unity

1.3 High Speed Chase

Steps:

Example of project by end of lesson

Length: 50 minutes

Overview: Keep your eyes on the road! In this lesson you will code a new C# script for your camera, which will allow it to follow the vehicle down the road and give the player a proper view of the scene. In order to do this, you'll have to use a very important concept in programming: variables.

Project Outcome: The camera will follow the vehicle down the road through the scene, allowing the player to see where it's going.

Learning Objectives: By the end of this lesson, you will be able to:

- Declare variables properly and understand that variables can be different data types (float, Vector3, GameObject)
- Initialize/assign variables through code or through the inspector to set them with appropriate values
- Use appropriate access modifiers (public/private) for your variables in order to make them easier to change in the inspector
- Use the Update and LateUpdate appropriately in order to call one action after another has already happened

Step 1: Add a speed variable for your vehicle

We need an easier way to change the vehicle's speed and allow it to be accessed from the inspector. In order to do so what we need is something called a variable.

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);
}
```

Step 2: Create a new script for the camera

The camera is currently stuck in one position. If we want it to follow the player, we have to make a new script for the 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 position, then test

- **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

```
public GameObject player;

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

Step 3: Add an offset to the camera position

We need to move the camera's position above the vehicle so that the player can have a decent view of the game.

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);
}
```

Step 4: Make the offset into a Vector3 variable

We've fixed the camera's position, but we may want to change it later! We need an easier way to access the offset.

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;
}
```

Step 5: Smooth the Camera with LateUpdate

You may have noticed that the camera is kind of jittery as the car drives down the road - let's fix that.

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.

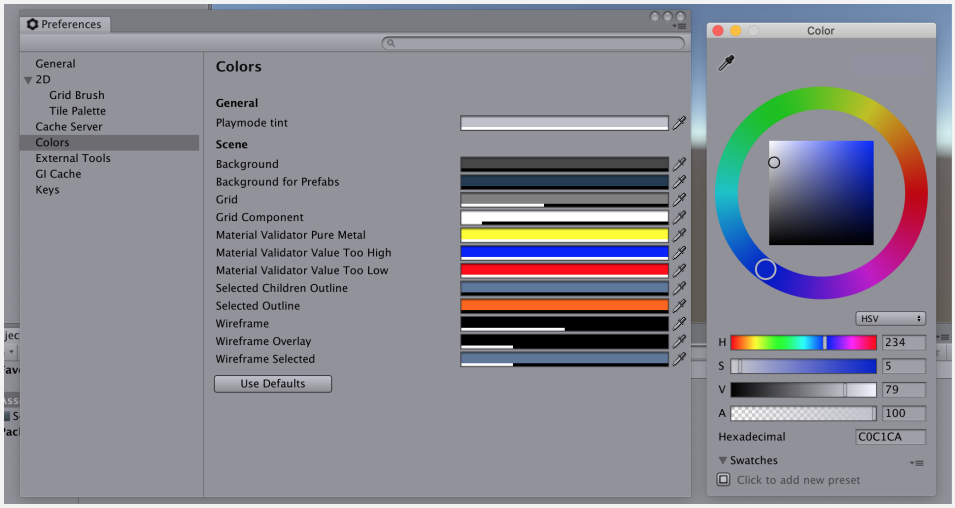
```
void LateUpdate()
{
  transform.position = player.transform.position + offset;
}
```

Step 6: Edit the playmode tint color

If we're going to be creating and editing variables, we need to make sure we don't accidentally try to make changes when in "Play mode"

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



Lesson Recap

New Functionality	• Camera follows the vehicle down the road at a set offset distance
New Concepts and Skills	• Variables • Data types • Access Modifiers • Declare and initialize variables • LateUpdate
Next Lesson	• In the next lesson, we'll add our last lines of code to take control of our car and be able to drive it around the scene.