UNIT2

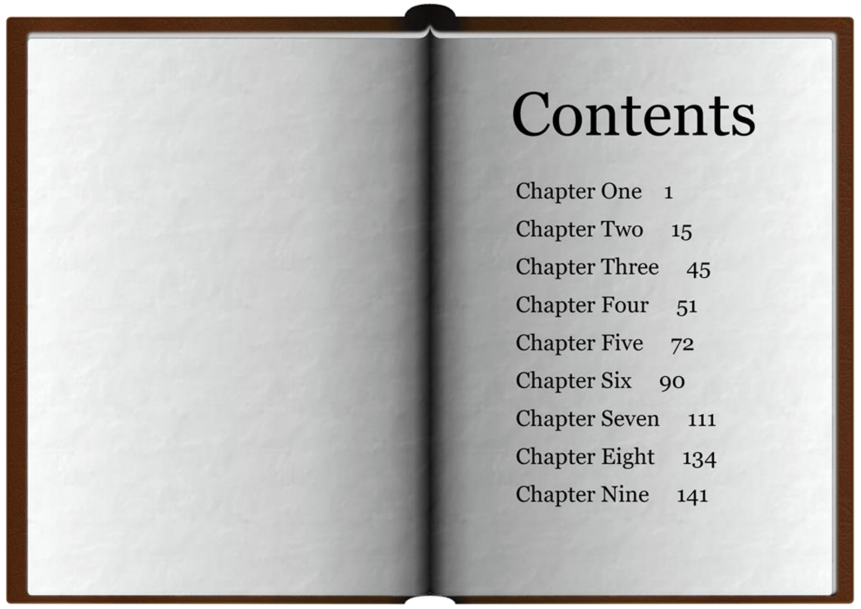
PLAYER CONTROL

PMDM - 2DAM

Àngel Olmos (a.olmosginer@edu.gva.es)

Jose Pascual Rocher (jp.rochercamps@edu.gva.es)

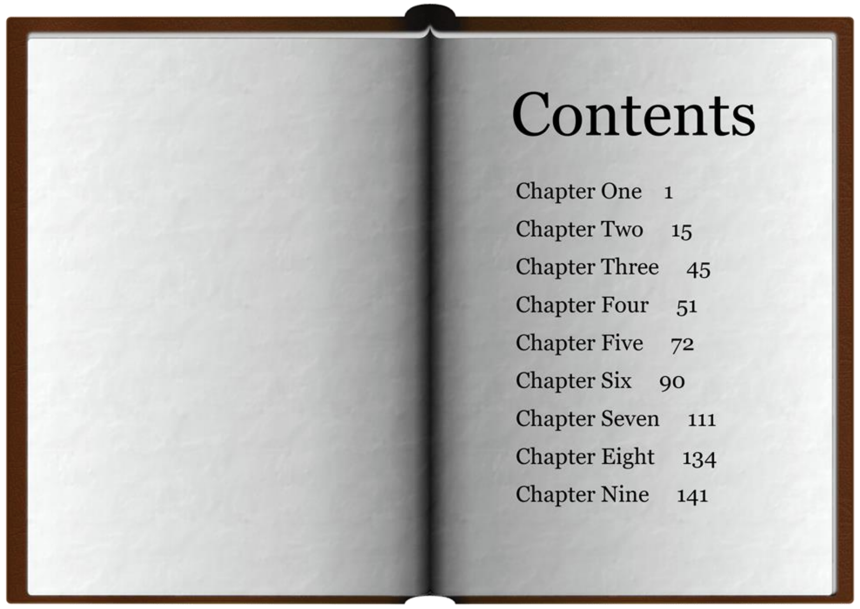
Content



Contents

Chapter One	1
Chapter Two	15
Chapter Three	45
Chapter Four	51
Chapter Five	72
Chapter Six	90
Chapter Seven	111
Chapter Eight	134
Chapter Nine	141

Content

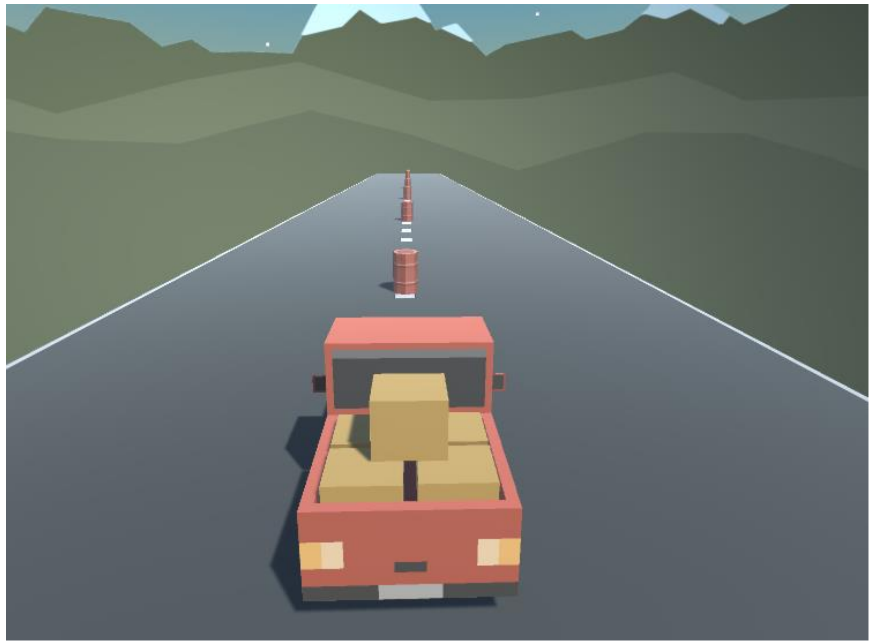


Contents

Chapter One 1
Chapter Two 15
Chapter Three 45
Chapter Four 51
Chapter Five 72
Chapter Six 90
Chapter Seven 111
Chapter Eight 134
Chapter Nine 141

Introduction

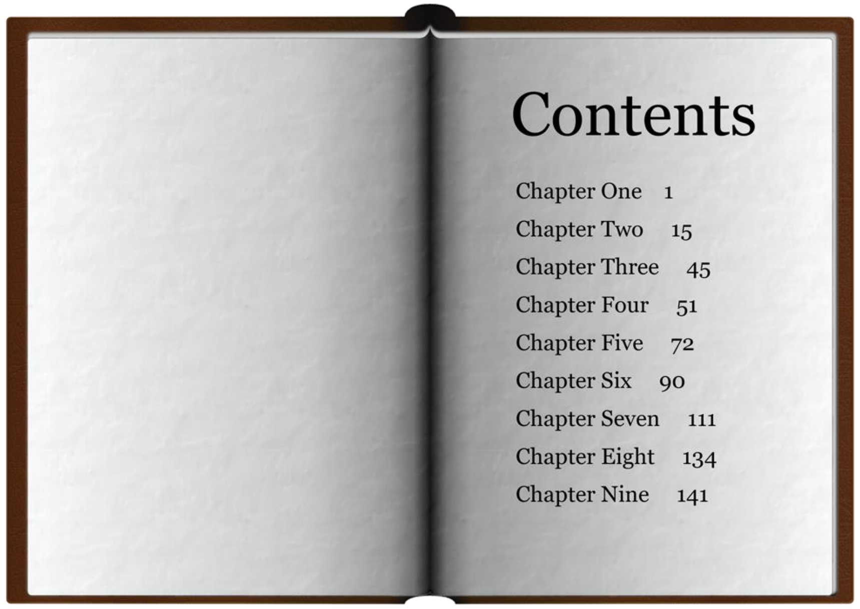
- In this lesson we will introduce the Unity **scripting with C#**
- From an already prepared scene, you will do modifications to it in order to control the player
- On the other hand, you will continue practicing:
  - the scene navigation
  - objects positioning
  - camera placement
  - components physic properties ...



Introduction

Video

Content

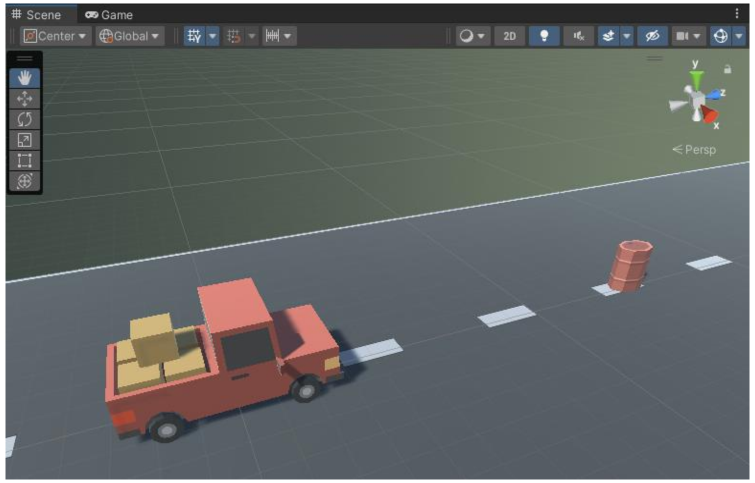


Contents

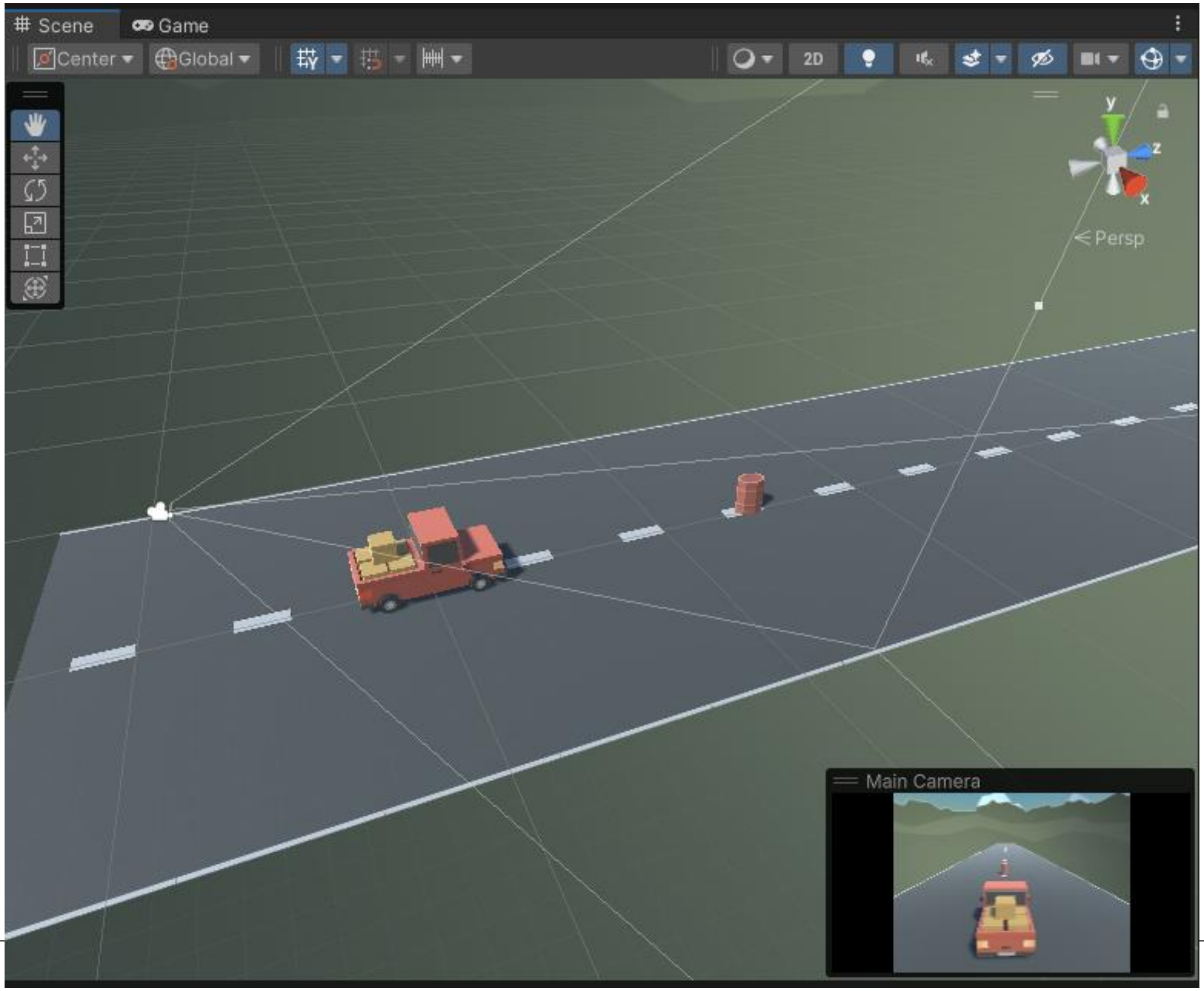
Chapter One	1
Chapter Two	15
Chapter Three	45
Chapter Four	51
Chapter Five	72
Chapter Six	90
Chapter Seven	111
Chapter Eight	134
Chapter Nine	141

Build the scene

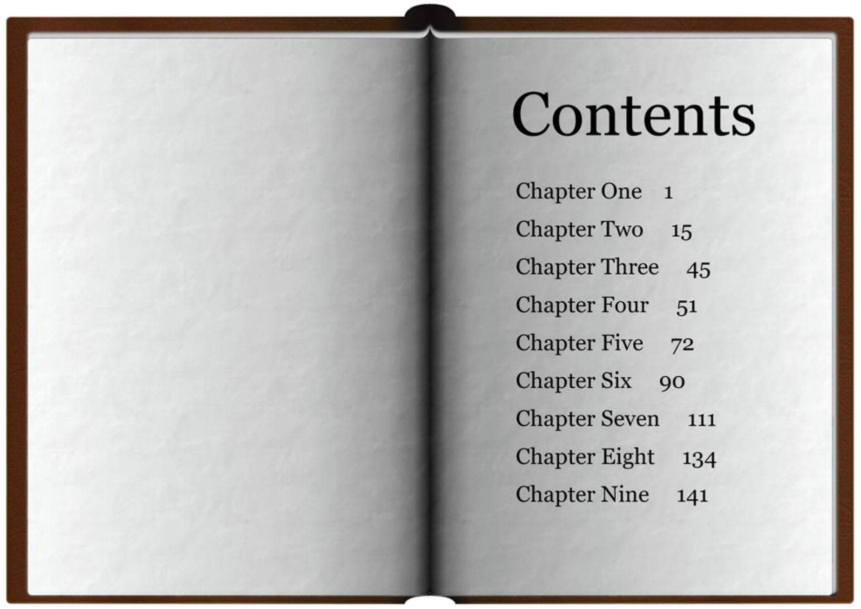
1) First thing to do is **import the assets** to a new empty project
   a) Create a project from a 3D template
   b) Import the assets from *Car.unitypackage* (menu Assets > Import Package > Custom Package)
2) Now **add a vehicle and an obstacle** to the scene. Find them in:
   a) folder **Assets > Course Library > Vehicles**
   b) folder **Assets > Course Library > Obstacles**
3) and **rename** them as "Vehicle" and "Obstacle"
4) **Move the camera** behind/above the vehicle



Build the scene



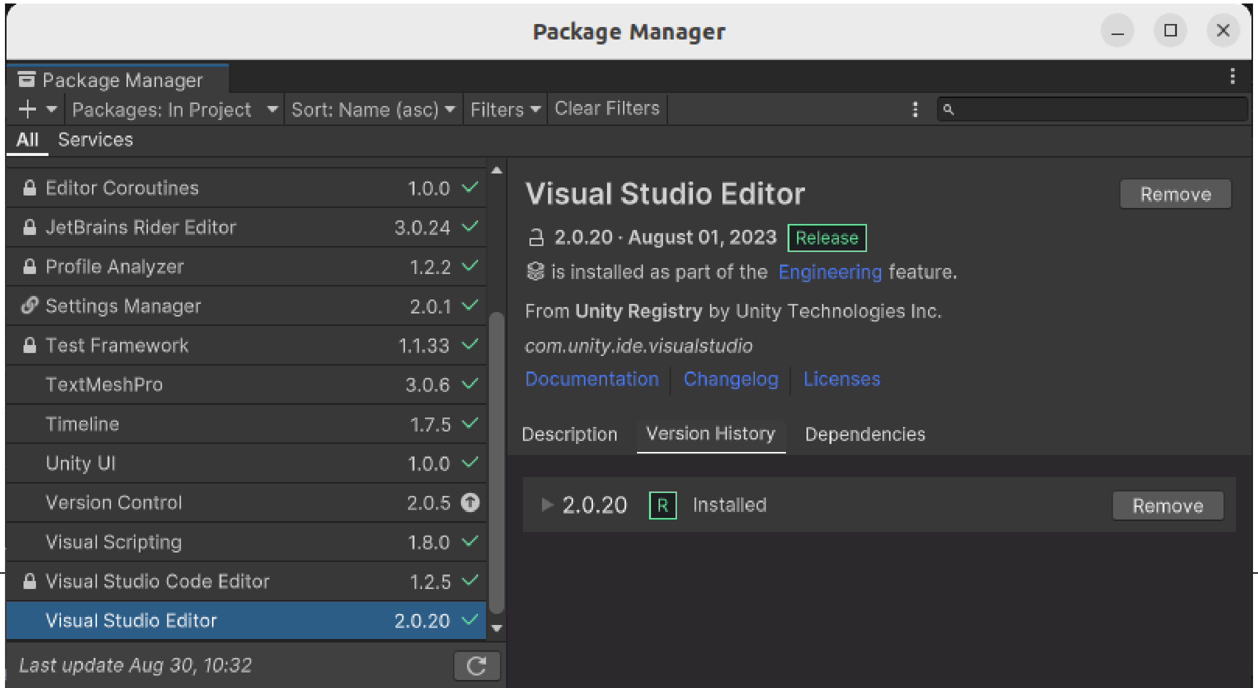
Content



Contents
Chapter One 1
Chapter Two 15
Chapter Three 45
Chapter Four 51
Chapter Five 72
Chapter Six 90
Chapter Seven 111
Chapter Eight 134
Chapter Nine 141

Create your first C# script

- Now we'll create at C# script to move the vehicle
- The scripting will be done in Visual Studio Code IDE
- One has to check that the project has the **Visual Studio Editor package** and the correct **version 2.0.20**
- If you have an older version, unlock and upgrade the package



Editor Coroutines	1.0.0 ✓
JetBrains Rider Editor	3.0.24 ✓
Profile Analyzer	1.2.2 ✓
Settings Manager	2.0.1 ✓
Test Framework	1.1.33 ✓
TextMeshPro	3.0.6 ✓
Timeline	1.7.5 ✓
Unity UI	1.0.0 ✓
Version Control	2.0.5
Visual Scripting	1.8.0 ✓
Visual Studio Code Editor	1.2.5 ✓
Visual Studio Editor	2.0.20 ✓

Last update Aug 30, 10:32

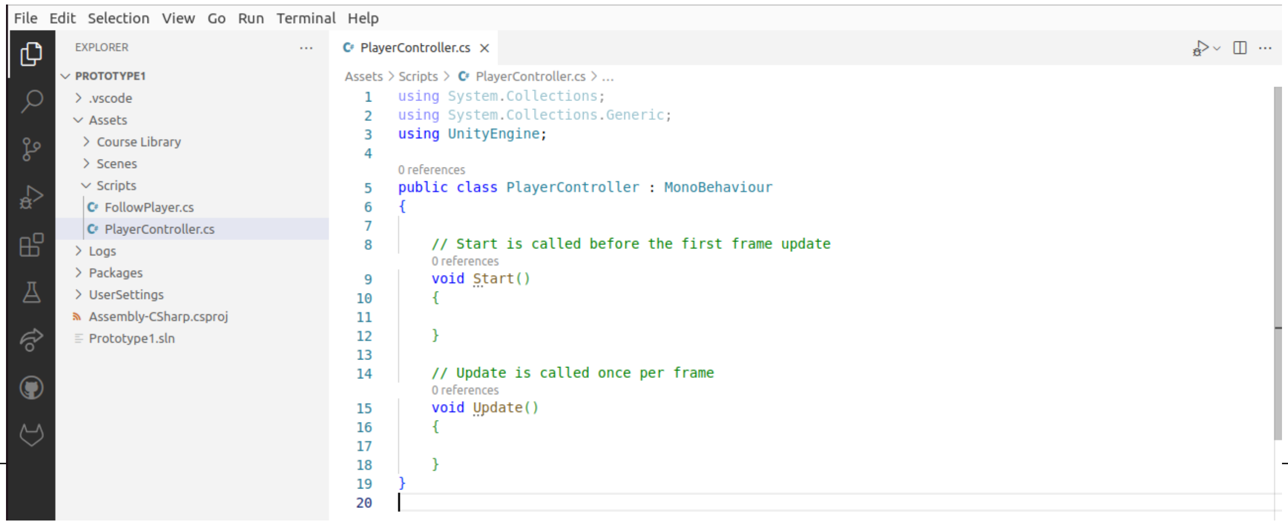
Visual Studio Editor
Remove
2.0.20 · August 01, 2023 Release
is installed as part of the Engineering feature.
From **Unity Registry** by Unity Technologies Inc.
com.unity.ide.visualstudio
Documentation | Changelog | Licenses

Description | Version History | Dependencies

2.0.20 R Installed Remove

Create your first C# script

1. Create a ***Scripts* folder** inside the *Assets* folder and **add a C# script** (Right-click > Create > C# Script)
2. Rename it as "**PlayerController**" and drag it to the Vehicle object
3. Double-click on the script to open the script in VSC
4. If VSC doesn't open, you'll have to set VSC as your default IDE (Edit > Preferences > External Tool)



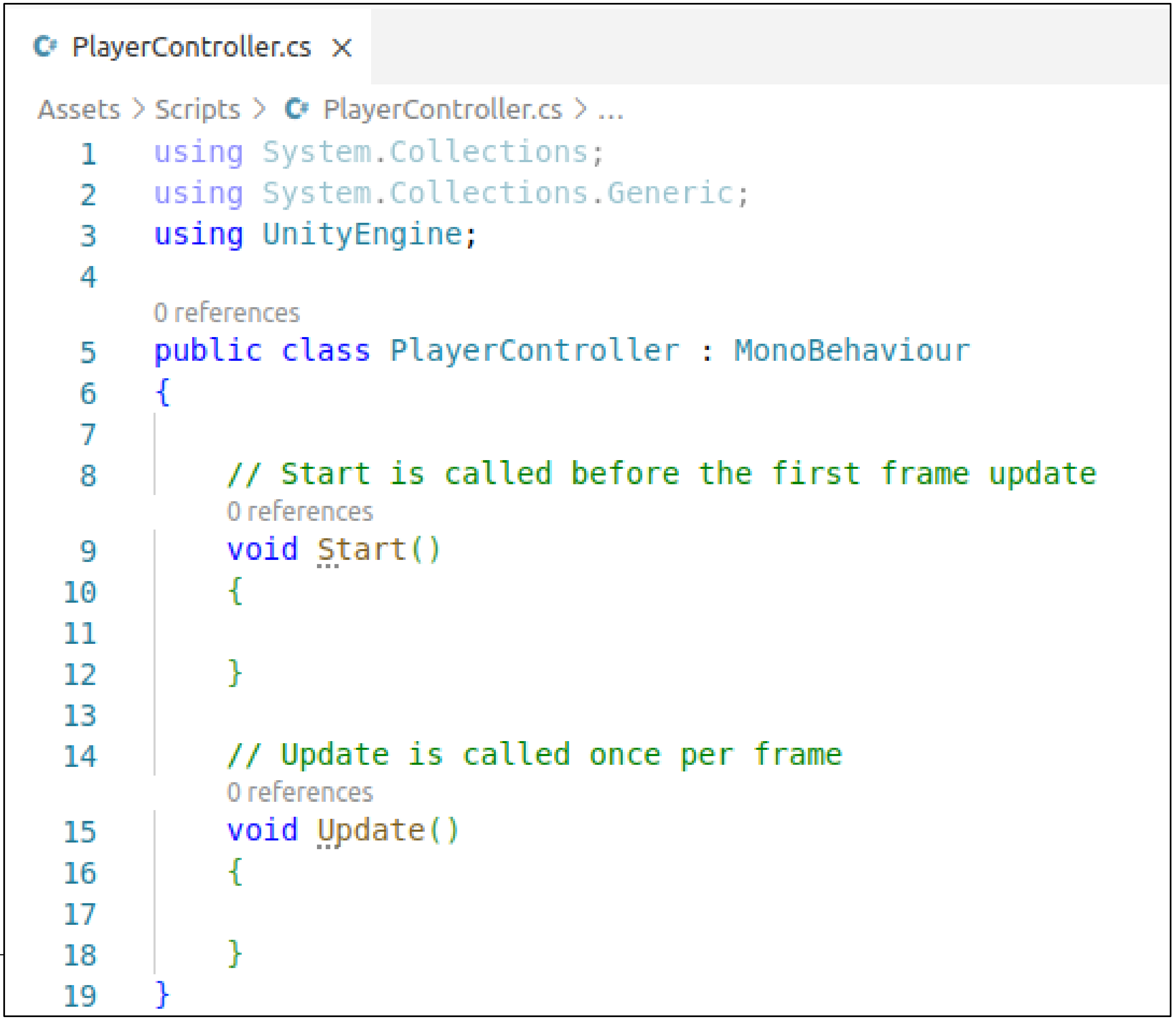
```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{

    // Start is called before the first frame update
    void Start()
    {

    }

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

    }
}
```

Create your first C# script

Unity creates a default C# script template

Can you explain its content?

What are the differences wrt Java?


PlayerController.cs ×

Assets > Scripts > PlayerController.cs > ...

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

0 references
public class PlayerController : MonoBehaviour
{

    // Start is called before the first frame update
    0 references
    void Start()
    {

    }

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

    }
}
```

Create your first C# script

1. Move the vehicle 1 meter forward on every frame and run the game

   transform.Translate(0, 0, 1);

2. We'll use Vector3 class to make it more "professional"

   transform.Translate(Vector3.forward); ← What's Vector3.forward?

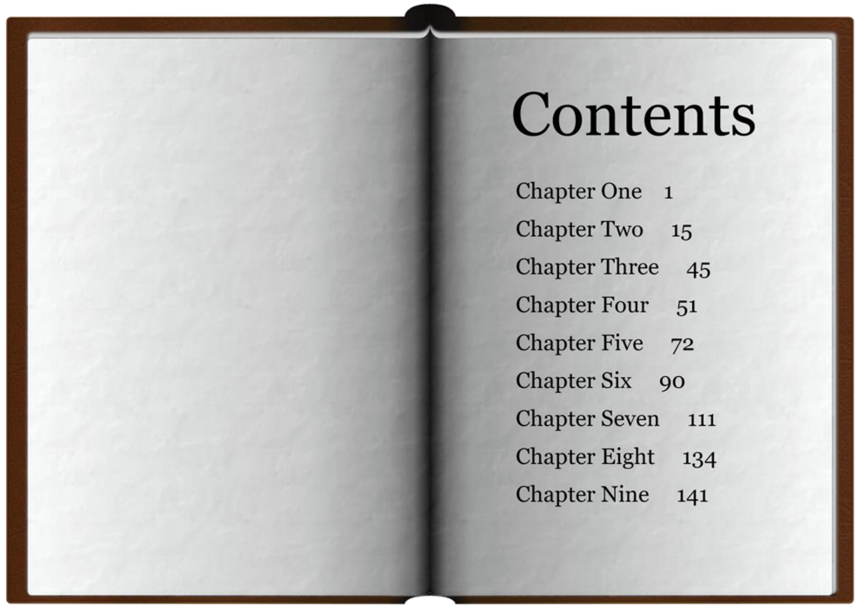
3. But the speed of the vehicle is still out of control because it advances 1m on every frame update → and that depends on the resources of the device where the game is running

4. Use *Time.delta* to changes the speed from 1m/frame to 1m/sec

   *transform.Translate(Vector3.forward * Time.deltaTime)*

5. What's the problem now? How to solve it?

Content



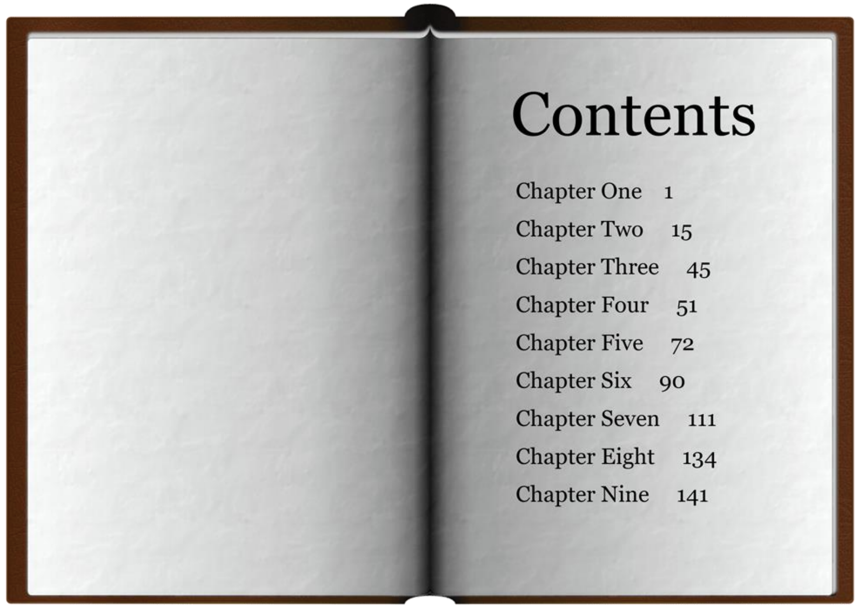
Contents

Adjust the player speed

1. Create a **class variable** (speed) with a default **float** value and use it in the *transform.Translate()* to adjust the vehicle speed
2. Setting its access modifier to **public** allows you to modify it in the **Inspector**
3. Run your game with different speed values and see the difference
4. Vary the speed while the game is running by dragging on the speed variable

But there're plenty of things still to fix … Does the vehicle crash to the obstacle? What is missing?

Content

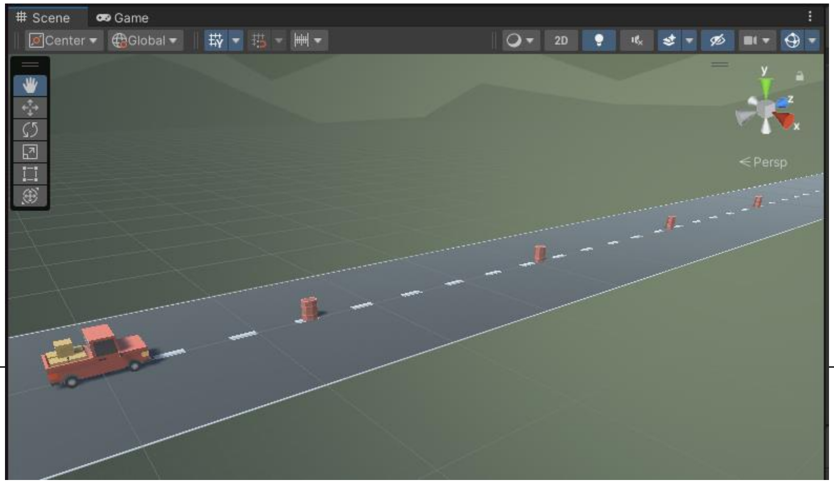


Contents

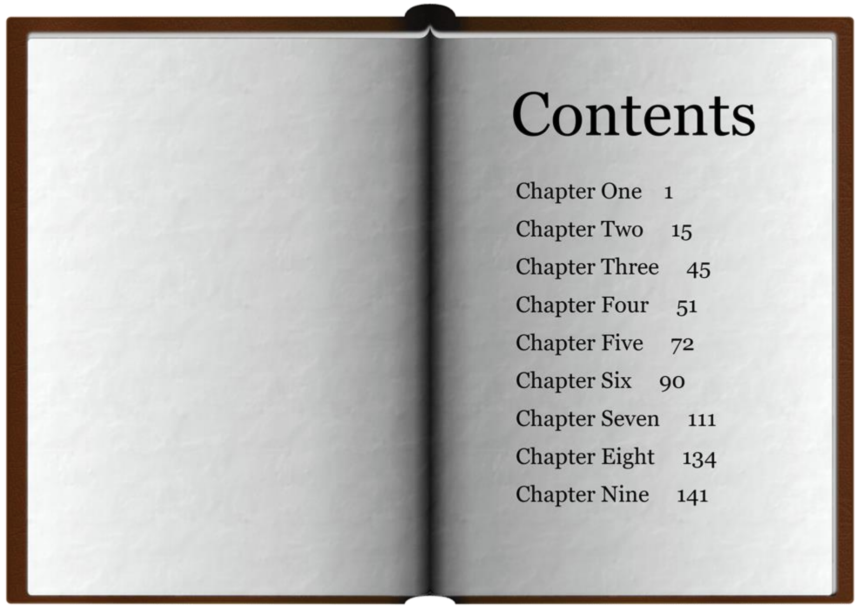
Chapter One	1
Chapter Two	15
Chapter Three	45
Chapter Four	51
Chapter Five	72
Chapter Six	90
Chapter Seven	111
Chapter Eight	134
Chapter Nine	141

Add physic properties to GameObjects

1. Add *Rigidbody* components to both the Vehicle and the Obstacle and test the game
2. Increase the mass of the vehicle and obstacle to be about what they would be in kilograms and run it
3. Try unrealistic masses to see the effect
4. Uncheck the Mesh Collider component … what should happen?
5. Duplicate the obstacles and move them down the Z axis



Content



Contents

Chapter One	1
Chapter Two	15
Chapter Three	45
Chapter Four	51
Chapter Five	72
Chapter Six	90
Chapter Seven	111
Chapter Eight	134
Chapter Nine	141

Follow the player: Camera adjustment

Follow the vehicle down the road and give the player a proper view of the scene

1. Create a new **C# script called "FollowPlayer"** and attach it to the camera
2. Open the script and add *public GameObject player* to the top of the script --> What is that for? What is that sentence doing?
3. Select the **Main Camera** and drag the Vehicle object onto the empty **player variable** in the Inspector --> What is that for?
4. In Update(), assign the camera's position to the player's position (Hint: *transform.position*)
5. Test your game --> Does the camera follow the player? Is it OK?

Follow the player: Camera adjustment

The camera is following the vehicle, but it is doing it right at the vehicle position

To have a better player experience, **you have to move the camera to an offset view wrt the vehicle** (above and behind) --> How will you do that?

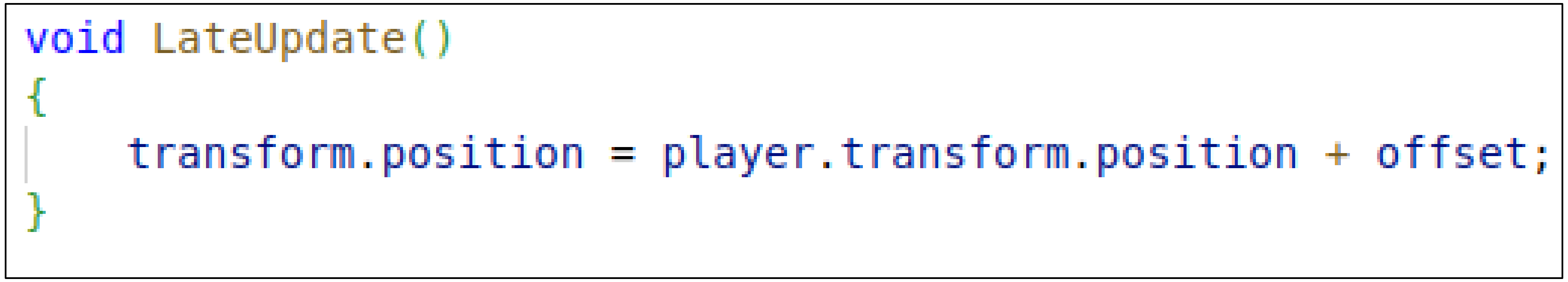
1. **Add a vertical and backwards offset to the camera position** on every update --> How?
2. **Instanciate a fixed Vector3 offset** (only Y and –Z) and add it to the camera position on every update
3. Try it and, if it works, move it to a variable

```
private Vector3 offset = new Vector3(0, 5, -8);
transform.position = player.transform.position + offset;
```

Follow the player: Camera adjustment

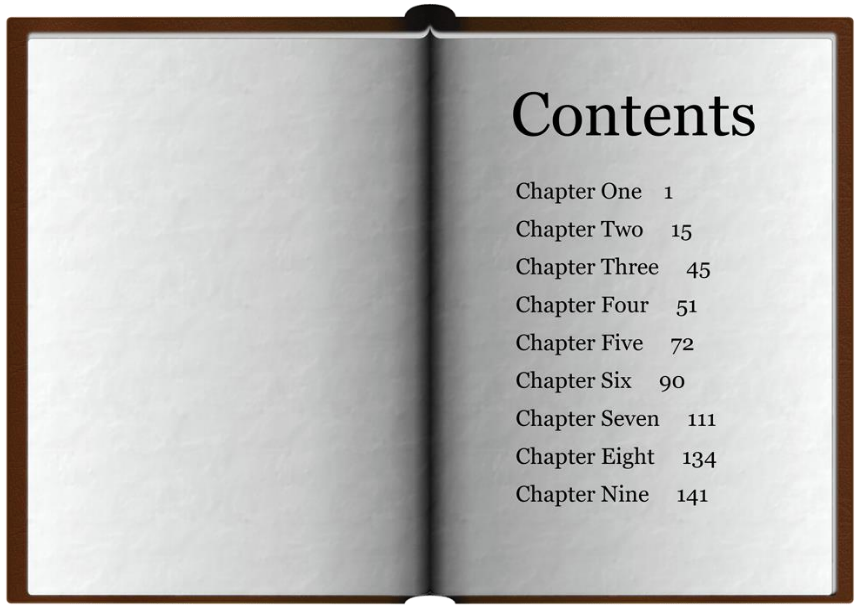
- You may have noticed that the **camera is kind of jittery** as the vehicle drives down the road
- That happens because **both the vehicle and the camera update at the same time**, sometimes the vehicle updates before, sometimes the camera updates before

1. In FollowPlayer.cs, **replace Update() with LateUpdate()**.



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

Content



Contents

Chapter One	1
Chapter Two	15
Chapter Three	45
Chapter Four	51
Chapter Five	72
Chapter Six	90
Chapter Seven	111
Chapter Eight	134
Chapter Nine	141

Input Manager

- We need to **detect when the player is pressing the arrow keys**, then accelerate or turn the vehicle based on that input

- Until now, the vehicle only has been able to move straight forward along the road. **We need it to be able to move left and right** to avoid the obstacles

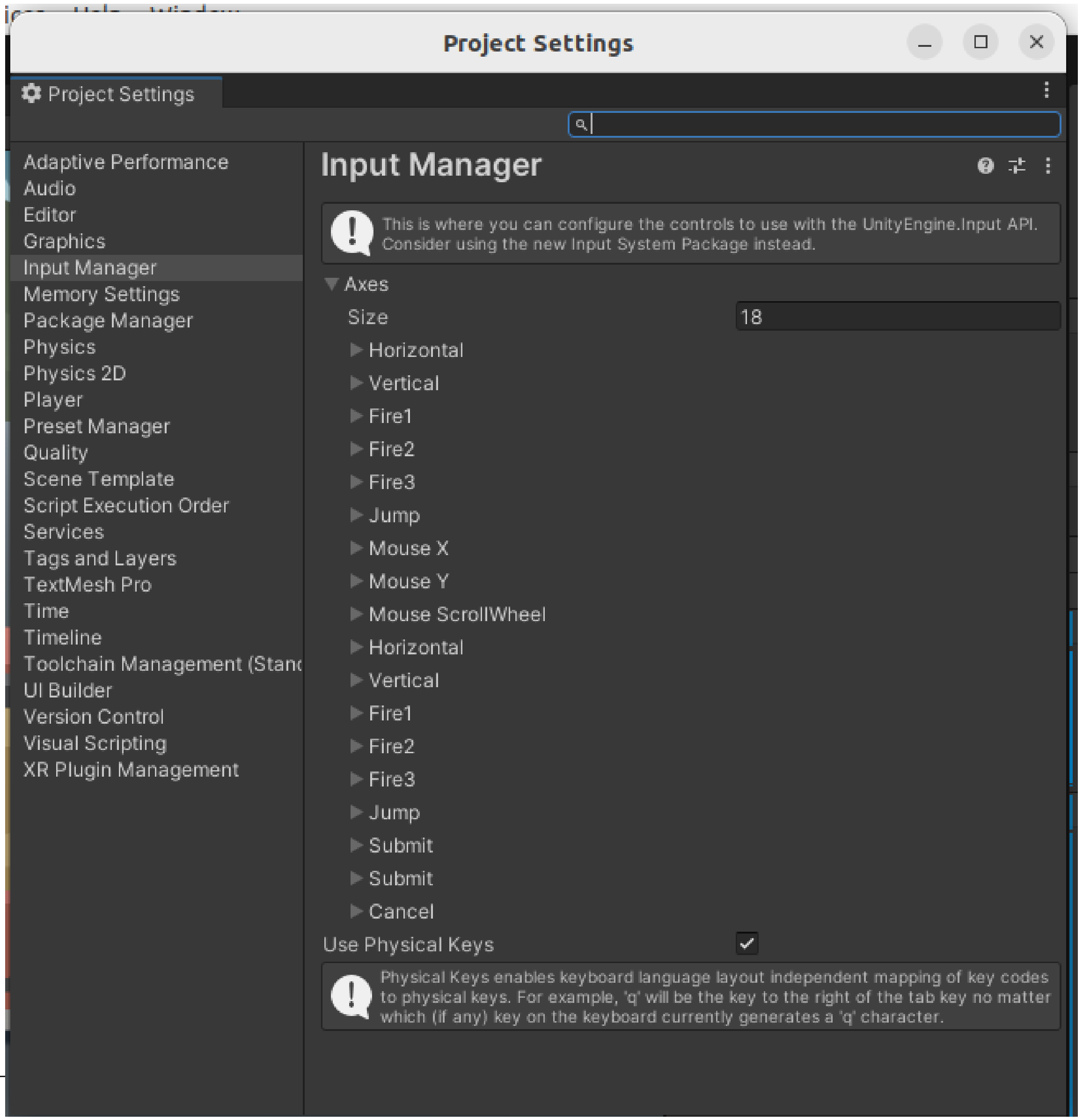
1. Add code to the PlayerController.Update() method to turn the vehicle left and right --> How will you do that?

```
transform.Translate(Vector3.right * Time.deltaTime * turnSpeed);
```

Run the game and use the *turnSpeed* variable slider to test it

Input Manager

- Currently, we can only control the vehicle's left and right movement in the inspector
- We need to grant some power to the players and allow them to control that movement using the keyboard: Input Manager
- Edit > Project Settings > Input Manager > Axes



Project Settings

Project Settings

Adaptive Performance
Audio
Editor
Graphics
Input Manager
Memory Settings
Package Manager
Physics
Physics 2D
Player
Preset Manager
Quality
Scene Template
Script Execution Order
Services
Tags and Layers
TextMesh Pro
Time
Timeline
Toolchain Management (Stanc
UI Builder
Version Control
Visual Scripting
XR Plugin Management

Input Manager

This is where you can configure the controls to use with the UnityEngine.Input API. Consider using the new Input System Package instead.

Axes
Size 18
Horizontal
Vertical
Fire1
Fire2
Fire3
Jump
Mouse X
Mouse Y
Mouse ScrollWheel
Horizontal
Vertical
Fire1
Fire2
Fire3
Jump
Submit
Submit
Cancel
Use Physical Keys ✓

Physical Keys enables keyboard language layout independent mapping of key codes to physical keys. For example, 'q' will be the key to the right of the tab key no matter which (if any) key on the keyboard currently generates a 'q' character.

Input Manager

- We will use the **Input class** and its methods to detect the player keyboard actions
- The **pressed keys will be stored in variables** and then used to move the vehicle left/right and back/forward
- The **Input.GetAxis([param]) method** allows String input parameters to determine the axis and **returns "a value" if the key** defined for that axis in the Input Manager **is pressed**

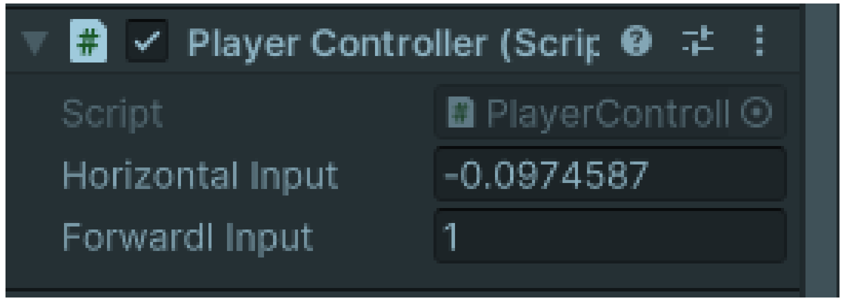
1. Check the method input/output parameters types and use it to move the car left / right --> How?
2. Once working, add also a forward/backward control for the vehicle using the up/down arrow keys

Input Manager

```
void Update()
{
    horizontalInput = Input.GetAxis("Horizontal");
    forwardlInput = Input.GetAxis("Vertical");

    transform.Translate(Vector3.forward * Time.deltaTime * speed * forwardlInput);
    transform.Translate(Vector3.right * Time.deltaTime * turnSpeed * horizontalInput);

}
```

Run the game, press the arrow keys and check the value of the *horizontalInput* and *forwardlnput* variables in the Inspector

--> Do you understand how is it working?

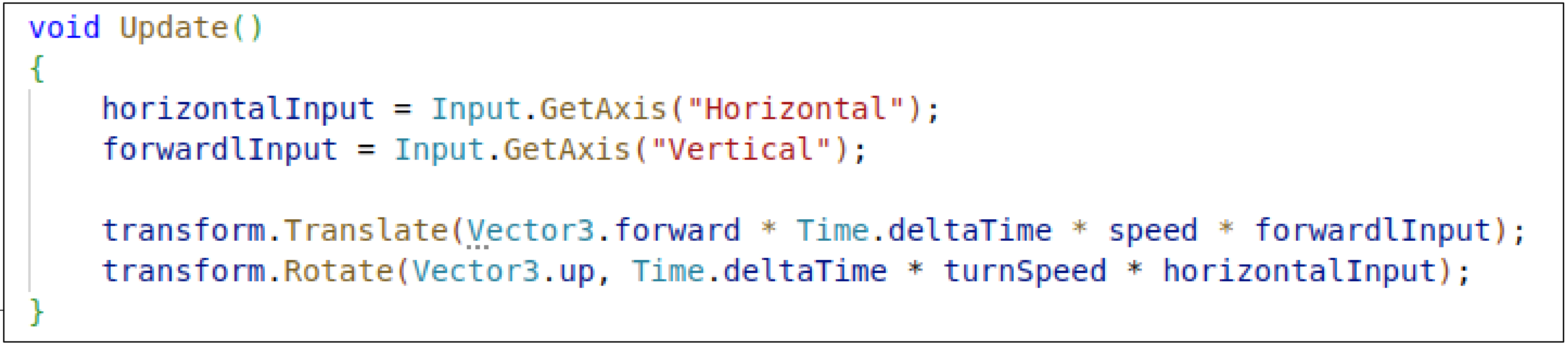

Player Controller (Scrip

Script	PlayerControll
Horizontal Input	-0.0974587
Forwardl Input	1

Input Manager

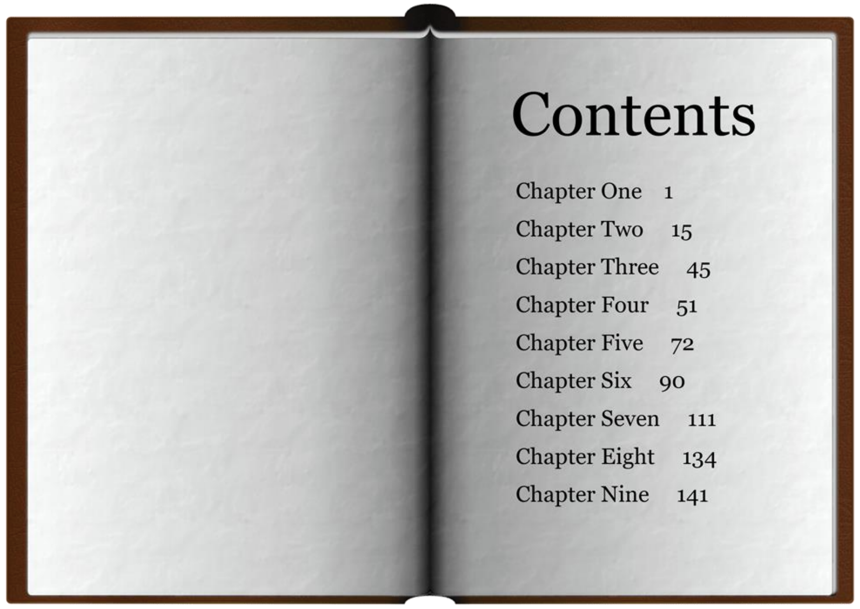
- Now we can control the 4 basic movements, but the vehicle is sliding instead of turning left/right --> How will you fix that?

1. Hint1: Rotate instead of Translate
2. Hint2: Do you know which is the required rotation axis?
3. Hint3: Rotate method input parameters are slightly different than Translate



```
void Update()
{
    horizontalInput = Input.GetAxis("Horizontal");
    forwardlInput = Input.GetAxis("Vertical");

    transform.Translate(Vector3.forward * Time.deltaTime * speed * forwardlInput);
    transform.Rotate(Vector3.up, Time.deltaTime * turnSpeed * horizontalInput);
}
```

Content


Contents
Chapter One 1
Chapter Two 15
Chapter Three 45
Chapter Four 51
Chapter Five 72
Chapter Six 90
Chapter Seven 111
Chapter Eight 134
Chapter Nine 141

ACTIVITY: Fix Plane Errors

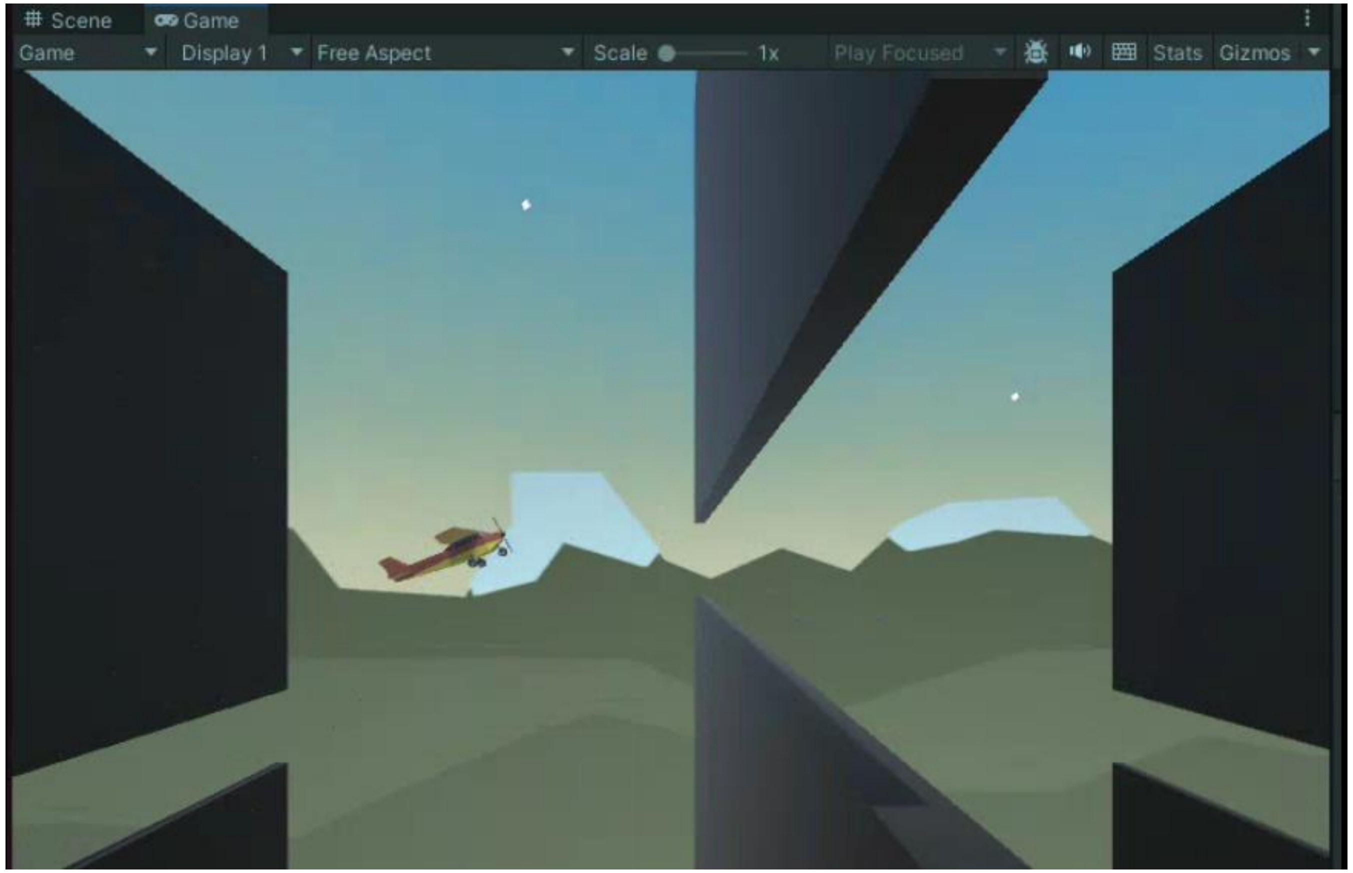
Fix the errors of a game where a plane flies around obstacles (*Plane.unitypackage*)

You have to implement modifications to:

1. Make the plane go forward
2. Slow the plane down to a manageable speed
3. Make the plane tilt only if the user presses the up/down arrows
4. Reposition the camera so it's beside the plane
5. Make the camera follow the plane
6. Make the propeller spin

ACTIVITY: Fix Plane Errors

Video



LICENSE