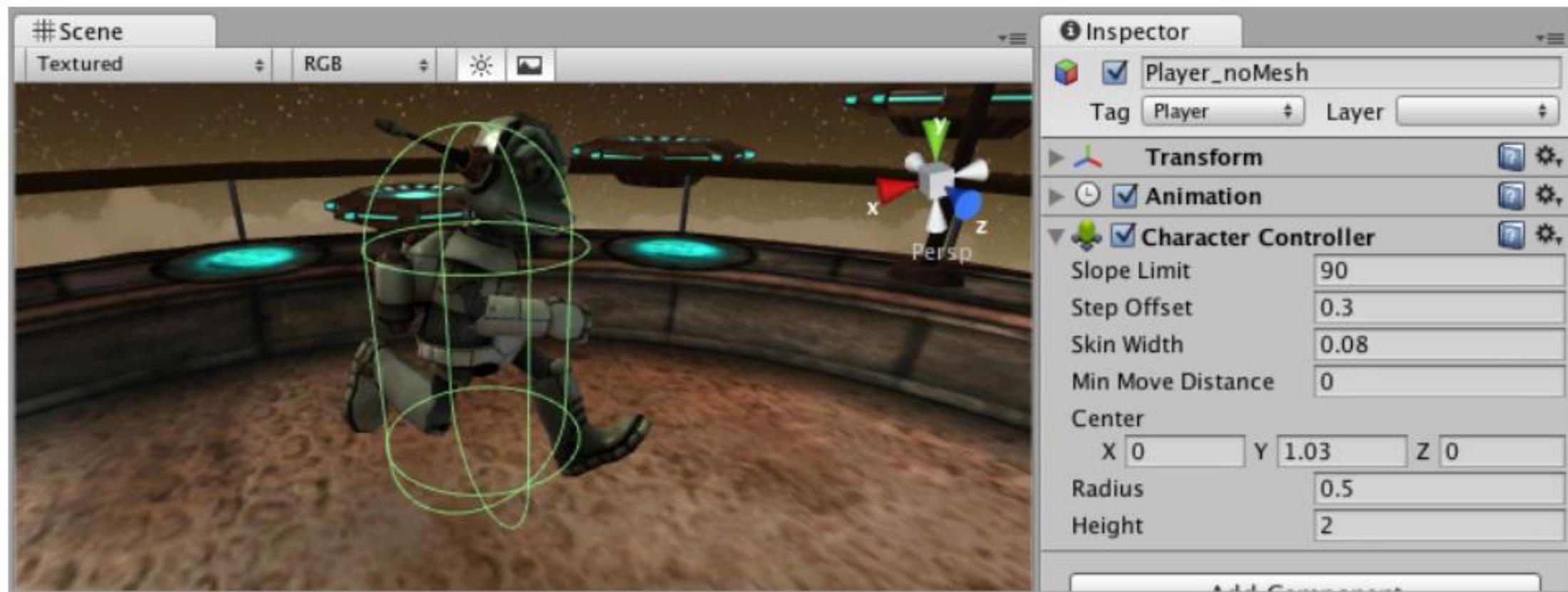
    void FixedUpdate()
    {
        rb.AddForce(Vector3.up);
    }
}
```

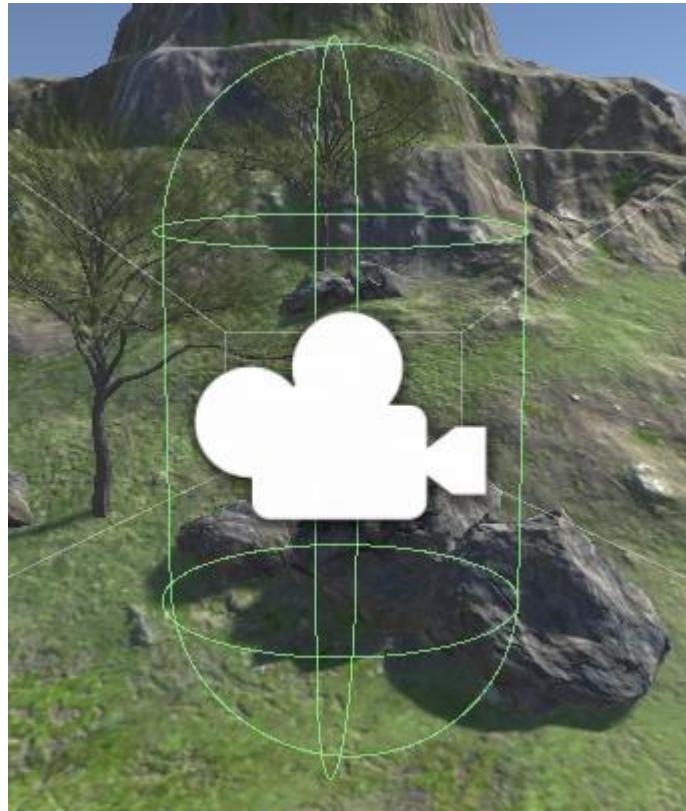
```
1 using UnityEngine;
2 using System.Collections;
3
4 [RequireComponent(typeof( AudioSource))]
5 public class JessicaScript : MonoBehaviour {
6     public float speed = 5f;
7
8
9     Animator animator;
10    CharacterController controller;
11    bool isDancing = false;
```

# Character Controller

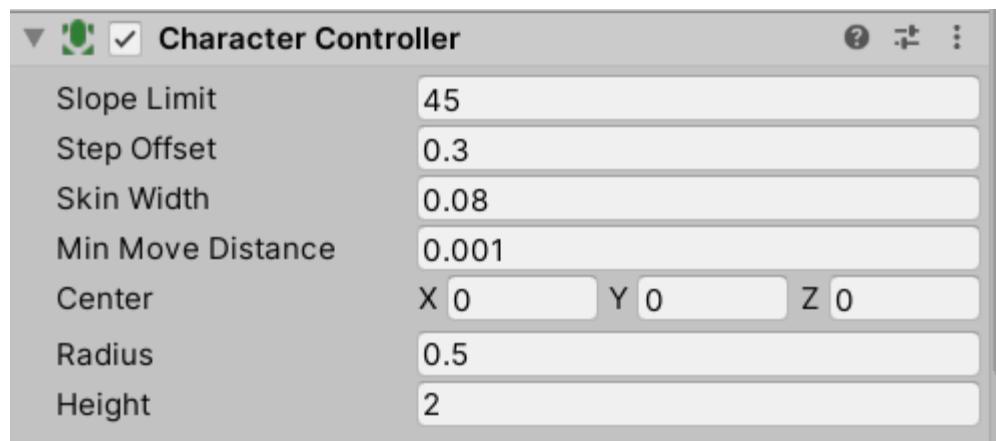
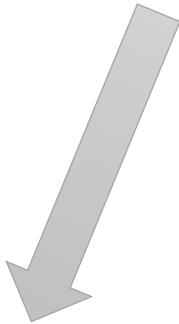
Компонент Character Controller в основном используется для управления от третьего или первого лица, где не требуется физика Rigidbody.

Controller не реагирует на силы сам по себе и не отталкивает Rigidbody-объекты автоматически. Если нужно толкать Rigidbody-объекты, использующие Character Controller, можно применить силы к любому объекту, в которого он врезается, через скрипting при помощи функции [OnControllerColliderHit\(\)](#).





Например, при добавлении к камере компонента **Character Controller** у нее появляется коллайдер в виде капсулы, со следующими свойствами:



Свойство:	Функция:
Slope Limit	Ограничивает возможность персонажа взбираться по склонам.
Step Offset	Персонаж переступит на поверхность, только если она ближе к земле, чем заданное значение.
Skin width	2 коллайдера могут пересечься друг с другом на глубину, равную значению Skin Width. Большее значение уменьшит степень тряски. Слишком низкое значение может вызвать застревание персонажа. Наилучший вариант = 10% от радиуса.
Min Move Distance	Если персонаж попробует сдвинуться ниже указанной величины, то он не сдвинется. Это может быть использовано, чтобы уменьшить тряsku. В большинстве ситуаций это значение стоит оставить равным нулю(0).

# CharacterController.Move

```
public CollisionFlags Move(Vector3 motion);
```

Скрипт для движения отсюда:

<https://docs.unity3d.com/ScriptReference/CharacterController.Move.html>

```
public class Example : MonoBehaviour
{
    private CharacterController controller;
    private Vector3 playerVelocity;
    private bool groundedPlayer;
    private float playerSpeed = 2.0f;
    private float jumpHeight = 1.0f;
    private float gravityValue = -9.81f;

    private void Start()
    {
        controller = gameObject.AddComponent<CharacterController>();
    }
    void Update()
    {
        groundedPlayer = controller.isGrounded;
        if (groundedPlayer && playerVelocity.y < 0)
        {
            playerVelocity.y = 0f;
        }
        Vector3 move = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
        controller.Move(move * Time.deltaTime * playerSpeed);

        if (move != Vector3.zero)
        {
            gameObject.transform.forward = move;
        }
        if (Input.GetButtonDown("Jump") && groundedPlayer)
        {
            playerVelocity.y += Mathf.Sqrt(jumpHeight * -3.0f * gravityValue);
        }
        playerVelocity.y += gravityValue * Time.deltaTime;
        controller.Move(playerVelocity * Time.deltaTime);
    }
}
```

В AssetStore можно найти готовые контроллеры персонажа от первого и третьего лица

# Modular First Person Controller

The image shows a large, central promotional image for the asset. It features a blue, capsule-shaped 3D model of a character or object resting on a solid orange cube. The cube has the number '2/5' printed on its front face. To the left of the main image is a small left arrow icon, and to the right is a small right arrow icon. At the top right of the main image are two small navigation icons: a white square with a black arrow pointing right and a white square with a black double-headed horizontal arrow.

**Modular First Person Controller**

JeCase ★★★★☆ (94) | ❤ (1905)

**FREE**

🕒 1905 views in the past week

Add to My Assets

**hugovanriet** ★★★★★ 4 days ago

**Great addition to simple 3D games**

Great beginner friendly FPS controller with lots of things to configure to make it perfect for your game!

[Read more reviews](#)

License agreement: Standard Unity Asset Store EULA

License type: Extension Asset

File size: 366.2 KB

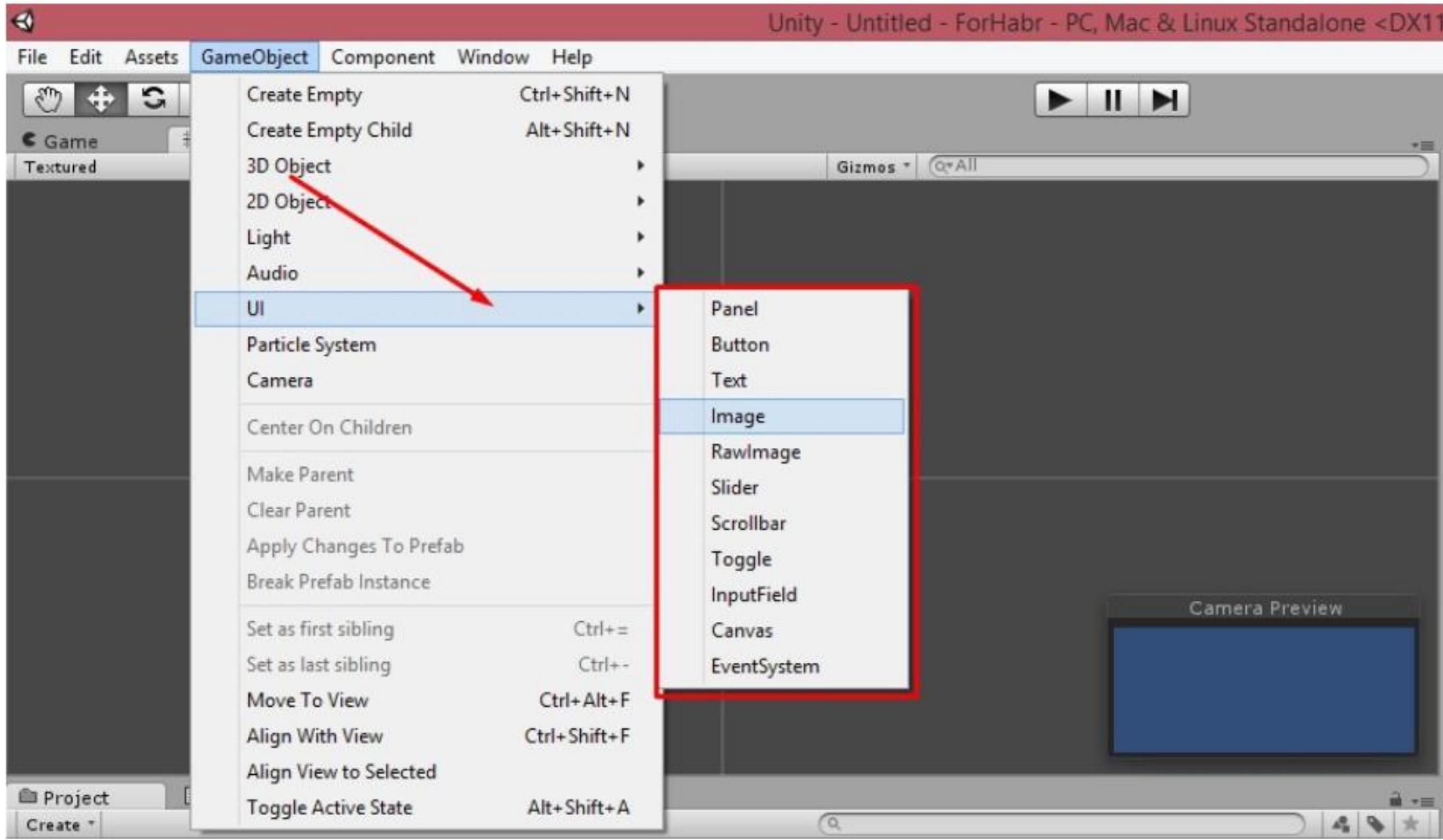
Latest version: 1.0.1

Latest release date: Apr 12, 2021

Below the main image, there is a row of five small thumbnail images, each showing a different aspect of the controller's functionality or configuration. From left to right, they are: a character in a desert environment, a character in a first-person view holding a blue capsule, a screenshot of a settings menu titled 'Modular First Person Controller', a screenshot of a settings menu titled 'Submodule Configuration' showing a list of items, and a screenshot of a settings menu titled 'Configure AI Values' showing a list of items.

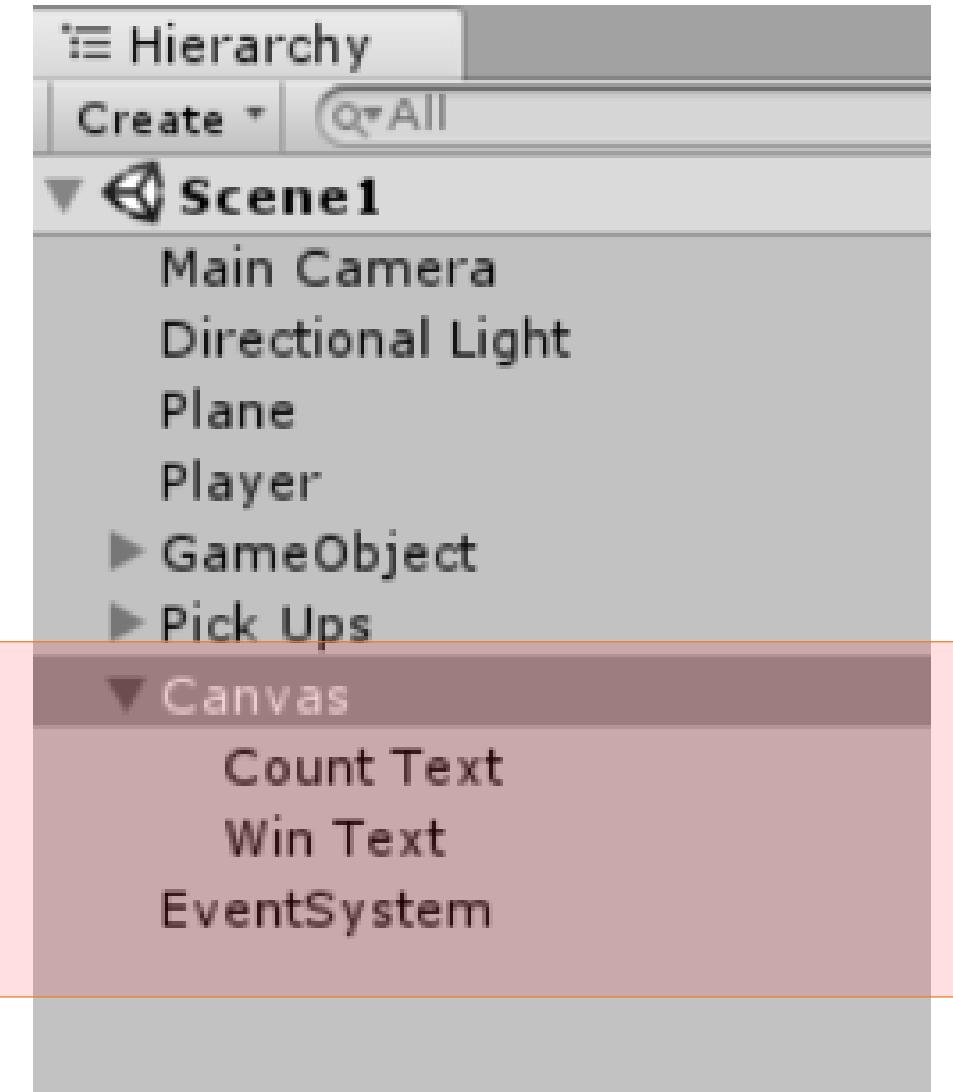
# Пользовательский интерфейс



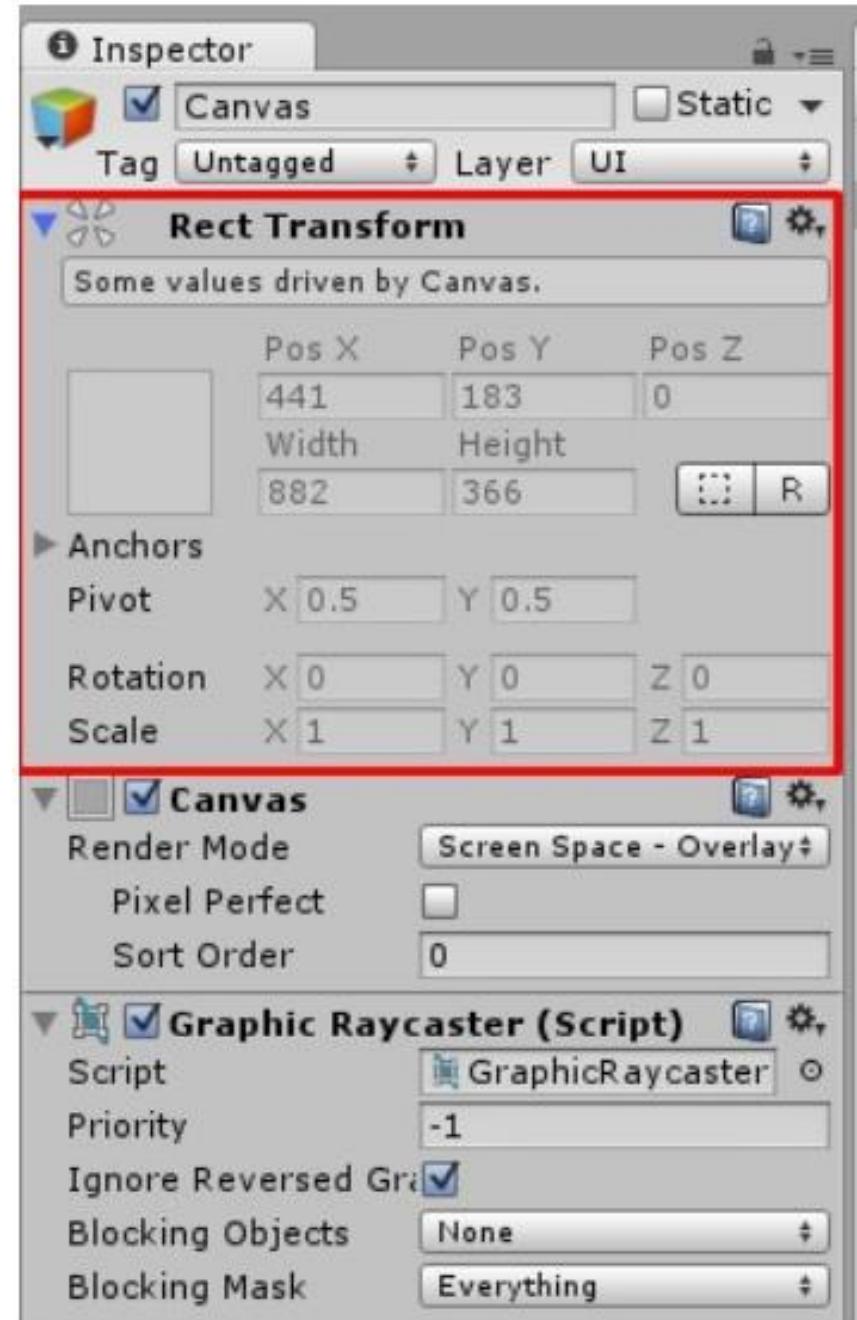


**Важно: каждый элемент УИ должен быть дочерним объектом Canvas.**

Canvas — это некое подобие рамки для хранения набора UI. Он может вмещать любое количество элементов. Также можно создавать любое количество этих самых Canvas'ов.



**Canvas** не имеет обязательного для всех объектов в Unity компонента **Transform**. Вместо этого у него, как и любого другого UI-элемента, имеется **Rect Transform**, который, по сути, является измененным под **UI Transform**'ом.



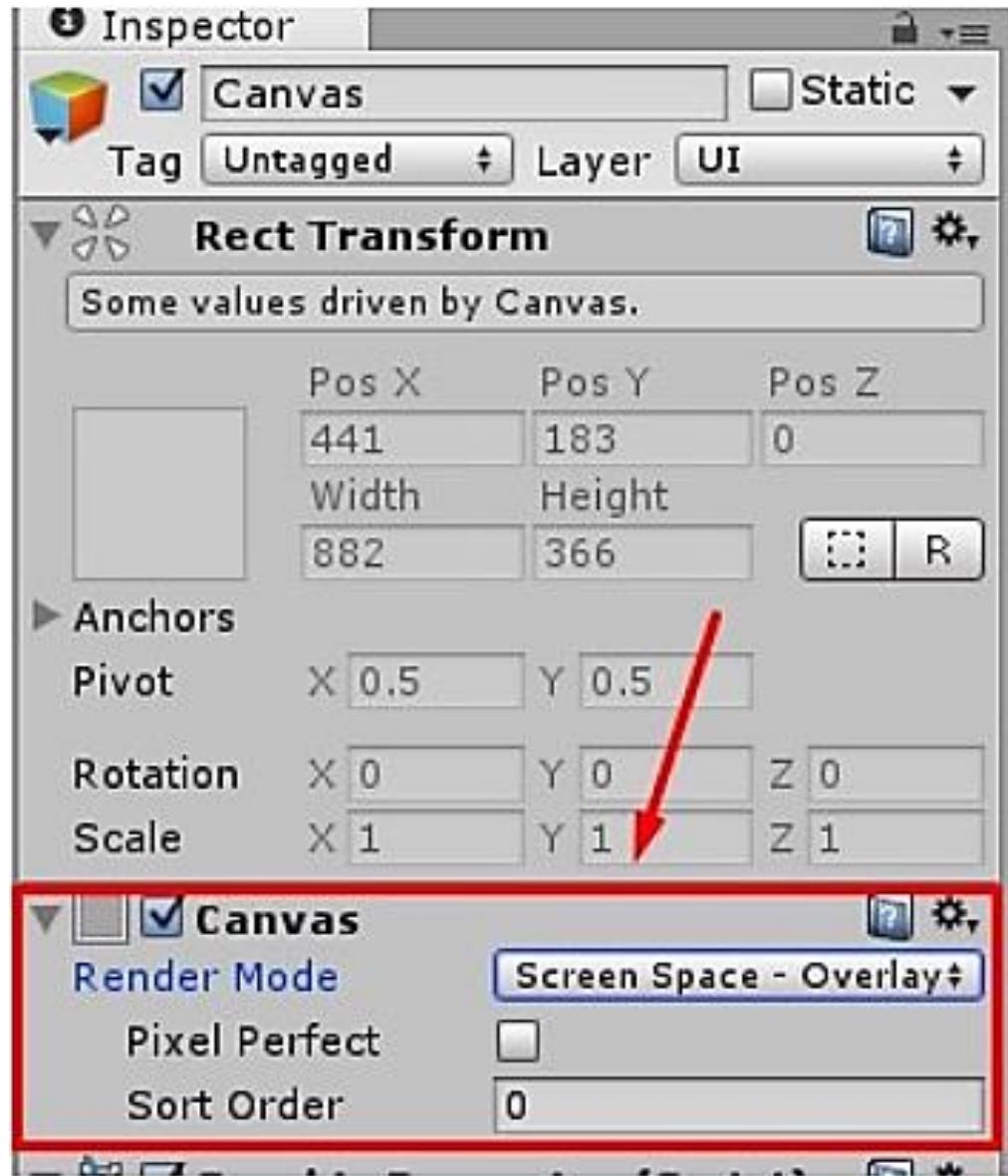
Persp

Count Text

Win Text

сцена





## Режимы отображения (Render Mode)

У полотна есть параметр Render Mode, который определяет, где оно будет отображаться: в пространстве экрана (screen space) или игрового мира (world space).



**Screen Space - Overlay**

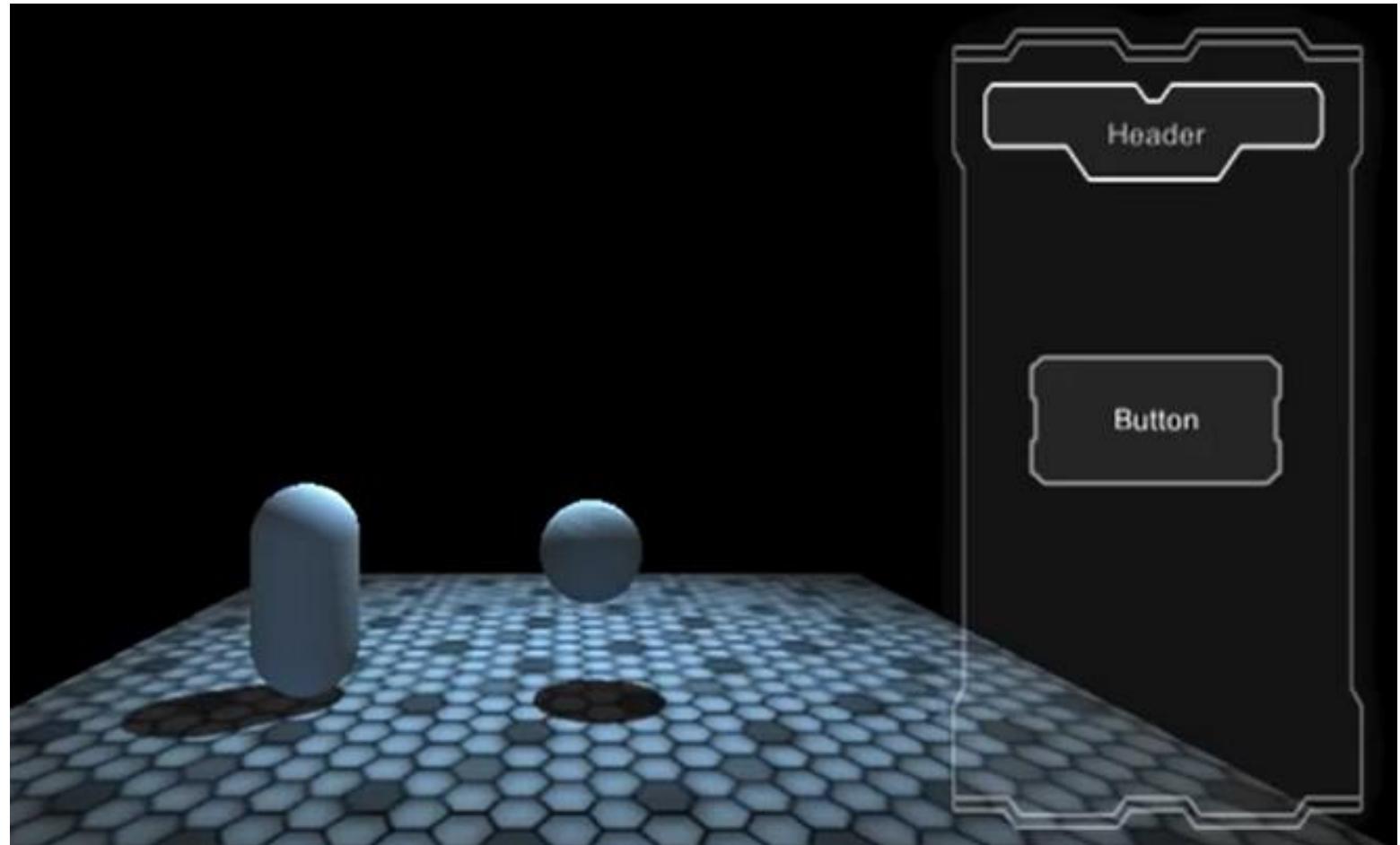
**Screen Space - Camera**

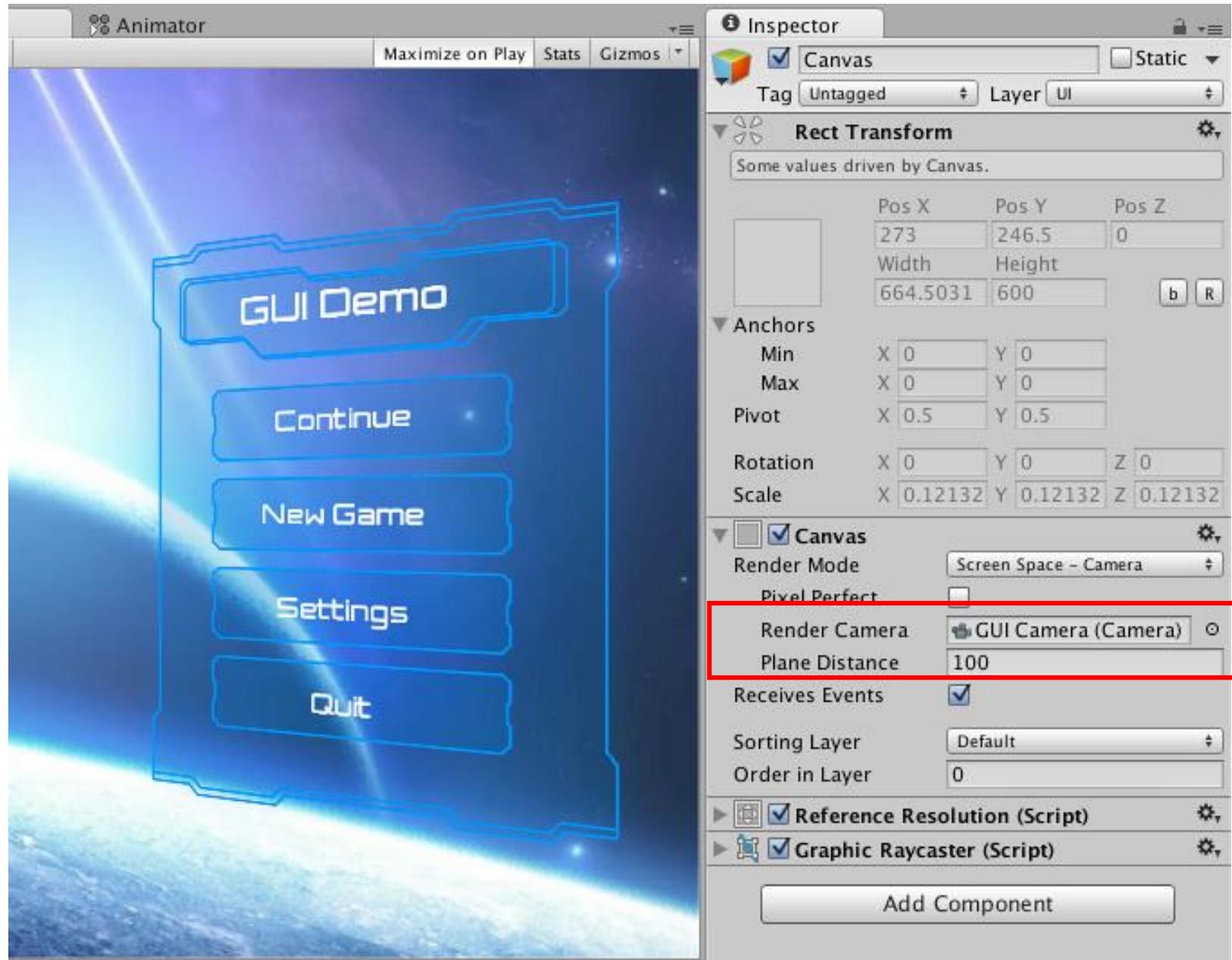
**World Space**

## Screen OverLay

По умолчанию в поле Render Mode установлено значение **Screen OverLay**, которое обозначает, что UI-элементы, прикрепленные к этому Canvas'у, будут отображаться поверх всего остального на сцене.

При таком режиме рендеринга Canvas равен пропорции экрана. При изменении параметров экрана, Canvas изменит свои размеры соответственно.





## Screen Space – Camera

Canvas помещается на заданное расстояние перед указанной камерой.

Элементы пользовательского интерфейса отображаются этой камерой, что означает, что настройки камеры влияют на внешний вид пользовательского интерфейса.

Если для камеры установлено значение Перспектива, элементы пользовательского интерфейса будут отображаться с перспективой.

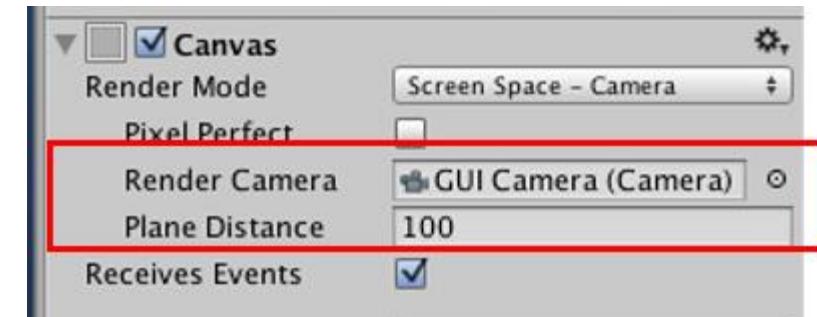
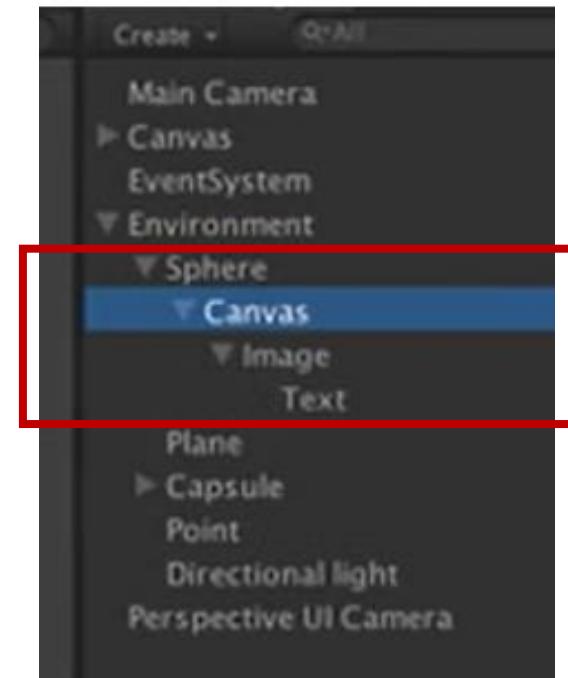
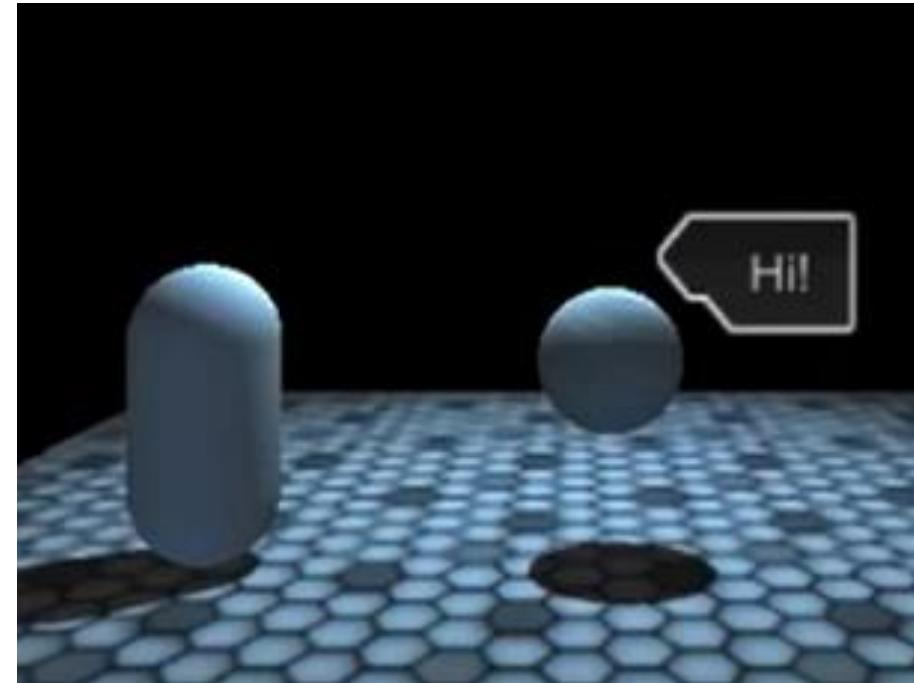
При этом режиме отображения Canvas ведет себя также, как и любой другой объект на сцене.

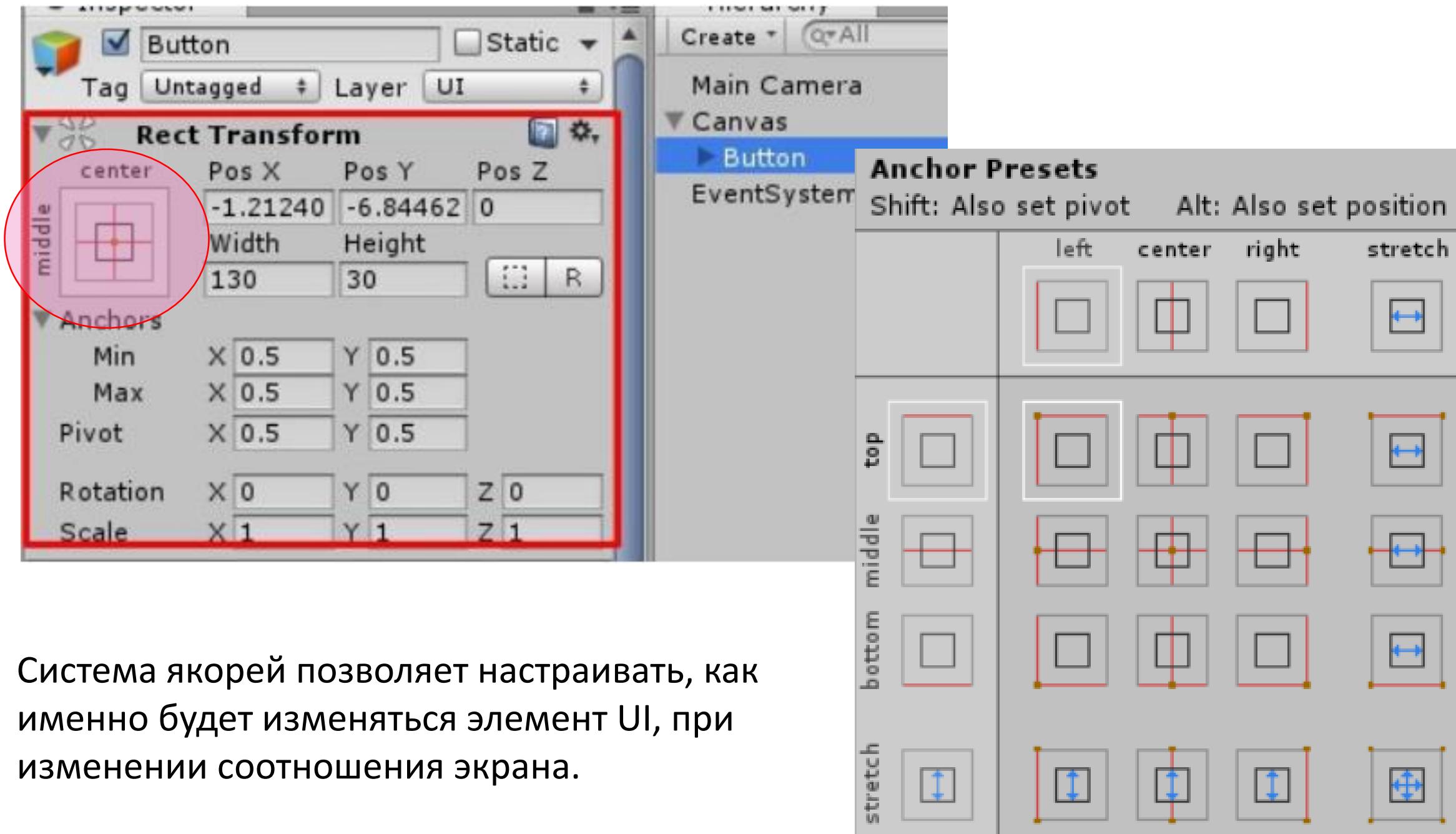
Размер Canvas может быть задан вручную при помощи Rect Transform, а элементы интерфейса будут отображаться перед или за другими объектами на сцене, в зависимости от их трехмерного расположения.

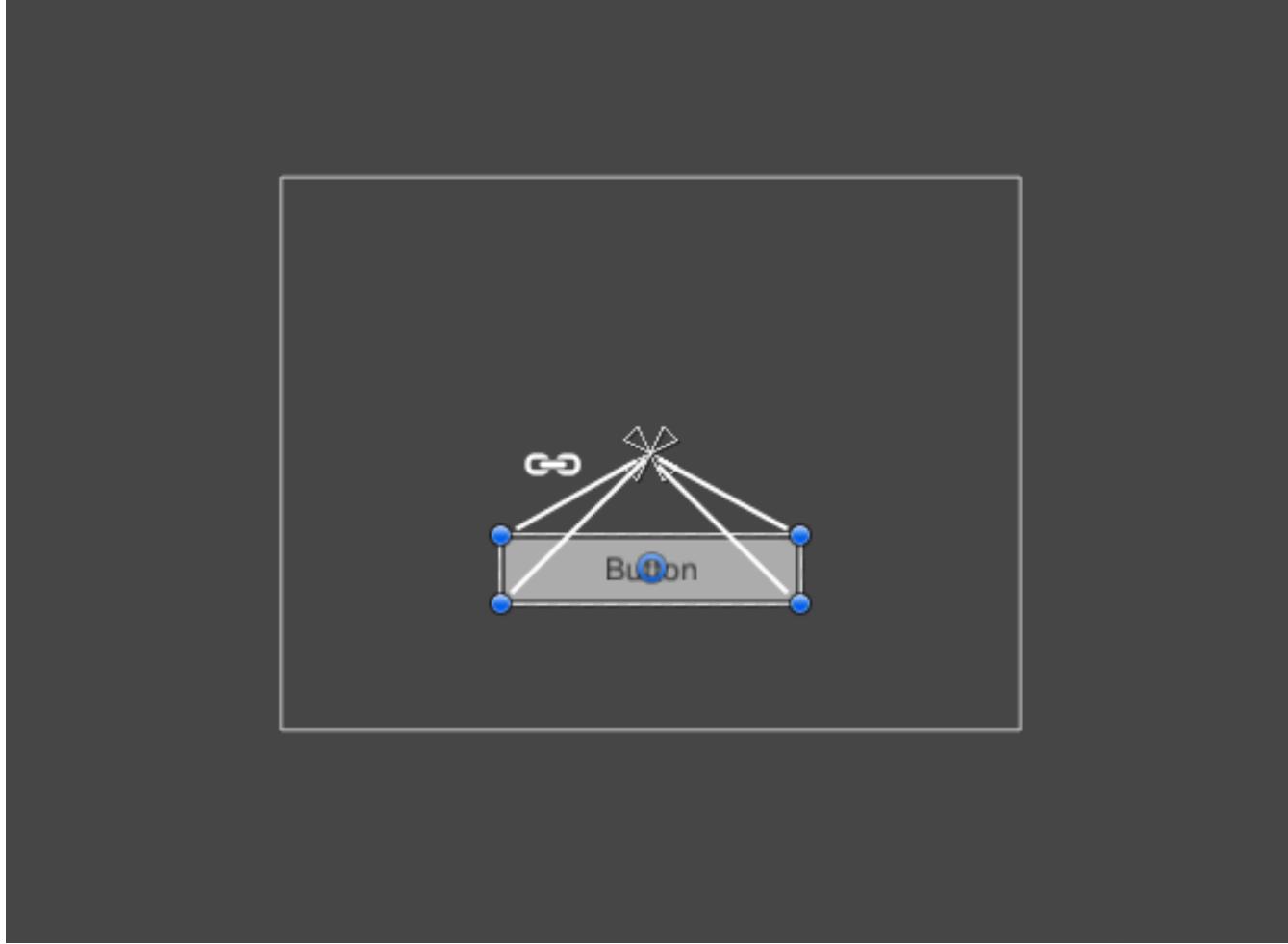
Этот режим удобен для тех интерфейсов, которые предполагаются как часть игрового мира (diegetic interfaces).

У холстов, у которых в параметре Render Mode выбраны Screen Space — Camera или World Space, всегда устанавливайте камеру. Если её не установить, то система пользовательского интерфейса в каждом кадре будет производить её поиск через Object.FindObjectWithTag, чтобы найти Camera.main, а это скажется на производительности.

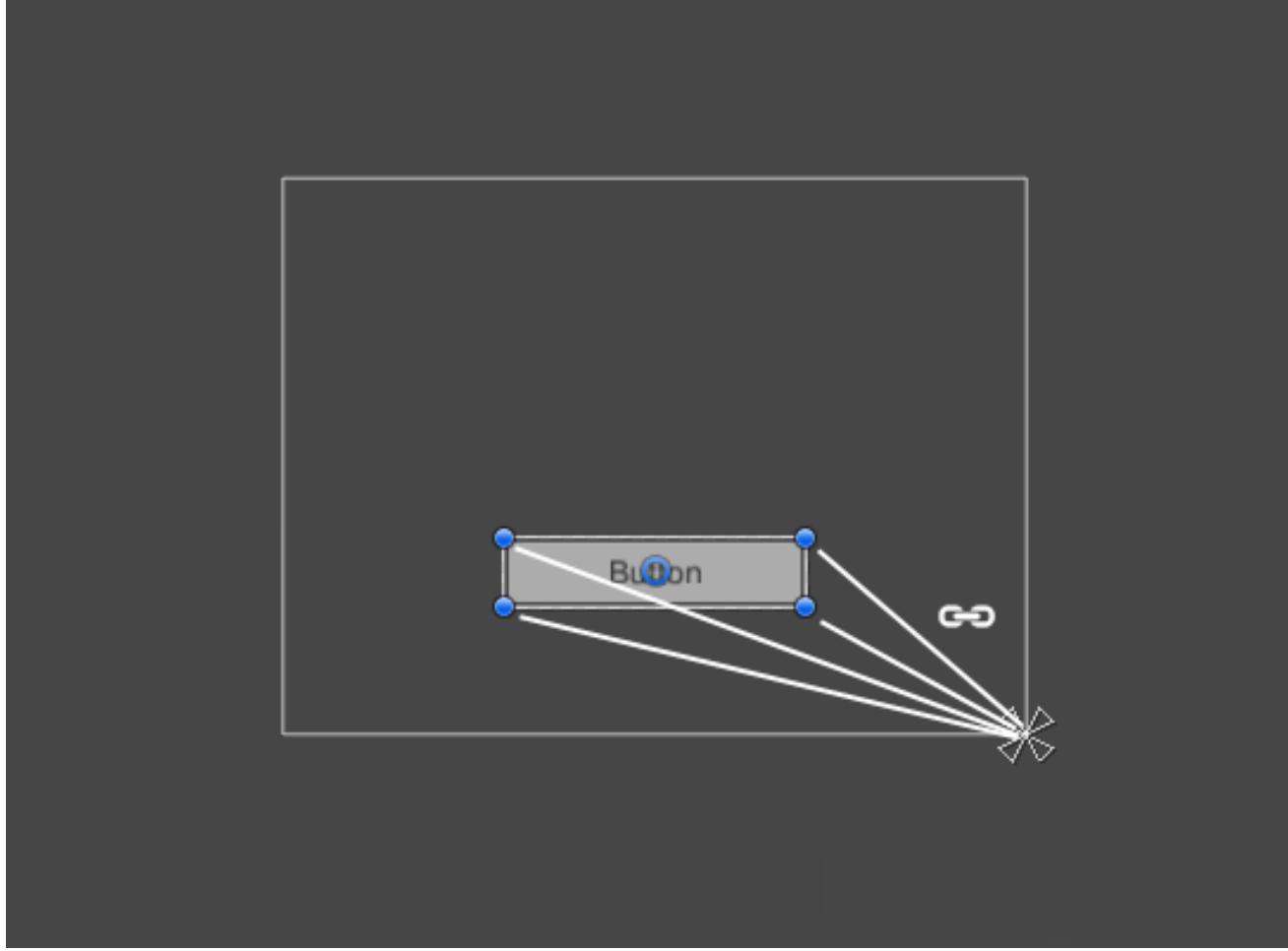
# World Space



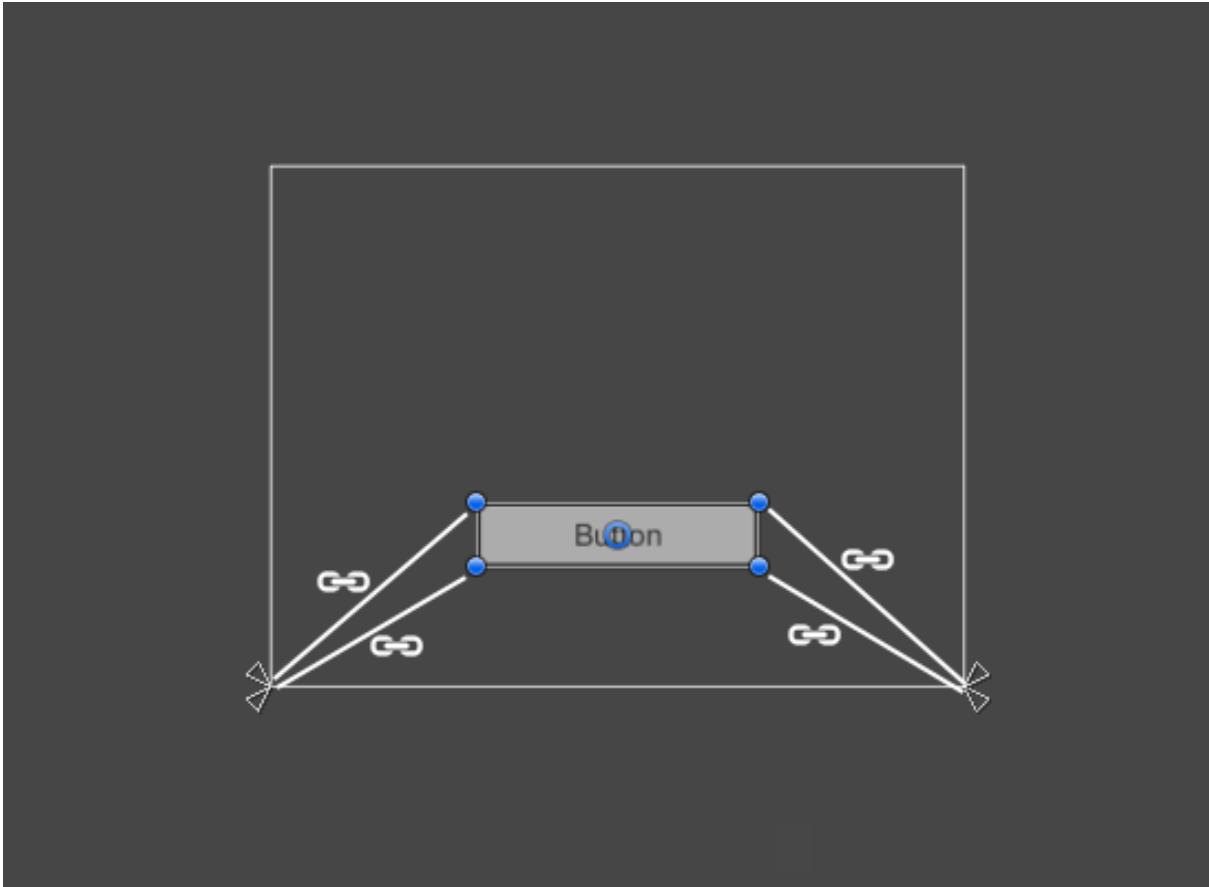


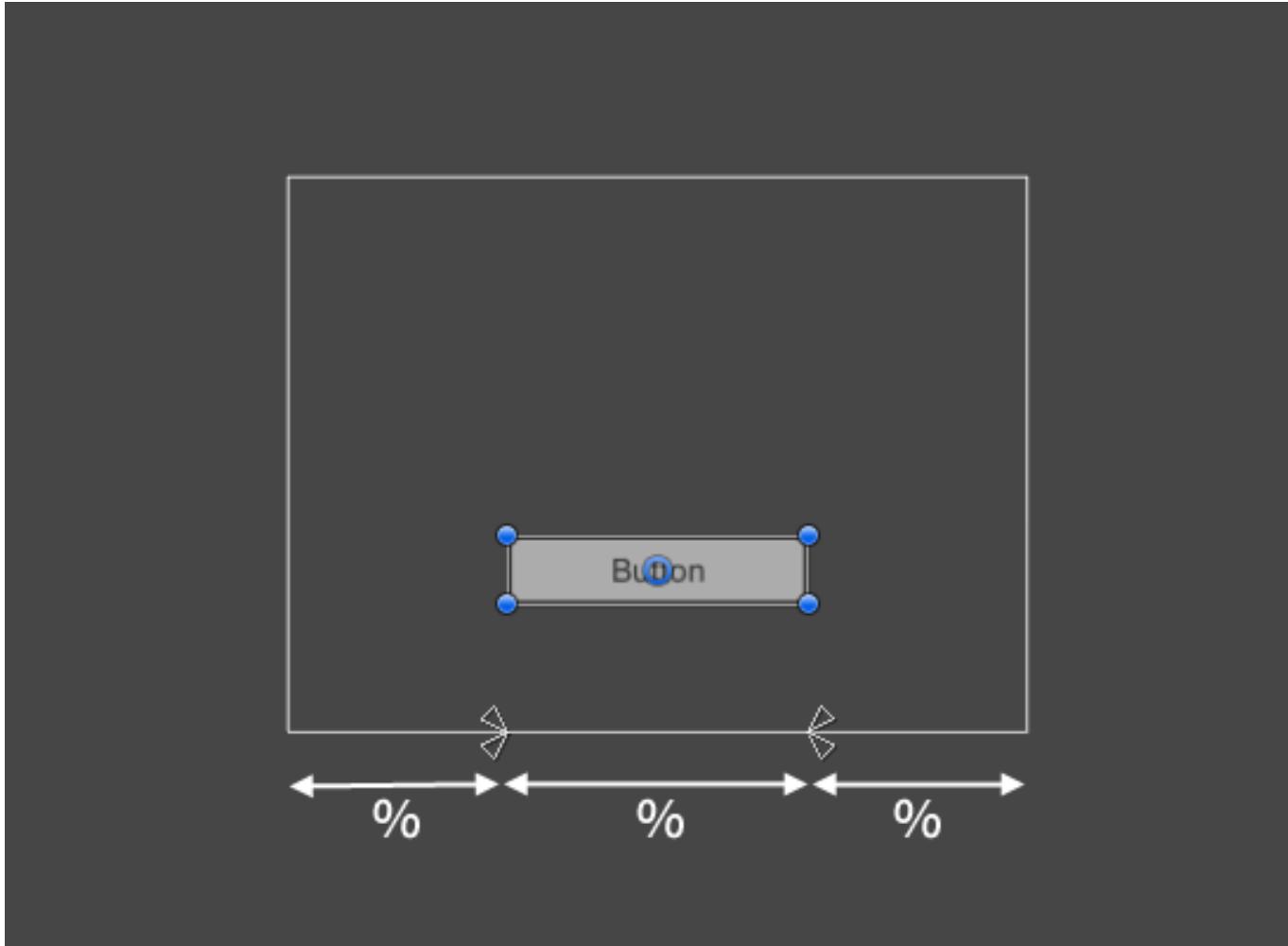


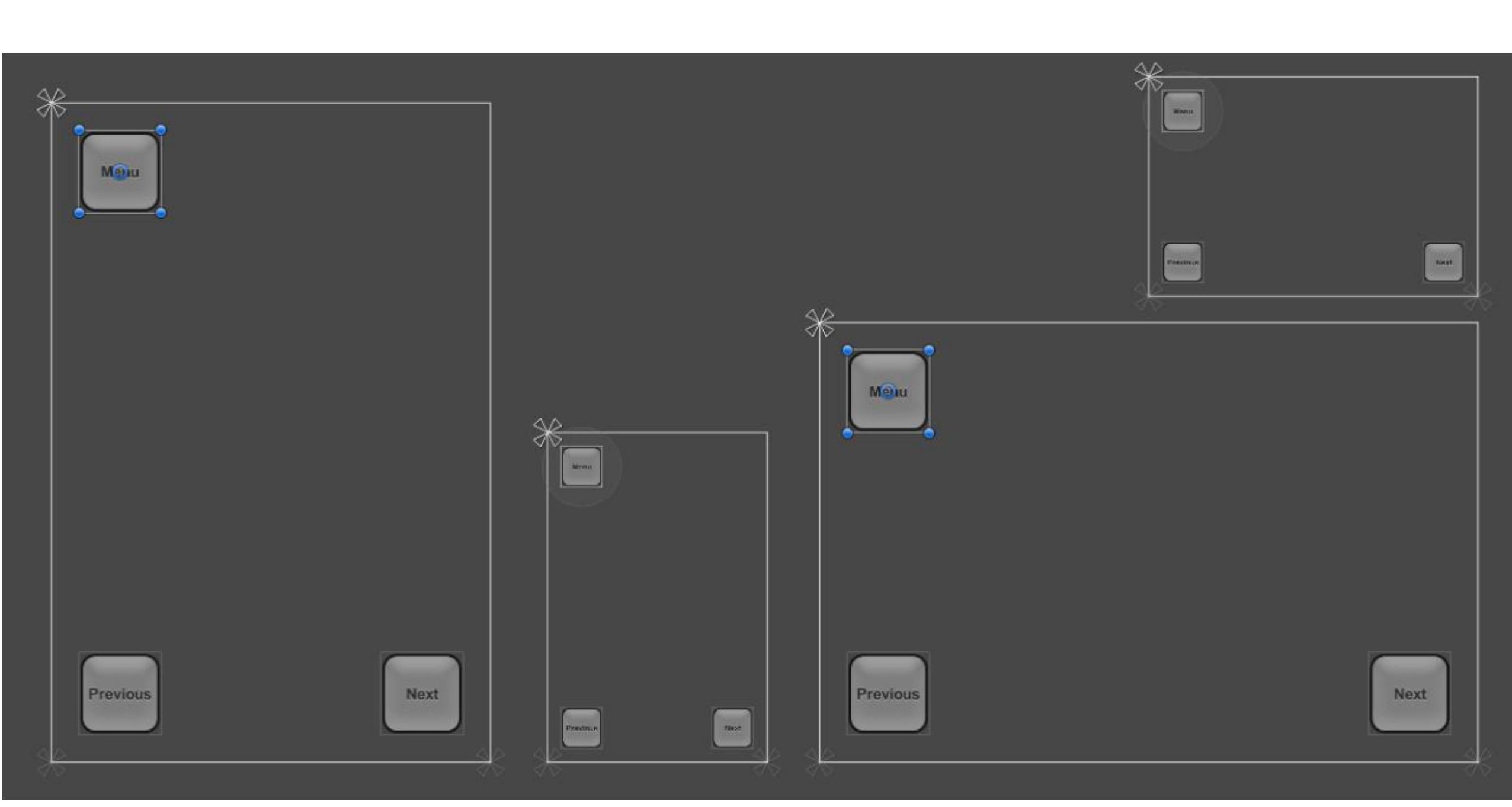
Здесь якоря были закреплены по центру объекта-родителя, благодаря чему при изменении соотношения экрана, расстояние от центра объекта-родителя до элемента остается неизменным.



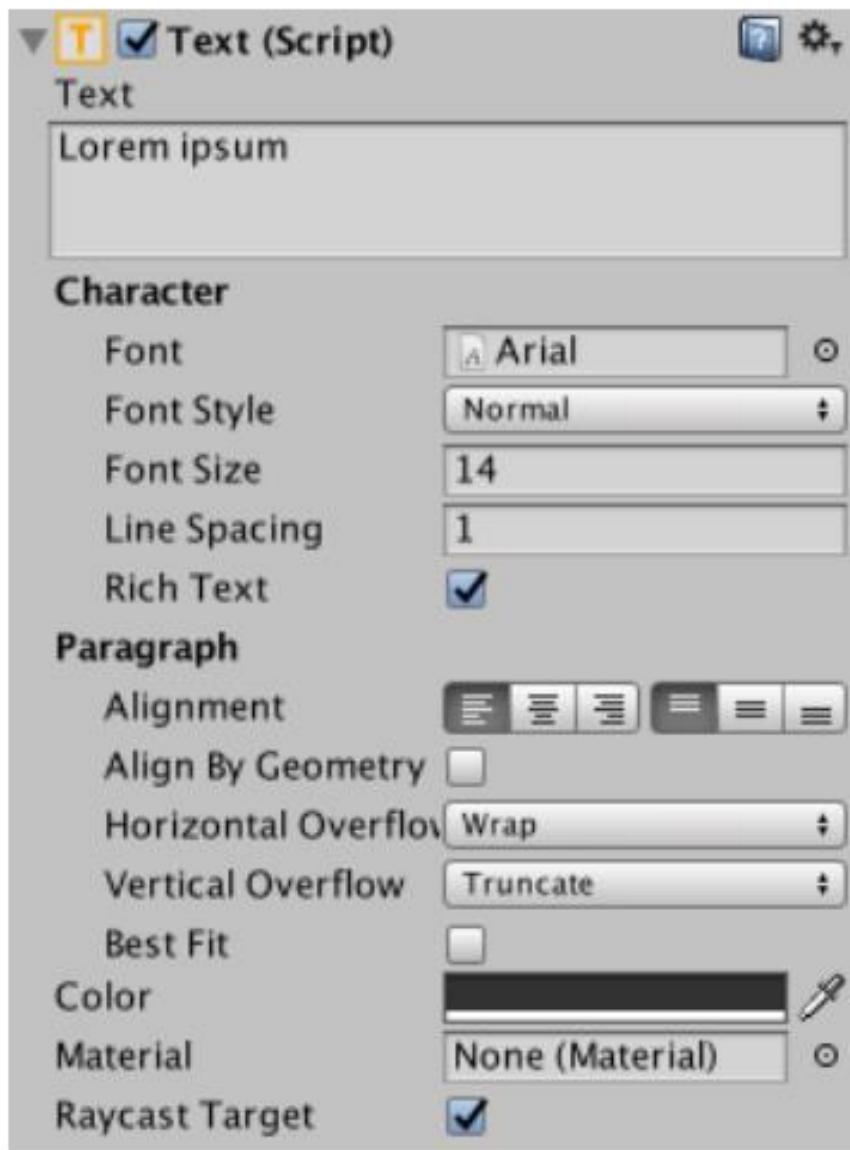
В данном случае якоря закреплены в правом нижнем углу объекта-родителя, поэтому при изменении соотношения сторон экрана, расстояние от элемента до правого нижнего угла остается неизменным.





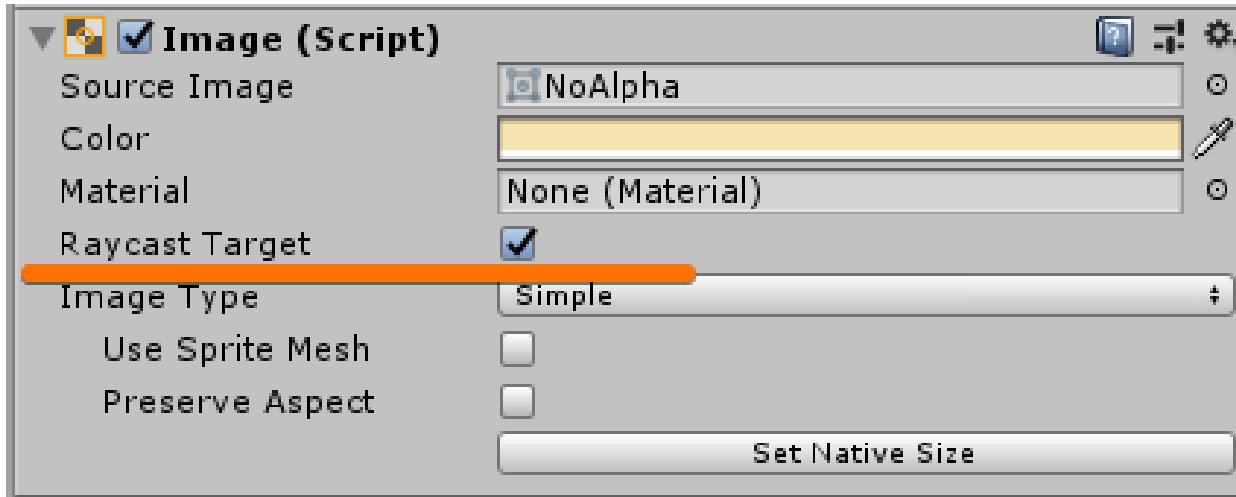


# Текст



Компонент Text, также известный как Label, имеет область для ввода текста, который будет отображен. Есть возможность задать шрифт, его стиль, размер и способность отображения Rich Text.

Присутствуют опции для установки параметров выравнивания; настройки горизонтального и вертикального переполнения, которые управляют поведением текста, когда он не влезает по ширине или высоте в отведенный ему прямоугольник; а также опция Best Fit, которая делает так, что текст занимает все доступное ему место.



Оставляйте флаг **Raycast** **Target** только на элементах, нуждающихся в событиях ввода, и снимайте у остальных.

По умолчанию Raycast target включён на многих элементах. Это затрудняет и замедляет работу Raycaster-компонента, который обрабатывает события ввода в UI Unity. При клике или тапе он проходит по всей иерархии элементов и ищет все графические компоненты с выставленным флагом Raycast Target, затем проверяет их на возможность событий ввода и после успешного прохождения проверок добавляет в список попаданий. После этого список попаданий сортируется по глубине, отбрасываются объекты, находящиеся за пределами экрана. В результате выдаёт окончательный список попаданий.

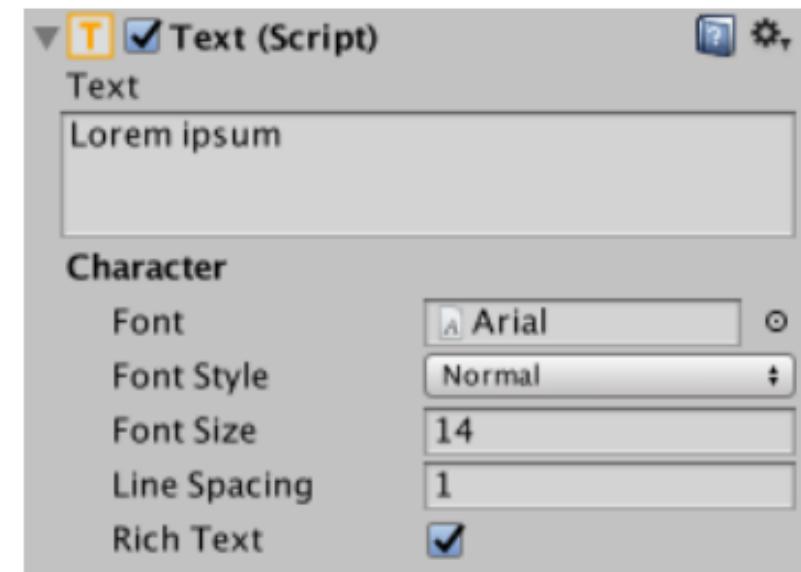
# Опция Rich Text

We are **<i>definitely not</i>** amused

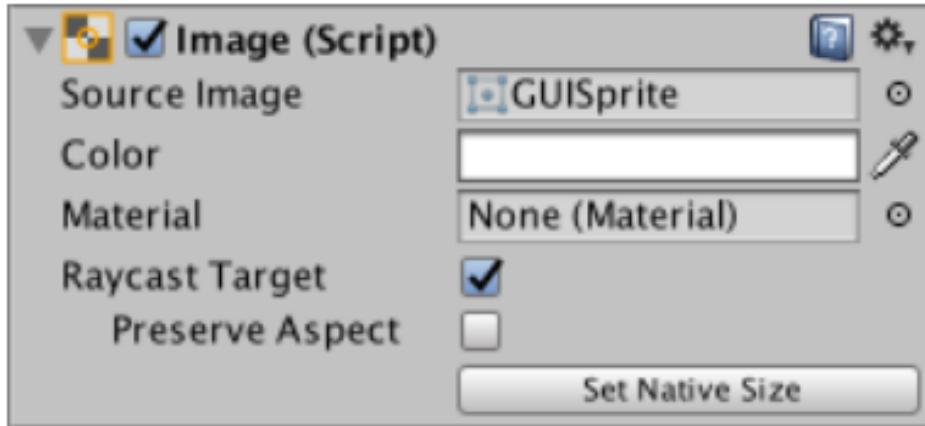
We are **absolutely definitely not** amused

```
GUIStyle style = new GUIStyle ();
style.richText = true;
GUILayout.Label("<size=30>Some <color=yellow>RICH</color> text</size>",style);
```

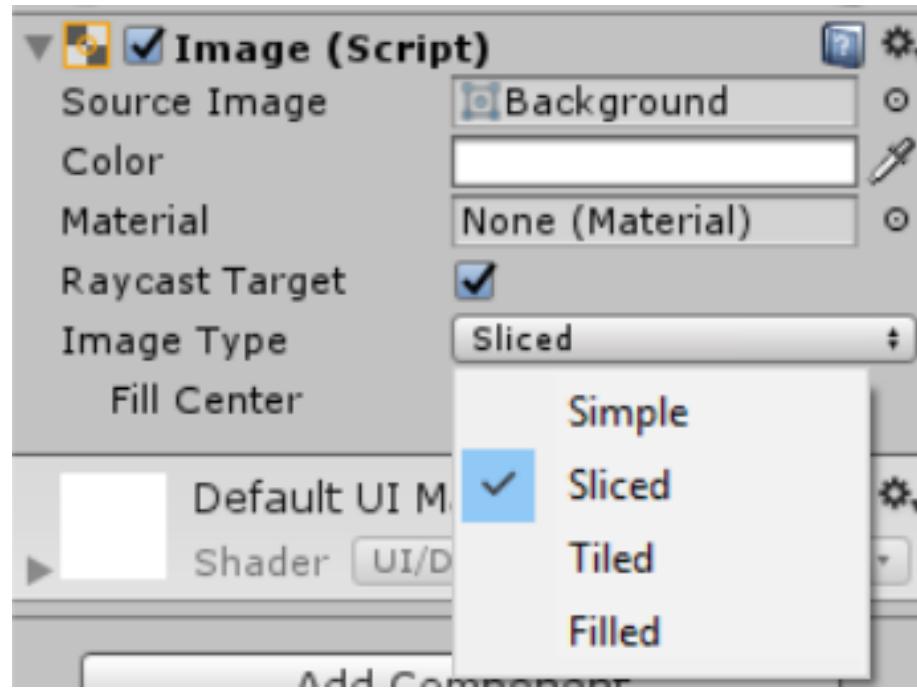
## Текст



# Изображение

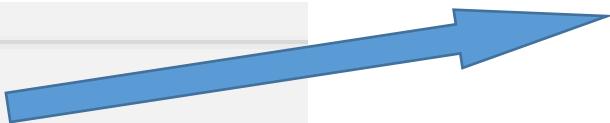
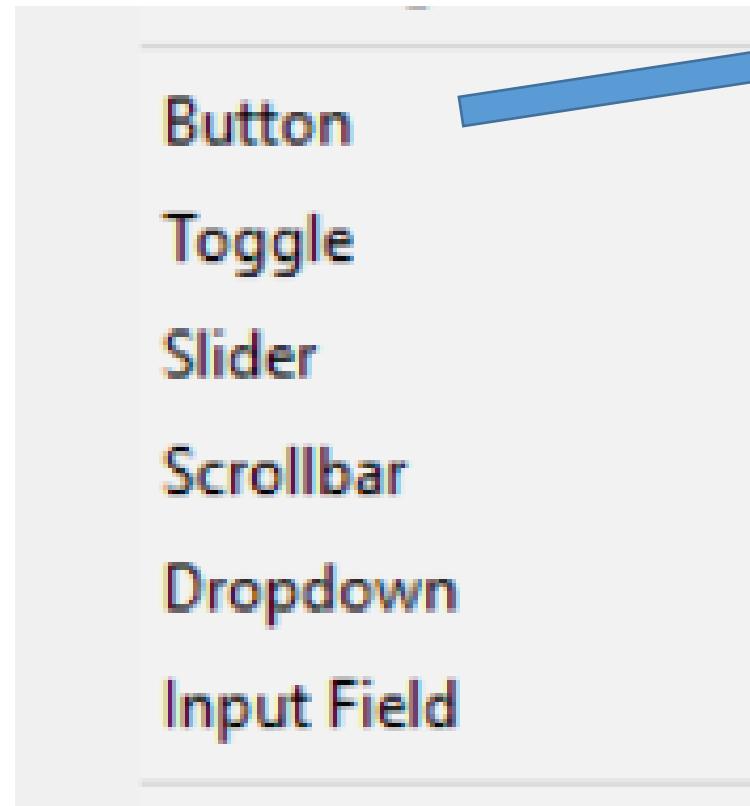


К **Image** может быть добавлен спрайт, а его цвет задан в поле Color.  
Также к компоненту **Image** может быть добавлен материал.



Режимы масштабирования спрайта

Элемент **Raw Image** принимает текстуру



▼ **OK**  **Button (Script)**

Interactable

Transition

Target Graphic  **Button (Image)**

Normal Color

Highlighted Color

Pressed Color

Disabled Color

Color Multiplier 2

Fade Duration 0.1

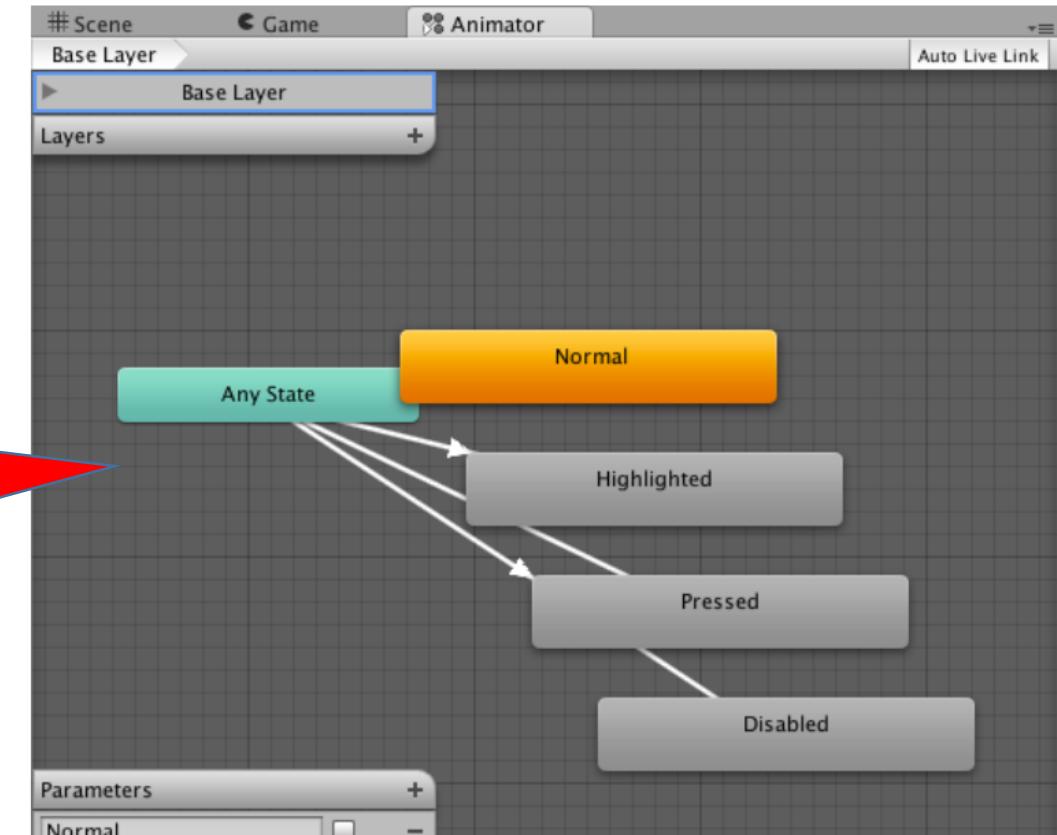
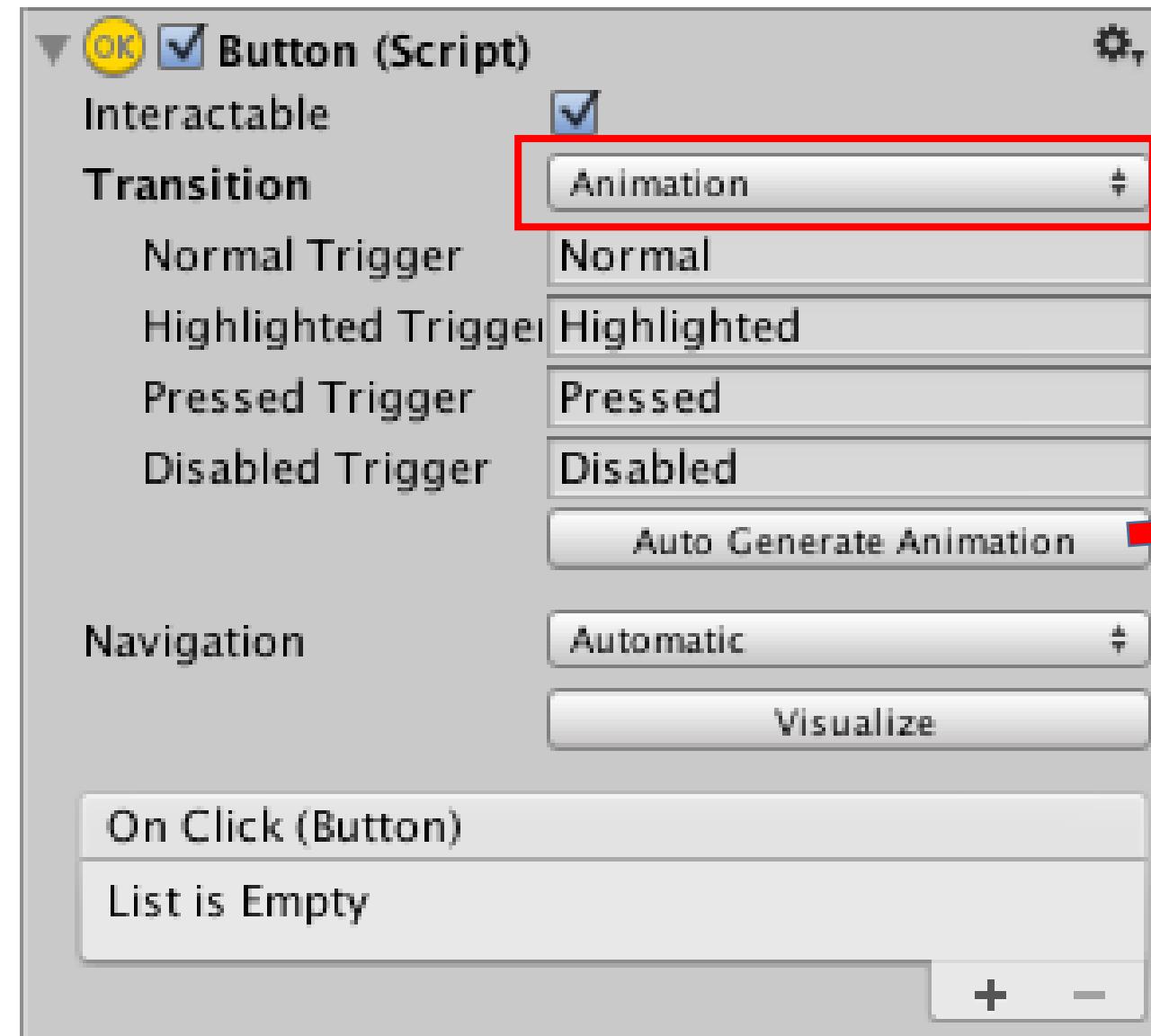
Navigation  Automatic   
Visualize

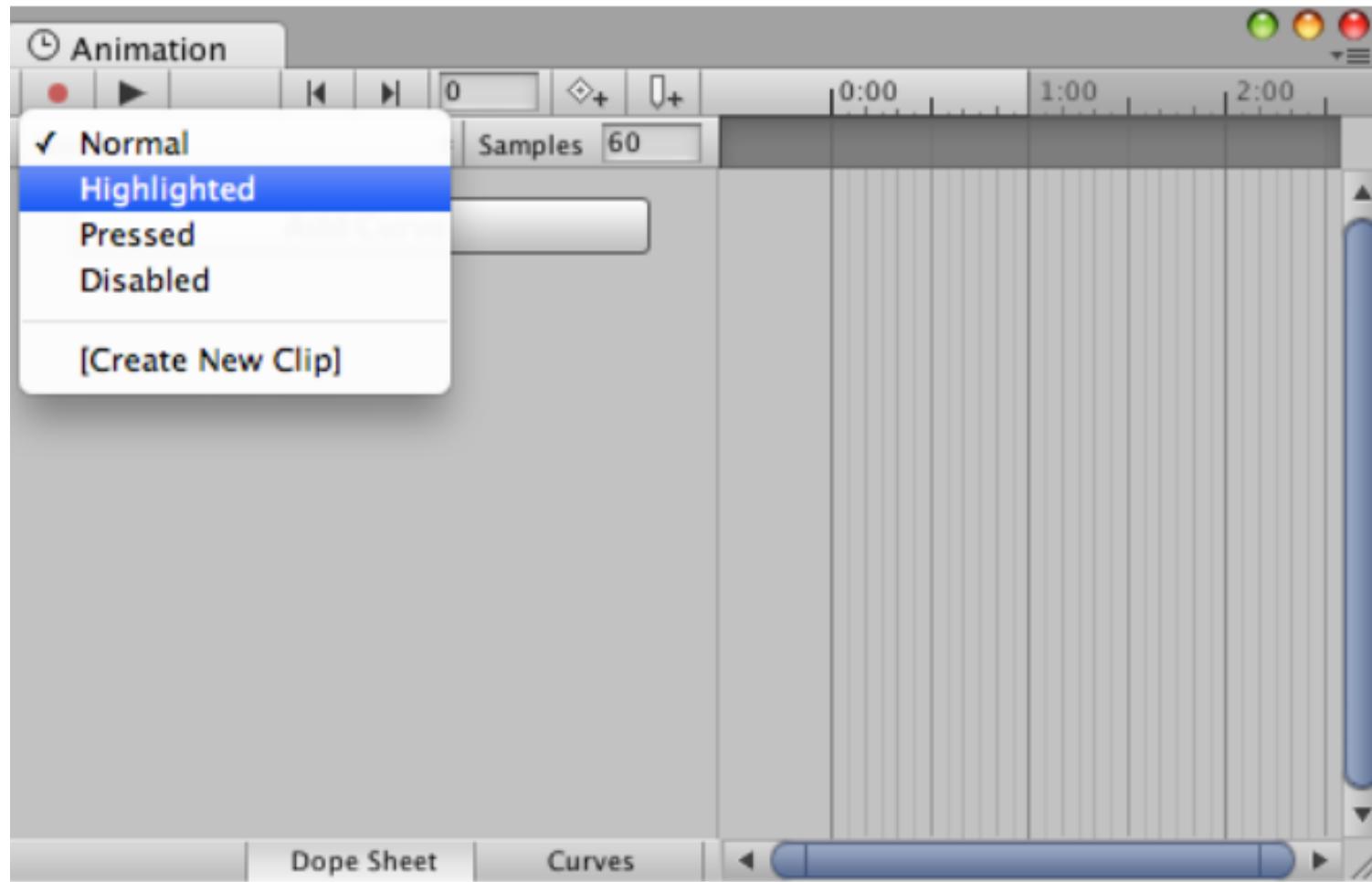
On Click (Button)

List is Empty

+ -

# Интеграция с анимацией





Например, если выбран элемент **Button** с прикрепленным контроллером Animator, анимации для каждого из состояний кнопки можно отредактировать, открыв окно Animation (Window>Animation). Во всплывающем меню **Animation Clip** вы можете выбрать нужный клип: «Обычный», «Выделенный», «Нажатый» и «Отключен».

# Советы по оптимизации UI

<https://habr.com/ru/company/funcorp/blog/470608/>

- Отключать невидимые и прозрачные объекты
- Минимизировать количество элементов
- Избегать пересечения объектов, не способных запечься друг с другом
- Распределять элементы по холстам в зависимости от частоты обновления
- У холстов с часто обновляющимися элементами отключить Pixel Perfect
- Отключить Raycast Target у не кликабельных элементов
- Удалить компонент Graphic Raycaster на холсте, у которого все элементы не кликабельны
- Для обычных текстовых компонентов не использовать Best Fit
- Для TextMeshPro В World Space использовать TextMeshPro вместо TextMeshProUGUI. TextMeshProUGUI используется в холстах
- Устанавливать камеру в настройках холста, если надо