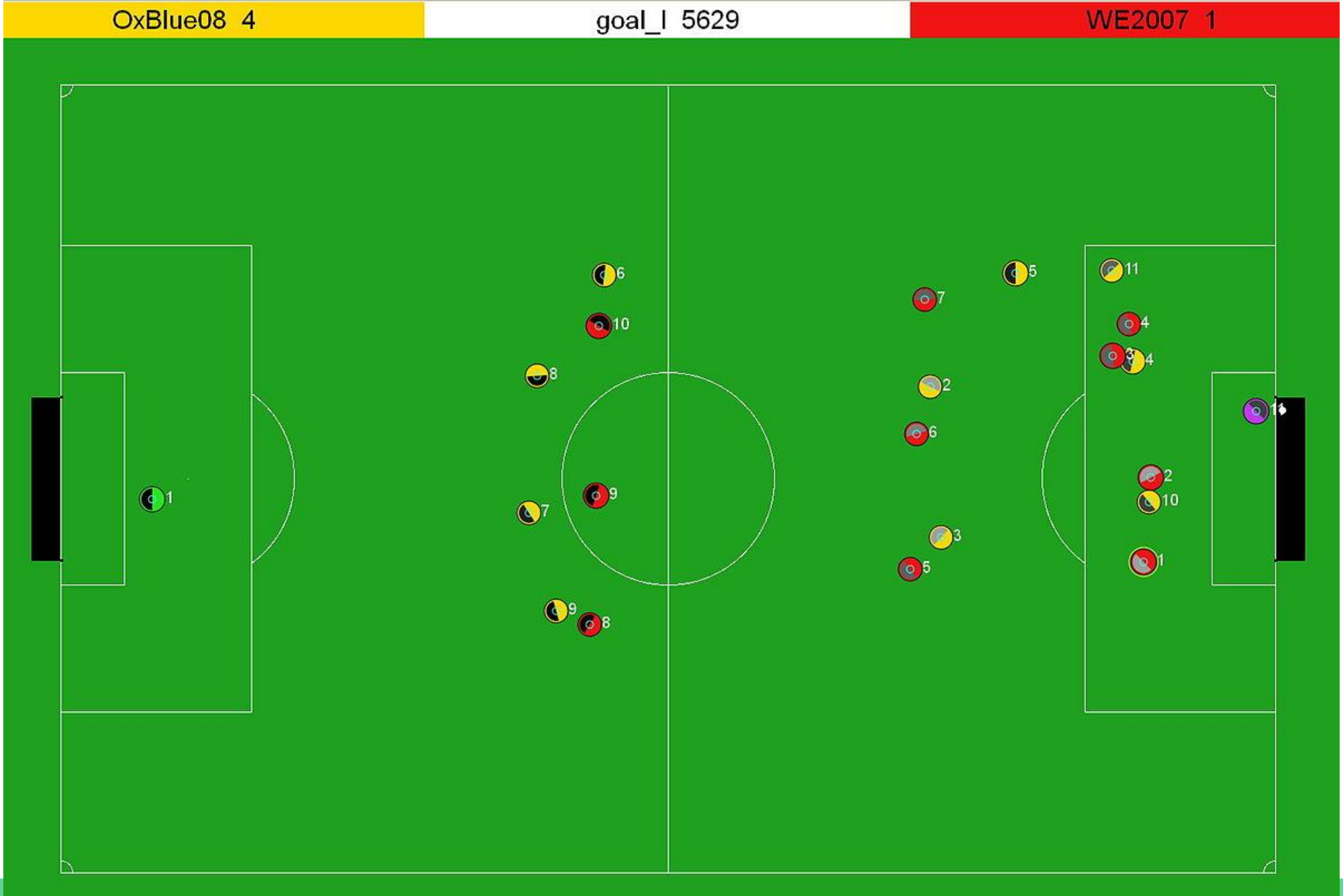
CIS367 - Computer Graphics
2D

Erik Fredericks - frederer@gvsu.edu

Based on material from: Unity tutorials and the gamedevelopment blog (https://www.gamedevelopment.blog/full-unity-2d-game-tutorial-2019/ - most Unity screenshots from here too!)

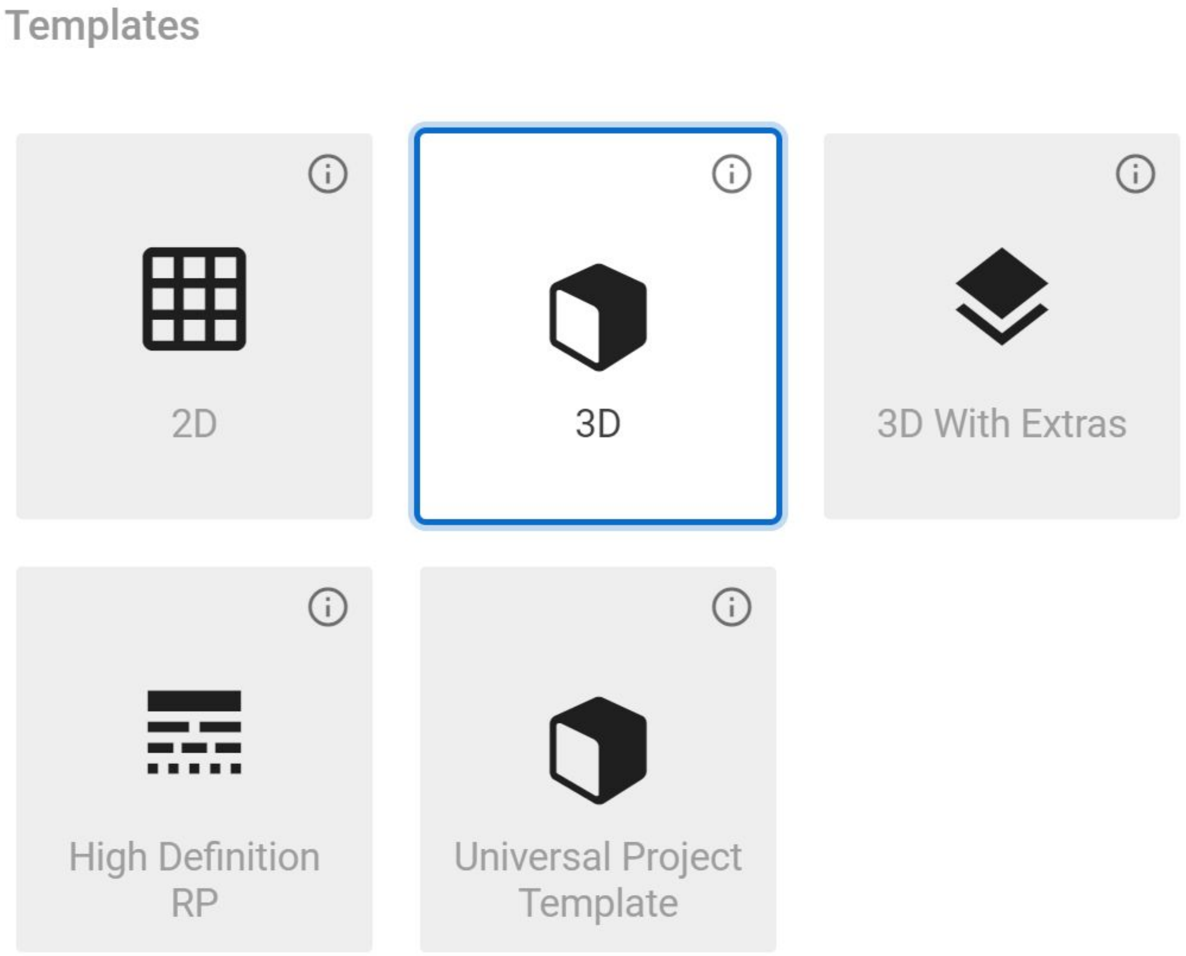

OxBlue08 4 | goal_l 5629 | WE2007 1

So...

Everything we've done so far in terms of 3D!

What about 2D!??!?!

- Or the other options as well?



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

2D:

- Instantiates wrt a 2D scene (sprites, ortho camera, etc)

3D:

- Instantiates wrt a 3D scene (models, rendering pipeline, etc)

3D with Extras:

- 3D + a post-processing stack

High Definition RP:

- Supports shader model 5.0 (DX11 and up)

Universal Project Template:

- Universal pipeline where support/optimization a concern

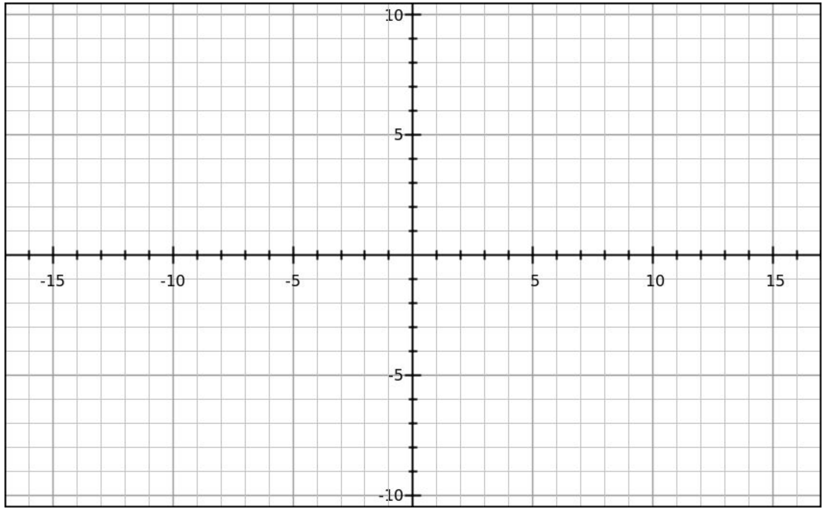
So, 2D games are a thing with Unity as well!

Wizard of Legend is a good example!

https://www.youtube.com/watch?v=DUxJAQ7KMjk (not the devs)

What are the differences here?

- Camera view
- Sprites instead of models
    - Tilemaps instead of terrains
    - etc.



Let's make a simple game then!

%
https://www.gamedevelopment.blog/unity-2d-game-tutorial-2019-player-movement/

Not going to do *all* of it though (12 part series)

- But we'll get started!
- We'll leave the rest as an *exercise to the student*
    - MUAHAHAHAHHAAHHAHAHAHA
        - hehe
            - heh

2D layout looks pretty similar!

Minus the 3D gizmo!

Let's create a player!

- New empty GameObject
  - On empty object, add a 2D Sprite
  - Select an asset (knob works) on the Inspector

- Add some control code (to PlayerObject, not sprite!)

Looks pretty similar so far!

Class vars:

```
public float
moveSpeed = 5f;

public float
hitPoints = 100f;
```

Update:

```
// check if user has pressed some input keys
if (Input.GetAxisRaw("Horizontal") != 0 ||
Input.GetAxisRaw("Vertical") != 0) {

    // convert user input into world movement
    float horizontalMovement = Input.GetAxisRaw("Horizontal") *
moveSpeed * Time.deltaTime;
    float verticalMovement = Input.GetAxisRaw("Vertical") *
moveSpeed * Time.deltaTime;

     //assign movement to a single vector3
     Vector3 directionOfMovement = new
Vector3(horizontalMovement, verticalMovement, 0);

    // apply movement to player's transform
    gameObject.transform.Translate(directionOfMovement);
}
```

Make it RIGID

Add a Rigidbody 2D to player object

- Turn off gravity otherwise player will fall through (like before)
  - Gravity scale 0

- Set Interpolate to Interpolate to minimize janky motion

- Add a Rigidbody2D reference in script
  - private variable and call to GetComponent in Start!

```
rb = gameObject.GetComponent<Rigidbody2D>();
if (rb == null) {
  Debug.LogError("Player::Start cant find RigidBody2D");
}
```

Add physics instead of raw movement

```
// Remove code in update for movement!!!
void FixedUpdate() {
    // check if user has pressed some input keys
    if (Input.GetAxisRaw("Horizontal") != 0 || Input.GetAxisRaw("Vertical") != 0) {
        // convert user input into world movement
        float horizontalMovement = Input.GetAxisRaw("Horizontal") * moveSpeed;
        float verticalMovement = Input.GetAxisRaw("Vertical") * moveSpeed;

        //assign world movements to a Vector2
        Vector2 directionOfMovement = new Vector2(horizontalMovement,
verticalMovement);

        // apply movement to player's transform
        rb.AddForce(directionOfMovement);
    }
}
```

Background

https://www.gamedevelopment.blog/wp-content/uploads/2019/01/spacebg.jpg

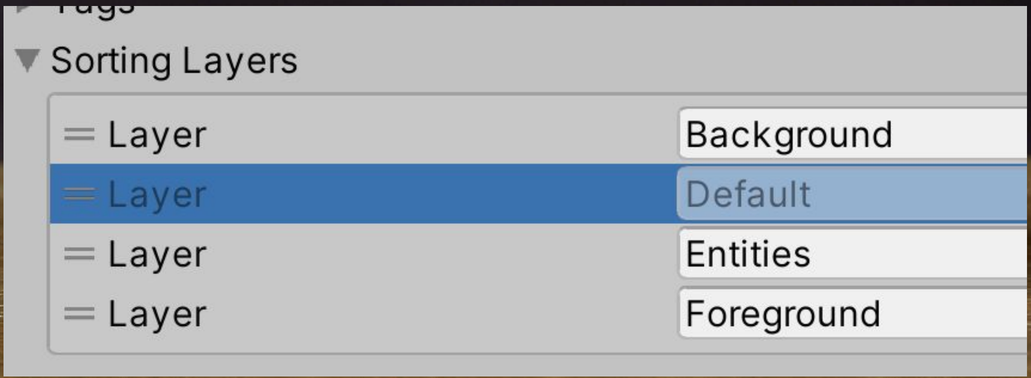
Add image to Assets folder and select

- Mesh type → Full rect
- Create an empty game object (BackgroundObject) → Reset!
- Add sprite to BackgroundObject, add image
  - Set mode to Tiled
    - Set width/height to be 100/100

Layers

On BG **sprite**, add a sorting layer!

- Consider it similar to a z-index in webpages
  - Add background, entities, foreground
  - Set Player to entities layer



Sorting Layers

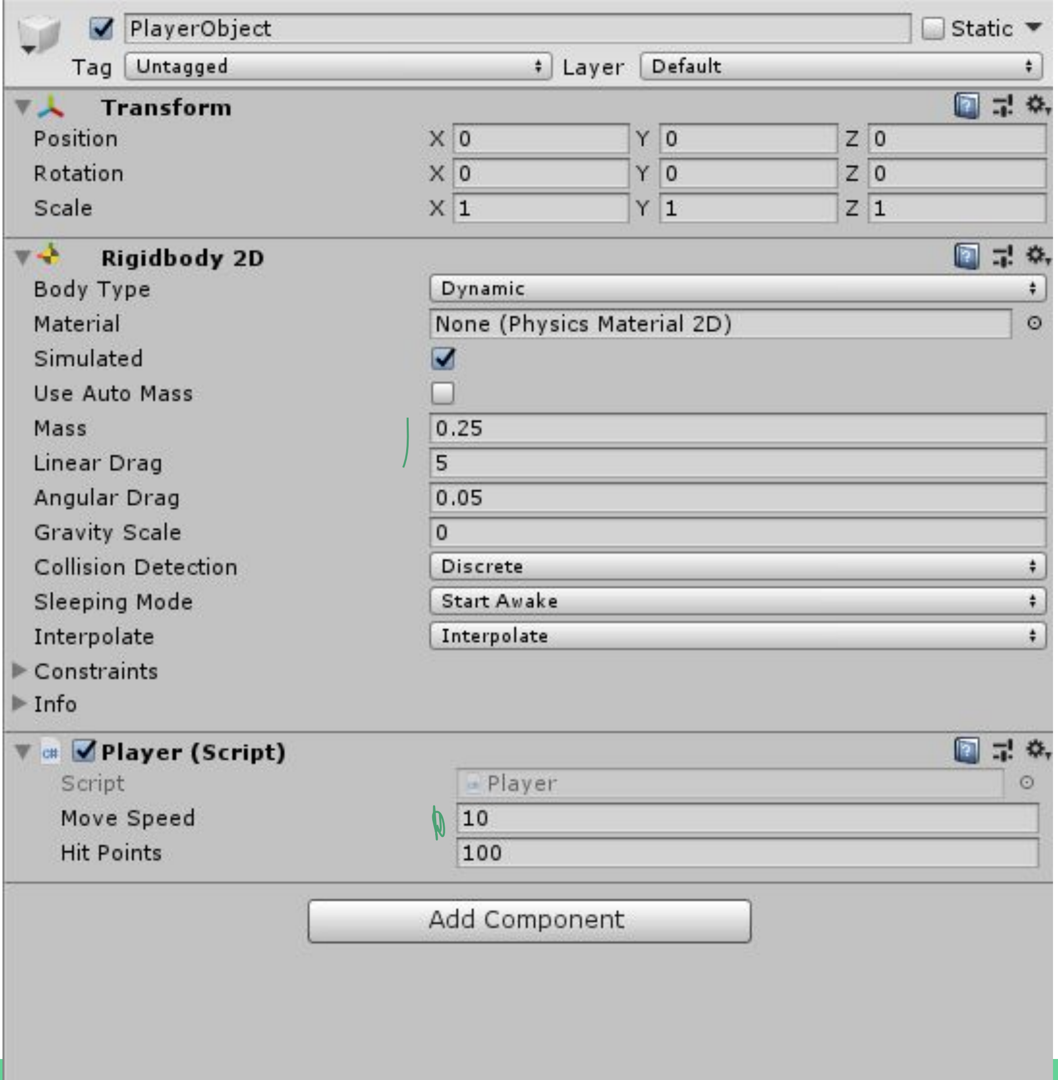
Layer	Background
Layer	Default
Layer	Entities
Layer	Foreground

Camera follow

Again, similar to 3D camera follower:

```
public class FollowPlayer : MonoBehaviour
{
    public GameObject playerObject;  // the player object to follow
    public float lerpSpeed = 0.5f;
    private Vector3 offset;

    void Start() {
        // get the current offset between player and camera positions
        offset = transform.position - playerObject.transform.position;
    }

    void LateUpdate() {
        transform.position = Vector3.Lerp(transform.position, // current camera position
    playerObject.transform.position + offset,   // new position plus our original offset
    lerpSpeed);                                 // the speed of smoothing
    }
}
```

Minor Rigidbody2D adjustments



PlayerObject ☐ Static

Tag	Untagged
Layer	Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Rigidbody 2D

Body Type	Dynamic
Material	None (Physics Material 2D)
Simulated	✓
Use Auto Mass	☐
Mass	0.25
Linear Drag	5
Angular Drag	0.05
Gravity Scale	0
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	Interpolate

Constraints

Info

Player (Script)

Script	Player
Move Speed	10
Hit Points	100

Firing bullets (i.e., managing dynamic content)

Add a new empty game object (BulletObject) and a sprite

- Add a RigidBody2D

```
public int speed = 50;          // The speed our bullet travels
public Vector3 targetVector;    // the direction it travels
public float lifetime = 10f;     // how long it lives before destroying itself
public float damage = 10;       // how much damage this projectile causes

void Start() {
    // find our RigidBody
    Rigidbody2D rb = gameObject.GetComponentInChildren<Rigidbody2D>();
    // add force
    rb.AddForce(targetVector.normalized * speed);
}
```

```
// Update is called once per frame
void Update() {
    // decrease our life timer
    lifetime -= Time.deltaTime;
    if (lifetime <= 0f) {
        // we have ran out of life
        Destroy(gameObject);    // kill me
    }
}
```

Now prefab it

Drag the BulletObject into a Prefabs folder

And let's test it out (in PlayerController):

- Add in a reference to the bullet prefab:
    - `public GameObject bulletPrefab;`

- And a handler (in Update):

```
if (Input.GetKeyDown(KeyCode.Space)) {
    // if the player pressed space (exclude holding key down)
    GameObject go = Instantiate(bulletPrefab, gameObject.transform);
    BulletController bullet = go.GetComponent<BulletController>();
    bullet.targetVector = new Vector3(1,1,0);
}
```

- If things look wonky, make sure the bullet object and sprite are at 0,0,0
    - In Prefab

Now prefab it

Drag the BulletObject into a Prefabs folder

And let's test it out (in PlayerController):

- Add in a refe prefab:
    - `publi etPrefab;`

Change to GetKey for repeating fire!

- And a hand

```
if (Input.GetKeyDown(KeyCode.Space)) {
    // if the player pressed space (exclude holding key down)
    GameObject go = Instantiate(bulletPrefab, gameObject.transform);
    BulletController bullet = go.GetComponent<BulletController>();
    bullet.targetVector = new Vector3(1,1,0);
}
```

- If things look wonky, make sure the bullet object and sprite are at 0,0,0
    - In Prefab

AI?!?!?

What do we need to implement enemy characters?

- Objects
  - 2D? sprite!
  - 3D? model!
  - Rigidbody

- Scripts
  - Autonomous movement…
  - Decision making
  - Health
  - …etc.

Enemy

Create an empty object (EnemyObject)

- Add a sprite
    - Make it a knob
    - Set its color

- Rigidbody2D
    - 0 gravity
    - Interpolate

- **Two** CircleCollider2D components
    - One to represent the physical object
    - One to represent a **trigger**

I made it bigger (X:10,Y:10)

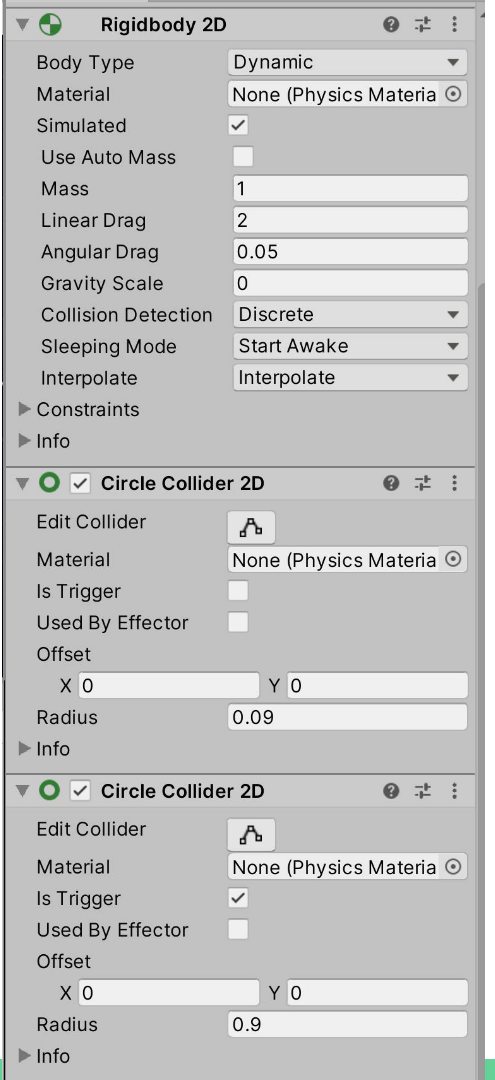
Set the enclosing collider to be bound around the object (r = 0.09)

- Setting this to be a trigger will create an invisible repulsion field

Set a large collider to be the enemy's "view radius" and make it larger (r = 0.9)

- Set "Is Trigger"

Add a ~~Circle~~ appropriate Collider to the Player as well - set the radius to bound the object (again, r = 0.09 and no trigger)



Rigidbody 2D

Body Type	Dynamic
Material	None (Physics Materia
Simulated	✓
Use Auto Mass	
Mass	1
Linear Drag	2
Angular Drag	0.05
Gravity Scale	0
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	Interpolate

Constraints

Info

Circle Collider 2D

Edit Collider	
Material	None (Physics Materia
Is Trigger	
Used By Effector	
Offset	X 0 Y 0
Radius	0.09

Info

Circle Collider 2D

Edit Collider	
Material	None (Physics Materia
Is Trigger	✓
Used By Effector	
Offset	X 0 Y 0
Radius	0.9

Info

Script

```
1)  Check if I have a target
    1.1) If I have, do a follow
    1.2) If not, do a nothing
    1.3) If a target comes in view, say I have a target
    1.4) If not, do nothing
```

Attributes

```
public float speed = 2.0f;    // Follow speed

[HideInInspector]
public bool hasTarget = false;  // do I have a target to move towards

[HideInInspector]
public GameObject target;   // the target i want to get closer to

private Rigidbody2D rb;
```

```
void Awake() {  // called immediately upon creation (start is slightly delayed)
     rb = gameObject.GetComponent<Rigidbody2D>();
}

private void Update() {
  if (hasTarget) {
    //get distance between me and my target
    float distance = Vector3.Distance(transform.position, target.transform.position);
    // am I further than 2 units away
    if (distance > 2) {
      // I am over 2 units away
      follow(target.transform); // do a follow
    }
  }
}
```

```
// if anything starts to collide with me I will run this method
private void OnTriggerEnter2D(Collider2D collision) {
  if (collision.name.Equals("PlayerObject")) {    // is the other object the player
    target = collision.gameObject;      // it is so set him as my target
    hasTarget = true;   // I have a target
  }
}

// if something is no longer coliiding with me I will run this code
private void OnTriggerExit2D(Collider2D collision) {
  if (collision.name.Equals("PlayerObject")) {
    target = null;
    hasTarget = false;
  }
}

private void follow(Transform target) {
  // add force to my rigid body to make me move
  rb.AddForce((target.transform.position - transform.position).normalized * speed);
}
```

"The last bit"

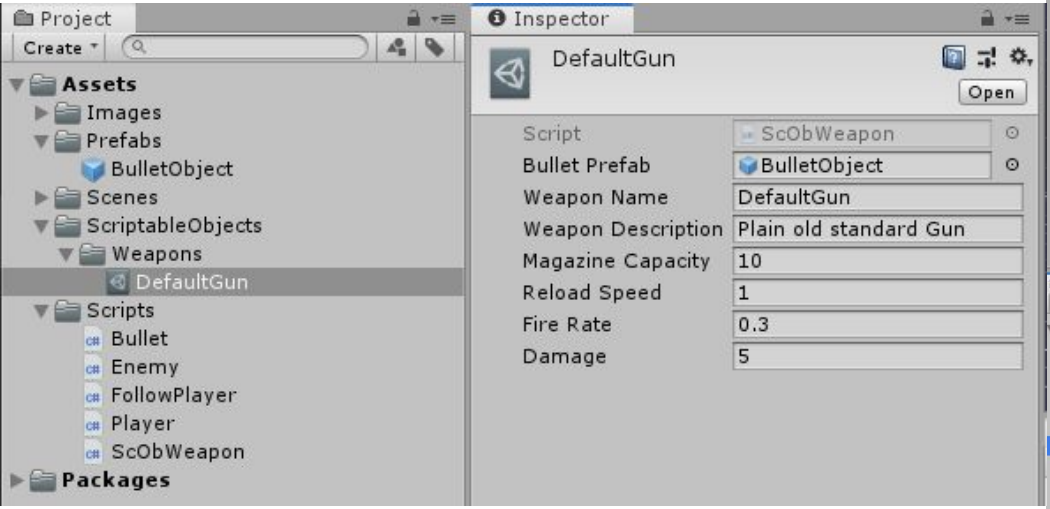
https://www.youtube.com/watch?v=S-XEINagmaU

Scriptable objects, collisions, and management

- The rest (scoring, UI, etc.) feel free to look at the tutorial
  - It isn't that tricky!

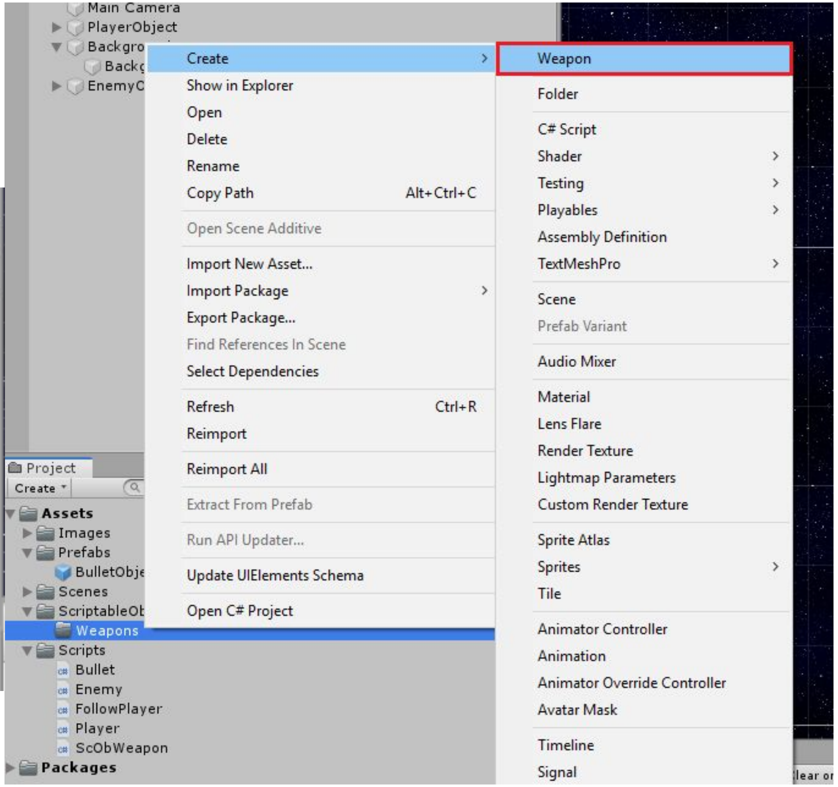
Scriptable objects

Makes the non-programmer life easier!



DefaultGun

Script	ScObWeapon
Bullet Prefab	BulletObject
Weapon Name	DefaultGun
Weapon Description	Plain old standard Gun
Magazine Capacity	10
Reload Speed	1
Fire Rate	0.3
Damage	5



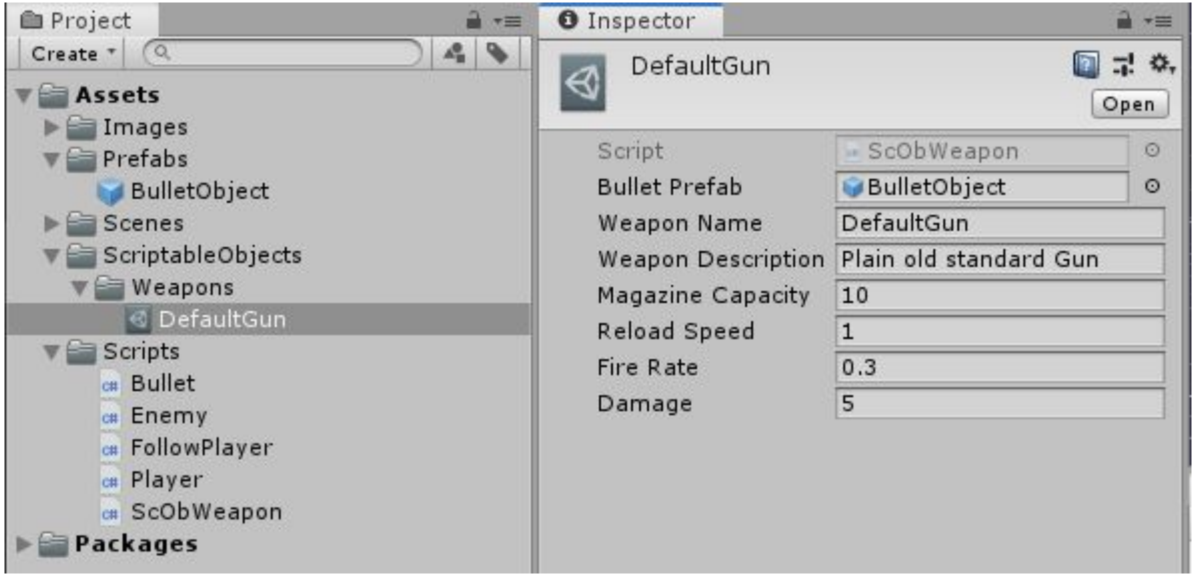
(Right click ➜ Create in Assets)

```
[CreateAssetMenu(fileName = "New Weapon", menuName = "Weapon")]
public class ScObWeapon : ScriptableObject
{
    public GameObject bulletPrefab; // Stores out Bullet Prefab

    public string weaponName;       // weapon name e.g  plasma cannon
    public string weaponDescription;// weapon description e.g "Fires plasma ball"

    public int magazineCapacity;    // amount of bullets per magazine e.g. 5
    public float reloadSpeed;       // time to reload   e.g 2.5f (2.5 seconds)
    public float fireRate;          // bullets shot per second 1f (1 per second)
    public float damage;            // damage per bullet (100)
}
```

Create a 'Weapons' folder to hold our assets

Create a 'Default' gun

- See params below
- Comment out reference to BulletObject in player script now!
    - Weapon will hold ref.



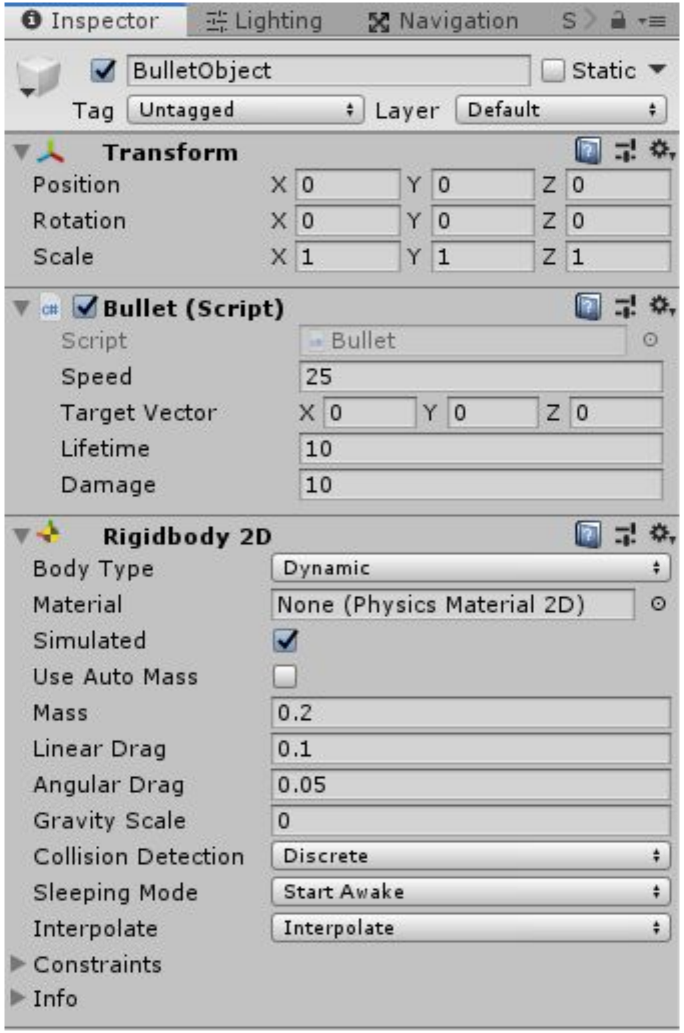
Script	ScObWeapon
Bullet Prefab	BulletObject
Weapon Name	DefaultGun
Weapon Description	Plain old standard Gun
Magazine Capacity	10
Reload Speed	1
Fire Rate	0.3
Damage	5

Change to fire "at" mouse pointer (in PlayerController)

```
public ScObWeapon currentWeapon;
…

void Update() {
  if (Input.GetMouseButton(0)) {
    Vector3 pointMouseVector = Camera.main.ScreenToWorldPoint(Input.mousePosition);
    pointMouseVector.z = 0; // set z to 0, this is 2D
    GameObject go = Instantiate(currentWeapon.bulletPrefab,
                                gameObject.transform.position,
                                Quaternion.identity);
    Bullet bullet = go.GetComponent<Bullet>();
    Vector3 targetVector =  pointMouseVector - gameObject.transform.position;
    bullet.targetVector = targetVector;
  }
}
```

Adjust mass, drag, and speed to make bullets go faster

(we'll leave the remainder of the weapons class to you if you want ... basically it involves checking the reload times, number of fired bullets, etc.)



BulletObject ✓ Static ☐
Tag Untagged Layer Default

Transform

Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Bullet (Script) ✓

Script	Bullet
Speed	25
Target Vector	X 0 Y 0 Z 0
Lifetime	10
Damage	10

Rigidbody 2D

Body Type	Dynamic
Material	None (Physics Material 2D)
Simulated	✓
Use Auto Mass	☐
Mass	0.2
Linear Drag	0.1
Angular Drag	0.05
Gravity Scale	0
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	Interpolate

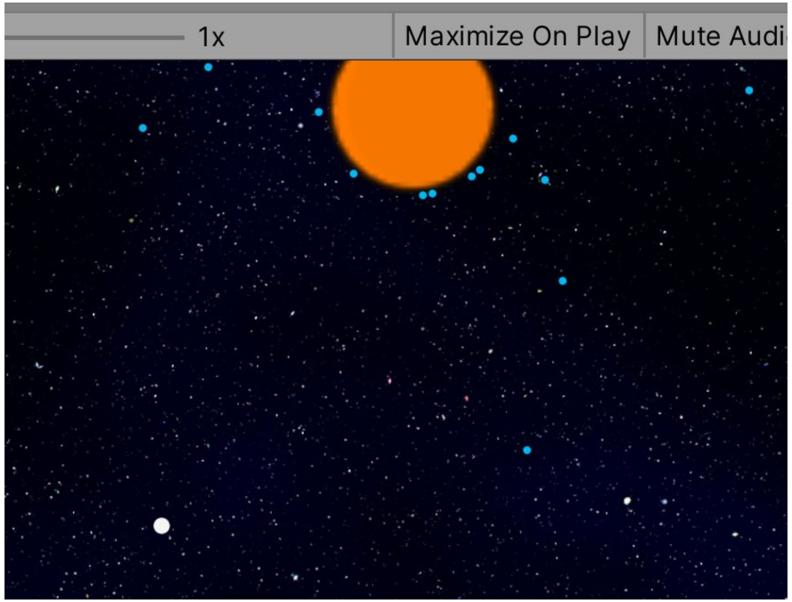
Constraints
Info

Collisions

Add a Circle Collider 2D to the bullet prefab (with a trigger)

- Though it is kind of fun without the trigger :)

What about the rest?



How would we finish this up?

Add collision triggers to have "something" happen when a bullet hits an object

- Decrement health
- "Kill" object
- Ensure that player doesn't hit self (unless you want that!)

Manage the game!

- Set spawn points/rates
- Change bullet types
- etc.

Add a menu system, handle user preferences, audio cues, etc.

Basically, the 'grunt work' in a way

Tilemaps:
https://www.raywenderlich.com/23-introduction-to-the-new-unity-2d-tilemap-system#toc-anchor-002

Collision:
https://www.reddit.com/r/Unity2D/comments/gldaxw/tilemap_colliders_not_working_properly/