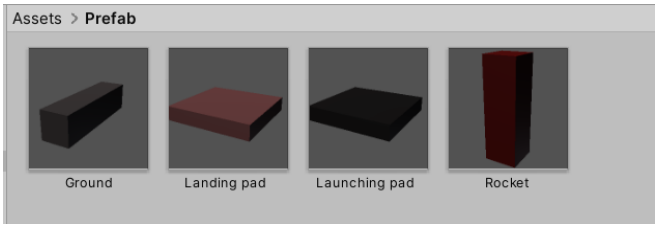
Create Prefabs from the scene objects

Step 1: Create prefab folder

- Go to Assets folder
- Right-Click
- Hover over the "create" option
- Then Choose "Folder"

Step2 : Create prefabs

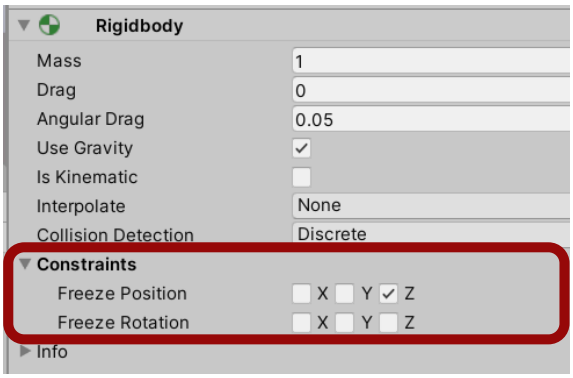
- Select and hold the object that you want to create a prefab to
- Then drag and drop it to the "Prefab" folder (Note : The object name in the scene will turn blue)



Rocket Movement and Rotation Constraints (1 of 2)

Step 1: Freeze the position constraint

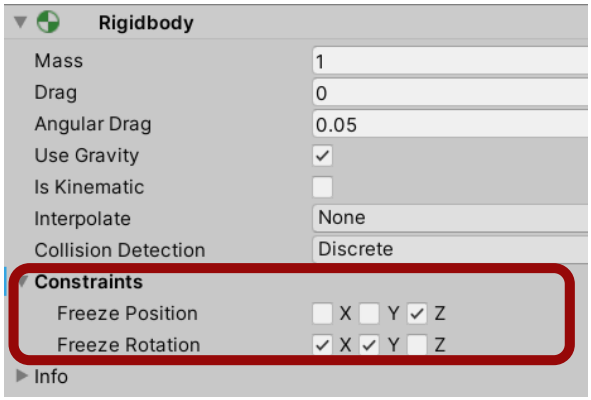
- Select the "Rocket" Object
- Go to the "Inspector"
- Scroll to the "Rigidbody" Component
- Within the "Rigidbody" Component, and under the Constraint option go to the "freeze position" fields and check the "z" box



Rocket Movement and Rotation Constraints (2 of 2)

Step 2: Freeze the rotation constraint

- Select the "Rocket" Object
- Go to the "Inspector"
- Scroll to the "Rigidbody" Component
- Within the "Rigidbody" Component, and under the Constraint option go to the "freeze rotation" fields and check "x" and "Y" boxes



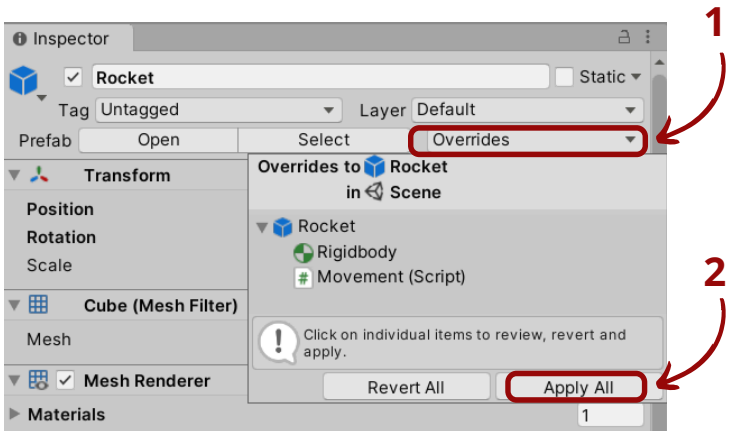
Step 3: Activate and deactivate the freezing constraints

- Open "Movement" Script
- Write the code below in the "ProcessRotation" function

```
void ProcessRotation()
{
    if (Input.GetKey(KeyCode.A))
    {
        rd.freezeRotation=true;
        transform.Rotate(Vector3.forward* RotateThrusting * Time.deltaTime);
        rd.freezeRotation=false;

    }

    else if (Input.GetKey(KeyCode.D))
    {
        rd.freezeRotation=true;
        transform.Rotate(-Vector3.forward* RotateThrusting * Time.deltaTime);
        rd.freezeRotation=false;

    }
}
```

Step 4: Update the "Rocket" object prefab

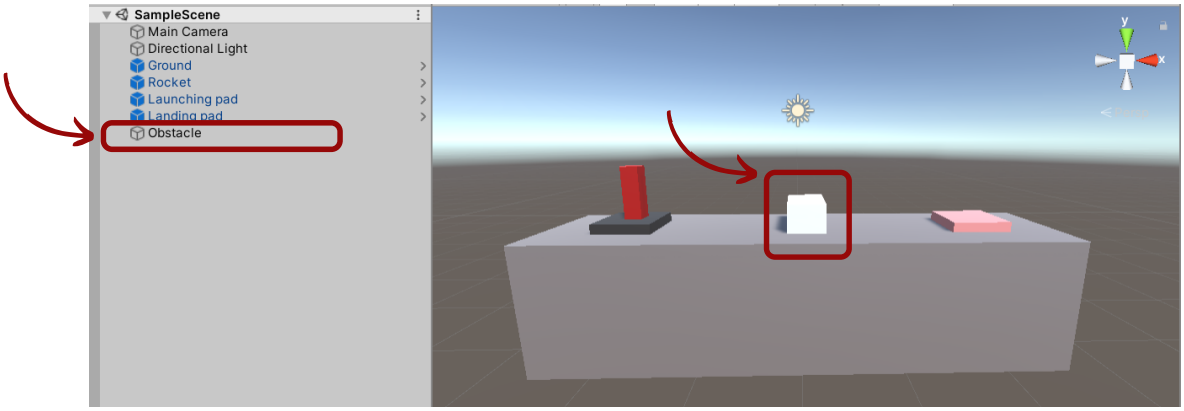
- Select the "Rocket" Object
- Go to the "Inspector"
- Click on the "Overrides" button
- Click on the "Apply All" button



Objects Collision (1 of 2)

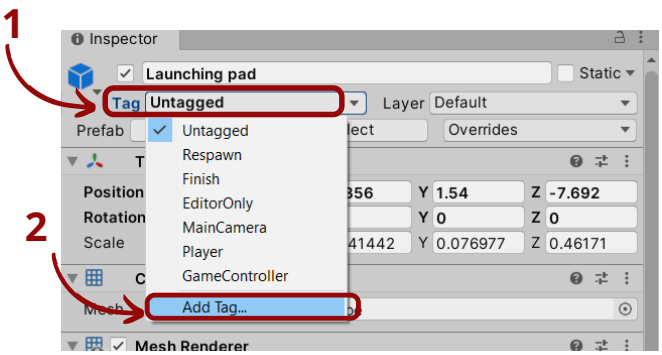
Step 1: Create an obstacle

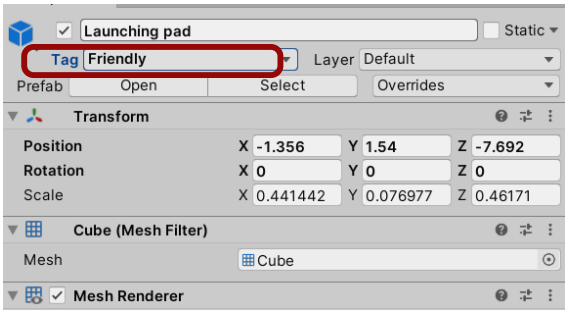
- Create a new 3D object using "Cube"
- Name it "Obstacle"
- Place it between the "launching pad" and the "landing pad" objects



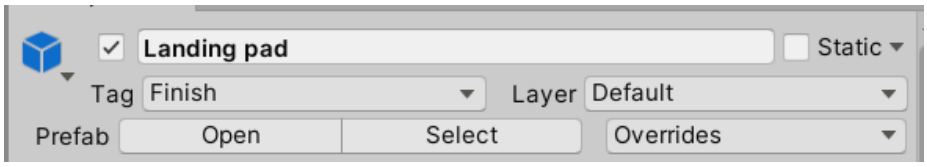
Step 2: Create New tags and Assign them to the objects

- Select the "launching pad" object
- Go to the "Inspector"
- Open the "Tag" list
- Choose "Add Tag"
- Click on the "+"
- Name the tag as "Friendly"
- Save the tag
- Go back and open the "Tag" list and choose "Friendly" tag





Step2 : Assign "Finish" tag to the "landing pad" object



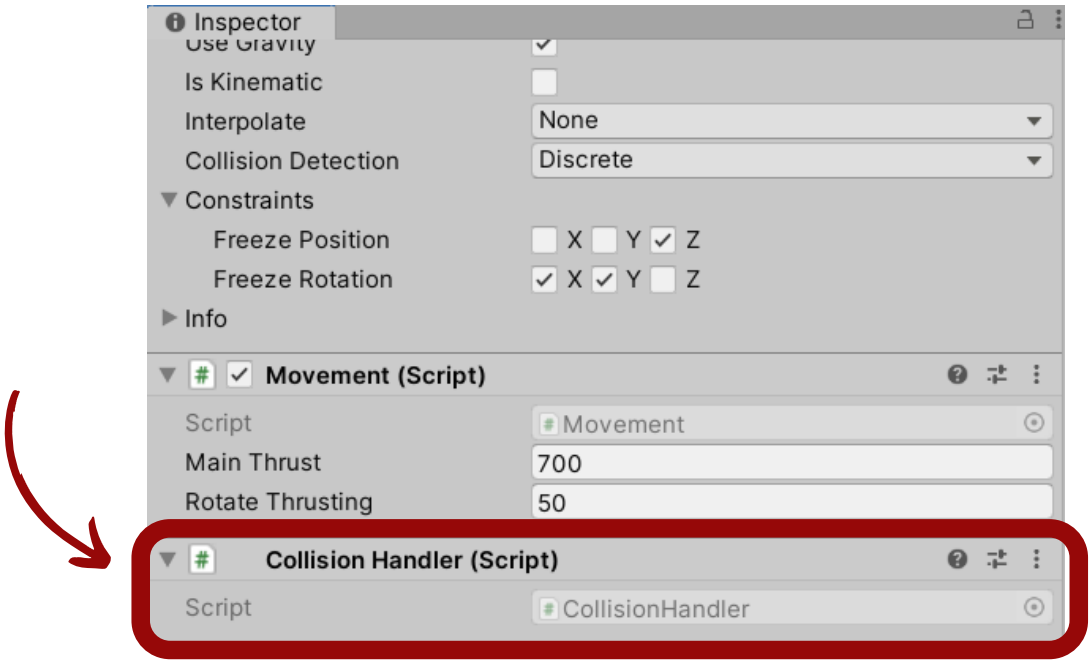
Objects Collision (2 of 2)

Step 1: Create the collision function

- Go to the Assets/Scripts folder
- Create a new C# script and name it "ColliderHandler"
- Open the Script and write the following

```
using UnityEngine;

0 references
public class CollisionHandler : MonoBehaviour
{
    0 references
    void OnCollisionEnter(Collision other)
    {
        if (other.gameObject.tag == "Friendly")
        {
            Debug.Log("This thing is friendly");
        }
        else if (other.gameObject.tag == "Finish")
        {
            Debug.Log("Finish, Add success audio sound");
        }
        else
        {
            Debug.Log("Add crash audio sound");
        }
    }
}
```

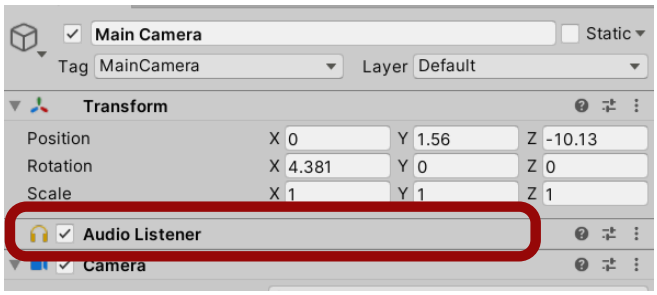
Step 2: Attach the "CollisionHandler" Script to the "Rocket" Object



Unity Audio (1 of 3)

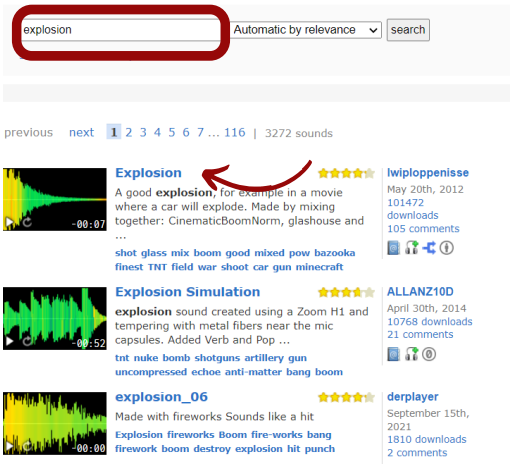
Step 1: Check that the camera has an audio listener

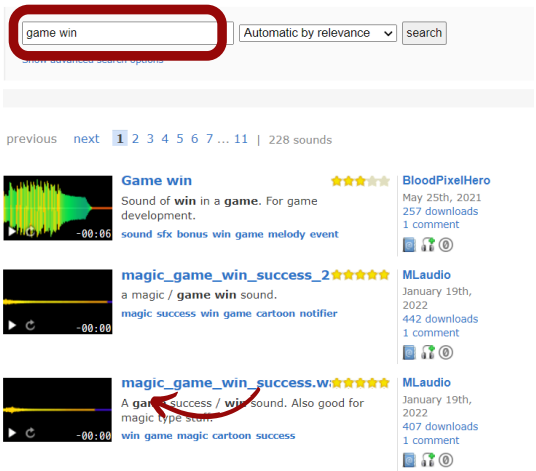
- Click on the "Main Camera" object
- Go to the Inspector and search for the "audio listener " component
- If not available, then add it (inspector-> Add component->audio listener)

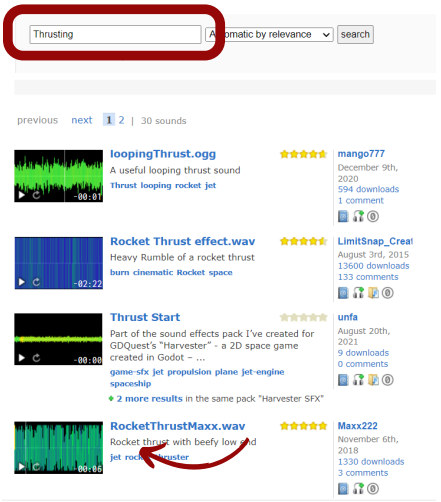


Step2 : Choose the background, success and crash sounds effect

- Go to https://freesound.org/
- Register using your preferred email and password
- Search for "explosion" and choose the first sound name "Explosion", then download it to your local drive (first figure to the left).
- Search for "game win" and choose the third sound name "magic_game_win_success", then download it to your local drive (the figure in the middle).
- Search for "thrusting" and choose the fourth sound name "RocketThrustMaxx", then download it to your local drive (the figure to the right).







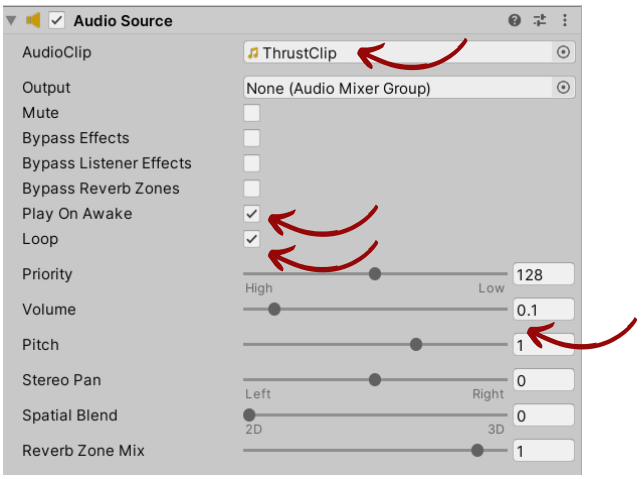
Step 3: Import and Create Sounds Folder

- Go to the project tab
- Go to the Assets folder
- Create new folder (right-click -> create -> folder), name it "Sounds"
- Import the sounds to "Sounds" folder

Unity Audio (2 of 3)

Step 4: Add the background sound effect

- Click on the "Ground" object
- Go to the Inspector
- Add "Audio Source" component (scroll-down -> Add component -> Audio Source)
- Add the "ThrustingClip" sound effect to the "AudioClip" field
- Check the "Play On Awake" option
- Check the "Loop" option
- Adjust the audio volume to be 0.1



Step 5: Add the success and crash sounds effect

- Click on the "Rocket" object
- Go to the Inspector
- Add "Audio Source" component (scroll-down -> Add component -> Audio Source)
- Uncheck the "Play on Awak" Option
- Open the "CollisionHandler" to modify its code

Step 6: Create a reference to the Audio source component and create clips fields

- Write the following code after the staring of the "CollisionHandler" class, and before the "OnCollisionEnter" function

```
public class CollisionHandler : MonoBehaviour
{
    [SerializeField] AudioClip success;
    1 reference
    [SerializeField] AudioClip crash;

    5 references
    AudioSource audioSource;

    0 references
    void Start()
    {
        audioSource = GetComponent<AudioSource>();
    }
    0 references
    void OnCollisionEnter(Collision other)
```

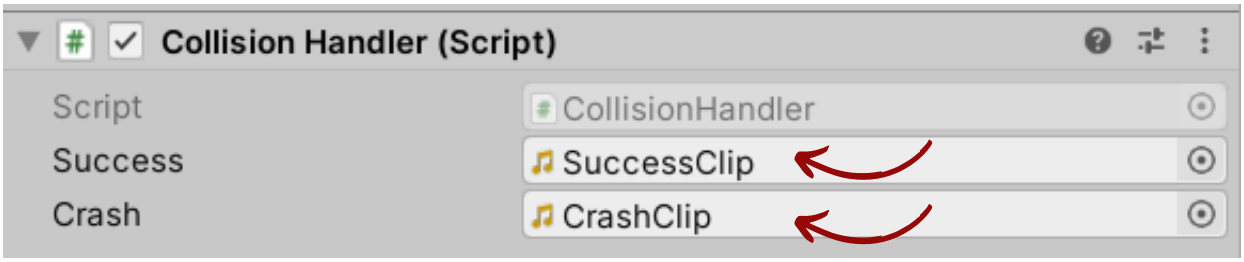
Unity Audio (3 of 3)

- Update the "OnCollisionEnter" function.

```
void OnCollisionEnter(Collision other)
{
    if (other.gameObject.tag == "Friendly")
    {
        Debug.Log("This thing is friendly");
    }
    else if (other.gameObject.tag == "Finish")
    {
        audioSource.Stop();
        audioSource.PlayOneShot(success);
    }
    else
    {
        audioSource.Stop();
        audioSource.PlayOneShot(crash);
    }
}
```

Step 7: Attach the Success and Crash sound effects

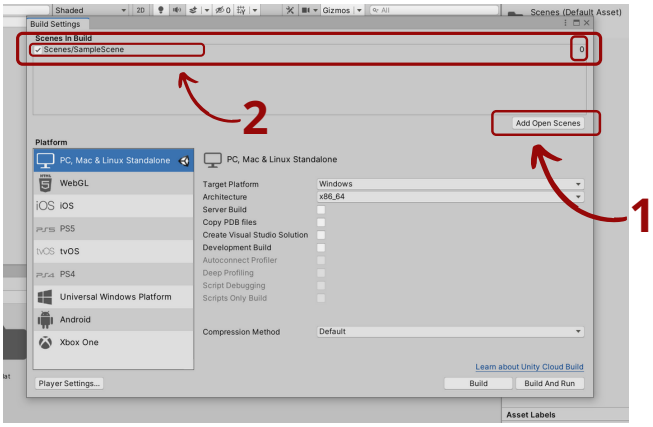
- Click on the "Rocket" object
- Go to the Inspector
- In the "CollisionHandler (Script)" Component, attach the "SuccessClip" to the "Success" field, and the "CrashClip" to the "Crash" field



Reload the Scene

Step 1: Add the current scene to the scene manager

- File -> Build settings
- Click on the "Add Open Scenes"
- Then the scene name will appear in the "Scenes In Build" Window with index "0" as the staring index.



Step 2: Update the "CollisionHandler" Script

- Open the "CollisionHandler" Script and first add the following statement before the beginning of the "CollisionHandler" class

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

0 references
public class CollisionHandler : MonoBehaviour
{
```

Step 3: Update the "CollisionHandler" Script

- Create "ReloadLevel" function after "OnCollisionHandler" function and Within the "CollisionHandler" class

```
void ReloadLevel()
{
    int C_Scence_I=SceneManager.GetActiveScene().buildIndex;
    SceneManager.LoadScene(C_Scence_I);
}
```

- Call the "ReloadLevel" function within the "OnCollisionHandler" function, more specifically within the "else" statement

```
void OnCollisionEnter(Collision other)
{
    if (other.gameObject.tag == "Friendly")
    {
        Debug.Log("This thing is friendly");
    }
    else if (other.gameObject.tag == "Finish")
    {
        audioSource.Stop();
        audioSource.PlayOneShot(success);
    }
    else
    {
        ReloadLevel();
        audioSource.Stop();
        audioSource.PlayOneShot(crash);
    }
}
```