

## Section Intro - Obstacle Course



Rick Davidson

```
[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"
```

# Game Design - Obstacle Course

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

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





```
[SerializeField] float runSpeed = 10f;
[SerializeField] float jumpSpeed = 8f;
[SerializeField] float climbSpeed = 6f;
```





# Game Design

## Player Experience:

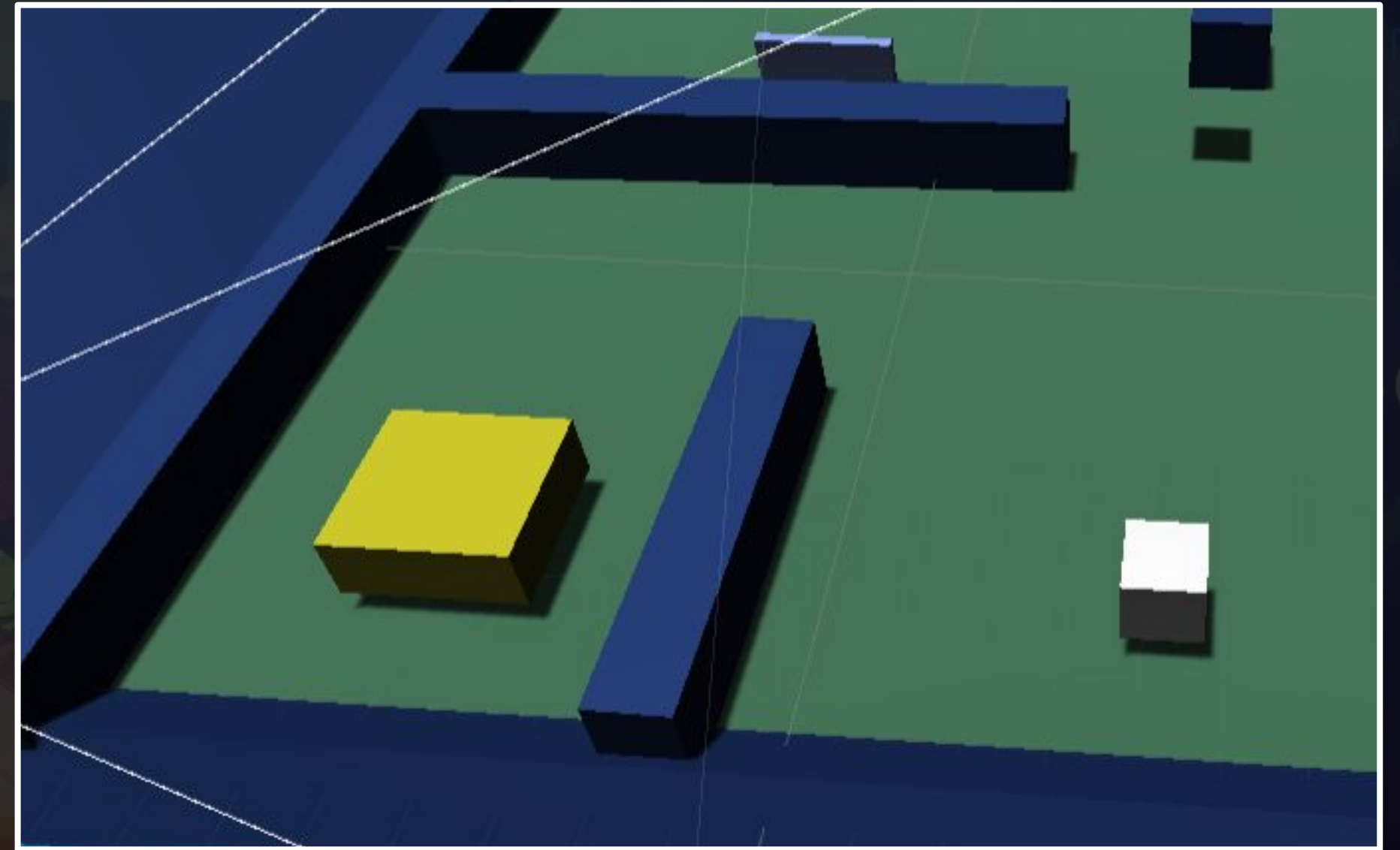
Careful? Clever? Nah  
Nimble / agile

## Core Mechanic:

Move & dodge obstacles

## Game Loop:

Get from A to B



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

```
const string CLIMB_BOOL = "Climb"
```

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

()

tp  
or

+

1

+



# A Quick, Silly Challenge

- Give your Player a name.
- Mine is going to be “Dodgy”.

```
[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();  
}
```





## Start() & Update()

```
[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 The Party Started

- Create a new Unity 3D project
- Add a ground plane
- Create your “player”
- Rename your player

```
[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();  
}
```





# Introducing Variables



Rick Davidson

```
[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();  
}
```



# Variables Are Like Boxes

- Variables help us store, manipulate and refer to information
- Each variable has a NAME
- Each variable contains DATA
- Each variable is of a particular TYPE

```
int hitPoints = 20;
```



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

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

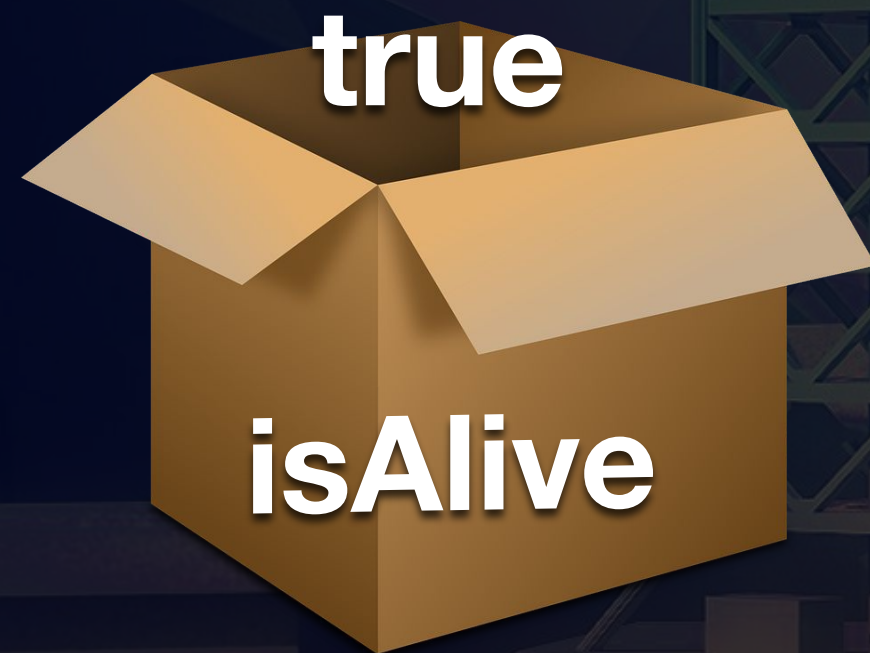
```
bool atLadder  
Vector  
Spriteke  
Animator
```

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

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



# Variables Are Like Boxes



```
bool isAlive = true;
```

```
string myName = "Rick";
```



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

```
const string CLIMB = "Climb"  
const string JUMP = "Jump"  
string LADDER
```

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

```
void Update
```

```
for (int i = 0; i < ladderCount; i++)
```

```
MoveHorizontal
```

```
ClimbLadder
```

```
ProcessJump();
```

```
SaveTheWorld();
```

```
1
```



# Create Two More Variables

- Create a variable for the y value and for the z value.
- Change the values so your player flies straight up in the air

```
[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 SerializeField

```
[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();  
}
```





# Serialize Your Other Variables

- Make all three of your variables serialized and therefore accessible in the inspector.
- While in play mode, make your player run away from the camera and then run back to the camera.





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

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

# C# Formatting & Input.GetAxis()

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

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



Rick Davidson



# Add Vertical Axis

- Update one of our variables so we are moving our player forwards and backwards (along the ground plane, not flying in the air).

```
[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();  
}
```





# Time.deltaTime

```
[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();  
}
```





# Using Time.deltaTime

- Using **Time.deltaTime** Unity can tell us how long each frame took to execute.
- When we multiply something by **Time.deltaTime** it makes our game “frame rate independent”.
- I.e. The game behaves the same on fast and slow computers

```
[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;  
Animator animator;
```

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





On Update (each frame) move 1 unit to the left

Slow  
Computer

Fast  
Computer

Frames per second

10

100

Duration of frame

0.1s

0.01s

Distance per second  $1 \times 10 \times 0.1 = 1$

$1 \times 100 \times 0.01 = 1$



# Multiply By A Speed Variable

- Create a new variable called moveSpeed. The value of moveSpeed does not need to update each frame.
- Make it available in the inspector.
- Multiply your xValue and zValue by moveSpeed.
- Tune your player movement (as best you can for now).





# Cinemachine Follow Camera

```
[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();  
}
```



Rick Davidson



# What Is Cinemachine?

- **Cinemachine** is a powerful package that lets us:
  - manage multiple cameras in our scene
  - Easily create rules for our cameras

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

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

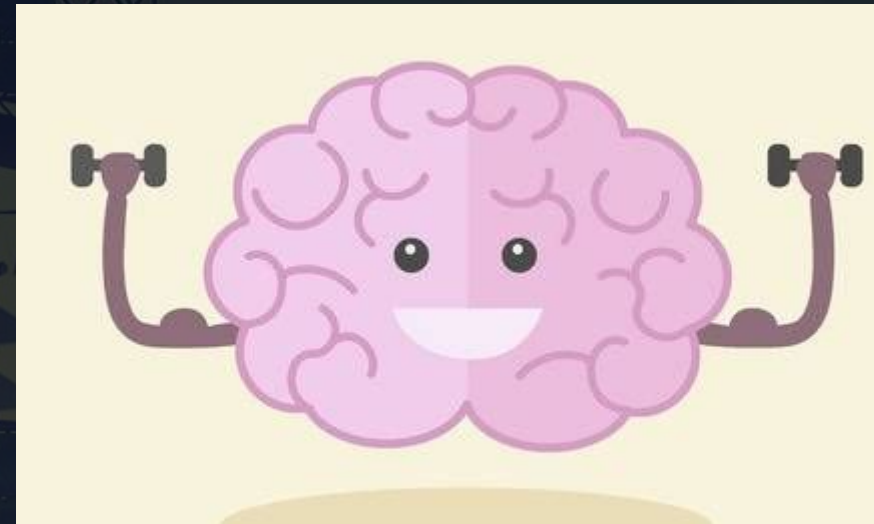
```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

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





Cinemachine  
Brain



Main Camera



Virtual  
Cameras



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

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

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

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





# Add A Follow Camera

- Install Cinemachine from Package Manager
- Add Cinemachine Brain component to camera
- Add Virtual Camera
- Rename Virtual Camera
- Point the Virtual Camera to follow the player
- Tune distance





# Basic Collisions

```
[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();  
}
```





# Make Some Walls

- Add some walls around the outside of your play area
- If you like, make them a different colour

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

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





# Introduction To Methods

```
[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 Are Methods?

- Methods (also called Functions) execute blocks of code that makes our game do things.
- To achieve this we must:
  1. DECLARE and define our method
  2. CALL our method when we want it to execute

```
[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();  
}
```





# Real World Example

## DECLARE Method

*CleanYourRoom*

- Pick up clothes
- Throw out garbage
- Kill the mutated science project that threatens to destroy life as we know it

## CALL Method



Hey, go and  
*CleanYourRoom*

“Go and do the steps we’ve defined and agreed to”

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





# Syntax Used For Declaring Method

```
void CleanYourRoom()
```

```
{
```

```
Things To Do;
```

```
}
```

Return value

void = return nothing

Function name

WHAT to do

Parameter

() = nothing required



# More Flavour Can Be Added

Two more common things when calling methods:

1. We can ask for some information to be **RETURNED**
2. We can specify some **PARAMETERS** are needed when calling the method

Using our real world example again...

```
[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();  
}
```





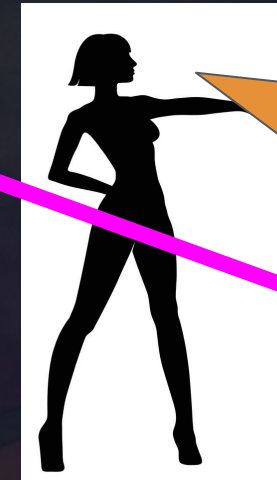
# More Flavour Can Be Added

## DECLARE Method

*CleanYourRoom*(*deadline*)

- Do it before (*deadline*)
- Pick up clothes
- Throw out garbage
- Kill the mutated science project that threatens to destroy life as we know it
- Is the room dirty?

## CALL Method



Hey, go and  
*CleanYourRoom*  
(*you've got 1 hour*).  
*isRoomDirty?*

\*Parameter / Argument

\*Return data



# Syntax Used For Declaring Method

```
void CleanYourRoom()
```

```
{
```

```
Things To Do;
```

```
}
```

Return value

void = return nothing

Function name

WHAT to do

Parameter

() = nothing required



# Syntax Used For Declaring Method

```
bool CleanYourRoom(int time)
{
    Things To Do;
    Deadline = time;
    return isRoomDirty;
}
```

Return value  
(Type = bool)

Return keyword  
(Type = bool)

Parameter  
Needs type  
and name



# Syntax Used For Calling Method

CleanYourRoom();

Remember  
semi colon

Executes statements defined in the code block...

{

...

}

```
[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;  
Animator animator;
```

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





# Defining & Calling

## DECLARE Method

```
void CleanYourRoom()  
{  
    Pick up clothes;  
    Clean garbage;  
    Kill mutated experiment;  
}
```

## CALL Method

```
CleanYourRoom();
```

```
[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();  
}
```





# But What About Start() & Update()?!

- We've been defining **Start()** & **Update()** but not calling them
- Where are they called?
- Unity's internal logic is taking care of calling them for us at the right time
- These are referred to as “callbacks”

```
[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 currentRenderer;  
Animator animator;  
  
void Update()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```



# Practicing With Methods

```
[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();  
}
```



Rick Davidson



# Call Our Method

- Call our method so that our instructions print out one time to the console.

```
[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 OnCollisionEnter()

```
[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

- Create a new C# script called **ObjectHit**
- Attach that script to all 4 walls

```
[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 GetComponent<>()

```
[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();  
}
```



Rick Davidson



# Incrementing A 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();  
}
```



Rick Davidson



# Prepare For Score Feedback

- Create a new c# script called **Scorer**
- Create a new **OnCollisionEnter()** method
- When we hit something, print to the console, “You’ve bumped into a thing this many times:”





# Using Time.time

```
[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();  
}
```





# Problem & Solution

## Problem to solve:

- Make an object fall after 3 seconds has passed

## Solution:

1. A timer - **Time.time**
2. A mechanism to “do a thing if 3 seconds has elapsed” - **if statement**
3. A way to start the object falling after 3 seconds - **disable / enable gravity**

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



# Print Out Time Elapsed

- On every frame, print out to the console how much time has elapsed since the game started.
- HINT: Use Time.time within debug.log
- Add your script to a game object

```
[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 onLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
SpriteRenderer spriteRenderer;  
Animator animator;
```

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





# If Statements

```
[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();  
}
```



Rick Davidson



# Conditional **if** Statement

```
if (some expression evaluates to true)
{
    // execute this code block only
}
```

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

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

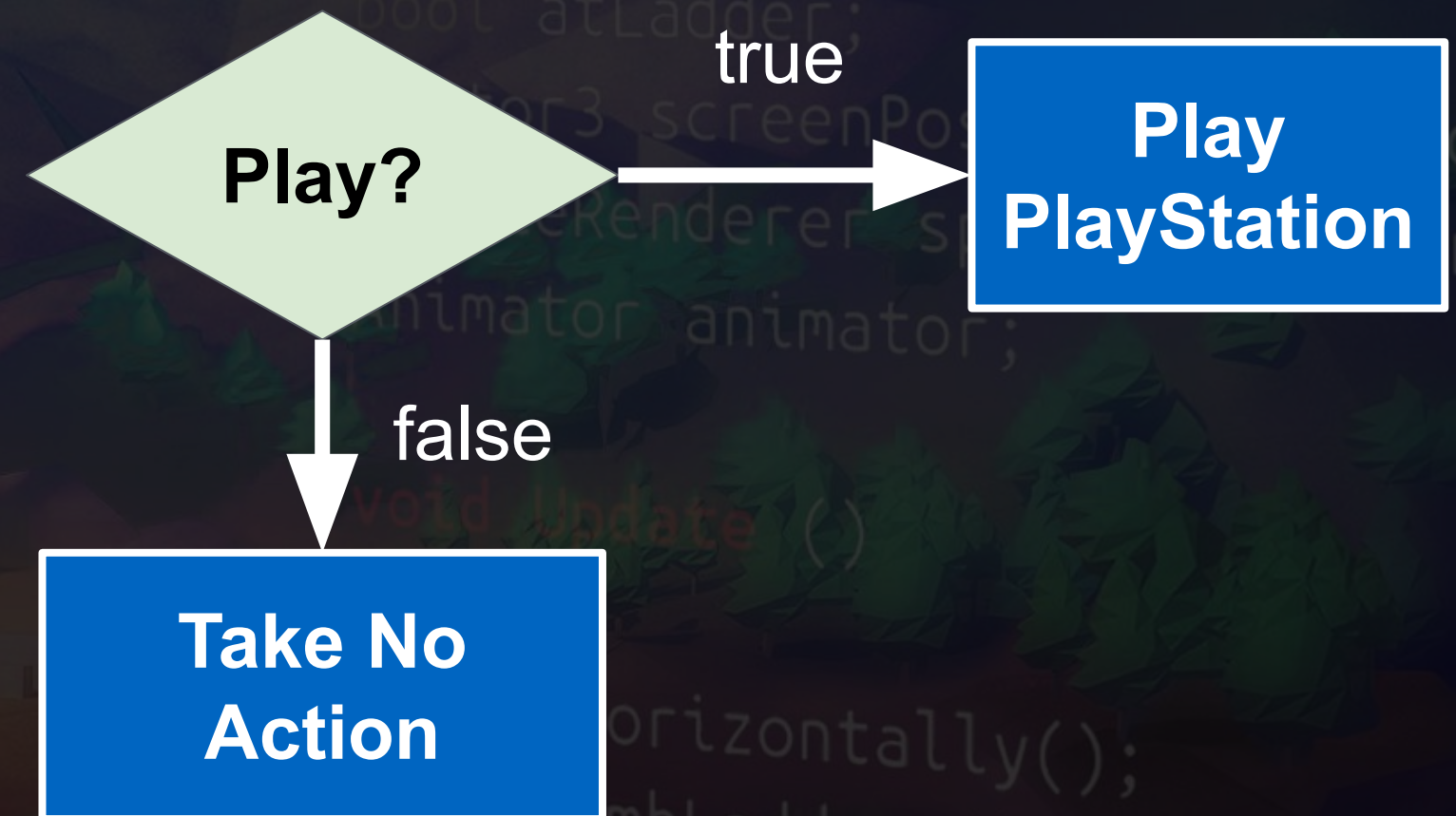
```
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;
```

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



# In Other Words...

```
if (in the mood for Pwning Noobs)
{
    // play some PlayStation
}
```



```
[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;  
for3_screenPos  
Renderer's  
Animator animator;
```

```
void Update ()
```

```
horizontally();
```

```
climbLadders();
```

```
ProcessJump();
```

```
SaveTheWorld();
```



# Use A Variable For Time To Wait

- Instead of “hard coding” 3 seconds, use a variable for the *time to wait* that can be easily changed in the Inspector.

```
[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();
}
```





# Caching A Reference

```
[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();  
}
```





# Simple Definition...

**Caching** is a technique of storing frequently used data/information in memory, so that it can easily be accessed when needed.

```
[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();  
}
```





# Make It Fall!

- Add 2 lines to our code that make our object become visible and fall after it has waited for the right amount of 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 ()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheGame();  
}
```





# Using Tags

```
[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();  
}
```





# Rotate An Object

```
[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 Variables For X, Y & Z

- Serialize 3 variables and pass those in to our Rotate method so that our object spins.

```
[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();  
}
```





# Prepare Our Prefabs



Rick Davidson

```
[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 Prefabs

- Create Prefabs for the following objects:
  - Dropper
  - Spinner
  - Roller
  - Obstacle

```
[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();  
}
```





# Build An Obstacle Course



```
[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();  
}
```



# Build Your Obstacle Course!

- Create a fun layout that makes the player go from A to B.
- Use your droppers, rollers and spinners to create interesting moments for the player.





# Wrap Up - Obstacle Course

```
[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();  
}
```