Programming in Unity

1

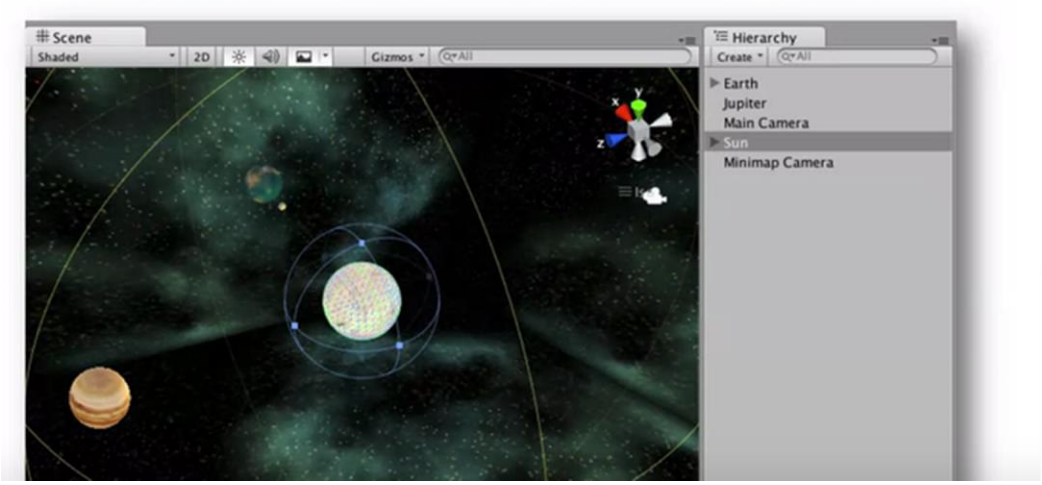
Object-Oriented Thinking

- Everything is an object
- Objects have properties (variables)
- Objects have methods (functions)

- GameObjects as Objects

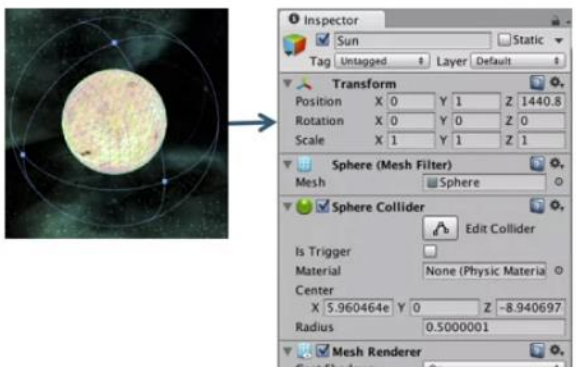
2

GAMEOBJECTS AS OBJECTS



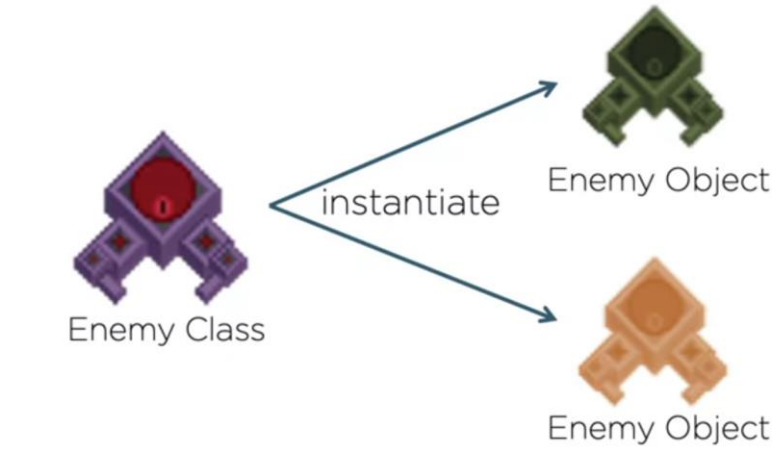
3

COMPONENTS AS OBJECTS



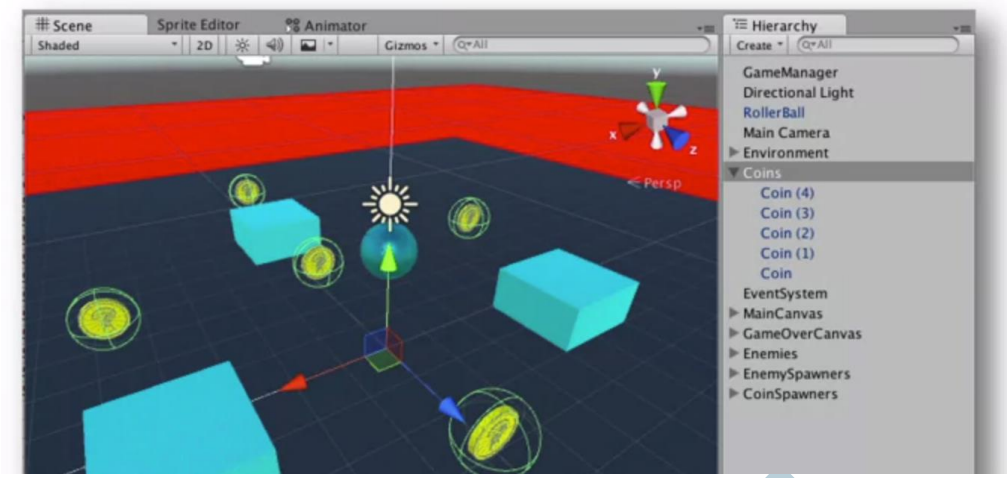
4

CLASSES



5

PREFAB = CLASS

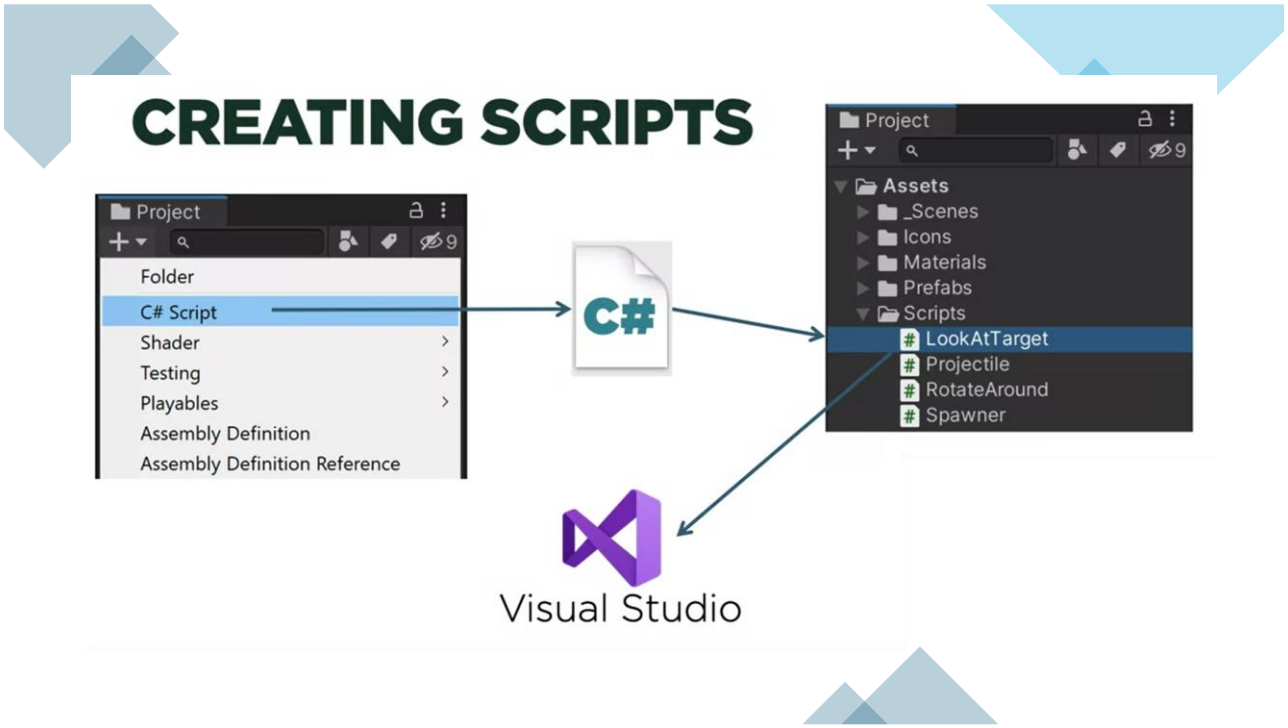


6

Object-Oriented Concepts

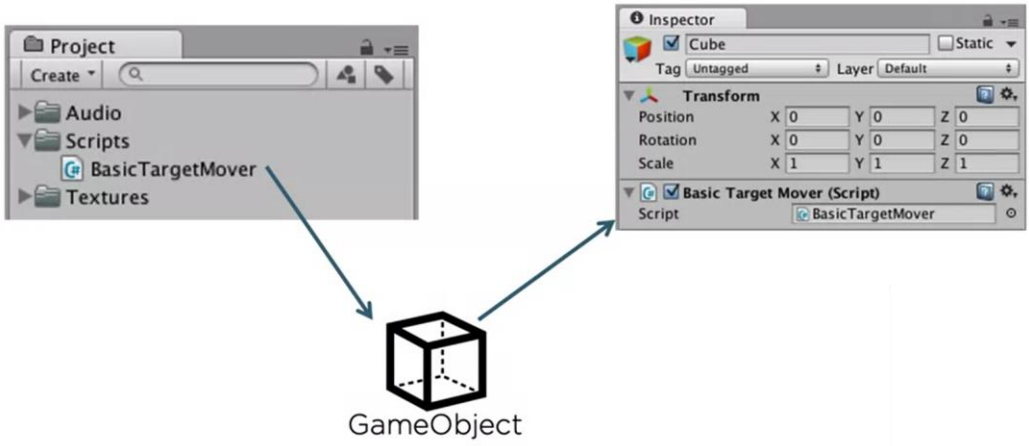
- **Classes** are templates for **objects**
- Classes are linked in a **hierarchy**
- A class can do anything its parent class can (**inheritance**)
- A class can override its parent's functionality (**polymorphism**)
- Objects are **instantiated** from a class
- An object gets all the **properties** and **methods** that are defined in the class

7



8

SCRIPTS AS COMPONENTS



9

ANATOMY OF A C# SCRIPT

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BasicTargetMover : MonoBehaviour
6  {
7      public bool doSpin = true;
8      public float spinSpeed = 180.0f;
9
10     private Transform mover;
11
12     private void Start()
13     {
14         mover = gameObject.transform;
15     }
16
17     private void Update()
18     {
19         if (doSpin)
20         {
21             mover.Rotate(Vector3.up * spinSpeed * Time.deltaTime);
22         }
23     }
24 }
```

Use this stuff

Class definition

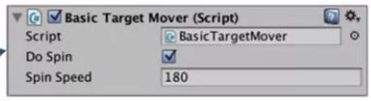
Class Variables

Class Functions

10

ANATOMY OF A C# SCRIPT

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BasicTargetMover : MonoBehaviour
6  {
7      public bool doSpin = true;
8      public float spinSpeed = 180.0f;
9
10     private Transform mover;
11
```



11

ANATOMY OF A C# SCRIPT

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BasicTargetMover : MonoBehaviour
6  {
7      public bool doSpin = true;
8      public float spinSpeed = 180.0f;
9
10     private Transform mover;
11
12     private void Start()
13     {
14         mover = gameObject.transform;
15     }
16
17     private void Update()
18     {
19         if (doSpin)
20         {
21             mover.Rotate(Vector3.up * spinSpeed * Time.deltaTime);
22         }
23     }
24 }
```

12

CLASSES

```
public class Player : MonoBehavior
```

Access — Class keyword — Class Name — Base Class

13

EVENT-DRIVEN

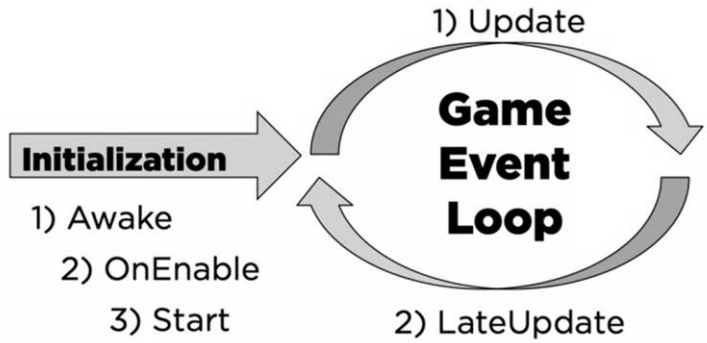
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DemoEvents : MonoBehavior
{
    void Start () {
        Debug.Log("Happens Once");
    }

    void Update() {
        Debug.Log("Happens Many Time");
    }
}
```

14

EVENT-DRIVEN



Other Events:
FixedUpdate
OnCollisionEnter
OnTriggerEnter
OnMouseDown
OnMouseUp
custom

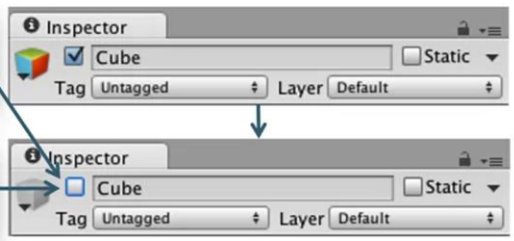
http://docs.unity3d.com/Manual/ExecutionOrder.html

15

REFERENCING GAMEOBJECTS

```
void Start () {
    // make this gameObject inactive (invisible)
    this.gameObject.SetActive(false);
}
```

```
void Start () {
    // make this gameObject inactive (invisible)
    gameObject.SetActive(false);
}
```



this.gameObject = gameObject

16

REFERENCING GAMEOBJECTS

```
1  public GameObject target1;
2
3  private GameObject target2;
4  private GameObject target3;
5
6  void Start () {
7      // target1 is set in the editor
8      // target2 is set based on the GameObject name
9      target2 = GameObject.Find("Boss");
10     // target3 is set based on the GameObject tag
11     target3 = GameObject.FindWithTag("Enemy");
12
13     // make the gameObjects inactive
14     target1.SetActive(false);
15     target2.SetActive(false);
16     target3.SetActive(false);
17 }
```

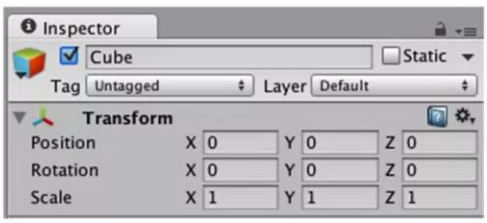
Public variable

Find based on name

Find based on tag

17

REFERENCING COMPONENTS



= gameObject.transform

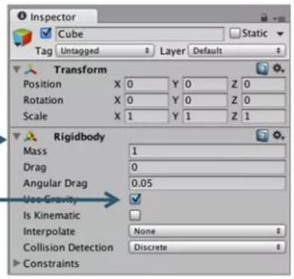
```
1  void Update () {
2      gameObject.transform.position.x += 1;
3  }
```

```
1  void Update () {
2      gameObject.transform.Translate(1,0,0);
3  }
```

18

REFERENCING COMPONENTS

```
1  private Rigidbody rb;
2
3  void Start () {
4      // get a reference to the Rigidbody component
5      rb = gameObject.GetComponent<Rigidbody>();
6
7      // turn gravity on
8      rb.useGravity = true;
9  }
```



gameObject.GetComponent<TYPE>()

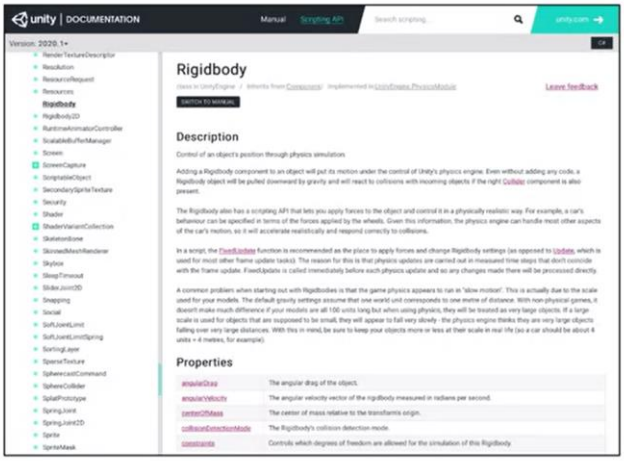
19

REFERENCING COMPONENTS

Types	Example
RigidBody	gameObject.GetComponent<RigidBody>()
RigidBody2D	gameObject.GetComponent<RigidBody2D>()
Collider	gameObject.GetComponent<Collider>()
Collider2D	gameObject.GetComponent<Collider2D>()
AudioSource	gameObject.GetComponent<AudioSource>()
SpriteRenderer	gameObject.GetComponent<SpriteRenderer>()
Animation	gameObject.GetComponent<Animation>()
Text	gameObject.GetComponent<Text>()

20

UNITY SCRIPTING API DOCS



21

SPAWNING GAMEOBJECTS

```
1  public class Spawn : MonoBehavior
2  {
3      public GameObject prefab;
4
5      void Update () {
6          if (Input.GetButtonDown("Fire1")) {
7              Vector3 spawnPosition = this.gameObject.transform.position;
8              Quaternion spawnRotation = this.gameObject.transform.rotation;
9
10             GameObject spawnObject = Instantiate(prefab, spawnPosition, spawnRotation);
11         }
12     }
13 }
```

22