การเขียนโปรแกรมคอมพิวเตอร์ขั้นสูงเพื่อควบคุมอุปกรณ์

Advance Computer Programming

[สัปดาห์ที่ 3]

สอนโดย พงศธร เกียรติเจริญพร (มิว)

Physic Game

Physic Game

- คือเกมที่มี Core Gameplay เกี่ยวกับแรงทางฟิสิกส์ เช่น การชน , แรงโน้มถ่วง , หมุน-กลิ้ง
- สามารถทำได้ง่ายโดยใช้ Physic Engine ใน Unity
- ตัวอย่าง

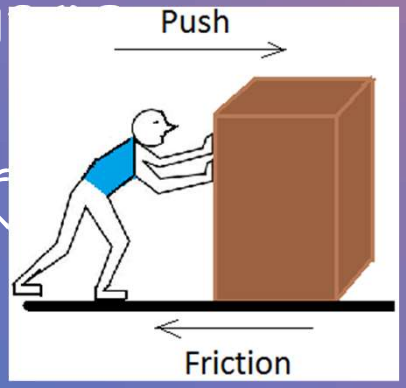
Components

Components

- การใส่ Component คือการใส่วิญญาณให้วัตถุ

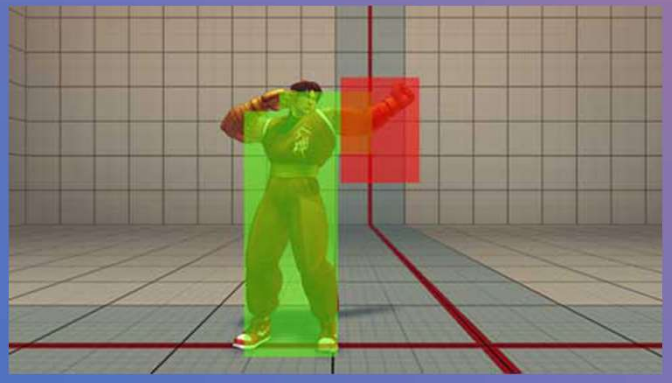
Rigidbody Component

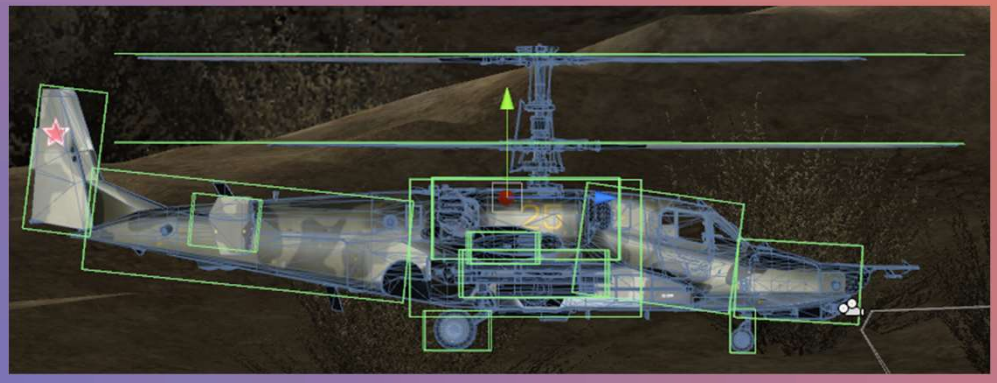
- ใส่เพื่อบอกให้ Physic Engine เข้ามาควบคุมการเคลื่อนไหวของวัตถุนี้
- ใช้งานร่วมกับ Collider Component ซึ่งทำ
บอกรูปทรงของ
*นอกจากนี้ ยังม หมวดฟิสิกส์
หลายตัว



Collider Component

- Collider หรือ Hitbox คือรูปทรงอย่างง่าย เช่น Box หรือ Sphere เพื่อเช็คการชนกันของวัตถุในเกม
- Collider ใน Unity จะมีอยู่ 2 ลักษณะ คือ แบบ Collision และ แบบ Trigger
  - Collision ใช้ร่วมกับ Rigidbody จะมีการชนเกิดขึ้นจริงๆ
  - Trigger ใช้เพื่อวางอาณาเขต

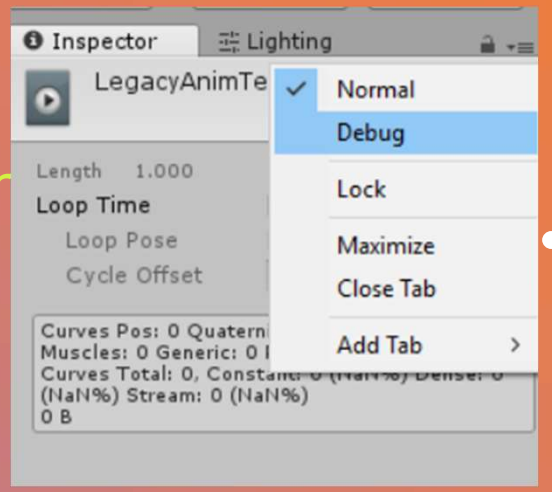


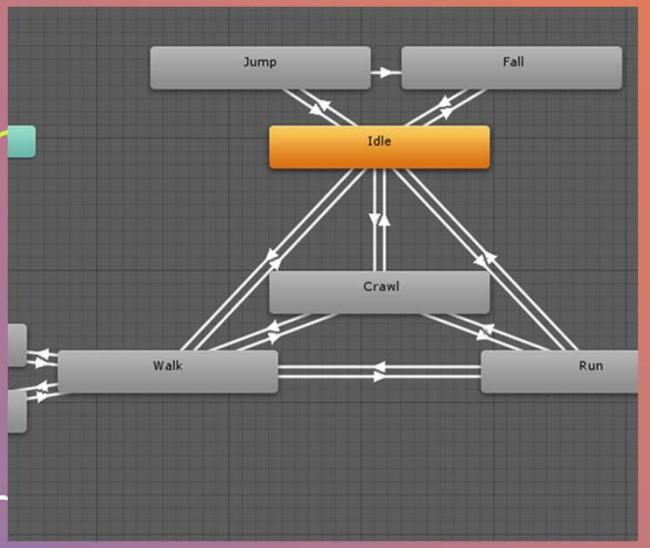


Animation Component

ระบบ Animation ใน Unity มี 2 ระบบ

- Legacy Animation System (Animation Componer
  - ใช้งานร่วมกับไฟล์ Animation (แบบ Legacy)
  - ใช้ง่าย ตรงไปตรงมา
  - อาจจะมีบัคในบางจุด
- Mechanism Animation System (Animator Cor
  - ใช้งานร่วมกับไฟล์ Animation (แบบปกติ) และไฟล์ Animation Controller
  - การทำงานใช้ระบบ State machine ซึ่งจะซับ





Custom Component

- คือ Component ที่เกิดจากการเขียนโ
- เราสามารถเนรมิตอะไรขึ้นมาก็ได้

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

public class Progress_Bar_script : MonoBehaviour
{
    public Image Progress_Bar;
    public ParticleSystem particleBurst;

    [SerializeField] private float currentAmount;
    [SerializeField] private float speed;

    private bool playedBurst = false;

    private void Update ()
    {
        if (!playedBurst) {
            CheckProgress ();
        }
    }

    private void CheckProgress ()
    {
        if (currentAmount < 100) {
            currentAmount += speed * Time.deltaTime;
            Progress_Bar.fillAmount = currentAmount / 100;
        } else {
            particleBurst.Play ();
            playedBurst = true;
        }
    }
}
```

Custom Component

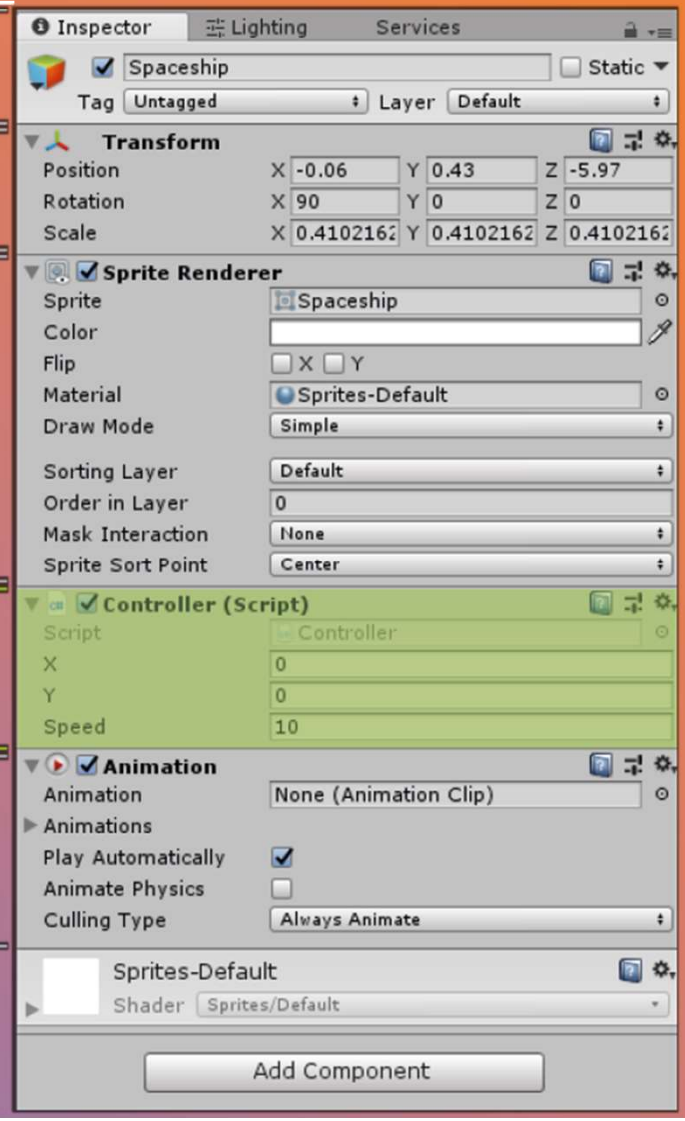
this.gameObject
ปิด/เปิด ทำลาย

this.transform
ใส่ hierarchy

this.GetComponent<SpriteRenderer>()

this

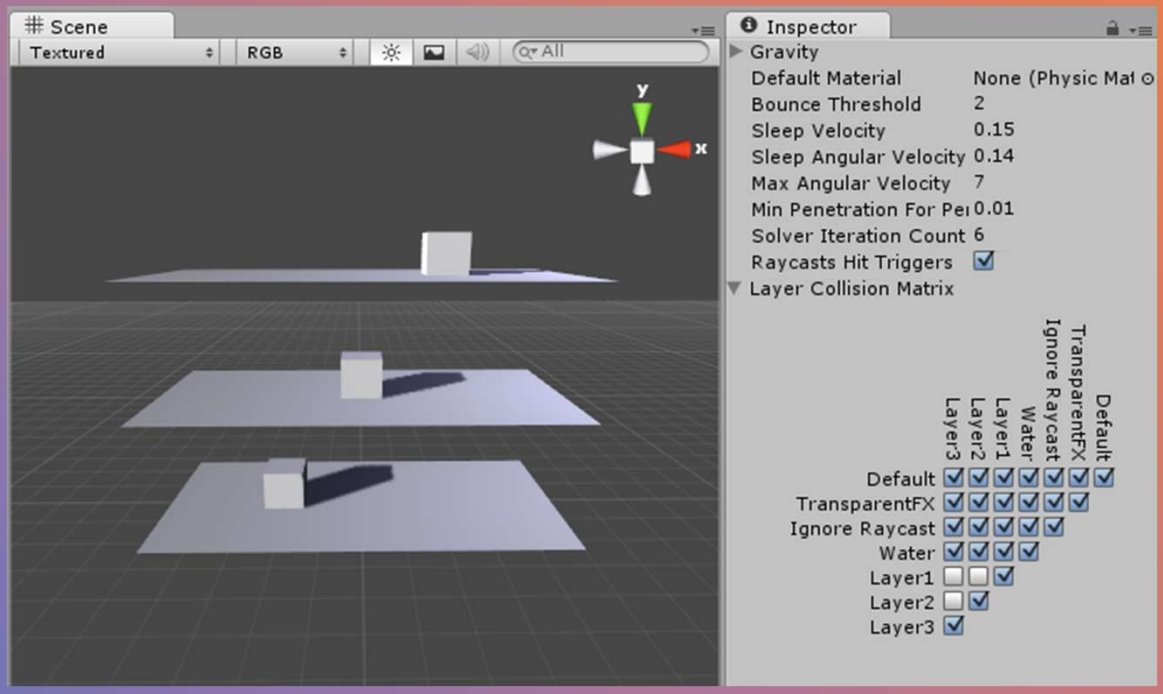
this.GetComponent<Animation>()



Miscellaneous

Physic Layer

- ใช้ควบคุมปฏิสัมพันธ์ของ Rigidbody ในเชิงลึก

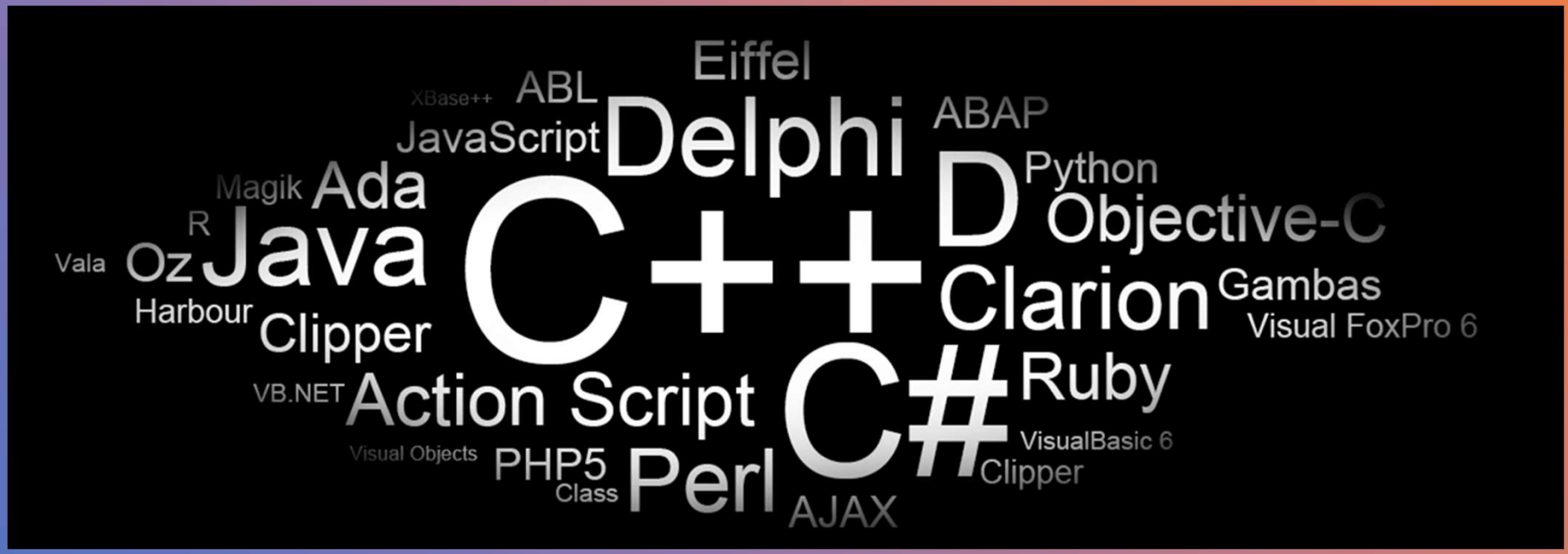


Physic Layer

- Local Space คือ Position / Rotation / Scale เมื่อเทียบกับ Parent Object
  - ปกติค่าแหน่งต่างๆบน Unity จะแสดงผลเป็น Local Space
  - เช่น ที่ Inspector , ที่ Animation Keyframe
- World Space คือ Position /

Coding เริ่มตรงไหนดี?

เลือกภาษา


Eiffel
XBase++
ABL
JavaScript
Delphi
ABAP
Magik
Ada
Python
R
Java
C++
D
Objective-C
Vala
Oz
Clarion
Gambas
Harbour
Clipper
Visual FoxPro 6
VB.NET
Action Script
C#
Ruby
Visual Objects
PHP5
Perl
VisualBasic 6
Class
AJAX
Clipper

โปรแกรมทุกภาษา มีความคล้ายกัน

```
number x = 175.3
text y = "false"
if x > 100 then
  y = "true"
end
else then
  y = "still false"
end
```

(a) Quorum

```
$x = 175.3;
$y = 'false';
if ($x > 100) {
  $y = 'true';
}
else {
  $y = 'still false';
}
```

(b) Perl

```
~ x \ 175.3
?? y \ ?false?
: x ` 100 {
  y \ ?true?
}
, {
  y \ ?still false?
}
```

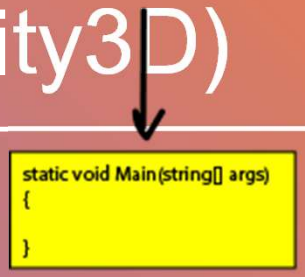
(c) Randomo

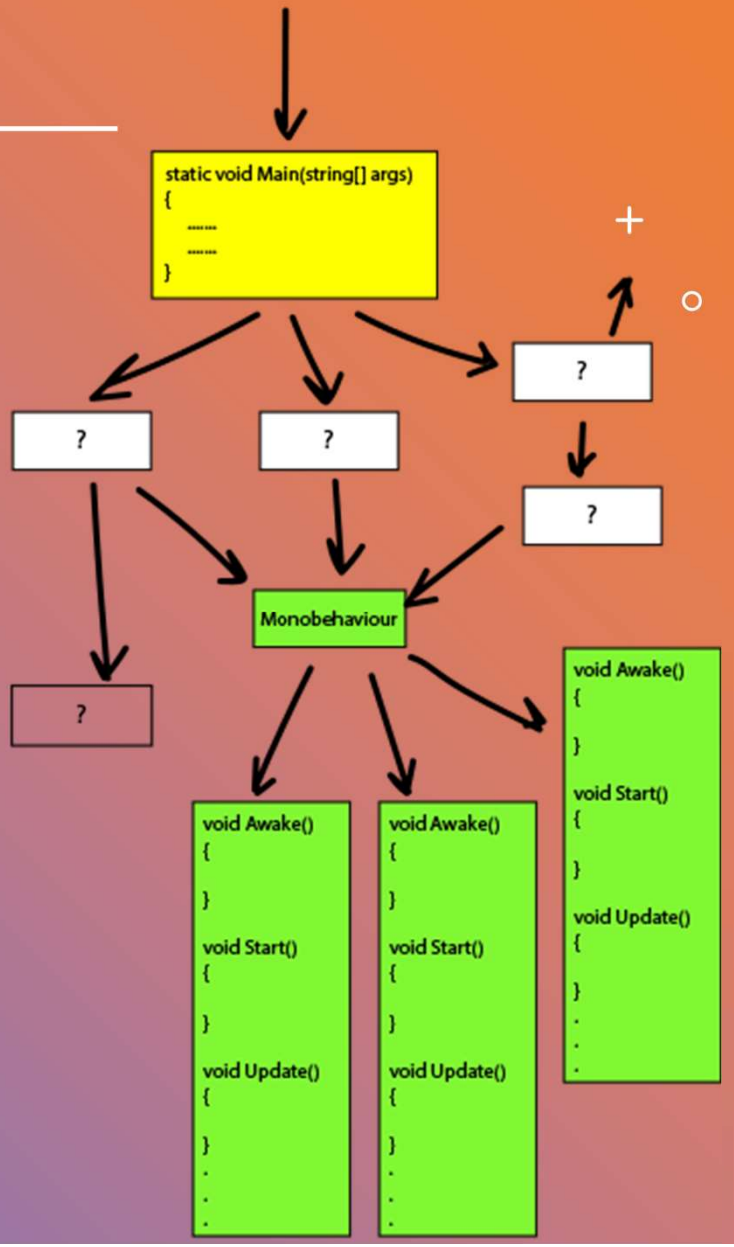
กลไกการทำงานเชิง Logic

- 1. Step by Step
- 2. Choice
- 3. Loop
- 4. Function

Function ที่ควรรู้จัก (Unity3D)

- ~~Main()~~
- Start()
- Update()

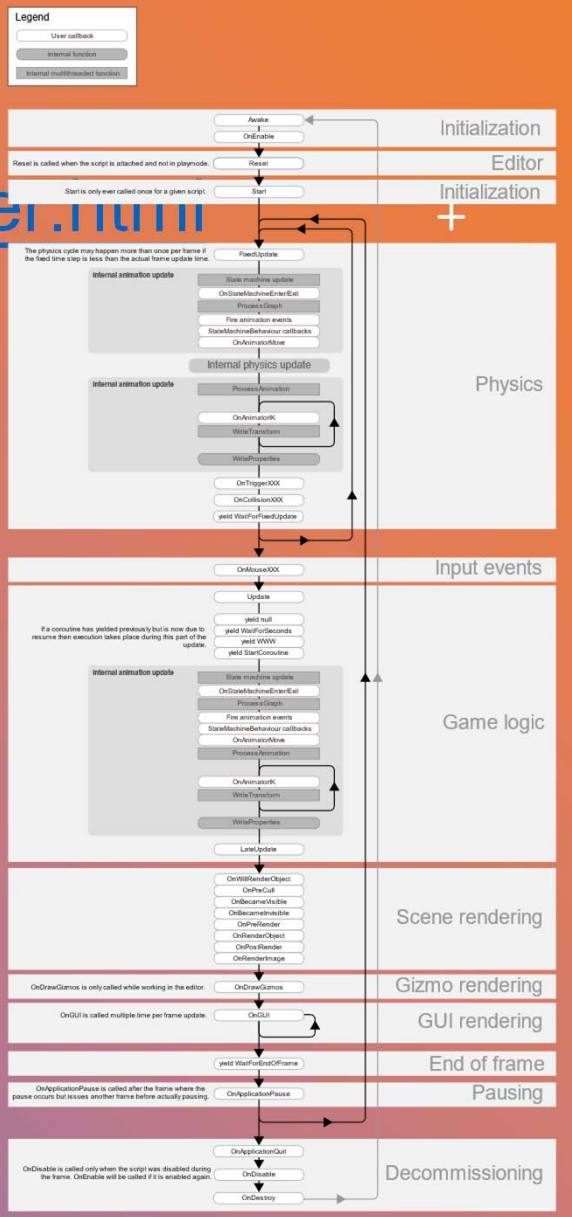




เขียน Logic การตื่นนอนไปทำงาน

ของ Unity

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



Data Types กับ Variables

Data Types

- Integer - จำนวนเต็ม

  -1999 , 0 , -99 , 16 , 300
- Float - จำนวน

  -150.4 , 16.0 , -34.4456 , 9004.55
- String - ตัวอักษร

  "hello" , "Game Over!!" , "400.55" , "yOu TuBE" , "131314"
- Boolean - จริง หรือ เท็จ (ค่าตรรกศาสตร์)

  true , false
- Class , Struct , etc.

  .....

Start() , Update()

```
//Run แค่ตอนเริ่ม
void Start ( )
{
        print("Run at
Start");
}
```

```
//Run ทุกๆ Frame
void Update ( )
{
        print("Run every
frame");
}
```

การควบคุม transform

```
//position
print(this.transform.localPosition);
print(this.transform.position);

//rotation
print(this.transform.localEulerAngles);
print(this.transform.eulerAngles);

//scale
print(this.transform.localScale);
print(this.transform.lossyScale);
```

```
//สร้างตัวแปร Vector
Vector3 vec = new Vector3(0 , 1 , 2);
vec.x = 10;
vec.y = vec.y + 1;
vec.z = vec.z * 10;

print(vec);
```

การควบคุม transform - ตัวอย่าง

```
print(this.transform.localPosition);
```

```
//-----Set ตำแหน่งใหม่------

//สร้างตัวแปร pos เก็บค่า (10,0,0)
Vector3 pos = new Vector3(10 , 0 , 0)
//ปรับค่าตำแหน่งไปที่จุด pos
this.transform.localPosition = pos;
```

```
//-----เลื่อนจากตำแหน่งเดิม-----

//สร้างตัวแปร pos เก็บค่าตำแหน่งเก่า
Vector3 pos = this.transform.localPosition;

//แก้ไขตำแหน่ง x y z ตามต้องการ
pos.x = pos.x + 1;
pos.y = 0;
pos.z = 0;

//ปรับค่าตำแหน่งไปที่จุด pos
this.transform.localPosition = pos;
```

Time.deltaTime

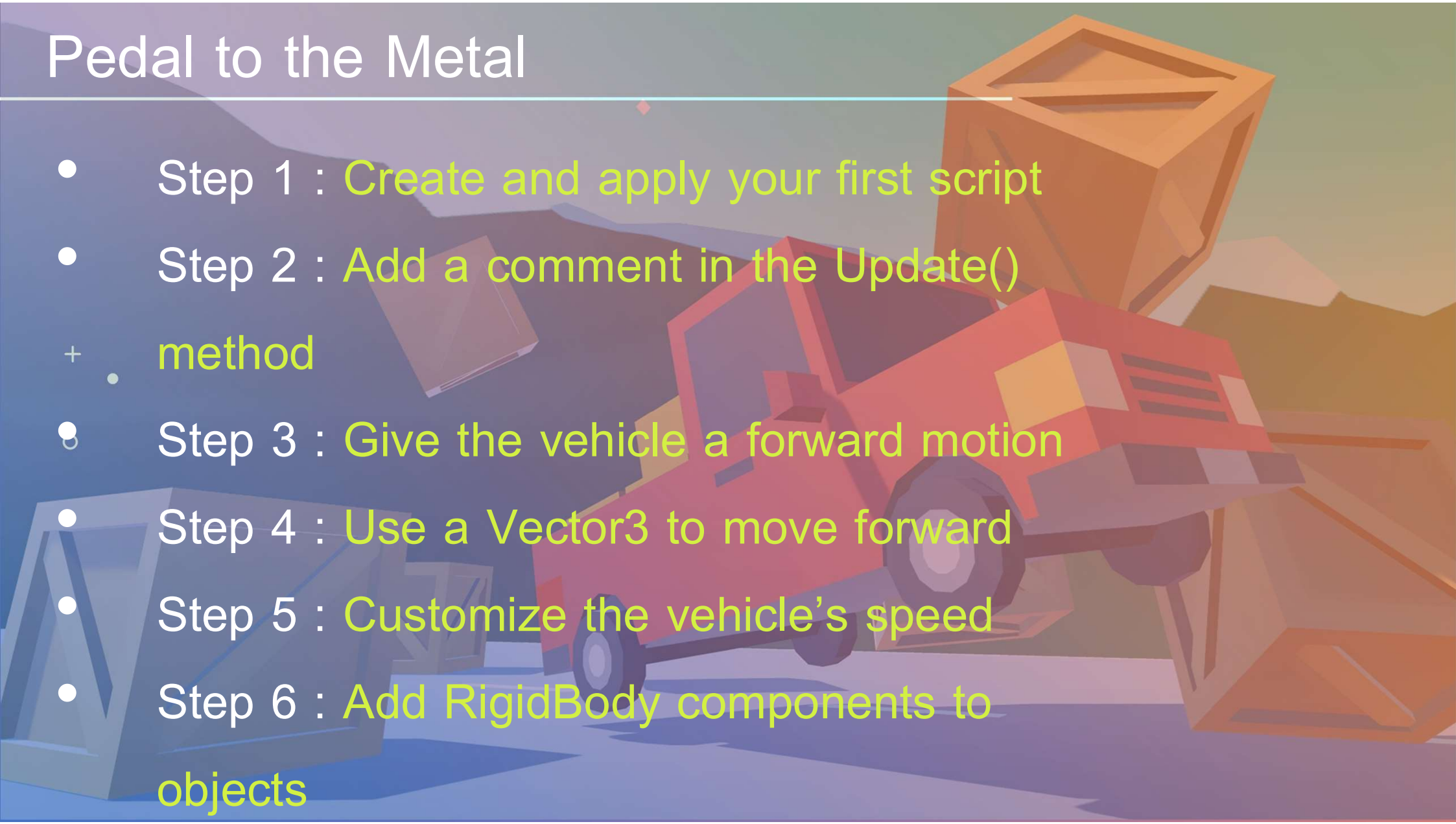
```
void Update()
{
        Vector3 pos = this.transform.localPosition;

        pos.x = pos.x + 1*Time.deltaTime;
        pos.y = 0;
        pos.z = 0;

        this.transform.localPosition = pos;
}
//----60 fps----
//1 frame เคลื่อนที่ 1/60 = 0.0167 หน่วย
//1 วินาที เคลื่อนที่ 0.0167 * 60 = 1 หน่วย
//----30 fps-----
//1 frame เคลื่อนที่ 1/30 = 0.0333 หน่วย
//1 วินาที เคลื่อนที่ 0.0333 * 30 = 1 หน่วย
```

```
void Update()
{
        Vector3 pos = this.transform.localPosition;

        pos.x = pos.x + 1;
        pos.y = 0;
        pos.z = 0;

        this.transform.localPosition = pos;
}
//----60 fps----
//1 frame เคลื่อนที่ 1 หน่วย
//1 วินาที เคลื่อนที่ 60 หน่วย
//----30 fps-----
//1 frame เคลื่อนที่ 1 หน่วย
//1 วินาที เคลื่อนที่ 30 หน่วย
```

Session 1 – Player Control

- Driving Simulator

Pedal to the Metal – ถึงเวลาเคลื่อนที่!



Pedal to the Metal

- Step 1 : Create and apply your first script
- Step 2 : Add a comment in the Update() method
- Step 3 : Give the vehicle a forward motion
- Step 4 : Use a Vector3 to move forward
- Step 5 : Customize the vehicle’s speed
- Step 6 : Add RigidBody components to objects

Step 1 : Create and apply your first script

1 . In the Project window, Right-click > Create > **Folder** named "Scripts"

2 . In the "Scripts" folder, Right-click > Create > **C# Script** named "PlayerController"

3 . **Drag** the new script onto the **Vehicle object**

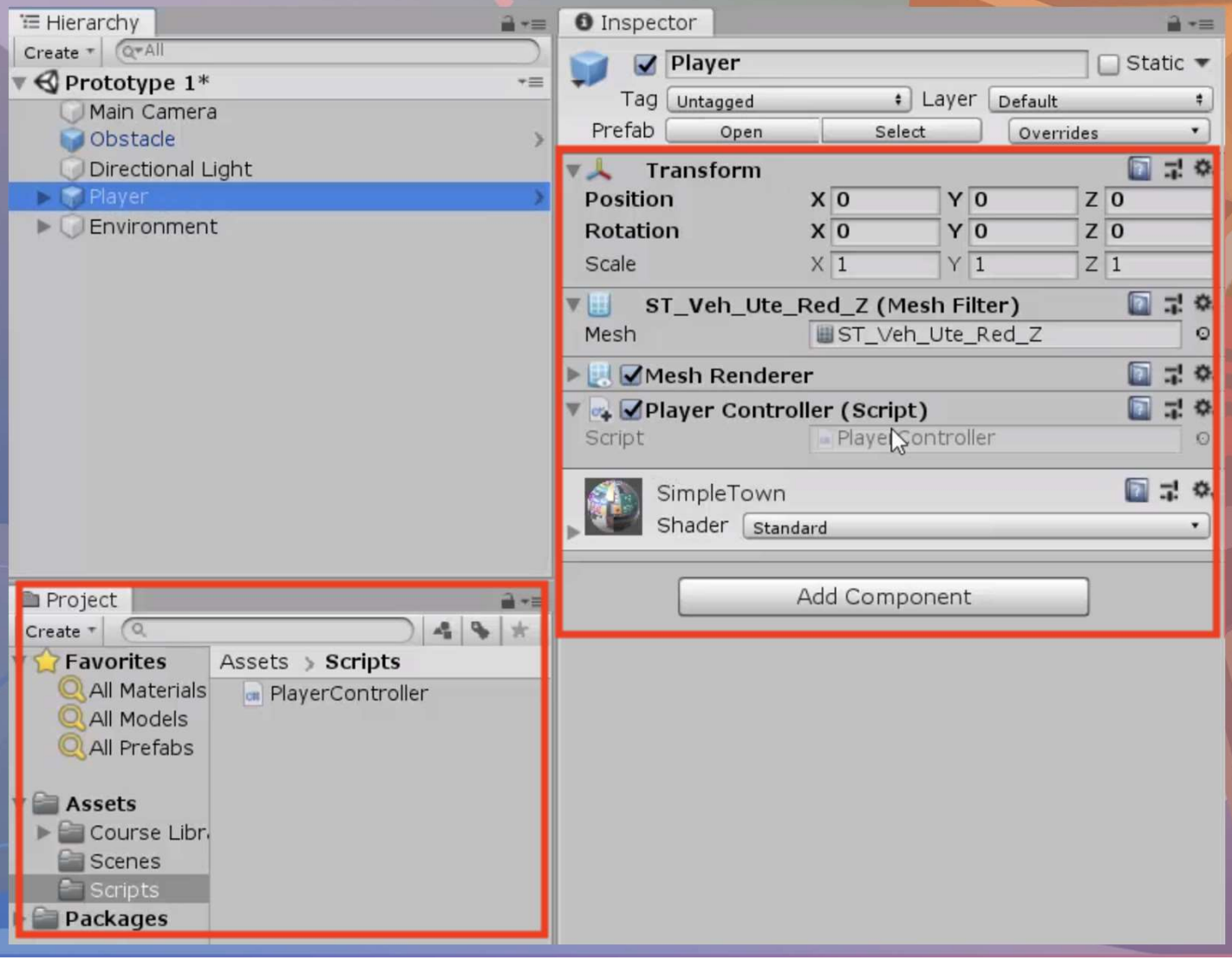
4 . **Click** on the Vehicle object to make sure it was added as a **Component** in the Inspector

New Concept: C# Scripts

Warning: Type the script name as soon as the script is created, since it adds that name to the code. If you want to edit the name, just delete it and make a new scripts

New Concept: Components

Step 1 : Create and apply your first script



method

1. **Double-click** on the script to open it in **Visual Studio**
2. In the **Update()** method, add a comment that you will: **// Move the vehicle forward**

```
void Update()
{
// Move the vehicle forward
}
```

- **New :** Start vs Update functions
- New : Comments

Step 3 : Give the vehicle a forward motion

- **New :** Start vs Update functions
- New : Comments

1. Under your new comment, type **transform.tr**, then select **Translate** from the autocomplete menu
2. Type (, add 0, 0, 1 between the parentheses, and
3.

```
void Update()
{
// Move the vehicle forward
transform.Translate(0, 0, 1);
}
```

Step 4 : Use a Vector3 to move forward

1. Delete the 0, 0, 1 you typed and use auto-complete to replace it with Vector3.forward

```
void Update()
{
// Move the vehicle forward
transform.Translate(0, 0, 1 Vector3.forward);
}
```

- **New Concept**: Documentation
- **New Concept**: Vector3
- **Warning**: Make sure to save time an use Autocomplete! Start typing and V Code will display a popup menu with recommended code.

Step 5 : Customize the vehicle's speed

1. Add ***Time.deltaTime** and run your game
2. Add ***20** and run your game

-**New Concept**: Math symbols in C#

-**New Function**: Time.deltaTime

```
void Update()
{
// Move the vehicle forward
transform.Translate(Vector3.forward * Time.deltaTime * 20);
}
```

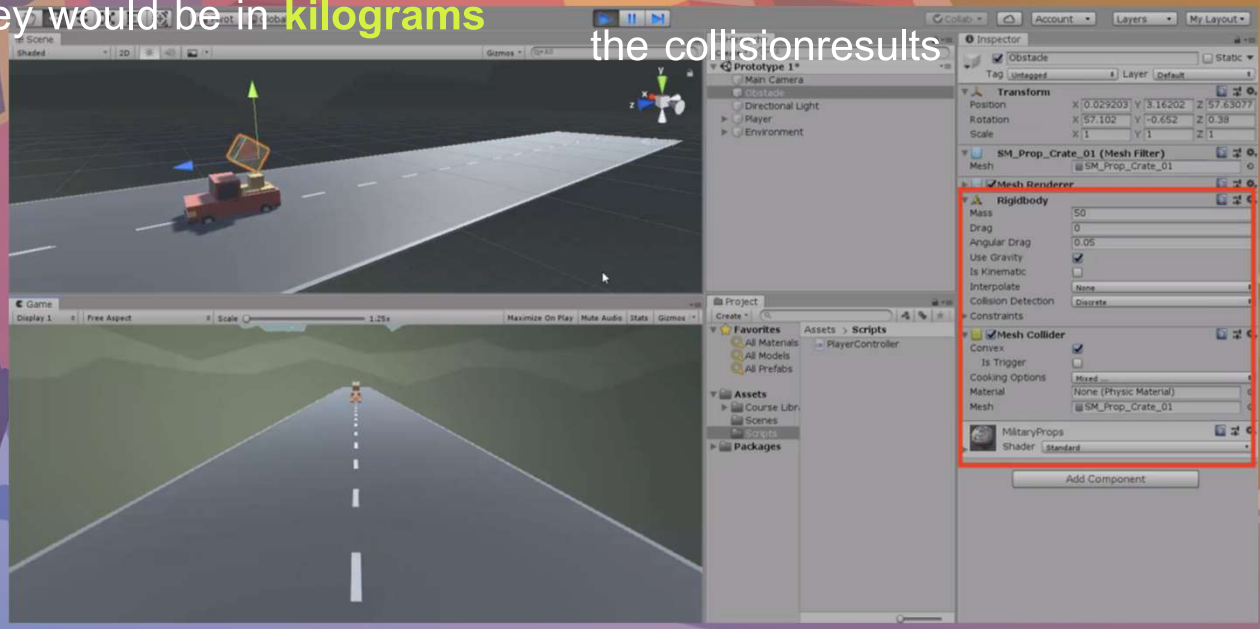
objects

1. Select the **Vehicle**, then in the hierarchy click **Add Component** and select **RigidBody**
2. Select the **Obstacle**, then in the hierarchy click **Add Component** and select **RigidBody**
3. In the RigidBody component properties, increase the **mass** of vehicle and obstacle to be about what they would be in **kilograms** and test again

-New Concept: Rigidbody Component

-New Concept: Collider Component

-Tip: Adjust the mass of the vehicle and the obstacle, and test the collisionresults



obstacles

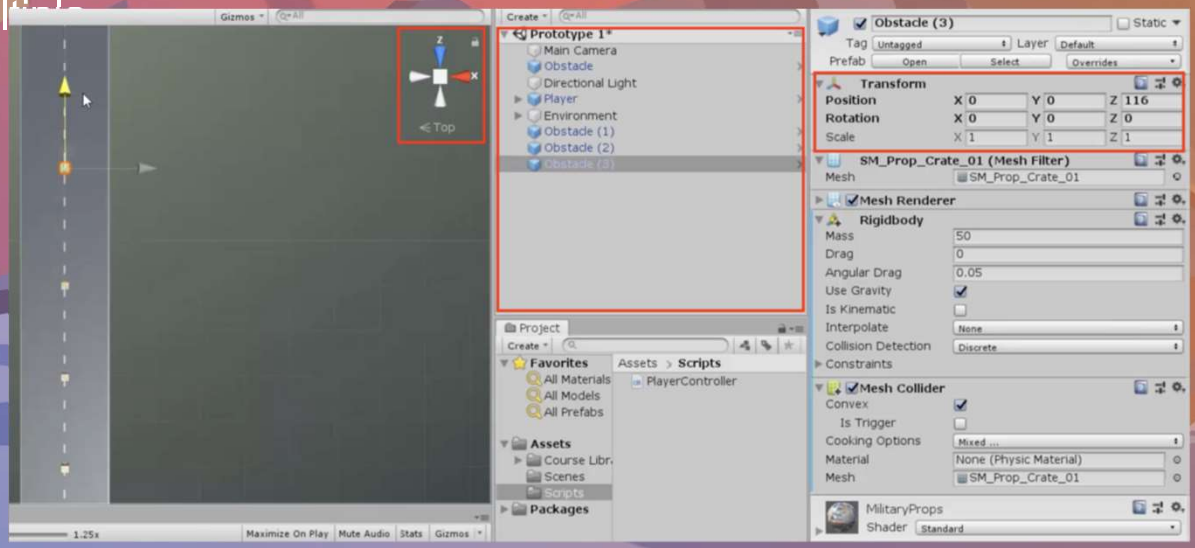
1. Click and drag your obstacle to the **bottom of the list** in the hierarchy
2. Press **Ctrl/Cmd+D** to duplicate the obstacle and move it down the Z axis
3. Repeat this a few more times to create more obstacles
4. After making a few duplicates, select one in the hierarchy and **hold ctrl + click** to select multiple obstacles, then **duplicate** those

-New Technique:

Duplicate(Ctrl/Cmd+D)

-Tip: Try using top-down view to make this easier

-Tip: Try using the inspector to space your obstacles exactly 25 apart



High Speed Chase – มาเร่งเครื่องกัน!

High Speed Chase

- Step 1 : Add a speed variable for your vehicle
- Step 2 : Create a new script for the camera
- Step 3 : Add an offset to the camera position
- Step 4 : Make the offset into a Vector3 variable
- Step 5 : Smooth the Camera with

vehicle

1.In PlayerController.cs, add **public float speed = 5.0f;** at the top of the **class**

2.Replace the **speed value** in the Translate method with the **speed variable**, then test

3.**Save** the script, then edit the speed value in the **inspector** to get the speed you want

-New Concept: Floats and Integers

-New Concept: Assigning Variables

-New Concept: Access Modifiers

```
public float speed = 20;
void Update()
{
transform.Translate(Vector3.forward * Time.deltaTime * 20 speed);
}
```

camera

1. Create a new **C# script** called FollowPlayer and attach it to the **camera**
2. Add **public GameObject player;** to the top of the script
3. Select the **Main Camera, then, drag** the player object onto the **empty player variable** in the Inspector
4. In **Update(),** assign the camera's position to the player's

```
public GameObject player;
void Update()
{
transform.position = player.transform.position;
}
```

-**Warning:** Remember to capitalize your script name correctly and rename it as soon as the script is created!

-**Warning**: It's really easy to forget to assign the player variable in the inspector

-**Don't worry:** The camera will be under the car... weird! We will fix that soon

position

1. In the line in the Update method add + new Vector3(0, 5, -7), then test

-**New Concept:** Vector3 in place of coordinates

-Tip: You need “new Vector3()” because 3 numbers in a row could mean anything

-**New Concept:** FixedUpdate

-**Warning:** Remember to update your comments and maintain their accuracy!

```
public GameObject player;
void Update()
{
        transform.position = player.transform.position + new Vector3(0, 5, -7);
}
```

variable

1. At the top of **FollowPlayer.cs**, declare **private Vector3 offset;**
2. Copy the **new Vector3()** code and **assign** it to that variable
3. **Replace** the original code with the **offset** variable
4. **Test** and **save**

-Don't worry: Pay no mind to the read only warning

Tip: Whenever possible, make variables!

You never want hard values in the middle of your code

```
public GameObject player;
private Vector3 offset = new Vector3(0, 5, -7);
void Update()
{
transform.position = player.transform.position + new Vector3(0, 5, -7) offset;
}
```

LateUpdate

1. Test your prototype to notice the jittering camera as the vehicle drives.
2. In **FollowPlayer.cs**, replace Update() with LateUpdate().
3. Save and test to see if the camera is less jittery.

- **New Concept**: LateUpdate is called after the Update method, which allows to more smoothly follow the player.

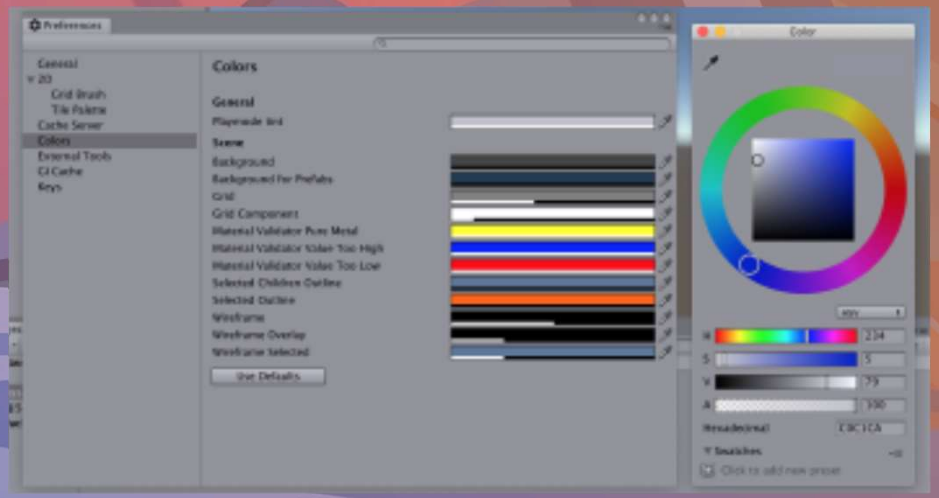
```
void LateUpdate()
{
transform.position = player.transform.position + offset;
}
```

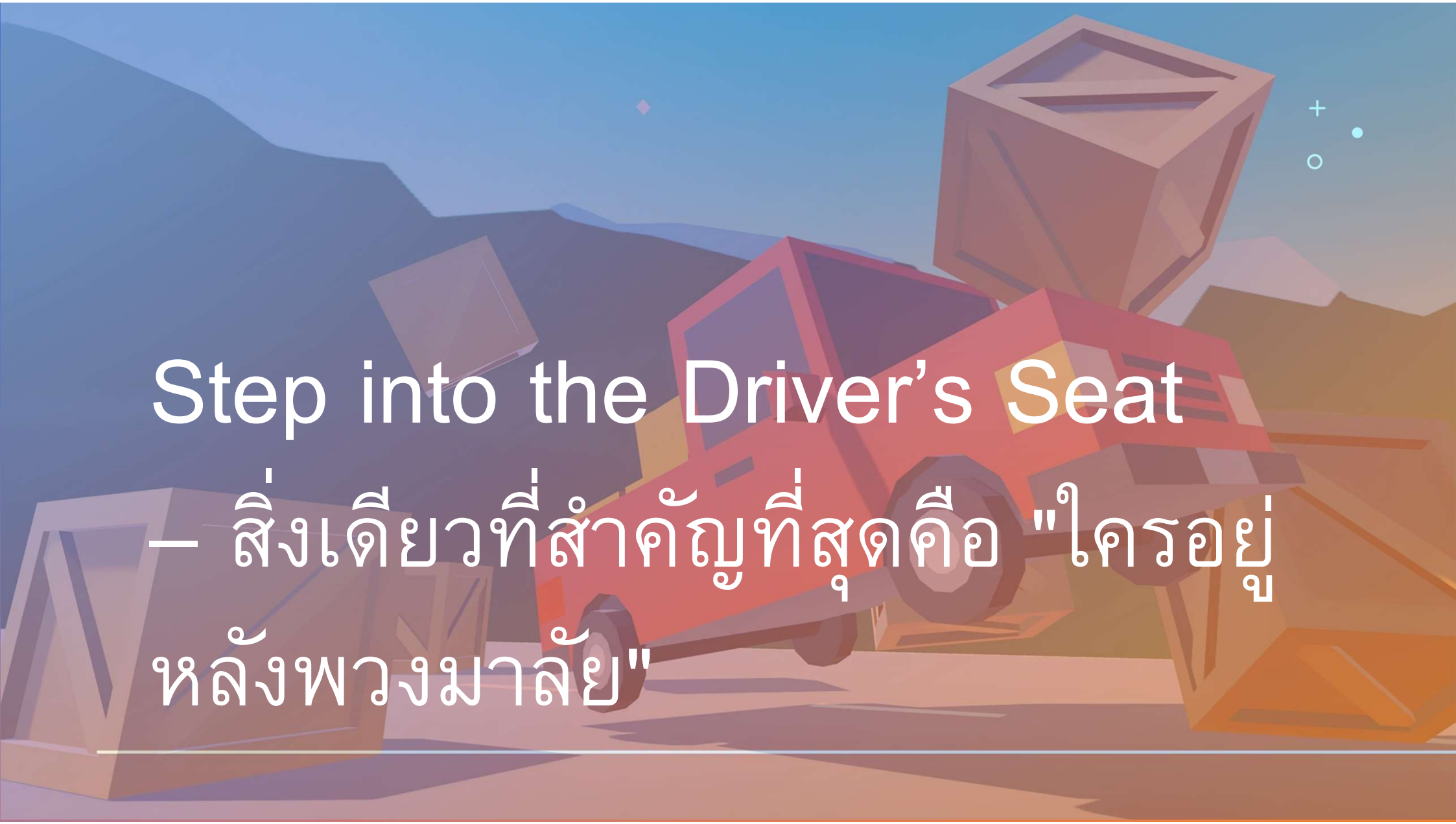
Step 6 : Edit the playmode tint color

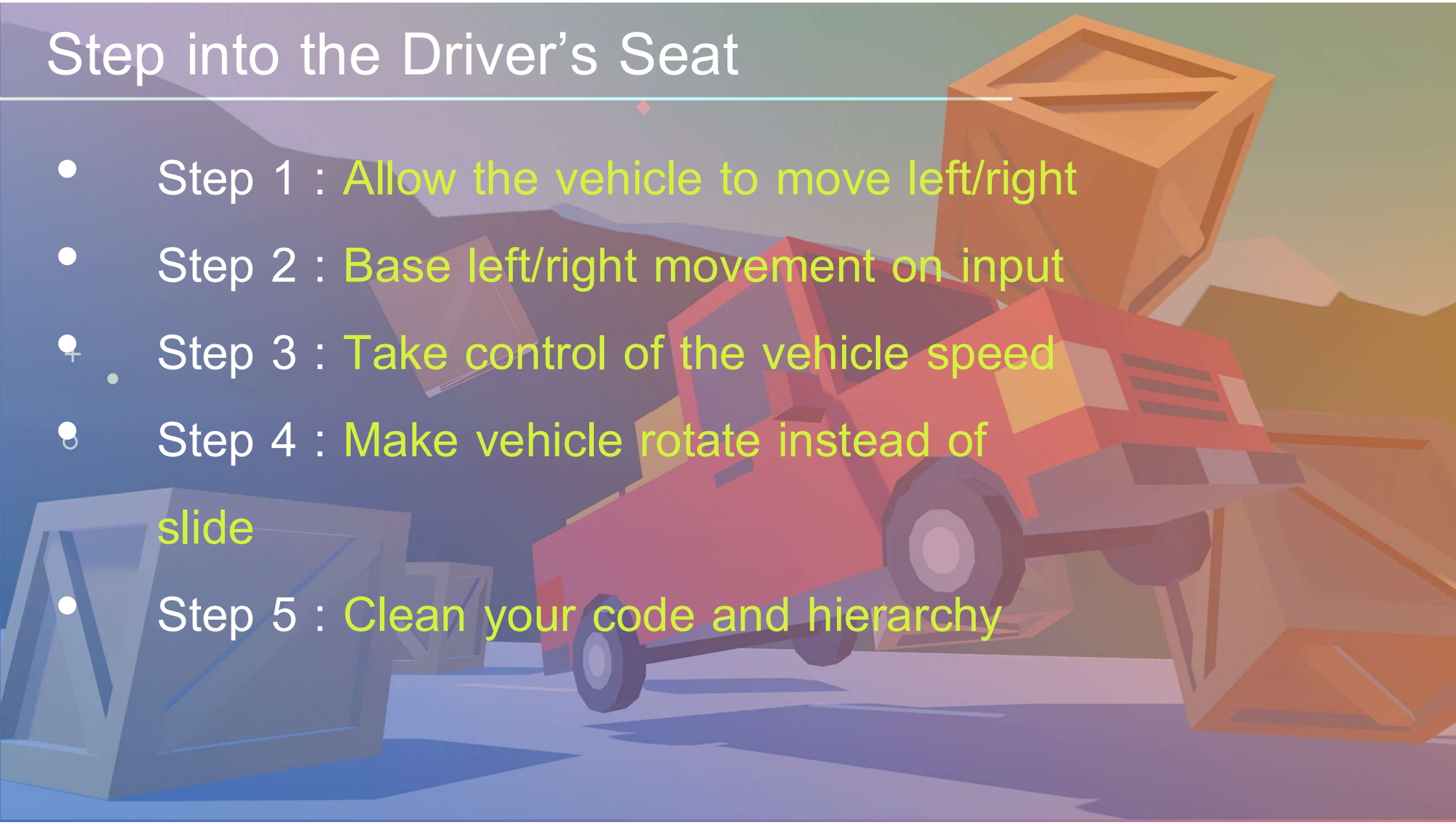
1. From the top menu, go to Edit > Preferences **(Windows)** or Unity > Preferences **(Mac)**
2. In the left menu, choose **Colors,** then edit the **“Playmode tint”** color to have a slight color
3. **Play** your project to test it, then close your preferences

-**Tip**: Try editing a variable in play mode, then stopping - it will revert

-**Warning**: Don’t go crazy with the colors or it will be distracting




Step into the Driver’s Seat
– สิ่งเดียวที่สำคัญที่สุดคือ "ใครอยู่
หลังพวงมาลัย"



Step into the Driver's Seat

- Step 1 : Allow the vehicle to move left/right
- Step 2 : Base left/right movement on input
- Step 3 : Take control of the vehicle speed
- Step 4 : Make vehicle rotate instead of slide
- Step 5 : Clean your code and hierarchy

Step 1 : Allow the vehicle to move left/right

- New Function: Vector3.righ

1. At the top of PlayerController.cs, add a **public float turnSpeed;** variable
2. In **Update(),** add **transform.Translate(Vector3.right *Time.deltaTime * turnSpeed);**
3. Run your game and use the turnSpeed variable slider to move the vehicle left

```
public float turnSpeed;

void Update()
{
        transform.Translate(Vector3.forward * Time.deltaTime * speed);
        transform.Translate(Vector3.right * Time.deltaTime * turnSpeed);
}
```

Step 2 : Base left/right movement on input

1. From the top menu, click Edit > Project Settings, select Input Manager in the left sidebar, then expand the Axes fold-out to explore the inputs.
2. In PlayerController.cs, add a new public float horizontalInput variable
3. In Update, assign horizontalInput = Input.GetAxis("Horizontal");, then test to see it in inspector
4. Add the horizontalInput variable to your left/right Translate method to gain control of the vehicle
5. [illegible]

- **New:** Input.GetAxis

-Tip: Edit > ProjectSettings > Input and expand the Horizontal Axis to show everything about it

-Warning: Spelling is important in string parameters. Make sure you spell and capitalize “Horizontal” correctly!

```
public float horizontalInput;
void Update()
{
        horizontalInput = Input.GetAxis("Horizontal");
        transform.Translate(Vector3.forward * Time.deltaTime * speed);
        transform.Translate(Vector3.right * Time.deltaTime * turnSpeed * horizontalInput);
}
```

Step 3 : Take control of the vehicle speed

1. Declare a new public forwardInput variable
2. In Update, assign forwardInput =Input.GetAxis("Vertical");
3. Add the forwardInput variable to the forward Translate method, then test

-**Tip:** It can go backwards, too!

-**Warning:** This is slightly confusing with forward Input and vertical axis

```
public float horizontalInput;
public float forwardInput;
void Update()
{
        horizontalInput = Input.GetAxis("Horizontal");
        forwardInput = Input.GetAxis("Vertical");
        transform.Translate(Vector3.forward * Time.deltaTime * speed * forwardInput);
        transform.Translate(Vector3.right * Time.deltaTime * turnSpeed * horizontalInput);
}
```

slide

1. In Update, call transform.Rotate(Vector3.up, horizontalInput), then test
2. Delete the line of code that translates Right, then test
3. Add * turnSpeed * Time.deltaTime, then test

-New: transform.Rotate

-Tip: You can always trust the official Unity scripting API documentation

```
void Update()
{
        horizontalInput = Input.GetAxis("Horizontal");
        forwardInput = Input.GetAxis("Vertical");
        transform.Translate(Vector3.forward * Time.deltaTime * speed * forwardInput);
        transform.Rotate(Vector3.up, turnSpeed * horizontalInput * Time.deltaTime);
    transform.Translate(Vector3.right * Time.deltaTime * turnSpeed * horizontalInput);
}
```

Step 5 : Clean your code and hierarchy

1. In the hierarchy, Right-click > Create Empty and rename it “Obstacles”, then **drag** all the obstacles into it
2. Initialize variables with values in PlayerController, then make all variables private (except for the player variables)
3. Use // to add comments to each section of code

-New: Empty Object

-Tip: You don’t actually need to type “private”, it defaults to that

-Tip: Comments are important, especially for your future self

```
public private float speed = 20.0f;
public private float turnSpeed = 45.0f;
public private float horizontalInput;
public private float forwardInput;

void Update() {
        horizontalInput = Input.GetAxis("Horizontal");
        forwardInput = Input.GetAxis("Vertical");
        // Moves the car forward based on vertical input
        transform.Translate(Vector3.forward * Time.deltaTime * speed * forwardInput);
        // Rotates the car based on horizontal input
        transform.Rotate(Vector3.up, turnSpeed * horizontalInput * Time.deltaTime);
}
```

Homework - Plane Programming

around obstacles in the sky. You will have to get the user's input from the up and down arrows in order to control the plane's pitch up and down. You will also have to make the camera follow alongside the plane so you can keep it in view.

Outcome:

- The plane moves forward at a constant rate
- The up/down arrows tilt the nose of the plane up and down
- The camera follows along beside the plane as it flies:

Objectives:

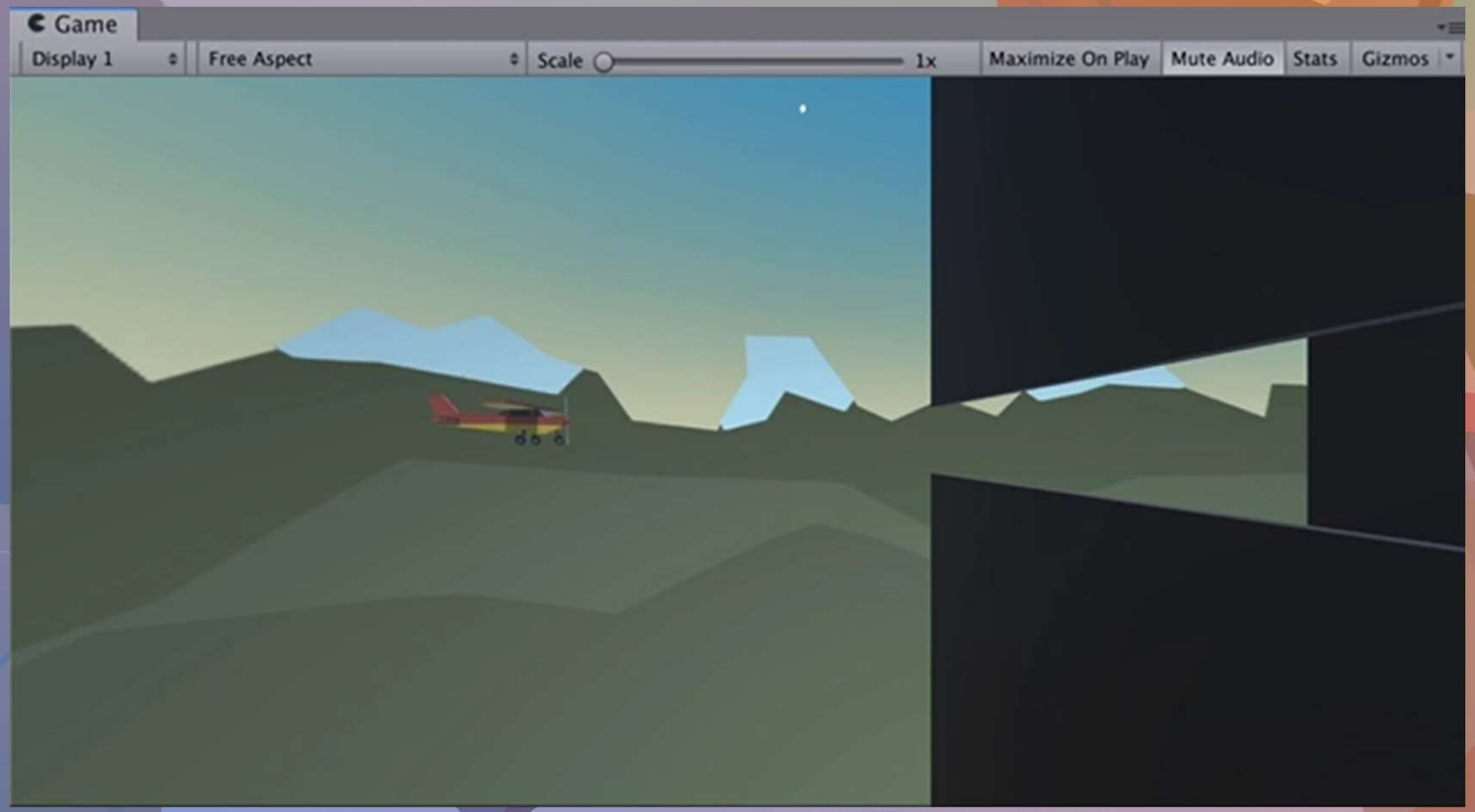
- Using the Vector3 class to move and rotate objects along/around an axis
- Using Time.deltaTime in the Update() method to move objects properly
- Moving and rotating objects in scene view to position them the way you want
- Assigning variables in the inspector and initializing them in code
- Implementing Input variables to control the movement/rotation of objects based on User input

Homework - Plane Programming

Challenge		Task	Hint
1	The plane is going backwards	Make the plane go forward	`Vector3.back` makes an object move backwards, `Vector3.forward` makes it go forwards
2	The plane is going too fast	Slow the plane down to a manageable speed	If you multiply a value by `Time.deltaTime`, it will change it from 1x/frame to 1x/second
3	The plane is tilting automatically	Make the plane tilt only if the user presses the up/down arrows	In PlayerControllerX.cs, in Update(), the `verticalInput` value is assigned, but it's never actually used in the `Rotate()` call
4	The camera is *in front of* the plane	Reposition it so it's beside the plane	For the camera's position, try `X=30, Y=0, Z=10` and for the camera's rotation, try `X=0, Y=-90, Z=0`
5	The camera is not following the plane	Make the camera follow the plane	In FollowPlayerX.cs, neither the plane nor offset variables are assigned a value - assign the `plane` variable in the camera's inspector and assign the `offset = new Vector3(30, 0, 10)` in the code

Bonus Challenge		Task	Hint
X	The plane's propeller does not spin	Create a script that spins the plane's propeller	There is a "Propeller" child object of the plane - you should create a new "SpinPropellerX.cs" script and make it rotate every frame around the Z axis.

Homework - Plane Programming



Gameplay