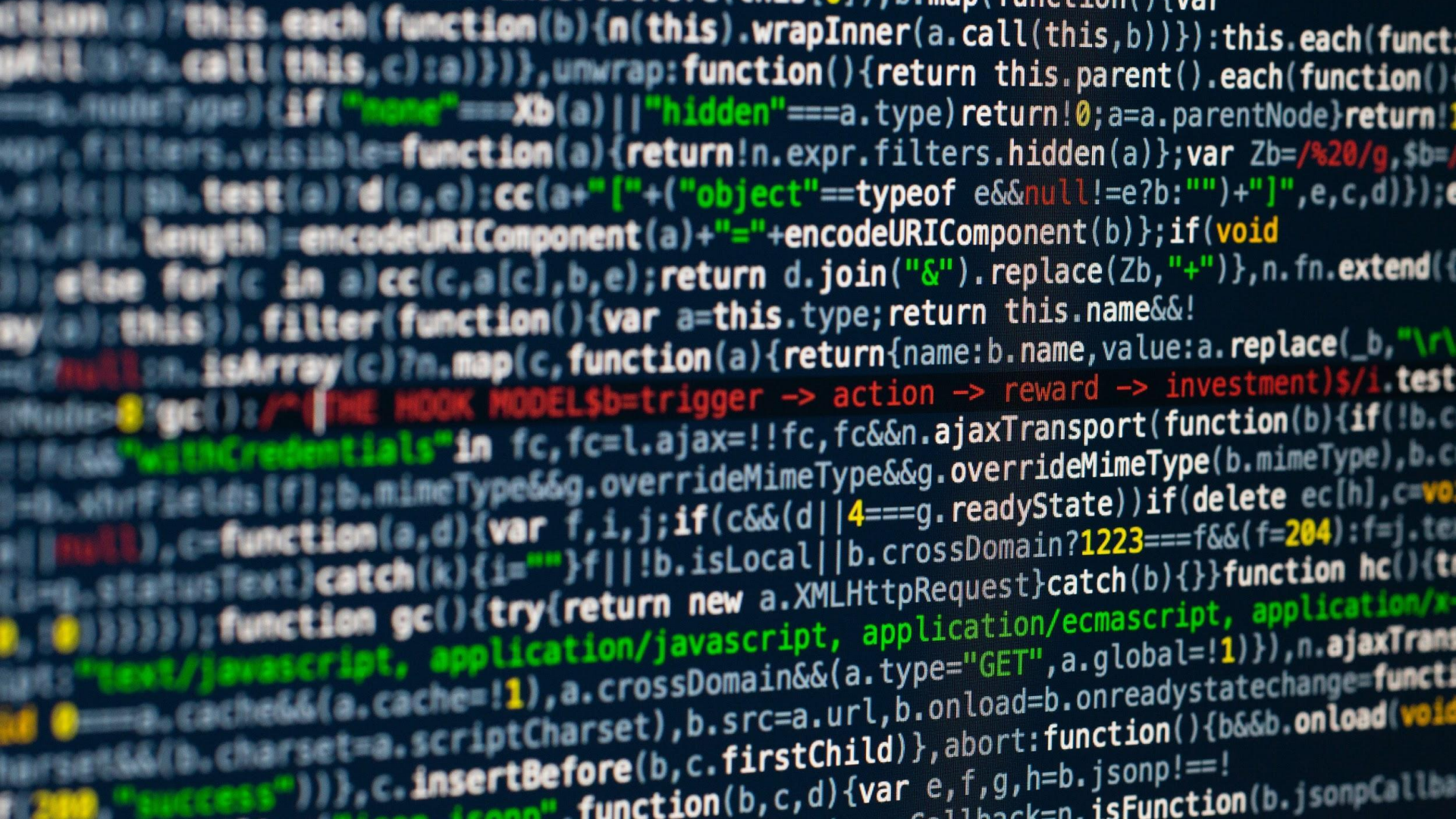
CIS367 - Computer Graphics

Unity Scripting

Erik Fredericks - frederer@gvsu.edu

Based on material from: Unity 2018 Game Development in 24 Hours // https://unity3d.com/learning-c-sharp-in-unity-for-beginners


THE HOOK MODEL$b=trigger -> action -> reward -> investment)$/i.test
function gc(){try{return new a.XMLHttpRequest}catch(b){}}function hc(){t
"text/javascript, application/javascript, application/ecmascript, application/x-
insertBefore(b,c.firstChild)},abort:function(){b&&b.onload(void
function(b,c,d){var e,f,g,h=b.jsonp!==!

Scripts

- Define behaviors
- Attach to Unity objects
- Enable interactivity

- etc.

Our steps:

1) Create a script
2) Attach to one (or more) game object(s)
3) Set script attribute properties

Prereq

Ensure that Preferences ➜ External Tools is set to open a code editor of your choice

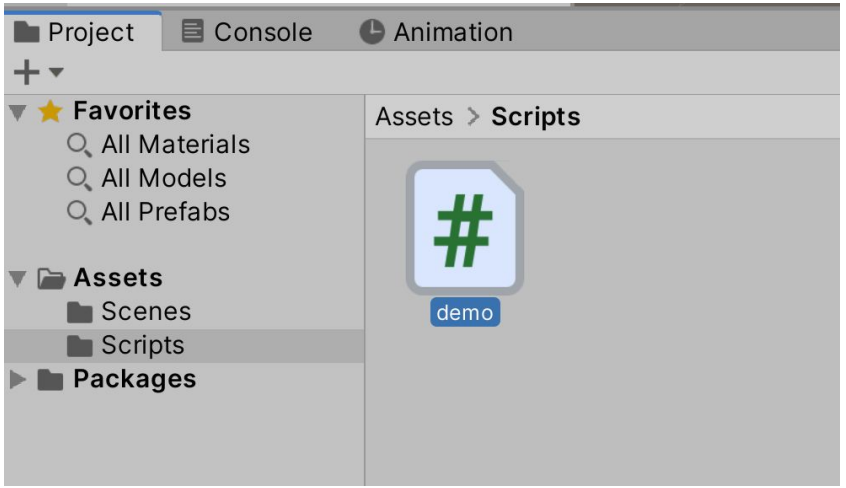
- Use an IDE of your choice

Create a script

(Generally a good idea to have a Scripts folder in your project)

➔ Right-click folder, C# Script

Double-click to open the C#/Mono IDE



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

public class demo : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        print("Hello there!");
    }

    // Update is called once per frame
    void Update()
    {

    }
}
```


NIKE SOCCER
LIFETIME

Saving (your script) will compile for Unity!

(you don't necessarily need to click compile/run in VS/Mono)

Don't forget to save your Scene as well!

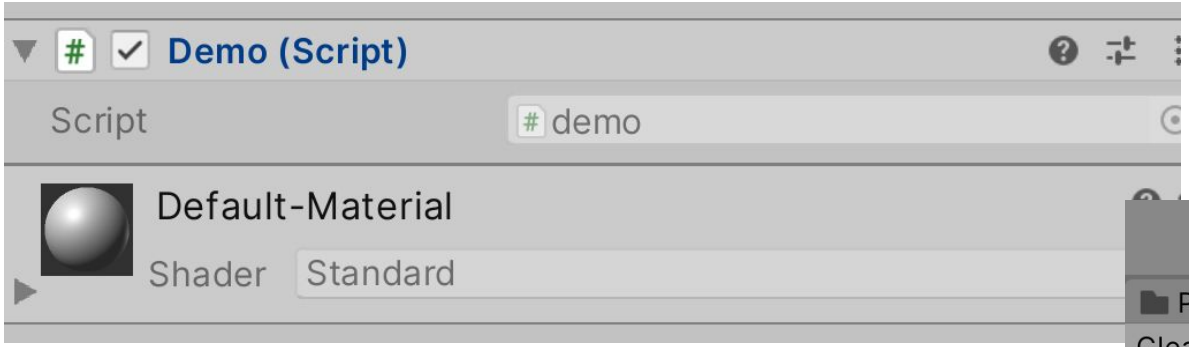
- Save often!!

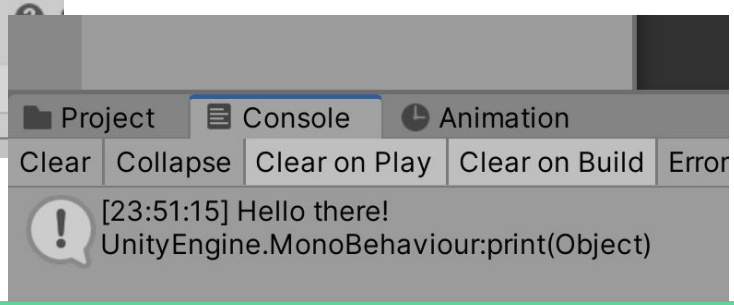
Click Play

...nothing happens?

Need to attach it to something!

- Drag it onto an object
  - Ideally, the one it is controlling/interacting with





...a script

Using

- Define necessary packages/collections to include

Class Declaration

- Define script class (very Java-like)

Contents

- Start
  - Runs upon Scene start
- Update
  - Runs as fast as possible (or however you constrain your program)

Types/scope

Standard C# types

- int, float, string, etc.

Scope

- Variables scoped to block they are defined in

public/private

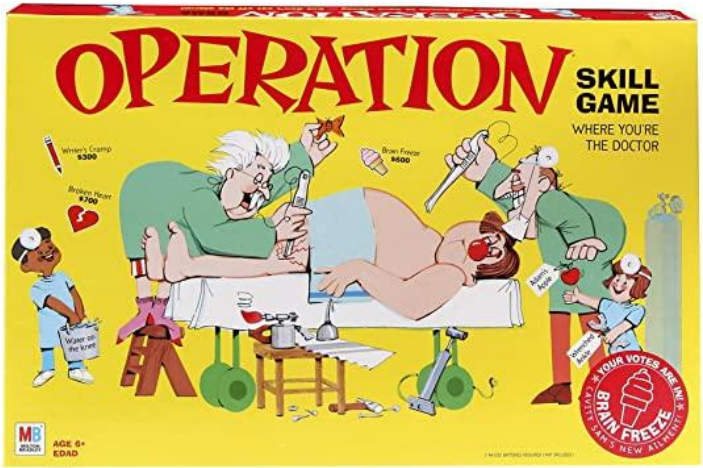
- Public variables visible to Unity editor!

```
public class demo : MonoBehaviour
{
    public int health = 100;
    ...
```

Other operations, equalities, etc.

Fairly similar to what you already know

- If you're coming from C/C++/Java, you basically know all the operations
- At this point, we assume you understand if statements, loops, etc.



Unity-specific functions

`Awake()`
- Called when GameObject is instantiated/active
  - Still called if component is not 'enabled!'
- Good for initializing variables

`Start()`
- Instantiated and **enabled**

`Update()`
- Called once per frame
- Animations, AI, other constant updates

Unity-specific functions

`FixedUpdate()`

- Physics-related updates
- Tutorial example: https://learn.unity.com/tutorial/update-and-fixedupdate

`LateUpdate()`

- Called at **end of frame**
- Unity will:
  - Find all gameobject updates
  - Then call `LateUpdates`

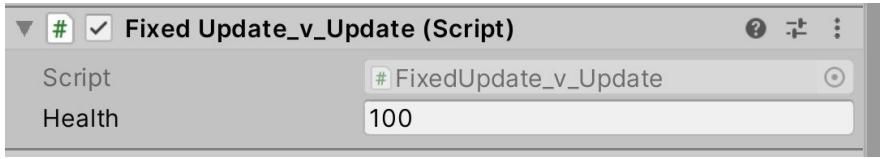
- *Let's say you want to move a character in your game. And then he's bumped into by another character and ends up in a different position. If we move the camera at the same time as the character, there would be a jiggle, and the camera wouldn't be where it needs to be. So, basically, it's a second loop that comes in very handy.*
  - *Unity site --* https://unity3d.com/learning-c-sharp-in-unity-for-beginners

Classes

Honestly I'm not going to go into C# classes too much, as (again) you should know this from the Java world

But...couple of caveats:

Classes

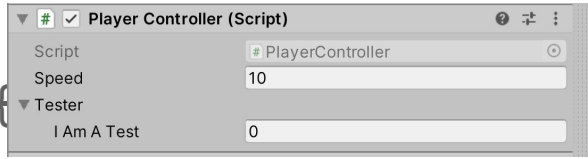


Class name must match filename:

```
public class FixedUpdate_v_Update : MonoBehaviour {
   …
}
```

~~If you want a custom class, it must be serialized for Unity to pick up on it~~

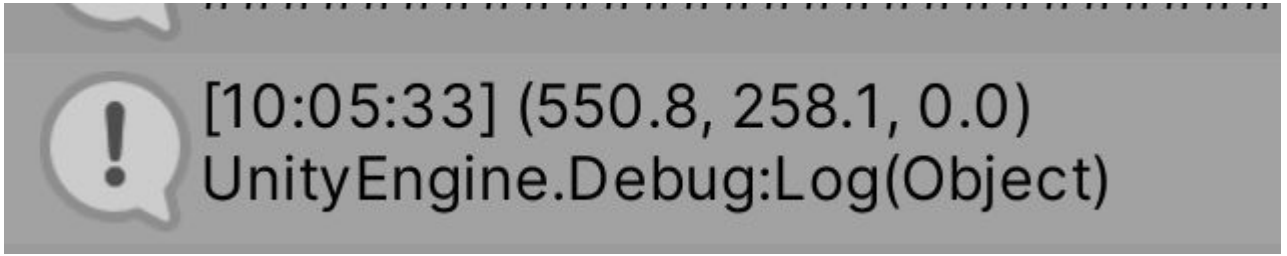
~~[System.Serializable]~~
~~public class MyNewClassThatDoesntMatchFilename {~~
~~   …~~
~~}~~



https://stackoverflow.com/questions/13362336/unity-serializable-class-custom-inspector

Input!

```
public void Update()
{
  if (Input.GetButtonDown("Fire1"))
  {
    Debug.Log(Input.mousePosition);
  }
}
```



```
void Update()
{
  if (Input.GetKeyDown(KeyCode.Space))
  {
    Debug.Log("Space key was pressed.");
  }

  if (Input.GetKeyUp(KeyCode.Space))
  {
    Debug.Log("Space key was released.");
  }
}
```

What if I want a repeating action?

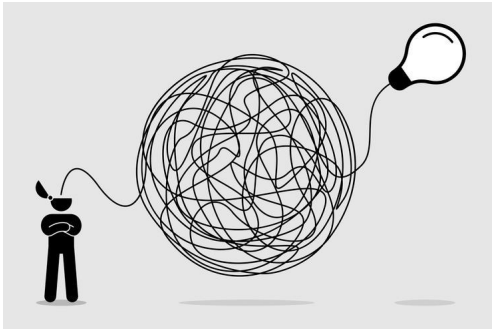
Like firin a lazer, pew pew

Just use `Input.GetKey`

Touchpad?

For instance, if you export to WebGL and somebody is using a phone?

- Could make a virtual D-pad
- Follow to picked point
- etc.



A bit more complicated, I'll put a link here for you to explore:

https://learn.unity.com/tutorial/touch-input-for-mobile-scripting#5cf1fef9edbc2a4cf8c28759

https://docs.unity3d.com/ScriptReference/Input.html


and
breathe

Framerate

```
// Make the game run as fast as possible
Application.targetFrameRate = 300;

// Lock to 60fps
Application.targetFrameRate = 60;
```

-1 sets framerate to application's default

- No guarantee a framerate will be achieved!!!

https://docs.unity3d.com/ScriptReference/Application-targetFrameRate.html

Sound

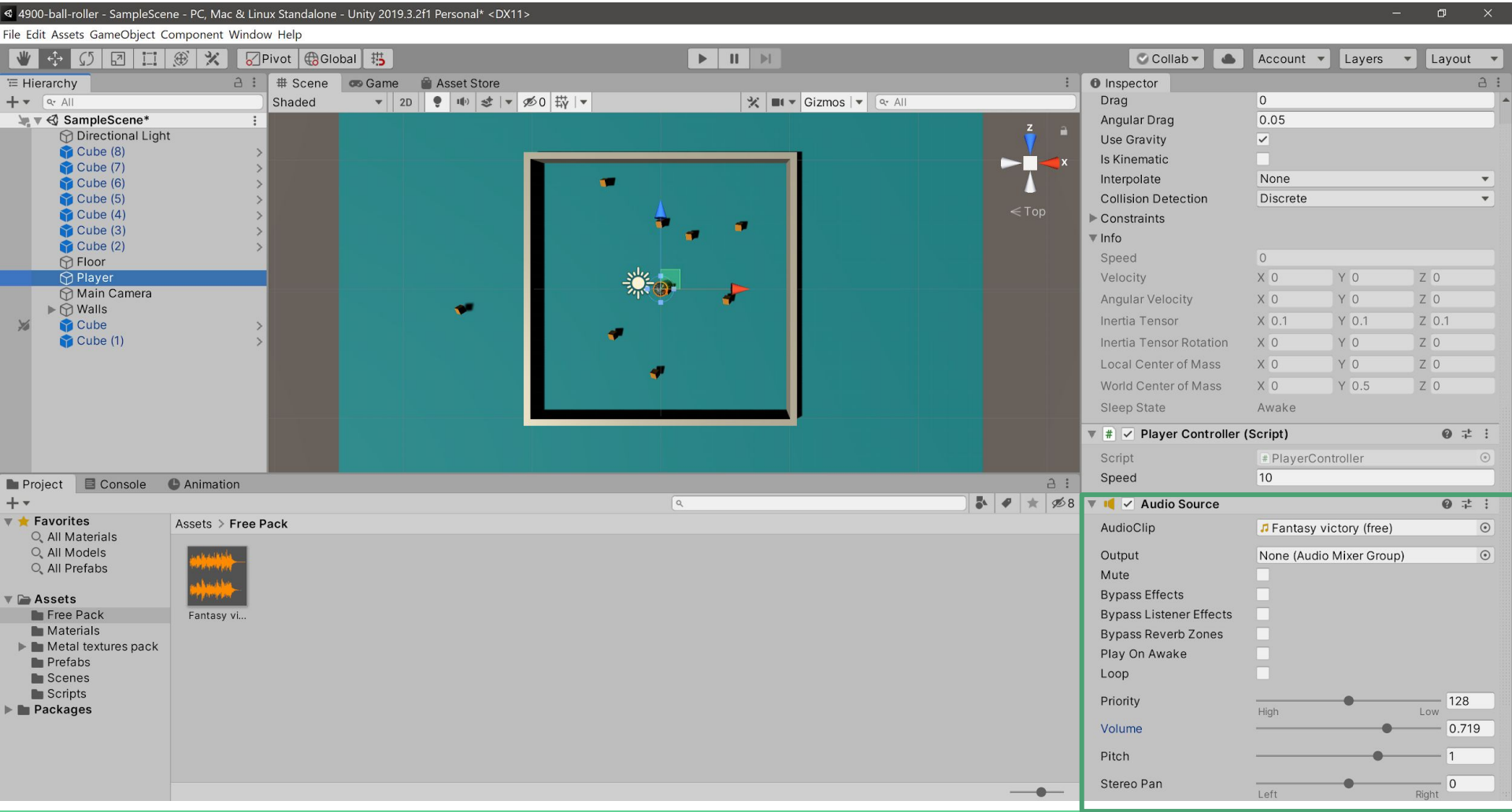
Basic sound is fairly straightforward

Attach an Audio Component to the object that your script is based on

- Otherwise, you'll get a missing component error

Import an audio clip into your project

- Place it into a proper folder location, say, Audio or Sounds


4900-ball-roller - SampleScene - PC, Mac & Linux Standalone - Unity 2019.3.2f1 Personal* <DX11>
File Edit Assets GameObject Component Window Help
Pivot
Global
Collab
Account
Layers
Layout
Hierarchy
SampleScene*
Directional Light
Cube (8)
Cube (7)
Cube (6)
Cube (5)
Cube (4)
Cube (3)
Cube (2)
Floor
Player
Main Camera
Walls
Cube
Cube (1)
Scene
Game
Asset Store
Shaded
2D
Gizmos
Top
Inspector
Drag 0
Angular Drag 0.05
Use Gravity
Is Kinematic
Interpolate None
Collision Detection Discrete
Constraints
Info
Speed 0
Velocity X 0 Y 0 Z 0
Angular Velocity X 0 Y 0 Z 0
Inertia Tensor X 0.1 Y 0.1 Z 0.1
Inertia Tensor Rotation X 0 Y 0 Z 0
Local Center of Mass X 0 Y 0 Z 0
World Center of Mass X 0 Y 0.5 Z 0
Sleep State Awake
Player Controller (Script)
Script PlayerController
Speed 10
Audio Source
AudioClip Fantasy victory (free)
Output None (Audio Mixer Group)
Mute
Bypass Effects
Bypass Listener Effects
Bypass Reverb Zones
Play On Awake
Loop
Priority 128
High
Low
Volume 0.719
Pitch 1
Stereo Pan 0
Left
Right
Project
Console
Animation
Favorites
All Materials
All Models
All Prefabs
Assets
Free Pack
Materials
Metal textures pack
Prefabs
Scenes
Scripts
Packages
Assets > Free Pack
Fantasy vi...

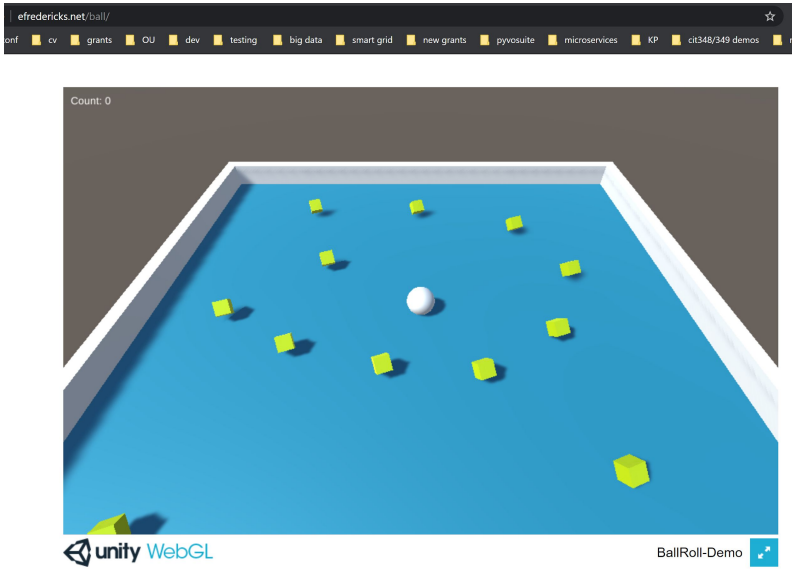
```
private AudioSource audioData;

void Start() {
  …
  audioData = GetComponent<AudioSource>();
  …
}

void Update() {
  if (Input.GetKeyUp(KeyCode.P))
    audioData.Play(0);
}
```

https://docs.unity3d.com/ScriptReference/AudioSource.Play.html


CAN I HAVE YOUR BALL?

Roll-a-ball!

Based on this: https://learn.unity.com/tutorial/introduction-to-roll-a-ball?projectId=5c51479fedbc2a001fd5bb9f&signup=true



http://efredericks.net/ball if you want to see it 'working'

Tasks necessary

Create environment
- Floor, walls, etc.

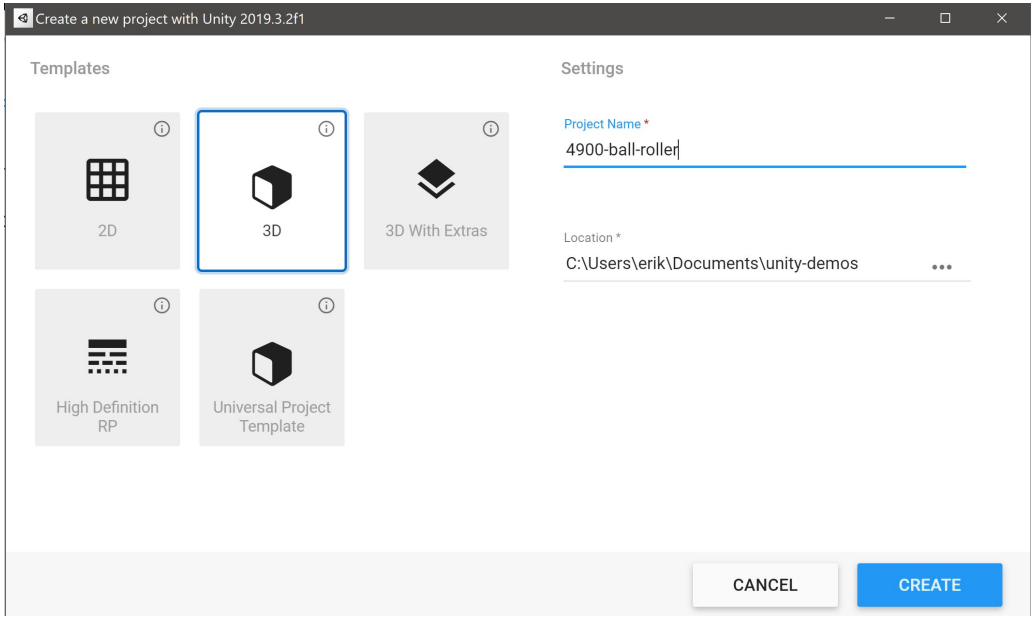
Create the player object
- Sphere
- Control movement

Create objects to pick up
- Cubes

Track and display score

First, create a new project

Must be 3D!

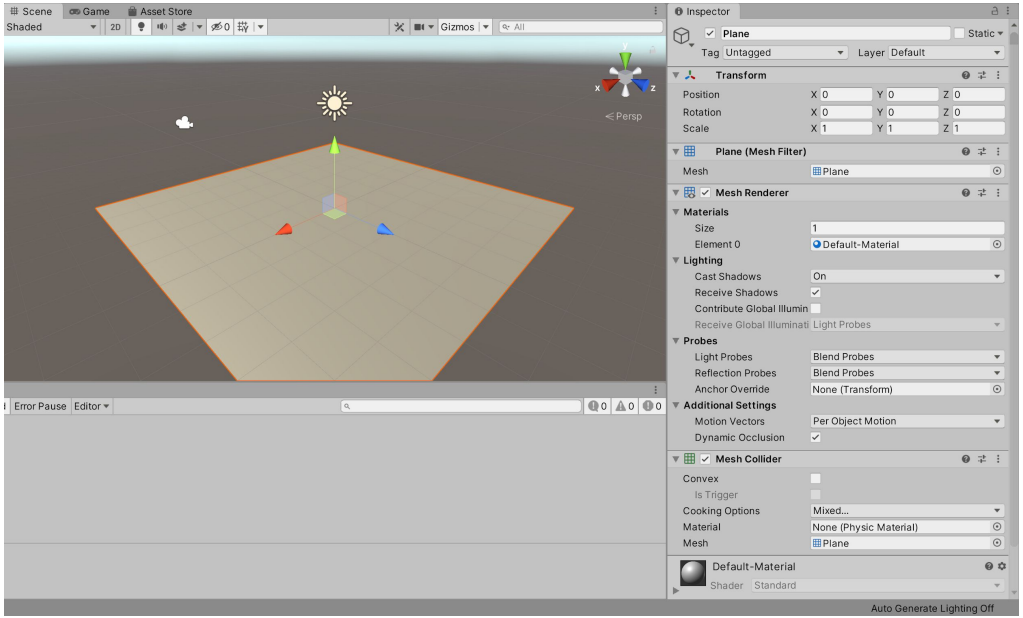


Add the floor

GameObject ➔ 3D Object ➔ Plane

Keep in mind, planes have no volume or physical properties by default!

- Scaling in Y doesn't do anything
- A mesh collider is applied by default though!
- *Add a cube and apply a RigidBody component to the cube to demonstrate collision!*

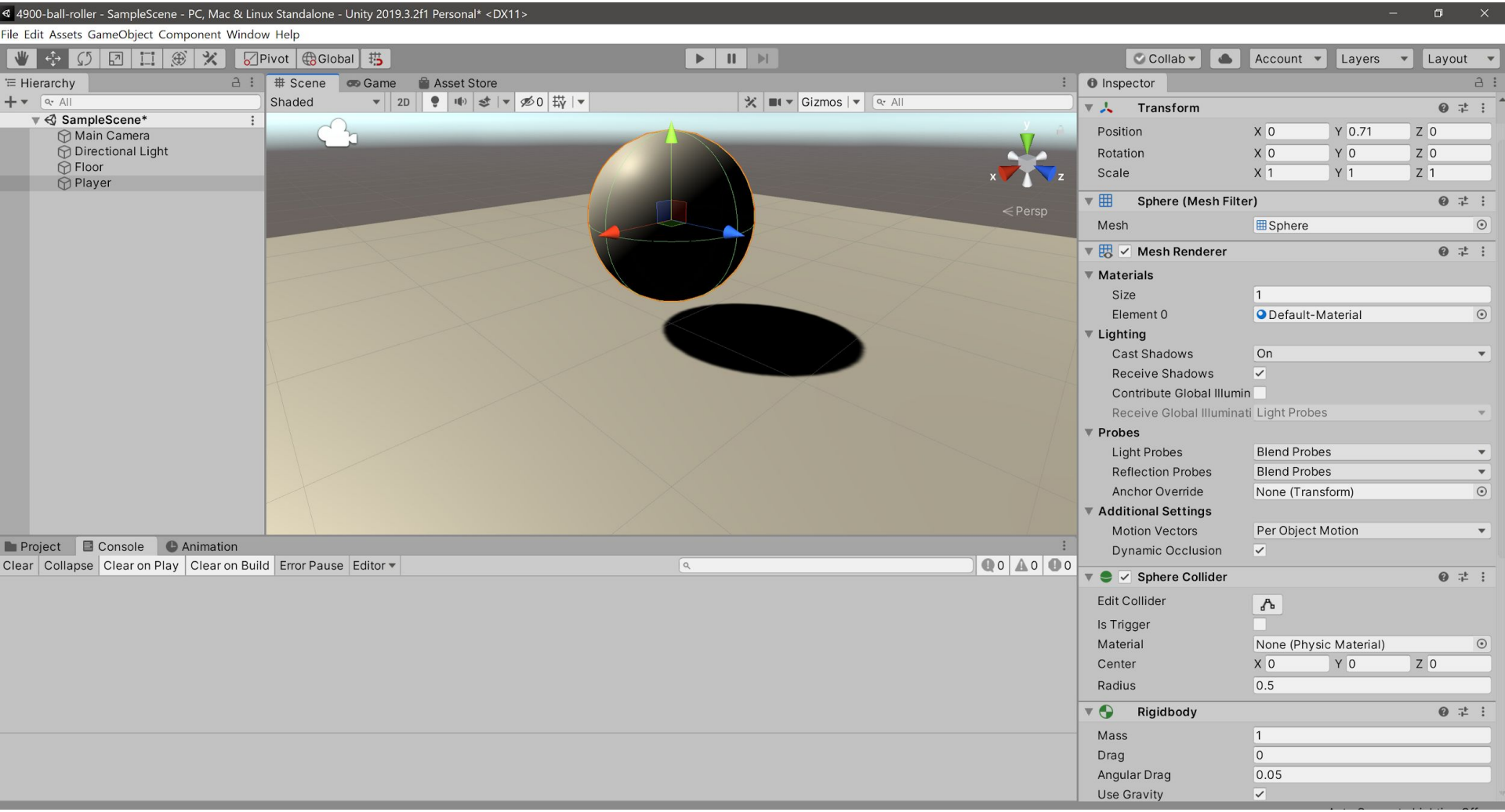


Add the "player"

We're an anthropomorphic, yet faceless, ball

Add the sphere (GameObject ➔ 3D Objects ➔ Sphere) and apply a RigidBody component to it

- Take the time to start renaming your objects in the Hierarchy!
- Put it on top of the plane with a transform
    - Up in Y by 0.5


4900-ball-roller - SampleScene - PC, Mac & Linux Standalone - Unity 2019.3.2f1 Personal* <DX11>
File Edit Assets GameObject Component Window Help
Pivot
Global
Collab
Account
Layers
Layout
Hierarchy
SampleScene*
Main Camera
Directional Light
Floor
Player
Scene
Game
Asset Store
Shaded
2D
Gizmos
Persp
Inspector
Transform
Position X 0 Y 0.71 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Sphere (Mesh Filter)
Mesh Sphere
Mesh Renderer
Materials
Size 1
Element 0 Default-Material
Lighting
Cast Shadows On
Receive Shadows
Contribute Global Illumin
Receive Global Illuminati Light Probes
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings
Motion Vectors Per Object Motion
Dynamic Occlusion
Sphere Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.5
Rigidbody
Mass 1
Drag 0
Angular Drag 0.05
Use Gravity
Project
Console
Animation
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor

Add some color

New folder ➔ Materials

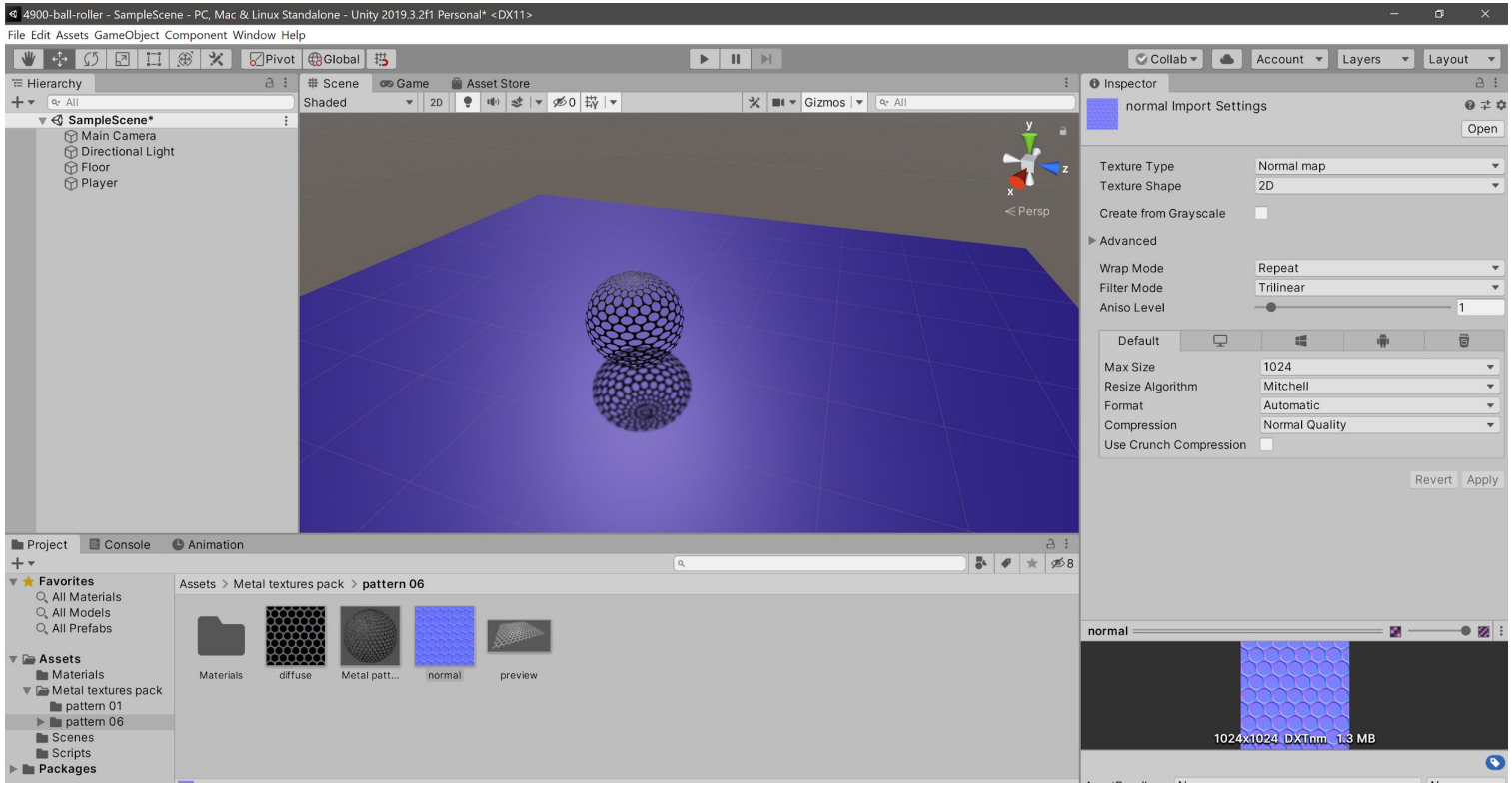
- Create ➔ Material
- Change to blue
- Apply to the plane

And rotate the light

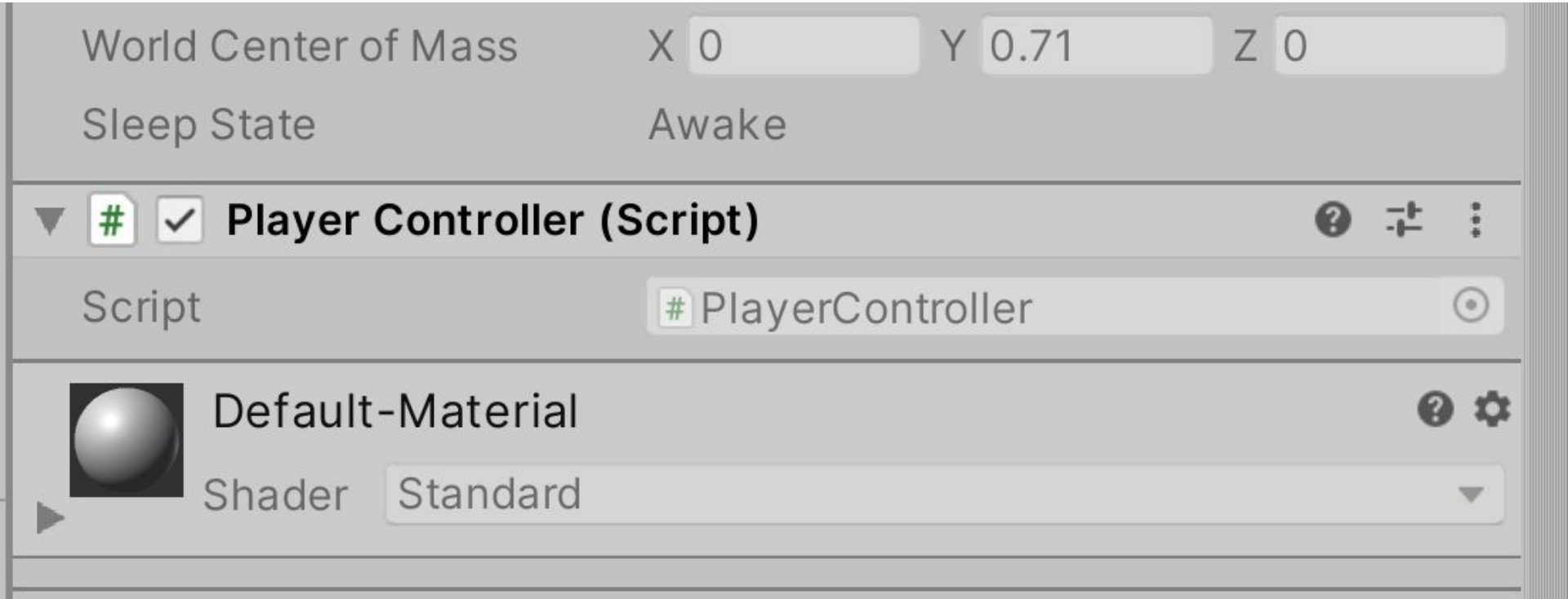
- Change Y rotation to 60

Add some texture to your ball if you want in a similar fashion!

The state of things so far



Add the control script for the player



Apply force to object

1) Create reference to rigidbody

```
private Rigidbody rb;

// Start is called before the first frame update
void Start()
{
   rb = GetComponent<Rigidbody>();
}
```

```
void FixedUpdate()
{
  // apply forces to rigidbody from inputs
  float moveHorizontal = Input.GetAxis("Horizontal");
  float moveVertical = Input.GetAxis("Vertical");

  Vector3 movement = new Vector3(moveHorizontal, 0.0f,
moveVertical);
  rb.AddForce(movement);
}
```

Add a modifier

```
public float speed;  // now an editable property
```

Multiply `movement` by `speed` in `AddForce`

- Set to 100
- Set to 10


Canon
EOS 5D

First...

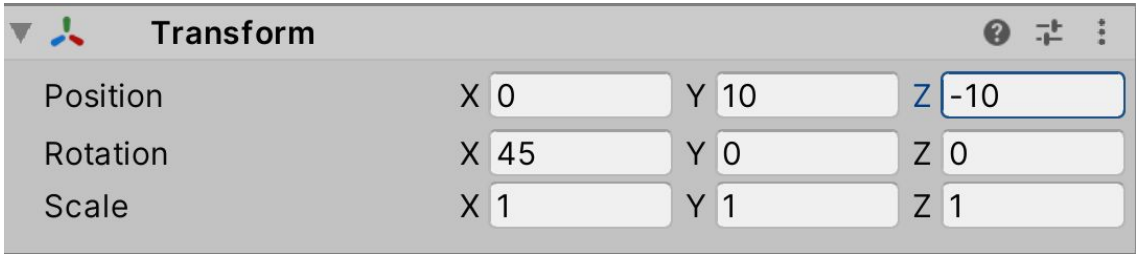
Position camera better:

- Move up by 10
- Rotate by 45deg

Child camera to ball

Test out!

- Scene view and running
- Ut oh!



Well...

Detach camera from player

Add script to camera and...

```
public GameObject player;
private Vector3 offset;


// Start is called before the first frame update
void Start()
{
    offset = transform.position - player.transform.position;
}

// Update is called once per frame
void LateUpdate()  // better than Update -- guaranteed to run after all Update
                   // items processed
{
    transform.position = player.transform.position + offset;
}
```

And then drag Player object into Camera's Player attribute slot

Try it out! We have a moving camera!

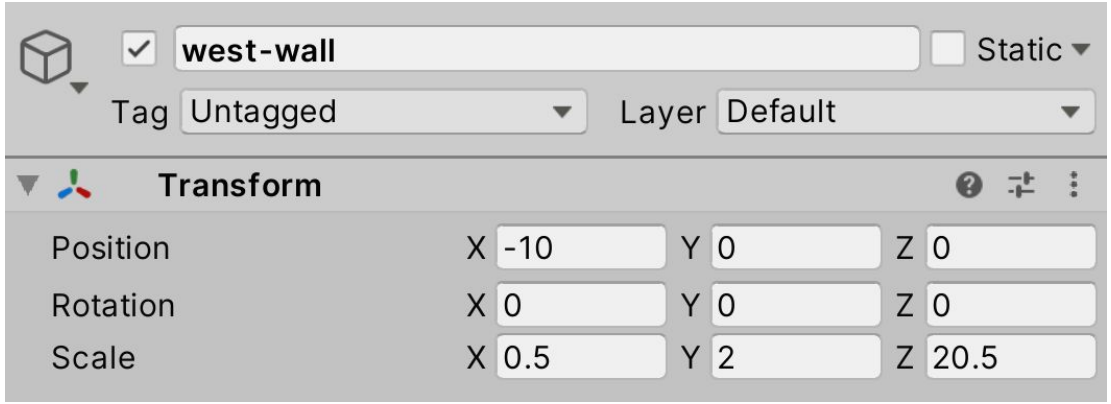
Walled gardens

New gameobject ➔ walls

- Organizer for our walls
- **Reset to origin --- very important!**

New cube

- West wall
- Reset
- Parent to Walls GameObject

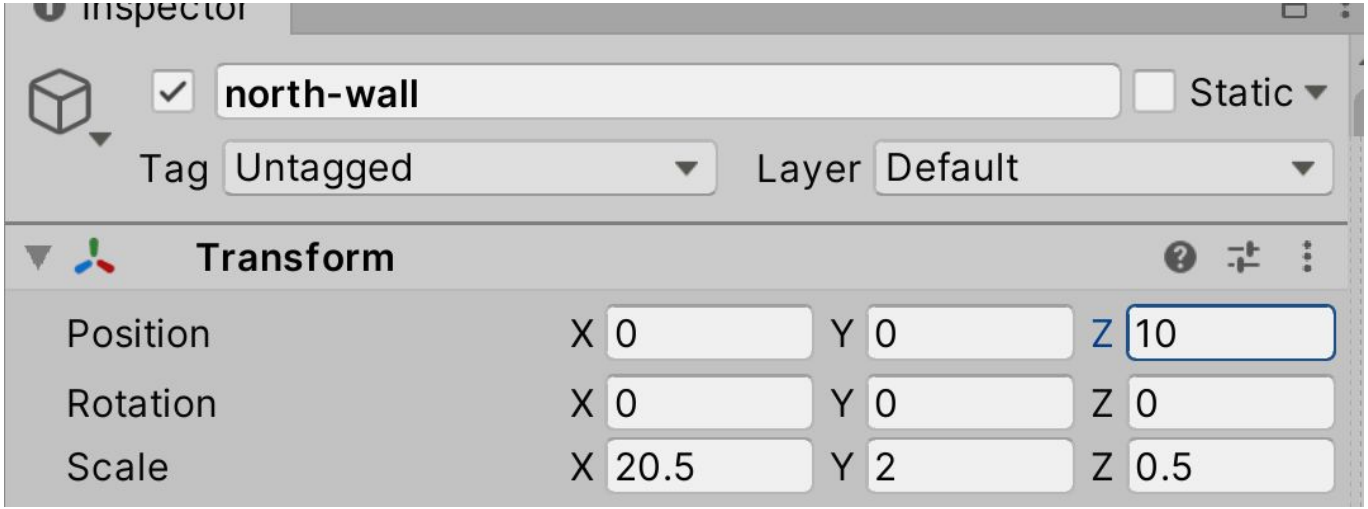


4 walls!

Duplicate west wall, remove negative from x value

- Name it east wall

Do the same for the other 2 walls, appropriately placing them and scaling



Collectibles!

Create cube ➜ rename pickup

- Reset to origin
- Focus on pickup

Uncheck player in Inspector to deactivate

- Lift up by 0.5 units
- Scale by 0.5 in all axes
- Rotate by 45 in all axes

Then create script for rotation/interaction

- No need for `Start`, normal `Update` is fine as well

Rotate away!

Rotate transform:

in `Update`:

```
// multipler smooths!
transform.Rotate(new Vector3(15, 30, 45) * Time.deltaTime);
```

Prefab

Prefab ➜ "template" of game object/family!

Turn cube into a prefab

- All instances will also change if you make a change!

New folder ➜ Prefabs (call Pickups)

- Move cube into Pickups
- Set to top-down view (click Y on gizmo)

Now place our collectibles

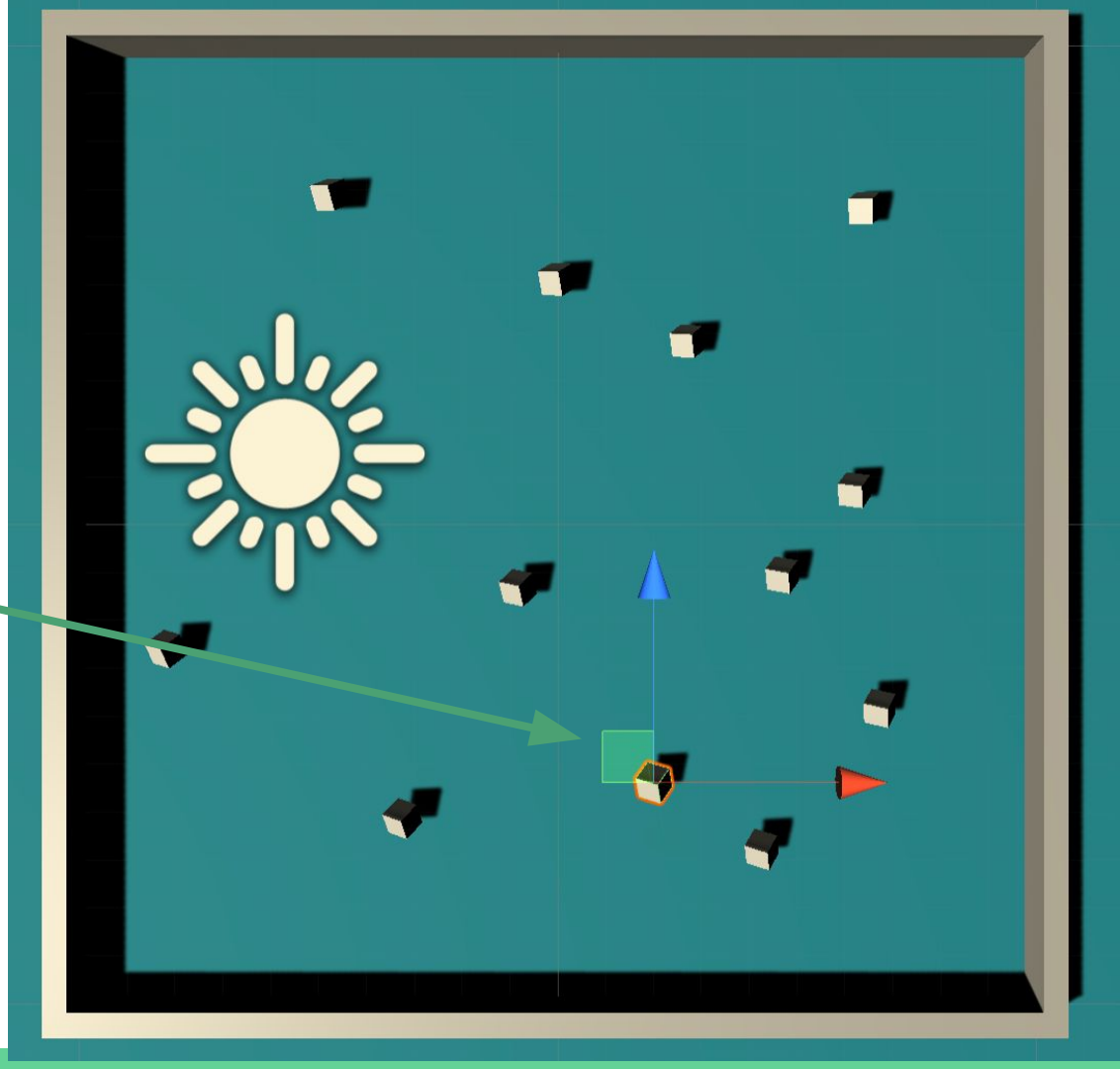
Select the cube (not parent) and duplicate (Ctrl+D)

Click the X-Z plane and move it around!

- May need to move to Global mode if not in it already

Change color of cubes!

- Add material (copy/paste background mat) to the **Prefab object**



And enable pickup (collision detection)

Look at sphere collider on player

- Click little question mark to go to reference
- `OnTriggerEnter` is what we're looking for
  - Called when something touches our collider

in PlayerController:

```
void OnTriggerEnter(Collider other) {
  // deactivate, not destroy collided object!

}
```

Tags

Can apply tags to objects to make them easily referencable

- E.g., "coin" "destructable"

```
if (other.gameObject.CompareTag("coin"))
  // pickup a coin and bleep a noise
```

```
void OnTriggerEnter(Collider other) {
  // deactivate, not destroy collided object!
  if (other.gameObject.CompareTag("Pick Up")) {
    other.gameObject.SetActive(false);
  }
}
```

Now set Pick Up on our Prefab!

Still bouncing off cubes!

Make collider a trigger instead of a physical object

Select Prefab and look at box collider

- Select **Is Trigger**

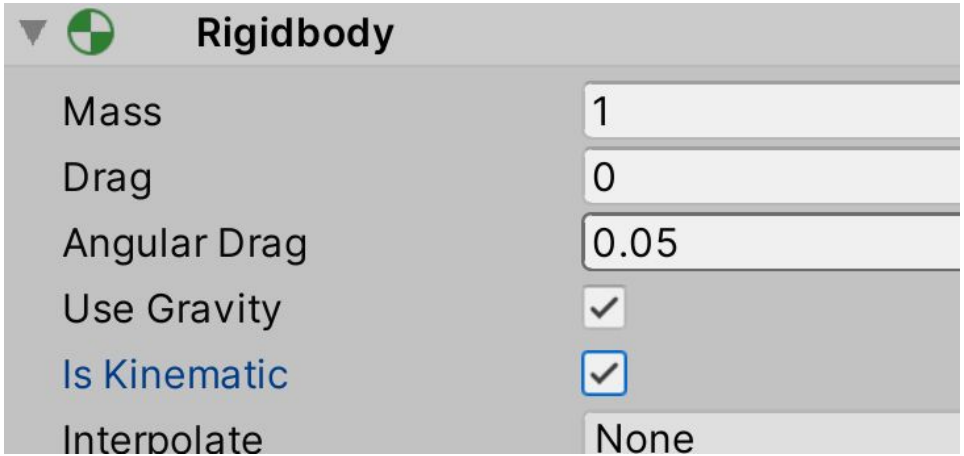
However, Unity calculates all physics calculation for static colliders in cache

- Rotation of static collider causes cache to be continually calculated
- Tell Unity that rotators are dynamic!
- Rigidbody + collider ➔ dynamic
- No body ➔ expected to be static
- Play!
    - Gravity!

Look at Rigidbody

Select "Is Kinematic" on Prefab

- Kinematic rigidbody does not react to physics forces!




1031740

Score

Add a private int to the Player Controller

```
private int count;
```

Increment each time we pickup an object

- Set equal to 0 in `Start`
- Increment in `onTriggerEnter` function when a collision occurs

Display text

Use UI toolset (TextMeshPro – UI toolset is now Legacy)

New Text -> TextMeshPro Element in Heirarchy

- Canvas and EventSystem auto-added and required
- UI elements must be child of Canvas!
    - (Look in Game view for how it 'really' looks)

Make text white to make it more visible

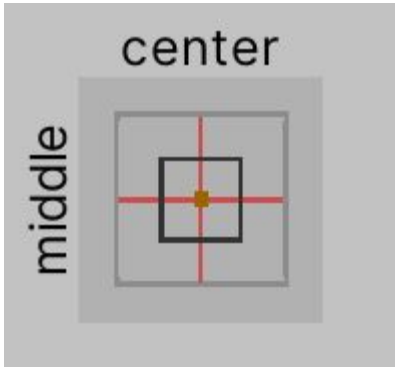
- Add placeholder text
- Note that canvas is different than other objects (Rect)

Move to upper left

Open Anchors and Presets

Hold Shift+Alt and then pick the upper left corner

Give it some breathing room with PosX and PosY values (10,-10, 0)



Tie in with our PC script

Make sure we include the TMPro namespace

Add reference to TextMeshPro

- `public TMP_Text countText;`

In `Start` (after initializing count):

- `countText.text = "Score: " + count.ToString();`

In `OnTriggerEnter`:

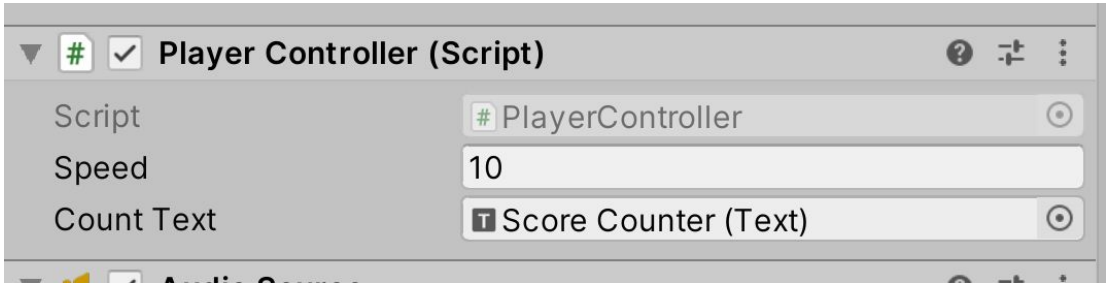
- `countText.text = "Score: " + count.ToString();`

Or create a function

```
void setCountText() {
  countText.text = "Score: " + count.ToString();
}
```

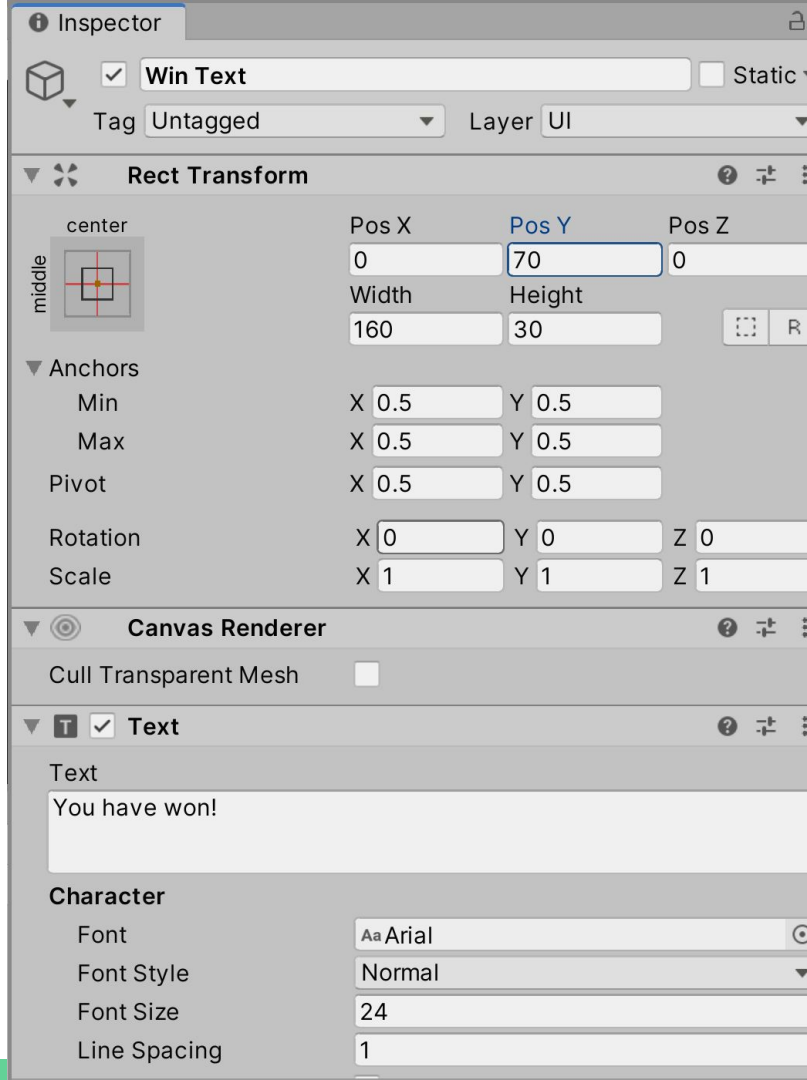
And replace other lines

Then, drag instance of Text element onto the appropriate PC script location



Win condition

- Add another text element, move up a bit, whiten, and embiggen
- Add another public text variable and initialize it to an empty string in the script
- Associate with the PC script again and then let's check for our win state



Win condition

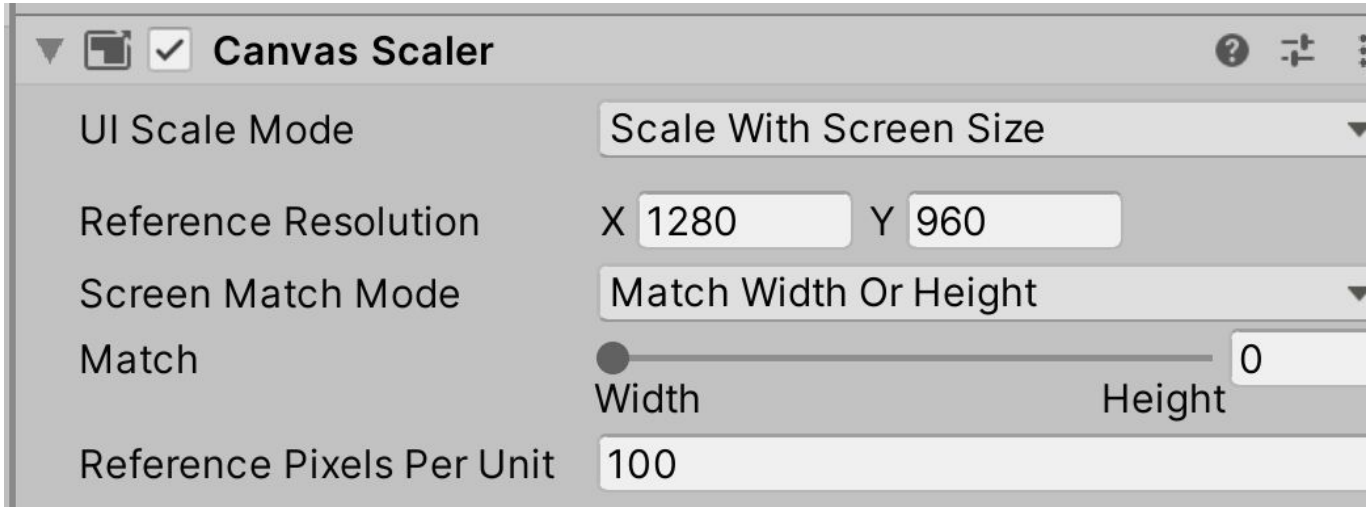
At our collision checker, check to see if we've gotten all the objects (simple method)

- With some PIZZAZZ

```
if (count >= 9) {
  winText.text = "!!YOU WIN!!";
  audioData.Play(0);  // the pizzazz
}
```

After build, text is tiny!

Text is set to pixel perfect ➜ change scale factor



http://efredericks.net/4900-ball-roller/

Dungeon Generator

https://docs.unity3d.com/Manual/InstantiatingPrefabs.html

http://efredericks.net/dungeonball/