Unity Game Engine

Introduction to Unity – Scene and Level Management

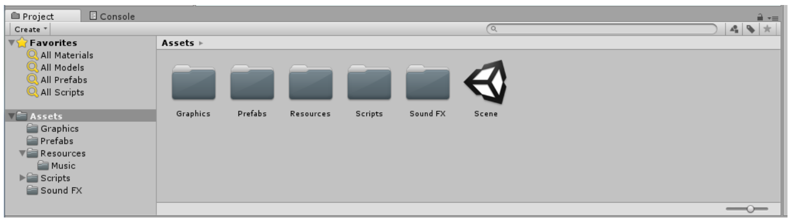
https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.html
https://docs.unity3d.com/ScriptReference/SceneManagement.Scene.html

Scenes

- Scenes contain the objects of your game. They can be used to create a main menu, individual levels, and anything else.
- In each Scene, you will place your environments, obstacles, and decorations, essentially designing and building your game in pieces.
- When you create a new Unity project, your scene view will show a new Scene.
  - This is an *untitled* and *unsaved* scene.
  - The scene will be empty except for default objects – a camera and a directional light.

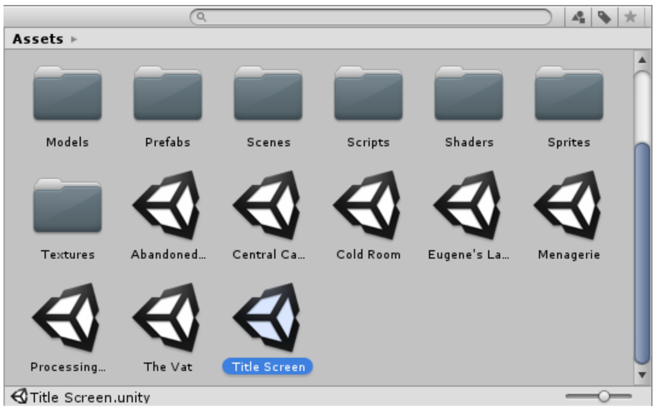
Save/Open Scenes

- To save the scene, choose **File > Save Scene** from the menu.
  - Scenes are saved as assets, into your project's Assets folder. Therefore they appear in the Project Window, just like any other asset.



- To open a scene, choose **File > Open Scene** from the menu or double-click the scene asset in the Project Window.

Saved Scene Assets visible in the Project window



Scene Management

- In Unity, a more coherent and logical scene management is provided through **`SceneManagement`** package.
- In order to manage scenes in script, include the package.

```
using UnityEngine.SceneManagement;
```

- This give access to three classes, `SceneManager`, `Scene` and `SceneUtilty` classes.

Scene

- Properties

`buildIndex`	Return the index of the Scene in the Build Settings.
`isDirty`	Returns true if the Scene is modifed.
`isLoaded`	Returns true if the Scene is loaded.
`name`	Returns the name of the Scene.
`path`	Returns the relative path of the Scene.

Load a Scene

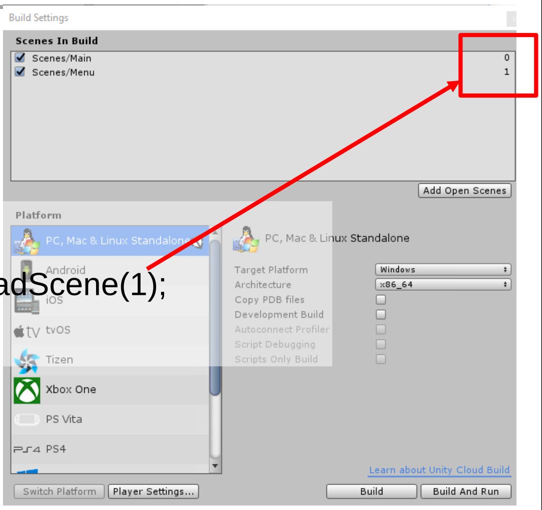
- `SceneManager.LoadScene()` loads the scene by its name or index in Build Settings (File > Build Settings.)

```
public static void LoadScene(string sceneName,
SceneManagement.LoadSceneMode mode =
LoadSceneMode.Single);
```

```
public static void LoadScene(int
sceneBuildIndex, SceneManagement.LoadSceneMode
mode = LoadSceneMode.Single);
```

Load a Scene - sceneBuildIndex

- Scene build index (File > Build Settings.)



✓ Scenes/Main	0
✓ Scenes/Menu	1

PC, Mac & Linux Standalone

Target Platform	Windows
Architecture	x86_64
Copy PDB files	☐
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐
Scripts Only Build	☐

SceneManager.LoadScene(1);

Load a Scene - sceneName

- The `sceneName` can either be the scene name only or the path as shown in the BuildSettings window.
- If only the scene name is given this will load the first scene in the list that matches.
- If you have multiple scenes with same name but different paths, you should use the full path.
- `sceneName` is case insensitive.

```
SceneManager.LoadScene("scene4")
SceneManager.LoadScene("path/to/scene/file/scene4")
```

Load a Scene - LoadSceneMode

- **`LoadSceneMode.Single` –** When a new scene in loaded with this method, the previous one is unloaded (all its objects destroyed). The game will freeze for a little bit while the new scene is being loaded and activated.
- **`LoadSceneMode.Additive` –** Adds the scene to the current loaded scenes.

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class ExampleClass : MonoBehaviour
{
  void Start () {
   SceneManager.LoadScene ("OtherSceneName", LoadSceneMode.Additive);
  }
}
```

SceneManager.GetActiveScene

- Gets the currently active Scene.

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class GetActiveSceneExample : MonoBehaviour
{
      void Start() {
        Scene scene = SceneManager.GetActiveScene();
        Debug.Log("Active scene is '" +
            scene.name + "'.");
      }
}
```

Reload the Current Scene

- In your game, you might need to reload the current scene. (e.g. restart the game when the player dies).
- Get the build index of the current scene, and pass it to `LoadScene()` as shown below:

```
SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);


public void restartCurrentScene()
{
  int scene = SceneManager.GetActiveScene().buildIndex;
  SceneManager.LoadScene(scene, LoadSceneMode.Single);
}
```

Scene Loading Bar Example

- Loading a scene takes time especially if your level is rather big or uses large assets.
- Let's delegate the loading of the new scene to Unity's dedicated background process. Then, periodically query the state of the loading and display it on the screen. When all the assets are loaded, we can finally trigger the activation of the scene. To achieve this, use `SceneManager.LoadSceneAsync`.

SceneManager.LoadSceneAsync

- Loads the scene asynchronously in the background.

```
public static AsyncOperation LoadSceneAsync(
    string sceneName,
    SceneManagement.LoadSceneMode mode =
          LoadSceneMode.Single);

public static AsyncOperation LoadSceneAsync(
    int sceneBuildIndex,
    SceneManagement.LoadSceneMode mode =
      LoadSceneMode.Single);
```

- Use the return value (of type `AsyncOperation)` to check if the operation has completed.

AsyncOperation

- Variables:
  - `isDone` - (Read Only) a boolean value that becomes true when the process is completed
  - `priority` - Priority lets you tweak in which order async operation calls will be performed.
  - `progress` - (Read Only) a number from 0 to 1 that indicates the current state of the process.
  - `allowSceneActivation` – If it is true, it allows scenes to be activated as soon as it is ready. If it is set to false, the scene is not activated when it is fully loaded, giving more control over the entire process.

- https://docs.unity3d.com/ScriptReference/AsyncOperation.html

Loading Bar Code Example

```
IEnumerator AsynchronousLoad (string scene)
{
  AsyncOperation ao = SceneManager.LoadSceneAsync(scene);
  ao.allowSceneActivation = false;

  while (! ao.isDone) {
    // [0, 0.9] > [0, 1]
    float progress = Mathf.Clamp01(ao.progress / 0.9f);
    Debug.log("Loading progress: " + (progress * 100) + "%");
    // Loading completed
    if (ao.progress == 0.9f) {
      Debug.Log("Press a key to start");
      if (Input.AnyKey())
        ao.allowSceneActivation = true;
    }
    yield return null;
  }
}
```

AsynchronousLoad

- Since we are required to update the loading bar in parallel with the loading process, we wrap out code into a coroutine.
- After invoking `SceneManager.LoadSceneAsync`, we force `allowSceneActivation` to false for a better control over the activation process.
- We then loop over `isDone`, starting the activation process when the loading is completed (progress == 0.9f).
- This code prints the current progress on the console, but it can be easily changed to update any UI you have designed.

- The code works very well if you have to load a single scene at a time.

Scene Fading Example

- When the player enters a trigger object (e.g. a door), LevelChanger will start cross fading.

- Scene Fading Scripts
  - `FadeInOut.cs`
  - `LevelChanger.cs`

LevelChanger.cs

```
using UnityEngine;
using System.Collections;

public class LevelChanger : MonoBehaviour {

    ScreenFader fadeScr;
    public int SceneNumb;

    void Awake() {
        fadeScr = GameObject.FindObjectOfType<ScreenFader>();
    }

    void OnTriggerEnter(Collider col) {
        if (col.gameObject.tag == "Player") {
            fadeScr.EndScene(SceneNumb);
        }
    }
}
```

When the player enters a trigger object (e.g. a door), LevelChanger will start cross fading.

FadeInOut.cs(1)

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

public class ScreenFader : MonoBehaviour {

  public Image FadeImg;
  public float fadeSpeed = 1.5f;
  public bool sceneStarting = true;

  void Awake(){
    FadeImg.rectTransform.localScale =
           new Vector2(Screen.width, Screen.height);
  }

  void Update(){
    // If the scene is starting...
    if (sceneStarting)
       // ...
       // call the StartScene function.
       StartScene();
  }
```

FadeInOut.cs(2)

```
  void FadeToClear(){
    // Lerp the colour of the image between itself and transparent.
    FadeImg.color = Color.Lerp(
          FadeImg.color, Color.clear, fadeSpeed * Time.deltaTime);
  }
  void FadeToBlack(){
    // Lerp the colour of the image between itself and black.
    FadeImg.color = Color.Lerp(
          FadeImg.color, Color.black, fadeSpeed * Time.deltaTime);
  }
  void StartScene(){
    // Fade the texture to clear color (0,0,0,0).
    FadeToClear();

    // If the texture is almost clear...
    if (FadeImg.color.a <= 0.05f) {
      // ... set the colour to clear and disable the RawImage.
      FadeImg.color = Color.clear;
      FadeImg.enabled = false;

      // The scene is no longer starting.
      sceneStarting = false;
    }
  }
```

FadeInOut.cs(3)

```
  public IEnumerator EndSceneRoutine(int SceneNumber) {
    // Make sure the RawImage is enabled.
    FadeImg.enabled = true;
    do {
      // Start fading towards black.
      FadeToBlack();
      // If the screen is almost black...
      if (FadeImg.color.a >= 0.95f) {
        // load the level
        SceneManager.LoadScene(SceneNumber);
        yield break;
      }
      else {
        yield return null;
      }
    } while (true);
  }

  public void EndScene(int SceneNumber) {
    sceneStarting = false;
    StartCoroutine("EndSceneRoutine", SceneNumber);
  }
}
```

Persistent Objects across Levels

- Because some object should persist in all scenes, it is best to separate them from individual levels of the game.
- There are several essential game elements that you must keep such as the player object, Controller, HUD Canvas, and etc.

Example Scenario

- The game consists of two scenes: *scene1* and *scene2*.
- *scene1* starts playing background music from an AudioSource.
- The music continues when *scene2* loads.
- Switch between scenes using a button.

- Example Scripts
  - `SceneSwap.cs`
  - `DontDestroy.cs`

Example Scenario Implementation

- Create two new Scenes, named *scene1* and *scene2*.
- Open *scene1*
  - Add the `SceneSwap.cs` script to an empty GameObject and name it **`Menu`**.
  - Next, add `DontDestroy.cs` to a new GameObject and name it **`BackgroundMusic`**.
  - Import an AudioClip into your Project.
  - Add an AudioSource to **`BackgroundMusic`** and
  - Assign the AudioClip to the AudioSource's AudioClip field. Create a tag, call it music, and add it to **`BackgroundMusic`**.
- Switch to *scene2*.
  - Again add `SceneSwap.cs` to a new GameObject and name it `Menu`.
- Save the Project.
- Return to *scene1* and run the Project from the Editor.

`Object.DontDestroyOnLoad`

- https://docs.unity3d.com/ScriptReference/Object.DontDestroyOnLoad.html

public static void **DontDestroyOnLoad**(Object **target**);

- Makes the object target not be destroyed automatically when loading a new scene.

```
using UnityEngine; using System.Collections;

public class ExampleClass : MonoBehaviour {
  void Awake() {
      DontDestroyOnLoad(transform.gameObject);
  }
}
```

```
public class SceneSwap : MonoBehaviour {
    private void OnGUI()     {
        ..

        Scene scene = SceneManager.GetActiveScene();

        if (scene.name == "scene1") {
             if (GUI.Button(new Rect(xCenter - width / 2,
                   yCenter - height / 2, width, height),
                   "Load second scene", fontSize)) {
                SceneManager.LoadScene("scene2");
            }
        }
        else {
             if (GUI.Button(new Rect(xCenter - width / 2,
                  yCenter - height / 2, width, height),
                   "Return to first scene", fontSize)) {
                SceneManager.LoadScene("scene1");
            }
        }
    }
}
```

3D Computer Game Programming

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


public class DontDestroy : MonoBehaviour
{
    void Awake()
    {
        GameObject[] objs =
            GameObject.FindGameObjectsWithTag("music");

        if (objs.Length > 1) {
            Destroy(this.gameObject);
        }

        DontDestroyOnLoad(this.gameObject);
    }
}
```

3D Computer Game Programming

Resources

- Scripts
  - `FadeInOut.cs`
  - `LevelChanger.cs`
  - `SceneSwap.cs`
  - `DontDestroy.cs`