VIA University College

Scripting

Learning to code in Unity

Scripts as Behaviour Components

Get an overview of how code is structured in Unity

Event Functions

Learn how Unity uses Inversion of Control to put you in control

Getting Started with Scripting

Get familiar with fundamental methods in the Unity API

Exercises

Use event functions, create character controllers and much more!

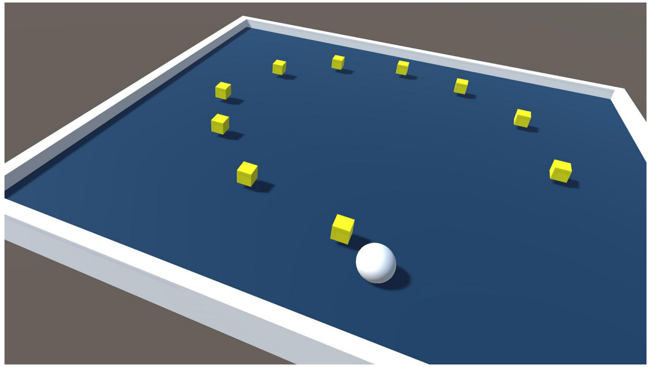
Last Week

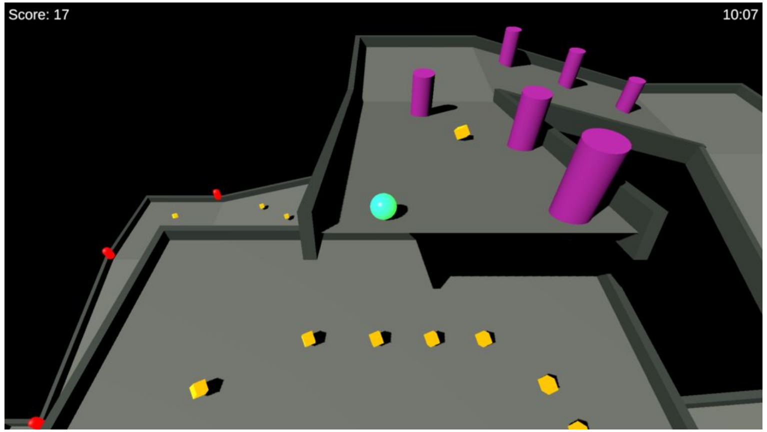
Do you feel comfortable with the editor yet?

Creating/Manipulating game objects?

Roll-a-ball: Did you finish the tutorial?

Did you extend your game with an obstacle course?





Last Week

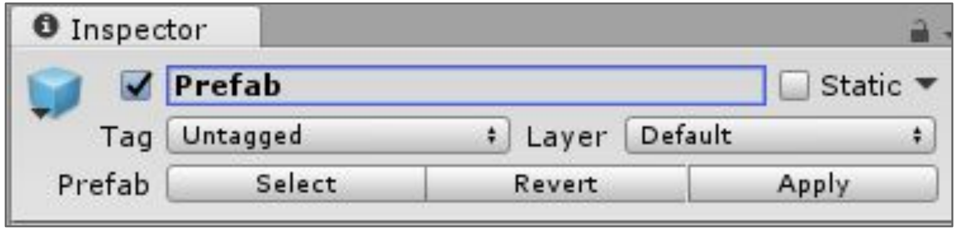
MDA?

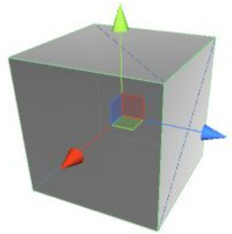
The engineer's role in the industry?

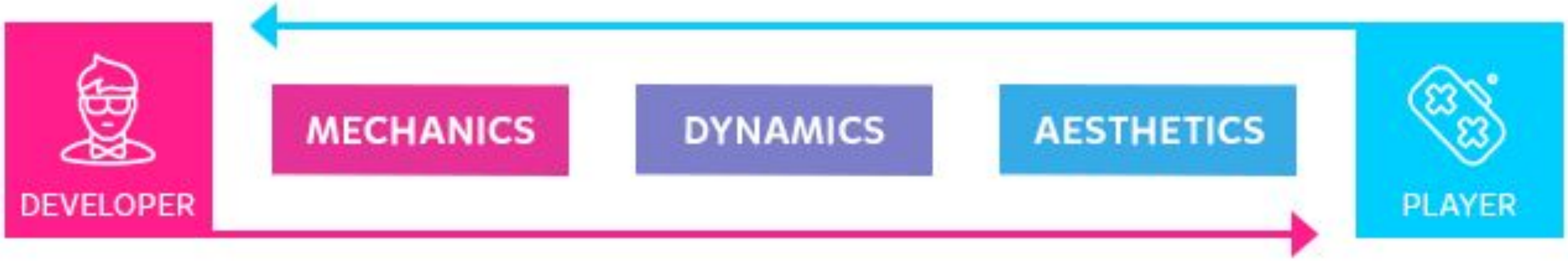
Core functionality in a game engine?

Important views in Unity?

GameObjects, Components and Assets?







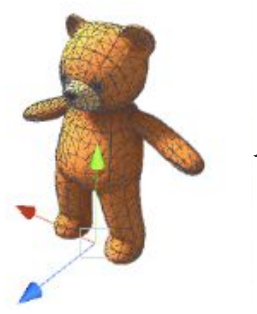
What we already know

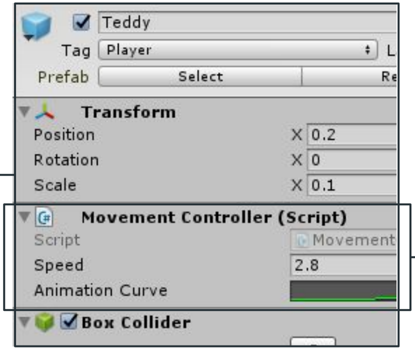
GameObjects live in Scenes

Components live on GameObjects

We can create our own components via scripting. By doing so we are adding custom behavior to our GameObject.

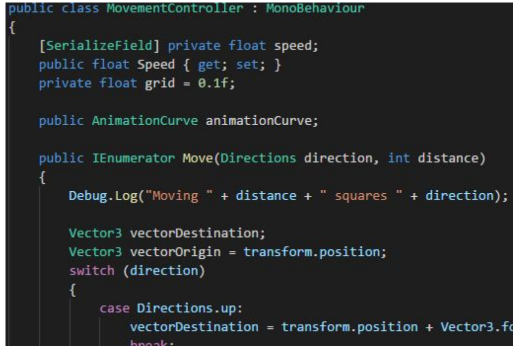
GameObject





```
Teddy
Tag     Player                     L
Prefab  Select                     Re
Transform
Position                  X 0.2
Rotation                  X 0
Scale                     X 0.1
Movement Controller (Script)
Script                    Movement
Speed                     2.8
Animation Curve
Box Collider
```

Custom Component / Script



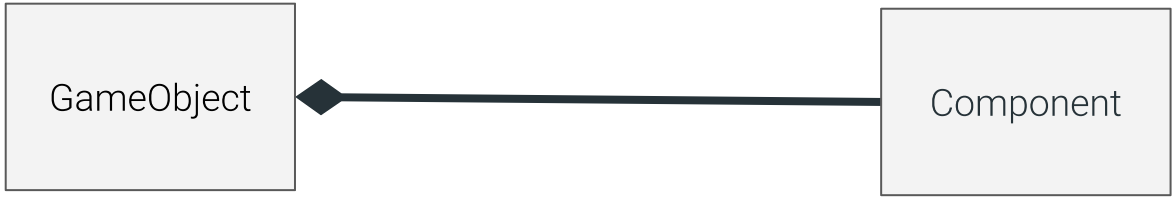
```
public class MovementController : MonoBehaviour
{
    [SerializeField] private float speed;
    public float Speed { get; set; }
    private float grid = 0.1f;

    public AnimationCurve animationCurve;

    public IEnumerator Move(Directions direction, int distance)
    {
        Debug.Log("Moving " + distance + " squares " + direction);

        Vector3 vectorDestination;
        Vector3 vectorOrigin = transform.position;
        switch (direction)
        {
            case Directions.up:
                vectorDestination = transform.position + Vector3.f
```

Scripting in Unity

A script is a .cs file, often consisting of just one C# class

MonoBehaviour is the base class from which (almost) every Unity script derives

The relationship between a GameObject and a Component is **composition**



A GameObject **HAS A** Component

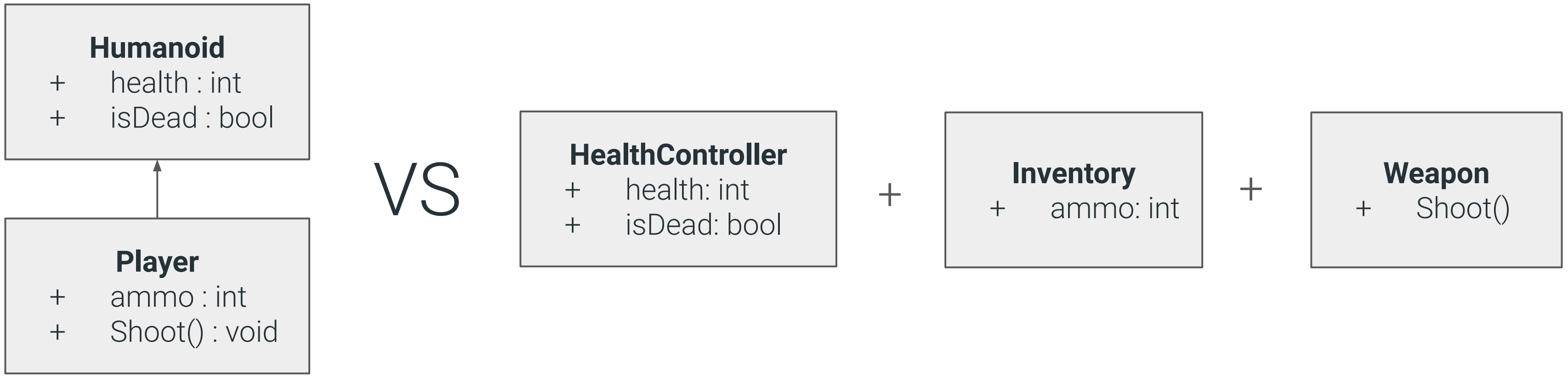
When writing code in Unity you...

Scripts as Behaviour Components

… add behavior to your GameObjects

… should use **composition over inheritance**

… should follow the single responsibility principle



Event Functions

Unity passes control to a script intermittently by calling certain functions that are declared within it

Once a function has finished executing, control is passed back to Unity

Don’t call us, we’ll call you

Learn more!

Event Functions

Regular Update Events

- Update
- FixedUpdate
- LateUpdate

Initialization Events

- Awake
- Start
- OnEnable

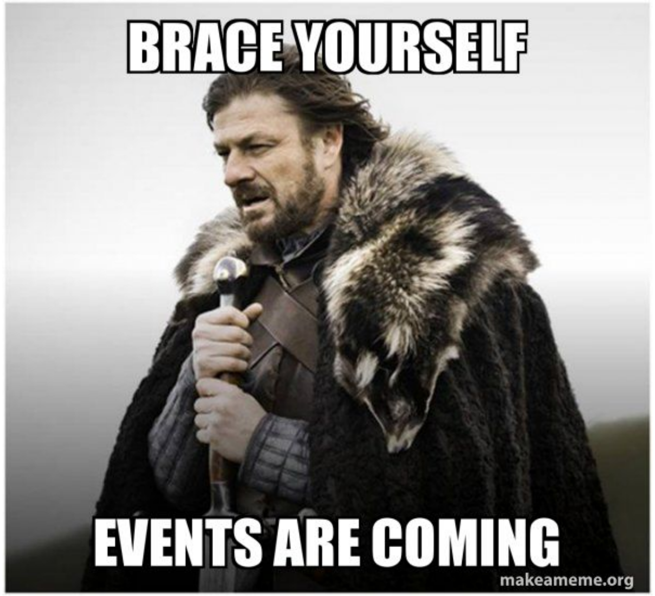
Input Events

- OnMouseOver, OnMouseDown

Physics Events

- OnCollisionEnter, OnCollisionStay, OnCollisionExit
- OnTriggerEnter, OnTriggerStay, OnTriggerExit

And many others… (OnDisable, OnDestroy, OnGUI, etc…)



Order of Execution for Event Functions

Primary Event Functions for Today

void Start()

Called before first frame update, if the script instance is enabled (once per lifetime of the script)

void Update()

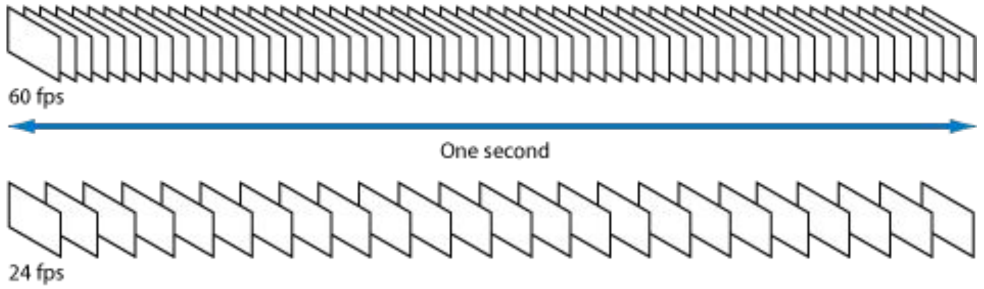
Called once per frame, before render

Used for non-physics objects

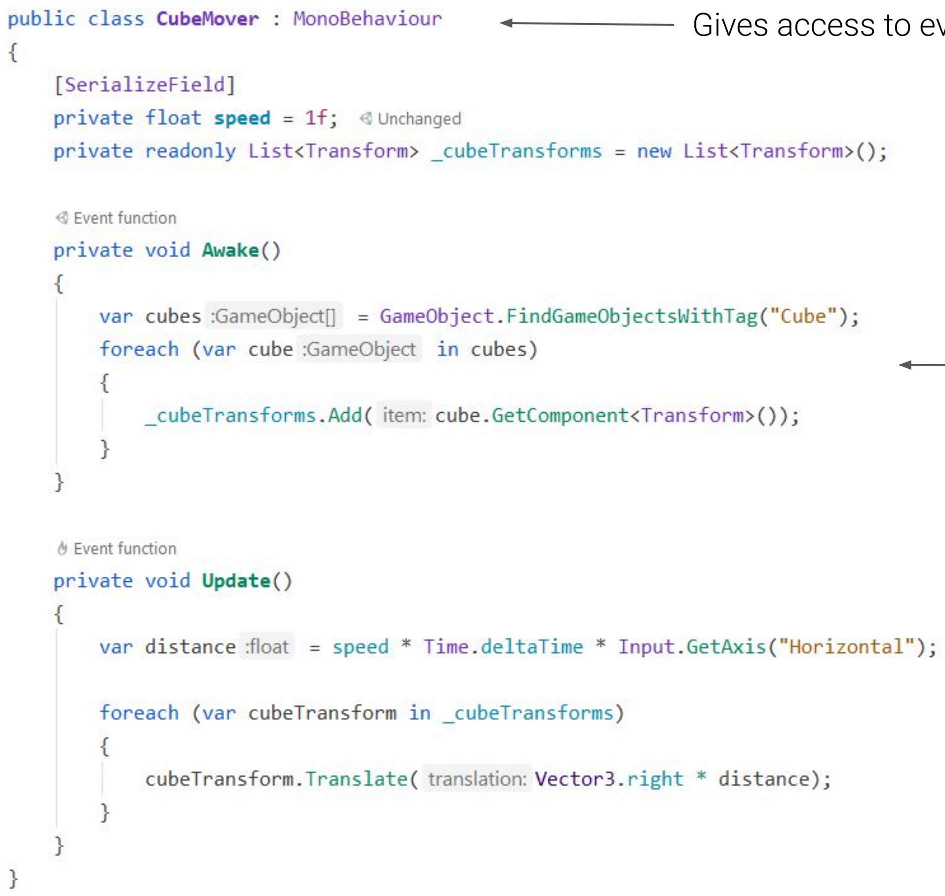
Simple Timers

Receiving Input

Careful: Update interval times vary!



Event Functions, Example Usage



```
public class CubeMover : MonoBehaviour
{
    [SerializeField]
    private float speed = 1f;  Unchanged
    private readonly List<Transform> _cubeTransforms = new List<Transform>();

    Event function
    private void Awake()
    {
        var cubes :GameObject[]  = GameObject.FindGameObjectsWithTag("Cube");
        foreach (var cube :GameObject  in cubes)
        {
            _cubeTransforms.Add( item: cube.GetComponent<Transform>());
        }
    }

    Event function
    private void Update()
    {
        var distance :float  = speed * Time.deltaTime * Input.GetAxis("Horizontal");

        foreach (var cubeTransform in _cubeTransforms)
        {
            cubeTransform.Translate( translation: Vector3.right * distance);
        }
    }
}
```

Gives access to event functions

Setup references, initialization

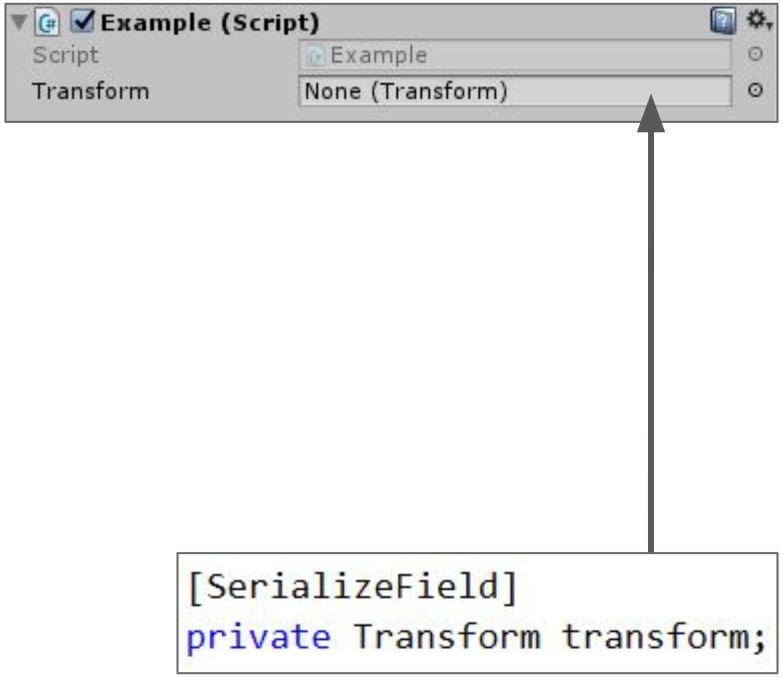
Called each frame

Accessing Data

Accessing variables in the editor

Public / **[SerializeField]**

- Easy and convenient way to change variable values
- Can be used for inter-object communication!



```
[SerializeField]
private Transform transform;
```

Accessing Components and GameObjects through code

gameObject, transform...

GetComponent<Type>()

GameObject.FindGameObjectWithTag(string tag)

Learn more!

Transforming GameObjects

Vector3 vector = new Vector3(x,y,z)

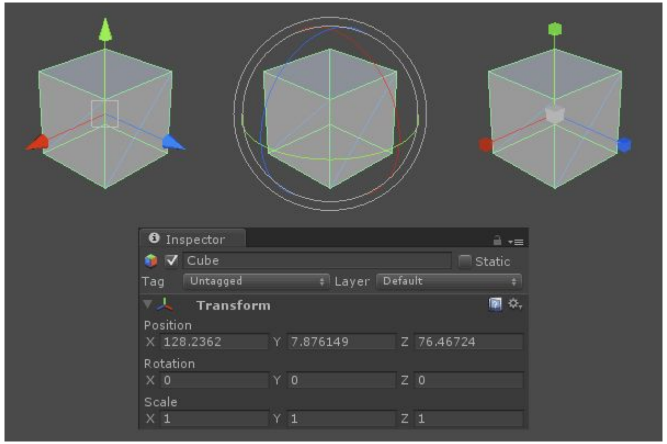
Object movement can be done **with or without the physics engine**

Without, using the transform of a GameObject:

Position: transform.Translate(vector)

Rotation: transform.Rotate(vector,angle)

Scale: transform.localScale = vector



Inspector
Cube Static
Tag Untagged Layer Default
Transform
Position
X 128.2362 Y 7.876149 Z 76.46724
Rotation
X 0 Y 0 Z 0
Scale
X 1 Y 1 Z 1

Learn more! Learn more!

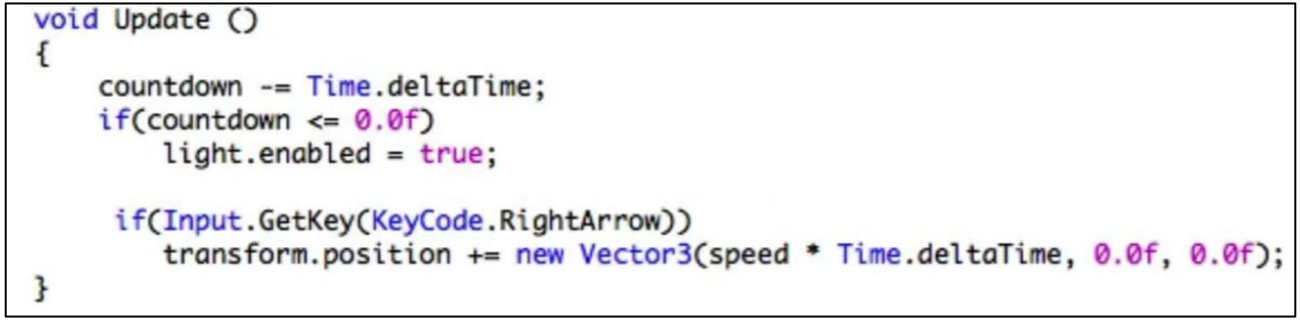
Time and Frame Management

Time.deltaTime = the time between each update call (between each frame)

Device Dependant! Not a constant!

When you multiply with Time.deltaTime, think of it as converting the data from per frame to **per second**.

Used to smooth out values used for movement, and other incremental calculations (not only for movement)



```
void Update ()
{
    countdown -= Time.deltaTime;
    if(countdown <= 0.0f)
        light.enabled = true;

     if(Input.GetKey(KeyCode.RightArrow))
        transform.position += new Vector3(speed * Time.deltaTime, 0.0f, 0.0f);
}
```

Learn more!

Input.GetKey()

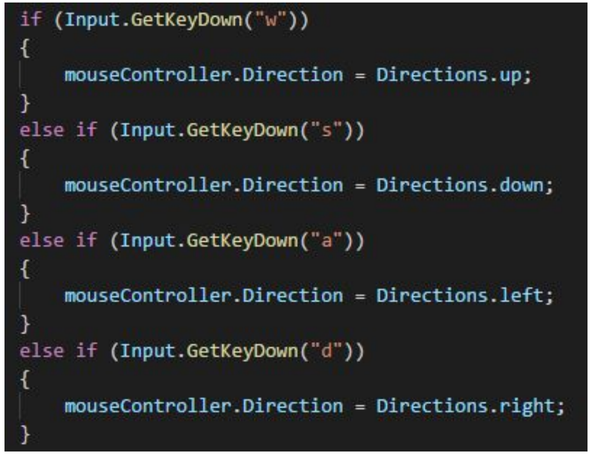
KeyCode or string name as argument. Examples: KeyCode.Space, “d”

GetKeyDown and GetKeyUp as well

Returns a boolean

Where do you think we should put this code?

More on proper input management later...



```
if (Input.GetKeyDown("w"))
{
    mouseController.Direction = Directions.up;
}
else if (Input.GetKeyDown("s"))
{
    mouseController.Direction = Directions.down;
}
else if (Input.GetKeyDown("a"))
{
    mouseController.Direction = Directions.left;
}
else if (Input.GetKeyDown("d"))
{
    mouseController.Direction = Directions.right;
}
```

Learn more!

Activating and Instantiating

Activate/deactivate GameObjects:

- gameObject.SetActive(true/false)

Instantiate/destroy GameObjects:

- Instantiate(prefab)
  - Instantiate returns a GameObject.
- Destroy(gameObject, time)

Enabling/Disabling Components:

- component.enabled = true/false

Learn more! Learn more! Learn more!

Invoking Methods

Used to schedule methods calls to occur at a later time.

Invoke("MethodName", delay)

- Only methods, that have no parameters and return type void can be invoked.

InvokeRepeating("MethodName", initialDelay, delayBetweenEachCall)

- Invoke a method over and over again!

CancelInvoke()

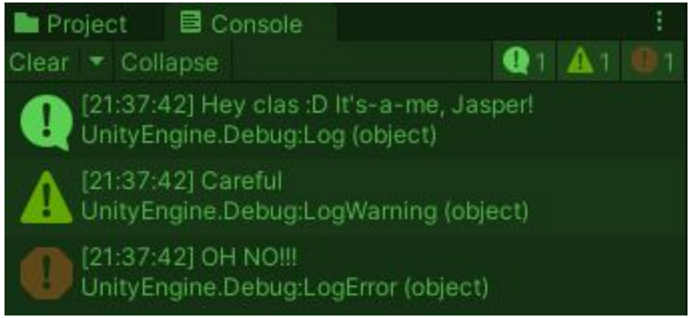
- Cancels all invoked methods
- Can also take a method name as an argument to specify what invoked method to cancel.

Learn more!

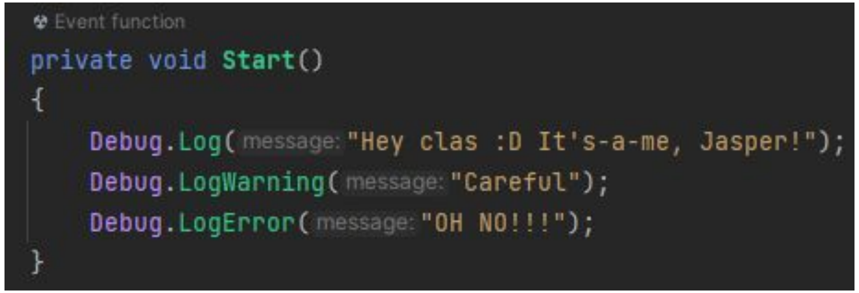
Logging

Use the Debug class to log to the console

Remember, you can double-click to go to the invocation in code

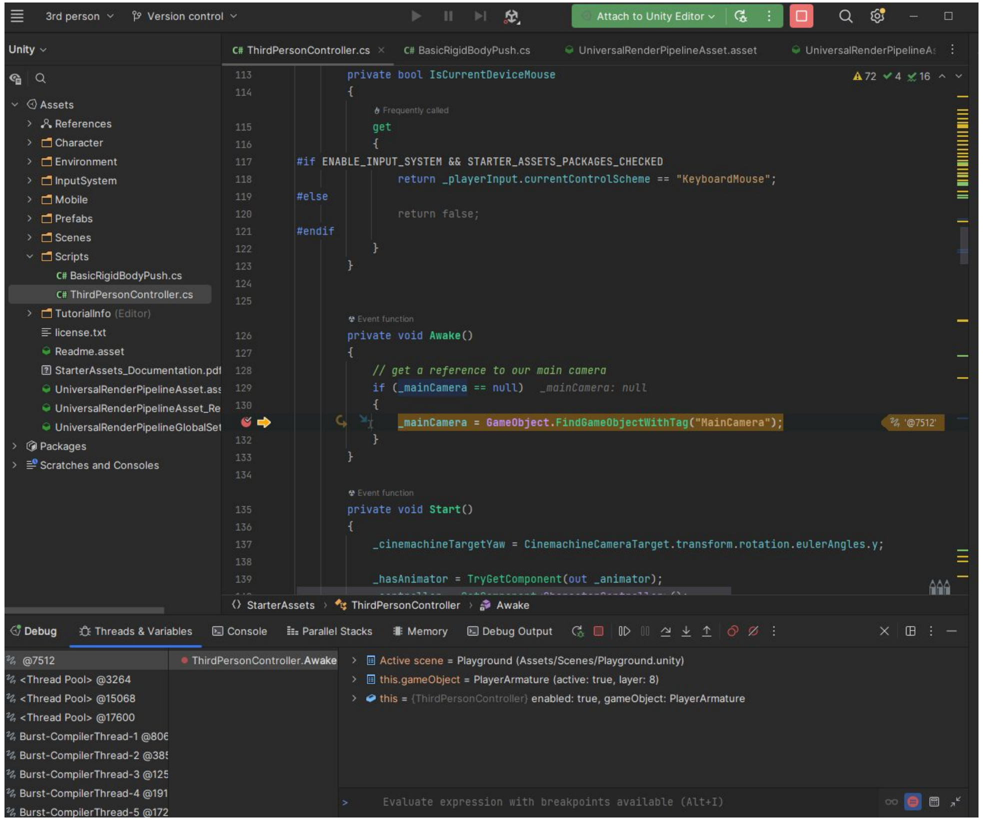


```
[21:37:42] Hey clas :D It's-a-me, Jasper!
UnityEngine.Debug:Log (object)
[21:37:42] Careful
UnityEngine.Debug:LogWarning (object)
[21:37:42] OH NO!!!
UnityEngine.Debug:LogError (object)
```



```
Event function
private void Start()
{
    Debug.Log( message: "Hey clas :D It's-a-me, Jasper!");
    Debug.LogWarning( message: "Careful");
    Debug.LogError( message: "OH NO!!!");
}
```

Debugging



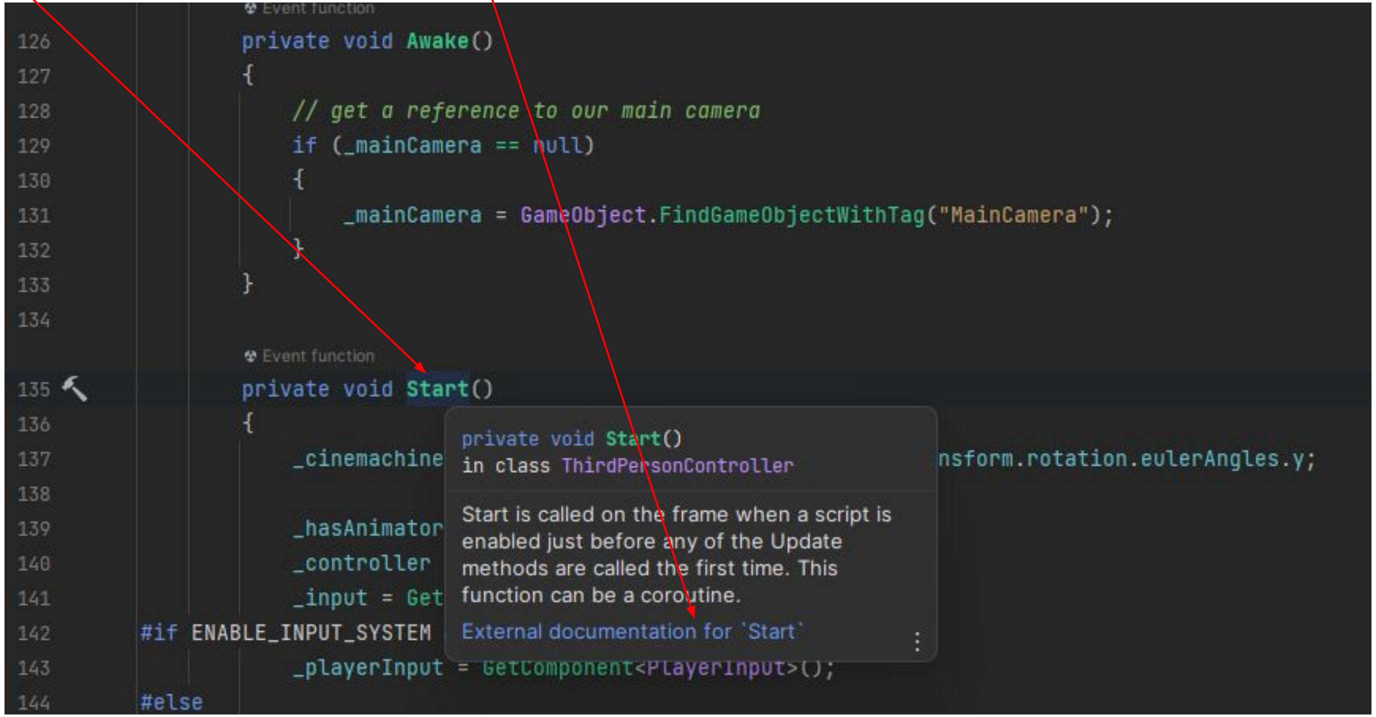
```
        private bool IsCurrentDeviceMouse
        {
            Frequently called
            get
            {
#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED
                return _playerInput.currentControlScheme == "KeyboardMouse";
#else
                return false;
#endif
            }
        }


        Event function
        private void Awake()
        {
            // get a reference to our main camera
            if (_mainCamera == null)  _mainCamera: null
            {
                _mainCamera = GameObject.FindGameObjectWithTag("MainCamera");   '@7512'
            }
        }

        Event function
        private void Start()
        {
            _cinemachineTargetYaw = CinemachineCameraTarget.transform.rotation.eulerAngles.y;

            _hasAnimator = TryGetComponent(out _animator);
```

```
@7512                          ThirdPersonController.Awake
<Thread Pool> @3264
<Thread Pool> @15068
<Thread Pool> @17600
Burst-CompilerThread-1 @806
Burst-CompilerThread-2 @385
Burst-CompilerThread-3 @125
Burst-CompilerThread-4 @191
Burst-CompilerThread-5 @172
```

```
> Active scene = Playground (Assets/Scenes/Playground.unity)
> this.gameObject = PlayerArmature (active: true, layer: 8)
> this = {ThirdPersonController} enabled: true, gameObject: PlayerArmature

>    Evaluate expression with breakpoints available (Alt+I)
```

Accessing Unity Docs

Hover with mouse - quick access to Unity doc



```
        Event function
        private void Awake()
        {
            // get a reference to our main camera
            if (_mainCamera == null)
            {
                _mainCamera = GameObject.FindGameObjectWithTag("MainCamera");
            }
        }

        Event function
        private void Start()
        {
            _cinemachine                                              nsform.rotation.eulerAngles.y;

            _hasAnimator
            _controller
            _input = Get
#if ENABLE_INPUT_SYSTEM
            _playerInput = GetComponent<PlayerInput>();
#else
```

```
private void Start()
in class ThirdPersonController
```

Start is called on the frame when a script is enabled just before any of the Update methods are called the first time. This function can be a coroutine.

External documentation for `Start`

XR Lab Access

Previous GMD Projects

Examples available on itslearning

First Milestone Coming Up!

Before next session…

- Create your GMD project git repository and host it on Github
    - For now, it can just be an empty .git project with a README.md file
- Add your first blog post: Roll-a-ball!

Exercises

The exercises assume that you have gone through the Beginner Scripting video tutorials

Finish early? Extend your Roll-a-ball with more features!

its learning