

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Section Intro - Argon Assault

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Argon Assault Game Design



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Core Gameplay Overview

999,999

Score:

- Points for killing enemies

Camera:

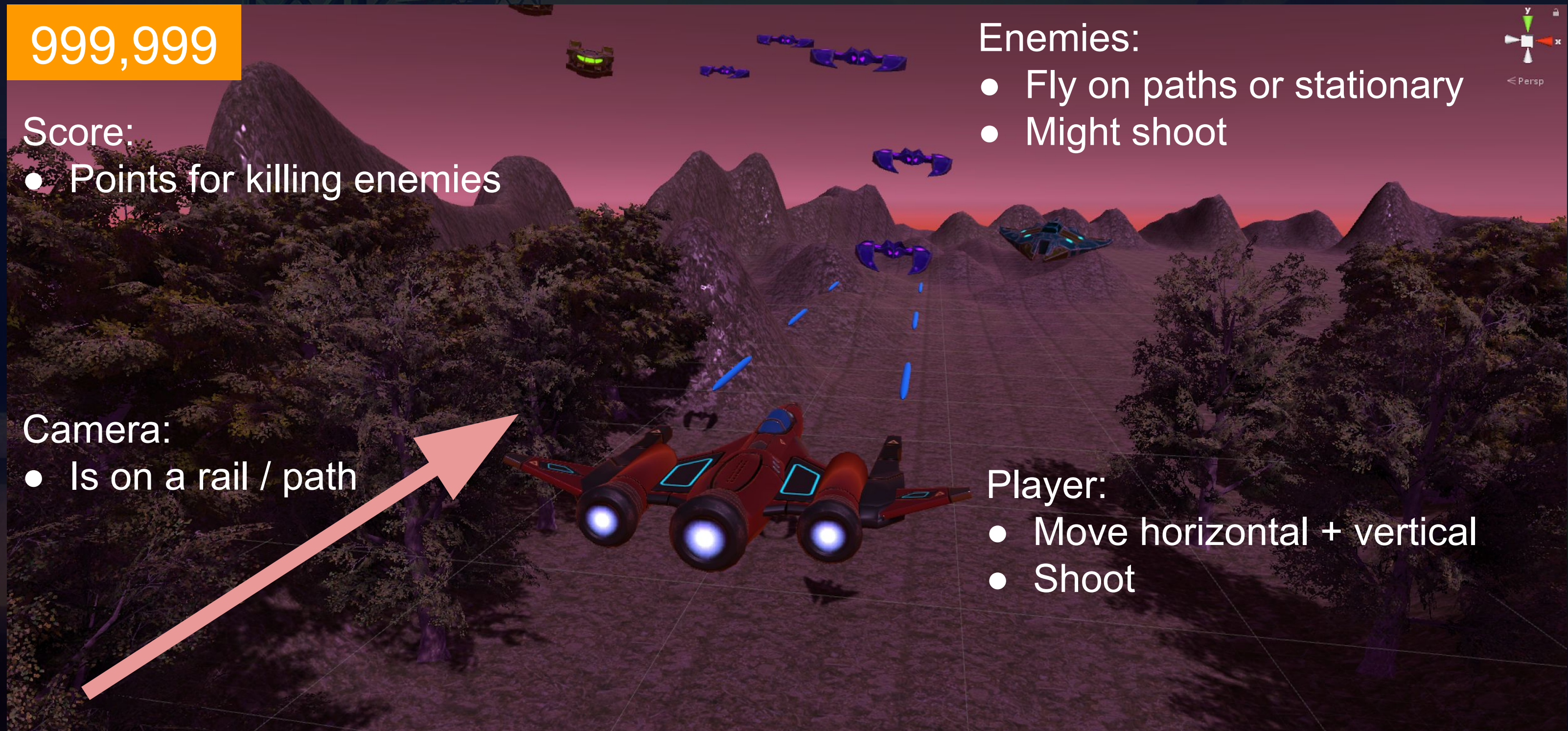
- Is on a rail / path

Enemies:

- Fly on paths or stationary
- Might shoot

Player:

- Move horizontal + vertical
- Shoot



Argon Assault Game Design

Player Experience:

Chaos

Core Mechanic:

Dodge and shoot

Core game loop:

Get as far as possible without dying in order to get the highest score possible. Start from beginning when die.

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Start()  
{  
    Horizontal();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Game Theme

- You must save your planet, Argon, from destruction by an invading force.

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Features - In Priority

- **Camera Rail:** Path through the level that the camera follows.
- **Player Movement:** Horizontal and vertical movement.
- **Shooting:** Player shoot bullets which do damage to opponent.
- **Health:** Enemies have health that depletes when shot.
- **Enemy Paths:** Enemies should travel on paths and be hand placed by the designer.
- **Score:** Points are given for killing enemies.
- **Game Loop:** Upon dying, player restarts the level.

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
atLadder;  
Vector3 moveDir = Vector3(), Vector3()  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Other Features You Can Consider

- **Multiple Levels:** Finish a level and load the next.
- **Player Shield:** When damaged, the player's shield depletes.
- **Pickups:** Flying over a shield pickup will increase the player's shields.
- **Momentary invulnerability:** After taking damage, the player cannot take damage again for X time.
- **Weapon Upgrades:** Flying over a pickup could increase weapon damage.

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed =
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Your Game Name And Theme?

- Think of a central theme for your game.
- Decide on a starting name for your game.
- Share with the community!



How To Add Terrain



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Create A Starting World

- Create a 600 x 600 terrain.
- Lower the terrain on y axis, then paint back up to 0 by setting your flatten value to 100.
- Play around with raising and lowering.
- Add some placeholder mountains.



Unity Terrain Tools



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Get Set Up With Terrain Tools

- Install the Terrain Tools package
- Add some canyons and ridges to your terrain

```
[SerializeField] float runSpeed;  
[SerializeField] float jumpSpeed;  
[SerializeField] float climbSpeed;
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

How To Use Unity Asset Store

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Download & Install Assets

- Find some assets on the asset store
- Download the assets
- Import the assets using package manager

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

Vector3 screenPos = new Vector3(
    SpriteRenderer.spriteRenderer;
    Animator animator;

void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTheGame();
}
```



Texturing Terrain In Unity



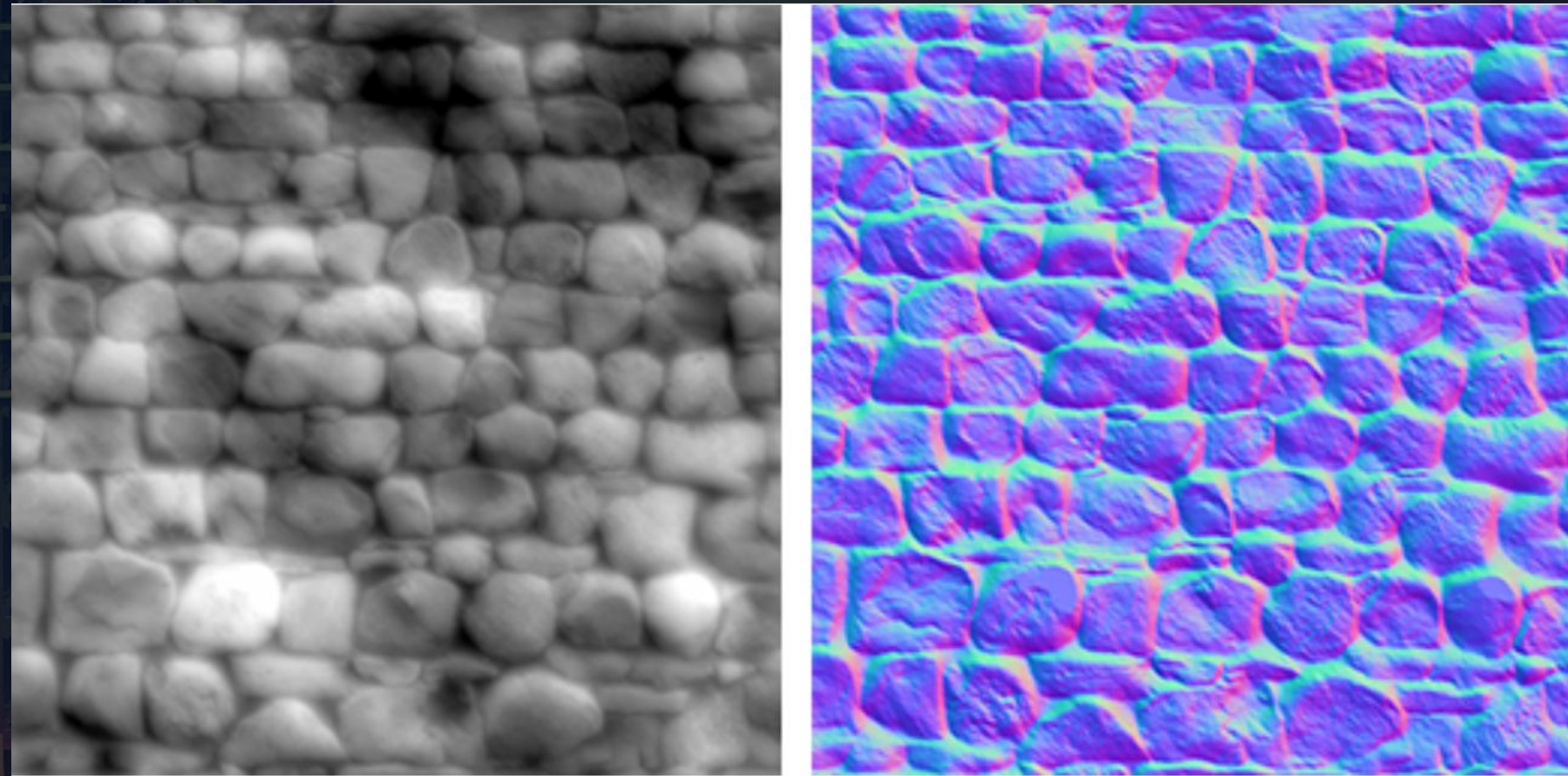
```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Bump Map, Height Map, Normal Map



- **Bump map:** texture with information on how light catches a shape.
- **Height map:** uses white-to-black scale to show height.
- **Normal map:** uses RGB values to indicate x, y, z facing direction.

Texture Your Terrain

- Download texture assets from asset store
- Set up your texture layers
- Paint some detail in your world
- Share a screenshot!

```
[SerializeField] float runSpeed;  
[SerializeField] float jumpSpeed;  
[SerializeField] float climbSpeed;
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```



Add Trees To Terrain



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Add Some Trees

- Find some trees on the asset store
- Import into your project

```
[SerializeField] float runSpeed;  
[SerializeField] float jumpSpeed;  
[SerializeField] float climbSpeed;
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Master Timeline For Player Rail

```
bool atLadder;  
Vector3 greenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Create Your First 5 Seconds

- Add Master Timeline
- Animate your player rig
- Adjust your grid colours if need be
- Create 5 seconds of your rail

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

bool atLadder;
Vector3 screenPos = new Vector3(
SpriteRenderer spriteRenderer;
Animator animator;

void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTheGame();
}
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Animate Enemy Using Timeline

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Create An Enemy Path

- Add an enemy placeholder (I'm adding a sphere)
- Using Timeline, animate that enemy so that the player narrowly misses it (and could shoot it)

```
[SerializeField] float runSpeed  
[SerializeField] float jumpSpeed  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
Vector3 screenPos = new Vector3(  
SpriteRenderer.spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
MoveHorizontally();  
ClimbLadders();  
ProcessJump();  
SaveTh...
```



Import Player Ship Asset



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Modular Asset Pack - Ebal Studios



Import / Create Player Ship

- Import our package or use your own assets.
- Ensure pivot point is central and object is prefabbed.

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

bool atLadder;
Vector3 screenPos = new Vector3(
SpriteRenderer spriteRenderer;
Animator animator;

void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTheGame();
}
```



Using GetAxis() For Movement

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Unity Input System

- I'll be showing you 2 different Input systems that Unity has available
- We only need to use one system for our game
- I'll be using the “old” system throughout this section because the “new” system isn't yet stable
- Our code **should** allow you to easily use the new system as it becomes more stable
- Feel free to use the new system if you prefer

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
Vector3 screenPos = new Vector3()  
SpriteRenderer spriteRenderer;
```

```
Animator animator;
```

```
void Update()  
{  
    if (Input.GetKeyDown(KeyCode.Space))  
    {  
        MoveHorizontally();  
        AddLadders();  
        ProcessJump();  
        SaveTheWorld();  
    }  
}
```

```
MoveHorizontally();  
AddLadders();  
ProcessJump();  
SaveTheWorld();
```

```
SaveTheWorld();
```




```
Input.GetAxis()  
InputAction.ReadValue<>()
```

Our Code

Input Manager / Input System

Use either Input Manager
OR Input System

Keyboard

Gamepad

Mobile

Unity listens,
uses the input
the player uses

Add The Same For The Vertical Axis

- Following the same flow, print out to the console when the player is pushing up or down.

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";
```

```
bool atLadder;
Vector3 screenPos = new Vector3(
SpriteRenderer spriteRenderer;
Animator animator;
```

```
void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveThePlayer();
}
```



Unity's New Input System



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Start Moving Our Player

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Move The Player To The Right

- Focus just on X axis for this lecture
- We'll use `localPosition` when moving player
- We need to add 2 things:
 - Where the player is located right now
 - +
 - How far we'd like to move it this frame
- Then we need to say, “move to this new position”

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Move The Player To The Right

- Change our new Vector3 so that the ship is moving slowly to the right.
- In other words: Current position + an x offset

```
[SerializeField] float runSpeed  
[SerializeField] float jumpSpeed  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(  
    SpriteRenderer.spriteRenderer;  
    Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveThePlayer();  
}
```



Move Player Using Input



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Move The Player Vertically

- Follow the same process to move the player vertically on the y axis.

```
[SerializeField] float runSpeed  
[SerializeField] float jumpSpeed  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(  
SpriteRenderer.spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveThePlayer();  
}
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Mathf.Clamp() To Constrain Movement

```
bool atLadder;  
Vector3 movePos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Apply The Same Logic For Vertical

- Clamp our player movement on the Y axis.

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

bool atLadder;
Vector3 screenPos = new Vector3(
SpriteRenderer spriteRenderer;
Animator animator;

void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTheGame();
}
```



How To Set Local Rotation

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

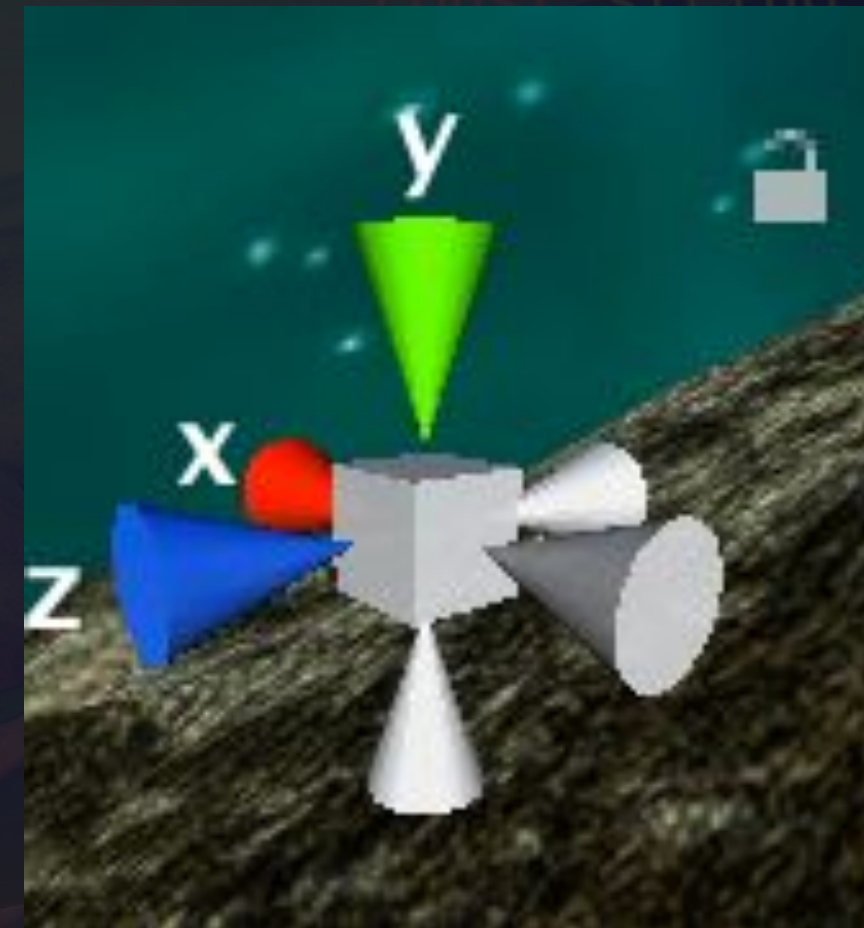
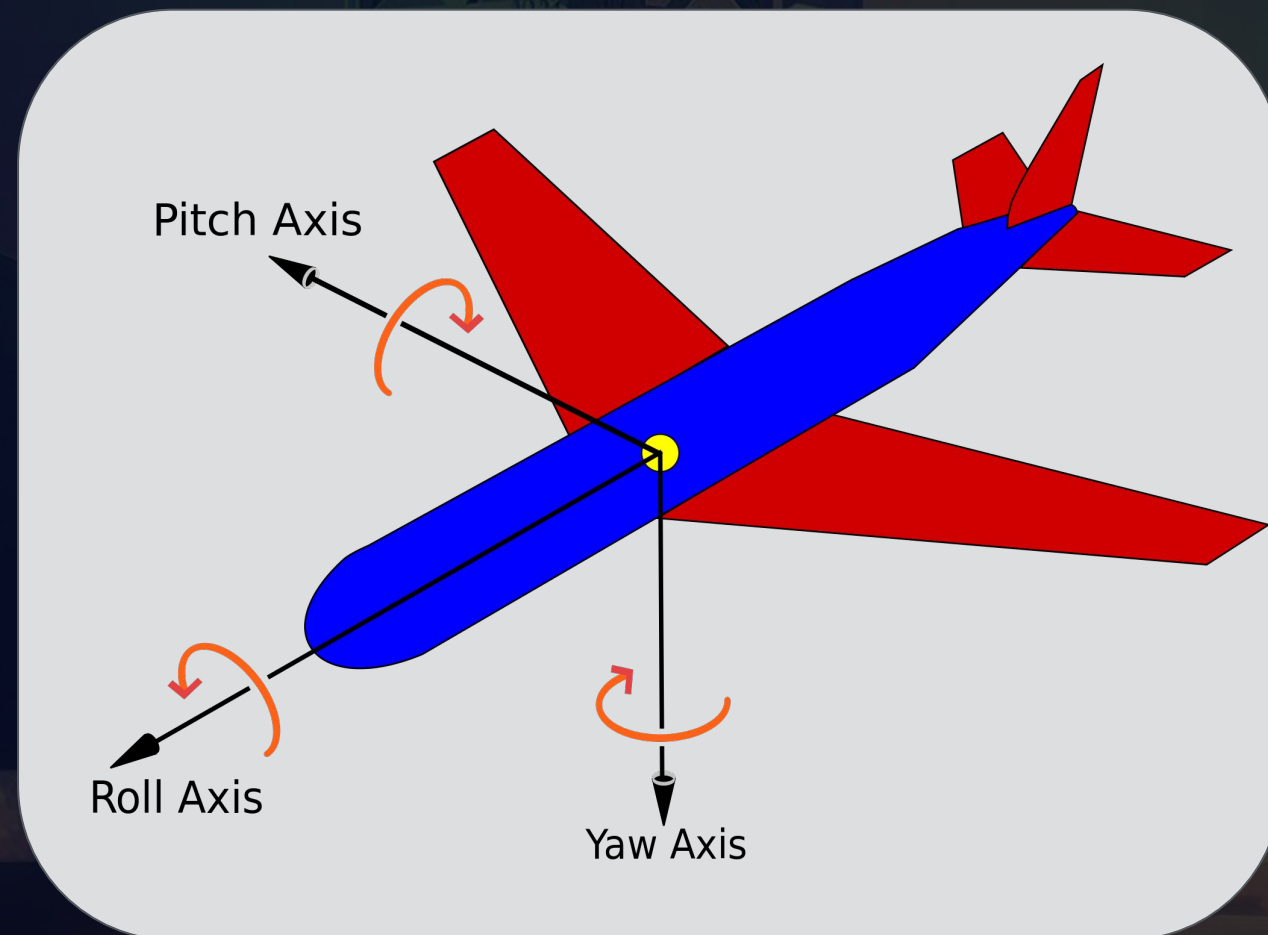
```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Roll, Pitch and Yaw



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
JUMP_TRIGGER = "Jump"  
LADDER_TAG = "Ladder"
```

```
Vector3 pos = new Vector3()  
spriteRenderer;  
ator;
```

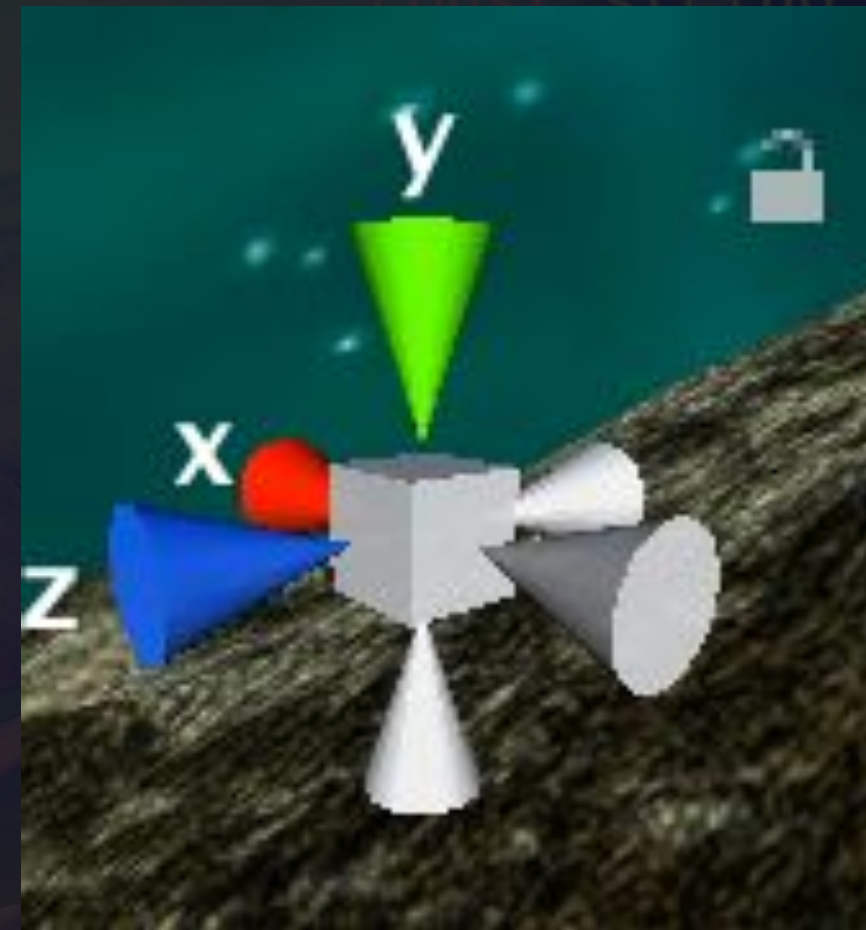
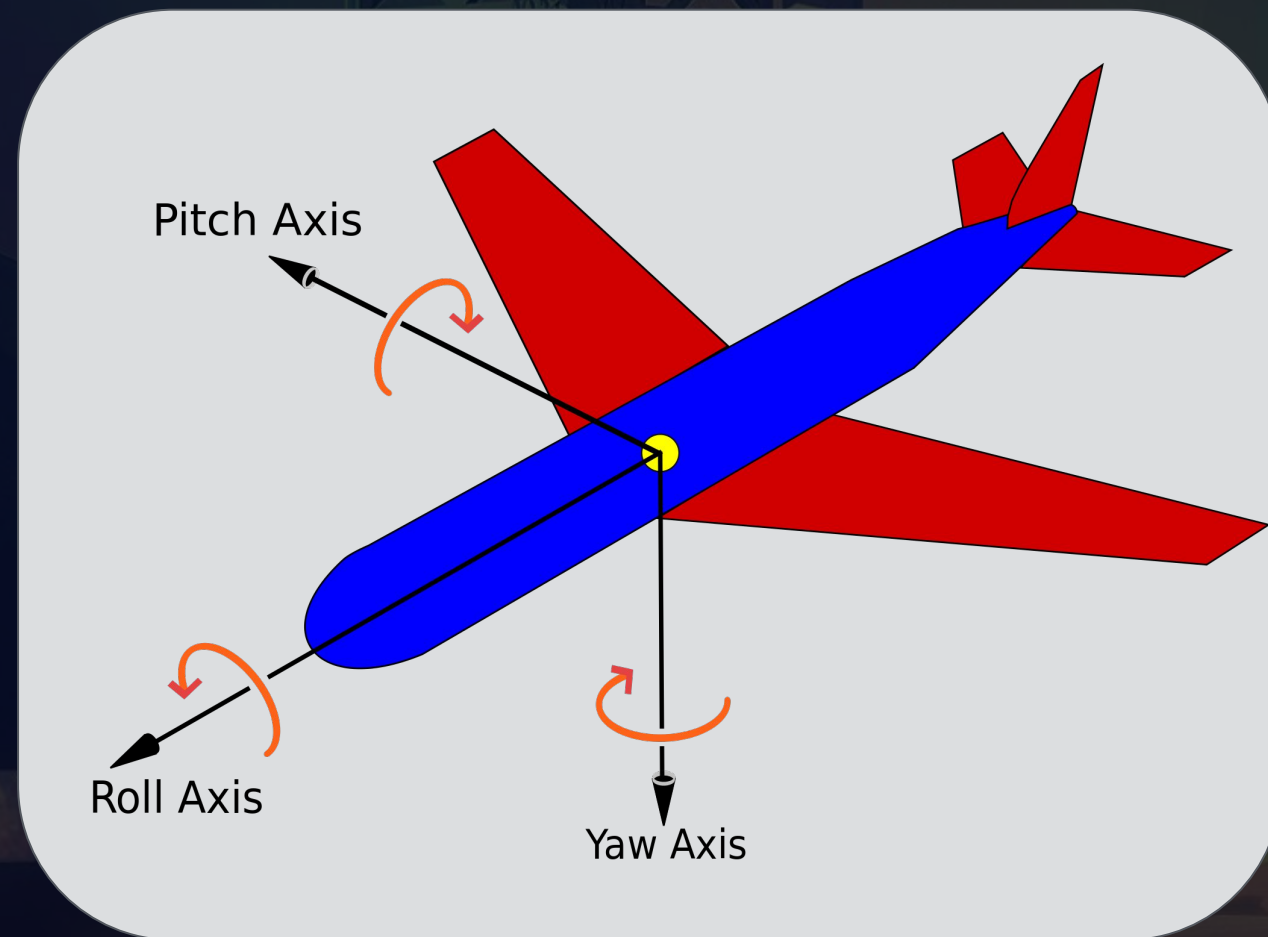
```
void Update()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Explore Rotations

- Have the editor in Local mode (x key).
- Use only the rotation tool in Scene view.
- Get to the following configuration...



Setting Local Rotation



```
transform.localRotation = Quaternion.Euler( x, y, z );  
transform.localRotation = Quaternion.Euler(pitch, yaw, roll);
```



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Rotate Ship With Position & Throw

```
bool isClimbing;  
Vector3 movement = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Our Rotation Couplings

	Position On Screen	Control Throw
Pitch	Coupled	Coupled
Yaw	Coupled	-
Roll	-	Coupled

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Finish Our Rotation Scheme

	Position On Screen	Control Throw
Pitch	Coupled	Coupled
Yaw	Coupled	-
Roll	-	Coupled

- Complete our rotation scheme as per the table
- Your parameters should feel good.
- Show off your work, maybe a 20s video.



Time To Tune And Tweak



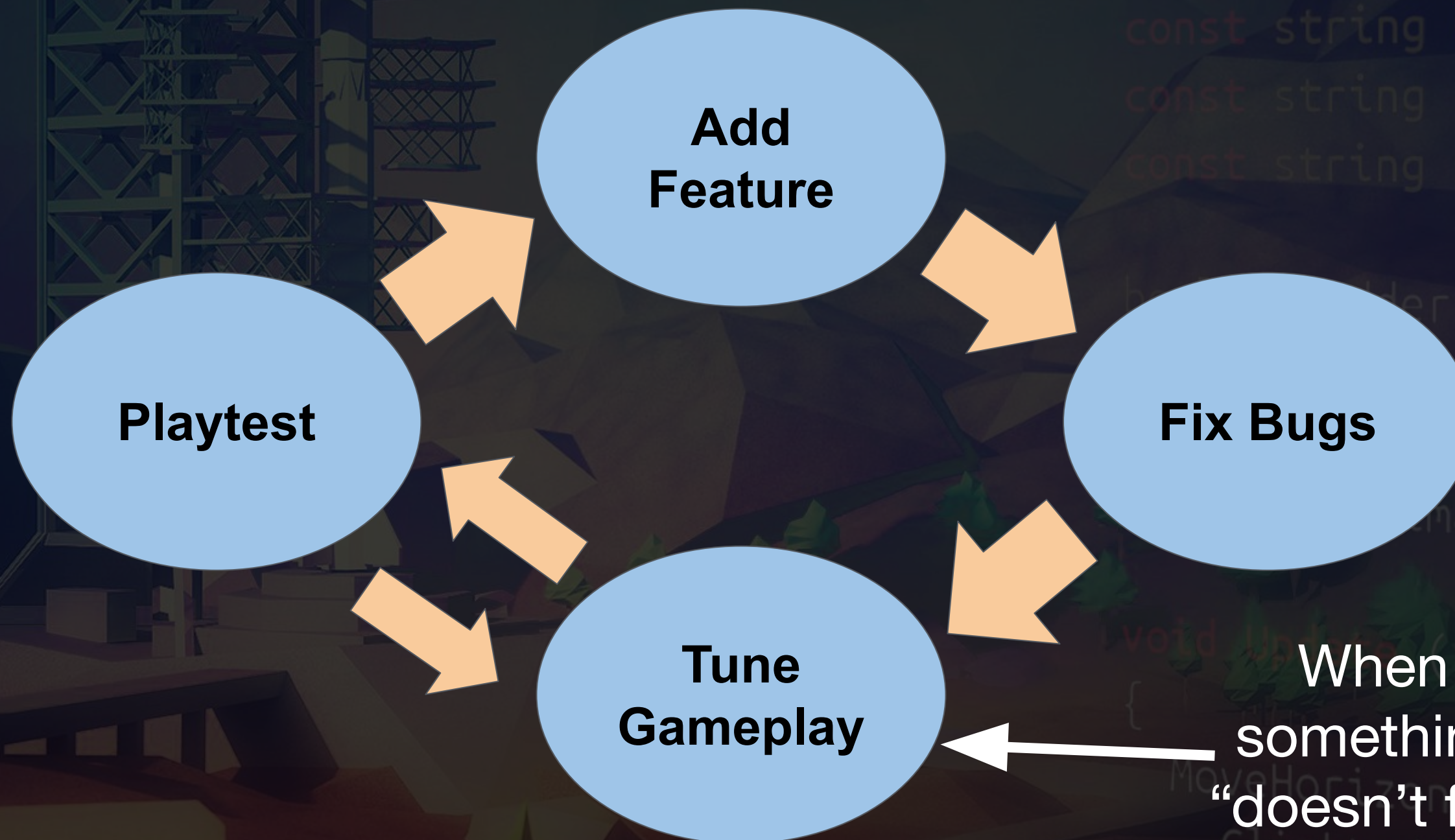
```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


When To Tune Your Gameplay



When
something
“doesn’t feel
right”

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed =  
  
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";  
  
void Update()  
{  
    if (ClimbBool) MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Tweak And Tune Ship Movement

- Spend some time tuning your game:
 - Add more length to your rail
 - Tune camera distance and FOV
 - Player speed, rotations, clamps
 - Obstacles to avoid in world

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

bool atLadder;
Vector3 screenPos = new Vector3(
SpriteRenderer.spriteRenderer;
Animator animator;

void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTheGame();
}
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Particle System Laser Bullets

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



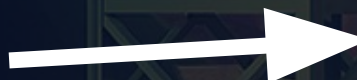
Particles System Component

Particle System is a Component added to a Game Object

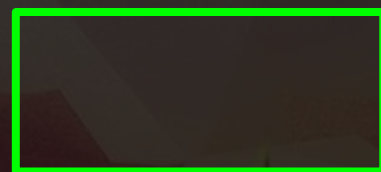
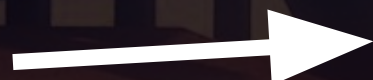
We use Modules for controlling behaviour

Each particle is not a Game Object

Particles



Emitter



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3()  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```

```
MoveHorizontally();  
ClimbLadders();  
ProcessJump();  
SaveTheWorld();
```



Create Bullet Particles

- Create projectiles to be shot from your ship
- Tune them to look like lasers
- Create a left and a right laser

```
[SerializeField] float runSpeed  
[SerializeField] float jumpSpeed  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Exploring Nested Prefabs In Unity

```
bool atLadder;  
Vector3 startPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Fix Your Nested Prefabbing

- We (me!) have created a bit of a mess
- Fix your nested prefabbing so each object is “owned” by the correct parent
- Set up your player ship so that editing the laser prefab will be updated within the game without issues



Set Up Firing Input

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Set Up Your Input System

- Using the old or the new input system (whichever is available to you), log to the console when the player pushes the fire button.

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

bool atLadder;
Vector3 screenPos = new Vector3(
SpriteRenderer spriteRenderer;
Animator animator;

void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTheGame();
}
```



New Input System

```
[SerializeField] InputAction fire;

void OnEnable()
{
    fire.Enable()
}
void OnDisable()
{
    fire.Disable()
}
```

```
[SerializeField] float runSpeed = 10f;
[SerializeField] float jumpSpeed = 10f;
[SerializeField] float climbSpeed = 10f;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

bool atLadder;
Vector3 screenPos = new Vector3();
SpriteRenderer spriteRenderer;
Animator animator;

void Update ()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        MoveHorizontally();
        ClimbLadders();
        ProcessJump();
        SaveTheWorld();
    }
}
```


Arrays & Foreach Loops

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



What Is An Array?

- If variables are like boxes, then arrays are boxes to store other boxes
 - I.e. Array allows us to store multiple objects in one variable
- We can only store the same Type of objects in the array variable

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
Vector3 screenPos = new Vector3()  
SpriteRenderer spriteRenderer;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Arrays Are Boxes For Variables



```
string[] names = { "Rick", "Ben", "Gary" };
```

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

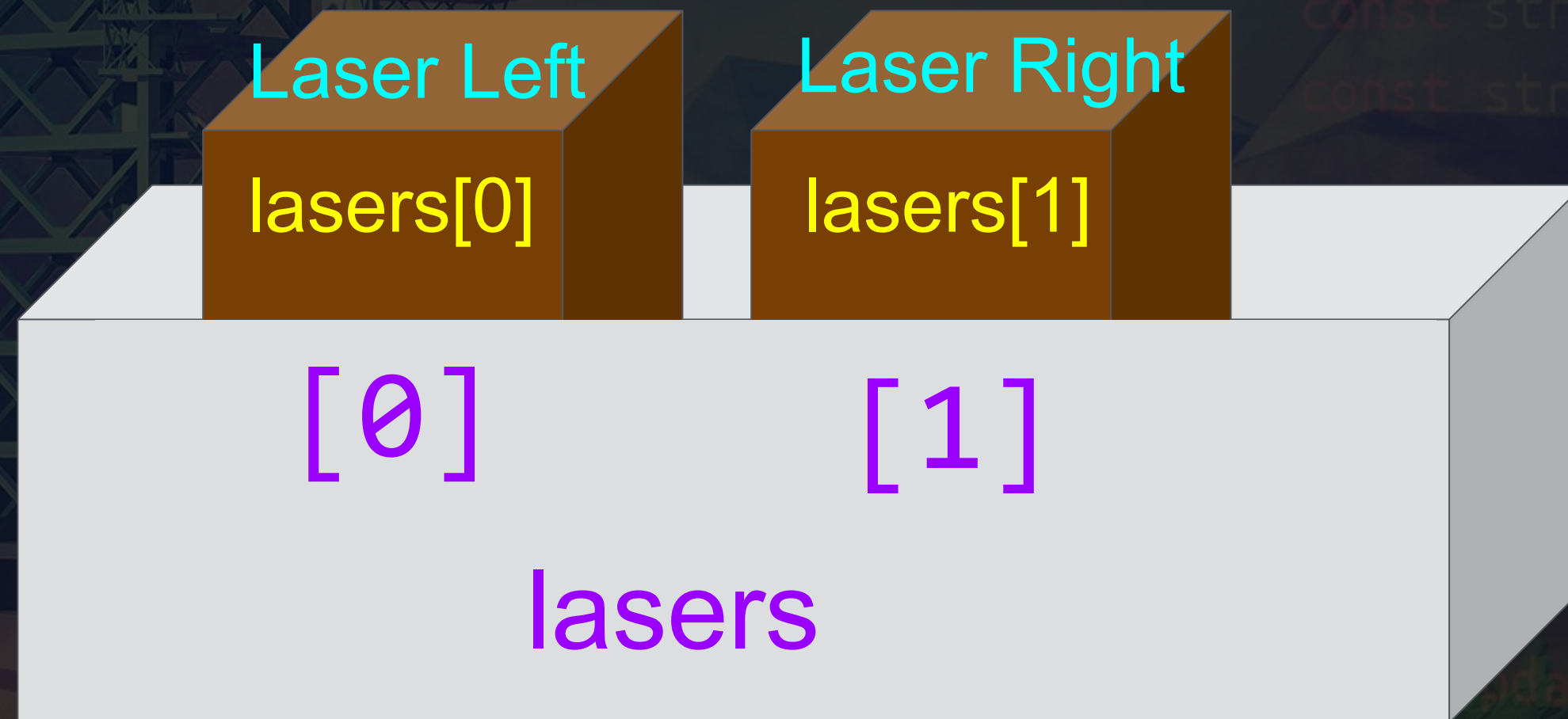
```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
LADDER_TAG = "Ladder"
```

```
= new Vector3();  
iteRenderer;
```

```
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Array For Our Laser GameObjects



```
lasers[0] = Laser Left;  
lasers[1] = Laser Right;
```

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed =  
  
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"  
  
Ladder;  
screenPos = new Vector3();  
renderer.spriteRenderer;  
animator;  
  
Update()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


What Is A foreach Loop?

- Control flow statement for traversing a collection
- It's a way of saying “do this to everything in our collection”

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



How to iterate over a collection

```
foreach (ObjectType item in things)
{
    item.DoSomething();
    // etc
}
```

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    // Move horizontally  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Deactivate The Lasers

- When you stop firing, the lasers should stop
- When you start firing again, the lasers should start
- Yes, we are duplicating some code, we'll improve on that at a later point

```
[SerializeField] float runSpeed  
[SerializeField] float jumpSpeed  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
Vector3 screenPos = new Vector3(  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update()  
{
```

```
MoveHorizontally();  
ClimbLadders();  
ProcessJump();  
SaveTheGame();
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Deactivating Particle System Emission

```
bool atLadder;  
Vector3 movementDirection = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Disable Particle System Emission

- Instead of enabling and disabling the entire Game Object, just enable or disable the emission module on the Particle System component




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Header & Tooltips Attributes

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Add Meaningful Attributes

- Group your variables using the **Header** attribute
- Add information about the variables using the **Tooltip** attribute

```
[SerializeField] float runSpeed;  
[SerializeField] float jumpSpeed;  
[SerializeField] float climbSpeed;
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    // Move horizontally  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Understanding Collisions & Triggers

```
bool isOnLadder;  
Vector3 startPos = new Vector3(  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
MoveHorizontally();  
ClimbLadders();  
ProcessJump();  
SaveTheWorld();  
}
```



Prepare Debug Statements

- Print to the console when the player
 - Collides with something
 - Triggers something
- Bonus points: make things clearer by printing the Game Object names of the 2 things that had the collision or trigger event (eg. Player Ship bumped into Enemy)



	Static Collider	Rigidbody Collider	Kinematic Rigidbody Collider	Static Trigger Collider	Rigidbody Trigger Collider	Kinematic Rigidbody Trigger Collider
Static Collider		Collision			Trigger	Trigger
Rigidbody Collider	Collision	Collision	Collision	Trigger	Trigger	Trigger
Kinematic Rigidbody Collider		Collision		Trigger	Trigger	Trigger
Static Trigger Collider		Trigger	Trigger		Trigger	Trigger
Rigidbody Trigger Collider	Trigger	Trigger	Trigger	Trigger	Trigger	Trigger
Kinematic Rigidbody Trigger Collider	Trigger	Trigger	Trigger	Trigger	Trigger	Trigger




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Detecting Particle Collisions

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



To Use Particles As Bullets...

1. Ensure particle collision & message sending is on.
2. Add a **non-trigger** collider to the target.
3. Use **OnParticleCollision()** on the target.

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 greenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Get Your Enemies Destroying

- Use `OnParticleCollision()` in `Enemy.cs`
- Call `Destroy()` to “blow up” the enemy `gameObject`.

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

bool atLadder;
Vector3 screenPos = new Vector3(
SpriteRenderer spriteRenderer;
Animator animator;

void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTh...
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Reload Scene After Collision

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Restarting...

- When the player crashes into something:
 - Disable player controls
 - Wait 1 second
 - Reload the level
- HINT: Looks at Project Boost repo

```
[SerializeField] float runSpeed  
[SerializeField] float jumpSpeed  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(  
    SpriteRenderer.spriteRenderer;  
    Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Create Explosion Particle Effect

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Create An Explosion

- Create a new particle effect and tune it to look like an explosion

```
[SerializeField] float runSpeed;  
[SerializeField] float jumpSpeed;  
[SerializeField] float climbSpeed;
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```



Trigger Player Explosion

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Trigger Explosion When Crashing

- Trigger the particle effect to play at the right time

```
[SerializeField] float runSpeed  
[SerializeField] float jumpSpeed  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    // Move horizontally  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```



Instantiate At Runtime

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Destroy Instantiated Objects

- Create SelfDestruct.cs
- Use Destroy() to ensure that none of our newly instantiated particle effects stick around

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

bool isClimbing;
Vector3 screenPos = new Vector3(0, 0, 0);
SpriteRenderer spriteRenderer;
Animator animator;

void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTheGame();
}
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Public Methods In Unity C#

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Remember Encapsulating Our Code?

- En-capsule-ating - putting in a capsule.
- Different parts of your code have a “need to know basis” level of access.
 - Ie. Don't let everything access everything else.

ClassA

Data & Statements

NO

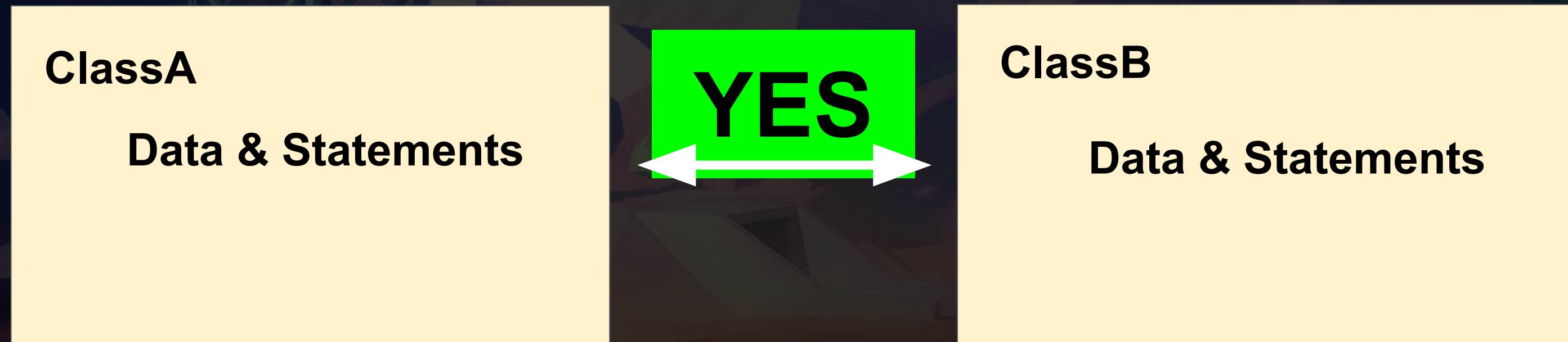
ClassB

Data & Statements



Remember Encapsulating Our Code?

- But now we'd like to let one class (Enemy) influence another class (ScoreBoard).



Public Versus Private Access Modifiers

ClassA

Private MethodOne()

Public MethodTwo()

ClassB

MethodOne();

NO

MethodTwo();

YES

Public Access Modifier

public

void

StartGame()

Access Modifier
Scope of use

Return type

void = return nothing

Method name

WHAT to do

Parameter

() = nothing

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()
```

```
MoveHorizontally();  
ClimbLadders();  
ProcessJump();  
SaveTheWorld();
```


Increase Player Score

- Call our public method to increase the score
- Create a tuneable variable for how much to increase score per hit
- Print to the console the score each time it updates

```
[SerializeField] float runSpeed  
[SerializeField] float jumpSpeed  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(  
    transform.position.x, transform.position.y, transform.position.z);  
Animator animator;
```

```
void Update ()  
{  
    // Move  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheScore();  
}
```




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Simple User Interface For Score

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Add A Simple UI

- Add your text element
- Pick a font
- Format color and size

```
[SerializeField] float runSpeed;  
[SerializeField] float jumpSpeed;  
[SerializeField] float climbSpeed;
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```



ToString() To Display Score

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Show Our Actual Score

- Our text field requires a string to display properly
- **Tostring()** is a method which converts other types of data into a String
- Use **Tostring()** to show our updated score when shooting enemies

```
[SerializeField] float runSpeed  
[SerializeField] float jumpSpeed  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(
```

```
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{
```

```
MoveHorizontally();  
ClimbLadders();
```

```
ProcessJump();  
SaveTheGame();
```



Enemy Hit Points



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Flex Your Programming Muscles!

- Give our enemies hit points
- Different enemies could have different hit points
- Each time the player hits an enemy, our score should still go up
- When the enemy's hitpoints reach zero, enemy should explode
- BONUS: Implement hit VFX



Set Up Enemy Prefabs



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Add A Rigidbody At Runtime

- Using **AddComponent<>()** add a Rigidbody to each enemy game object when the game starts
- Bonus points: Ensure that Use Gravity is disabled

```
[SerializeField] float runSpeed;
[SerializeField] float jumpSpeed;
[SerializeField] float climbSpeed;

const string CLIMB_BOOL = "Climb";
const string JUMP_TRIGGER = "Jump";
const string LADDER_TAG = "Ladder";

bool isClimbing;
Vector3 screenPos = new Vector3(0, 0, 0);
SpriteRenderer spriteRenderer;
Animator animator;

void Update ()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTheGame();
}
```



Create At Least 5 Enemies

- Using the same process, create a total of at least 5 enemies

```
[SerializeField] float runSpeed;  
[SerializeField] float jumpSpeed;  
[SerializeField] float climbSpeed;  
  
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```



Using FindWithTag()

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Set Parent Reference

- Create a new tag for our Spawn At Runtime game object
- Set the reference for our parent game object using **FindWithTag()**
- Update other areas of our code accordingly to make this work
- Note: Pay attention to 'Type'




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Control Tracks For Enemy Waves

```
bool atLadder;  
Vector3 spawnPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Set Up Enemy Waves

- Create at least 5 waves of enemies
- Add those waves to your timeline
- Tune the timing so that the player can shoot and dodge the enemies

```
[SerializeField] float runSpeed;  
[SerializeField] float jumpSpeed;  
[SerializeField] float climbSpeed;
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
Vector3 screenPos = new Vector3(  
    SpriteRenderer.spriteRenderer;  
    Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();
```



Timeline For Dialogue

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Add Some Character Dialogue

- Either using my assets, or creating your own, place instructions or information for the player using Timeline.
- Share a screenshot or video of your game with us.




```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

Singleton Pattern For Music Player

```
bool isLadder;  
Vector3 movement = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Find Some Music For Your Game

- Search for royalty free music to add to your game.
- Try to match the player experience (eg. hectic experience = hectic music).



Sneaky Explosion SFX



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Adding Explosion SFX

- What is the simplest way you can think of to implement an explosion SFX?

```
[SerializeField] float runSpeed;  
[SerializeField] float jumpSpeed;  
[SerializeField] float climbSpeed;
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```



Skybox & Fog



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Grab A Skybox

- Head over to the asset store and grab a skybox asset which matches your game.
- If you don't want to or can't get to the asset store, you can just use the procedural skybox.



Unity Post-Processing



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

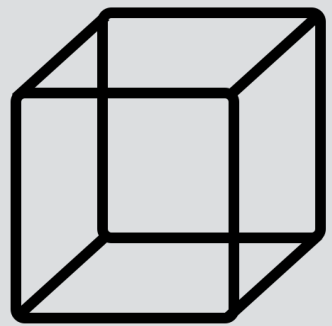
```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```

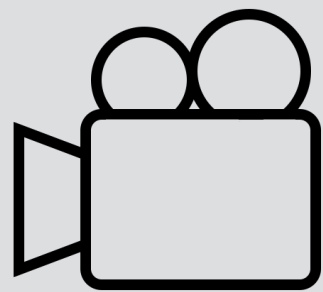

Post Processing

- Add effects to your Camera to change the look of your game



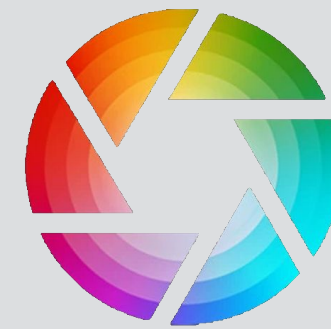
Volume

Where in
the world



Layer

Which
camera
triggers PP



Profile

Specific settings
for effects

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool  
Vect  
Spri  
Animator animator;  
void Update()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Experiment With Post Processing

- Add each of the post processing effects to see if it gives your game the feel you want.
- Take a screenshot and share in the community forum.



Your 3 Minute Experience



```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


Creating Your 3 Minute Experience

- Your level should be a roller coaster with moments from 1 to 10 in intensity
- Consider your “beat chart”...

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Level Design Beat Chart

Intensity



Time

1. We're under Attack!
2. Incoming, sector 5.
3. They're hitting our shield generators!
4. Watch out near the mines!
5. Phew, I think that's the worst of them.
6. Help! Help! They're everywhere!
7. Holy mother of mercy, what is that!
8. You did it.



Creating Your 3 Minute Experience

- Variety is king:
Each 10 second moment should be unique
- Always be thinking of your player experience
- Give your player choices

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool isClimbing;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    // Move  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Variety

- There are thousands of combinations of variables to create variety:
 - Enemies: appearance, size, speed, hit points, movement patterns, FX, score
 - Player rail: speed of movement, where you fly, rotation, movement patterns
 - Environment: Textures, shapes, trees, obstacles, explosions, colours, lighting, visibility
 - Player: Laser speed and colour, controls, camera

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climb"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3()  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



Create Your 3 Minute Experience

- Take the time now to create a three minute experience.
 - Create a player path you're happy with
 - Add enemies and enemy waves
 - Add voice overs / characters / story
 - Tune the game so that its the right difficulty
- Create a build of your game, upload to sharemygame.com and share with your friends and with our community



Wrap Up - Argon Assault

```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```





```
[SerializeField] float runSpeed =  
[SerializeField] float jumpSpeed =  
[SerializeField] float climbSpeed
```

```
const string CLIMB_BOOL = "Climbing"  
const string JUMP_TRIGGER = "Jump"  
const string LADDER_TAG = "Ladder"
```

```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```