Introduction into Game Programming (CSC329)

Ubbo Visser

Department of Computer Science
University of Miami

March 10, 2021

UNIVERSITY
OF MIAMI

Outline

Creating and Using Scripts

Creating and Using Scripts

- Standard anatomy of a script file: `Start()`, `Update()`
- Controlling a GameObject: `Debug.Log("text")`, or `print("text")`, e.g. in `Start()`
- Variables: public: can be seen in Inspector of game object, private: hidden from Inspector.

Controlling GameObjects

Controlling GameObjects Using Components

- Accessing Components: e.g.,
  `Rigidbody rb = GetComponent<Rigidbody>();`
- Accessing Other Objects
  - Linking objects with variables: `public GameObject player;`
  - Finding child objects: child objects can be retreived using the parent's Transform Component: e.g. `transform.Find("Hero");`
  - Finding objects by name or tag: e.g. `player = GameObject.Find("MainHeroCharacter");`
- Creating and Destroying GameObjects: `Instantiate`, `Destroy`

Event functions

Event functions

- Regular update events: `Update`, `FixedUpdate`, `LateUpdate`, e.g. for camera settings.
- Initialization Events: `Start()` (called before the first frame or physics update), `Awake()` (called for each object in the scene at the time when the scene loads)
- UI events: called periodically; e.g. `void OnGUI()`
- Physics events:
  - `OnCollisionEnter`, `OnCollisionStay`, `OnCollisionExit`.
  - `OnTriggerEnter`, `OnTriggerStay`, `OnTriggerExit`
- See flowchart:
  Unity Flowchart

Coroutines

Coroutines

- Example: consider the task of gradually reducing an object's alpha (opacity) value until it becomes completely invisible
- `StartCoroutine`
- WaitForSeconds, e.g. `yield return new WaitForSeconds(.1f);`

Platform Dependent Compilation

Software example

```
using UnityEngine;
using System.Collections;

public class PlatformDefines : MonoBehaviour
{ void Start () {
   #if UNITY_EDITOR
      Debug.Log("Unity Editor");
   #endif
    #if UNITY_IPHONE
      Debug.Log("IPhone");
   #endif
   #if UNITY_STANDALONE_OSX
      Debug.Log("Stand Alone OSX");
   #endif
   #if UNITY_STANDALONE_WIN
      Debug.Log("Stand Alone Windows");
   #endif
}
```

Special Folders and Script Compilation Order

Special Folders and Script Compilation Order

- Example: script written in one language must refer to a class defined in another language (say, a UnityScript file that declares variables of a class defined in a C# script). The rule here is that the class being referenced must have been compiled in a earlier phase.
- Phase 1: Runtime scripts in folders called Standard Assets, Pro Standard Assets and Plugins.
- Phase 2: Editor scripts in folders called Standard Assets/Editor, Pro Standard Assets/Editor and Plugins/Editor.
- Phase 3: All other scripts that are not inside a folder called Editor.
- Phase 4: All remaining scripts (ie, the ones that are inside a folder called Editor).
- Any script inside a folder called WebPlayerTemplates at the top level of the Assets folder will not be compiled at all.

Attributes

Attributes

- Attributes are markers that can be placed above a class, property or function in a script to indicate special behaviour.
- For example, the `HideInInspector` attribute can be added above a property declaration to prevent the property being shown in the inspector, even if it is public.
- `HideInInspector` public float strength;

Camera Settings

1. option: follow game object by parenting

- In project hierarchy: drag main camera object as a child to the object to follow. Avoid doing this for objects that rotate.

2. option: follow game object without parenting

- In `Start()` or `Awake()` find game object to follow, use `FindGameObjectWithTag` for example.
- Get the offset between object to follow and actual camera position.
- Track changes in the position, not the rotation of the game object.
- In `Update()` or `LateUpdate()` assign new transform position + offset.

Prefabs

Prefabricated objects with all characteristics in scene view

- Choose your sprite from sprites folder and drag it into scene view to make a new GameObject.
- Goto Anim folder and drag object's annimation onto GameObject in scene view.
- Add collision component, physics, circle/sphere collider, trigger', adjust collision box etc.
- Make folder Prefab under Assets.
- Right-click to make new prefab, change name of prefab.
- Select GameObject from hierarchy and drag it onto prefab symbol in prefab folder.
- Drag new prefabs into scene.
- When changes are necessary: add componentn onto **one** prefab.
- Hit `apply` at prefab section in Inspector.
- Very quick way of updating prefabs.

UI Feedback

Simple feedback using UI text on screen

- Create game object for UI Text in scene.
- Adjust font, color etc, check best fit to makes a difference when screen resolutions vary.
- Set location of text on screen with anchor.
- Create script on the (empty) game object (e.g. PowerUpCounter) using public variable.
- Drag text game object from project hierarchy onto public variable in inspector of game object.
- Increase/decrease counter in different class.

Killbox

Simple killbox to restart game when lost

- Needed when player looses the game.
- Basic idea: place a trigger outside of the game scene. If player hits that trigger, go back to start of the game.
- Create killbox object, long, wide, deep, use 2D object in 2D game as 2D and 3D objects use different physics engines.
- Create killbox script.
- OnTrigger: Application: load scene (0) with SceneManager.

Finishbox

Simple finish for a game

- Idea: show player that they have won.
- Create UI text object that will appear first (winner's text).
- Create game object that triggers finish.
- Create script for finish object.
- Drag script onto Finish object.
- Drag finishText object onto winner's text variable in inspector (public variable).