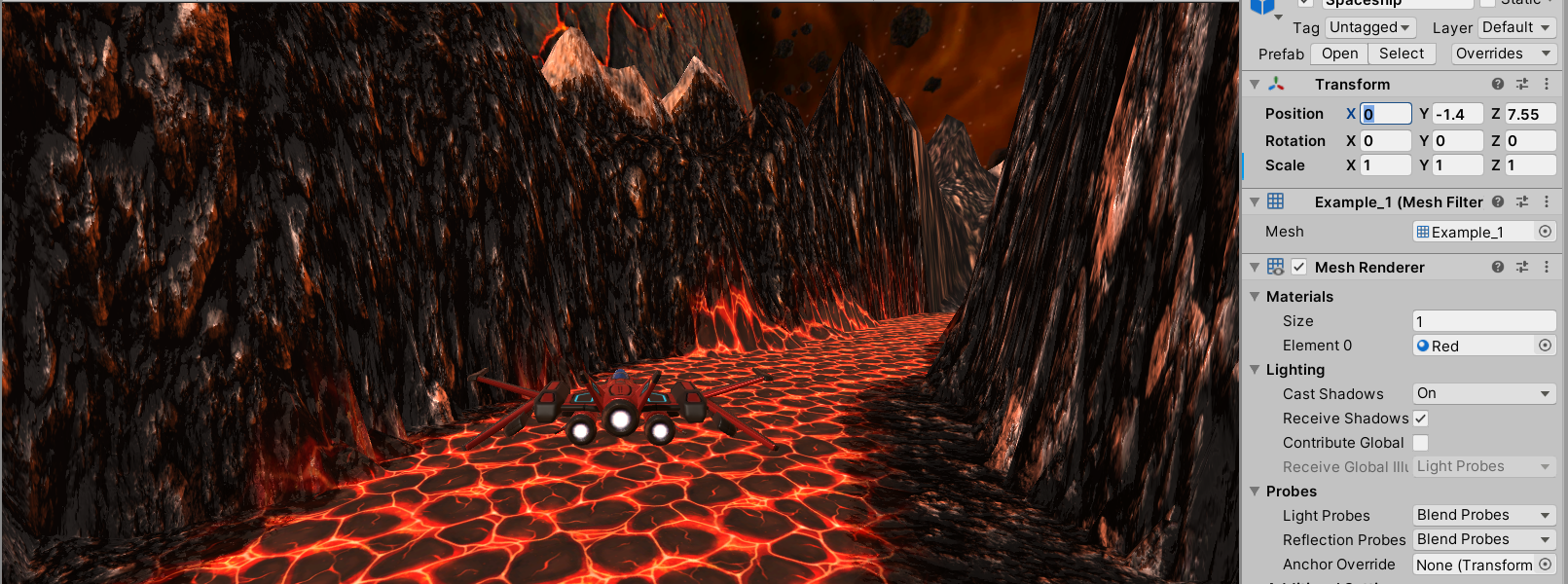
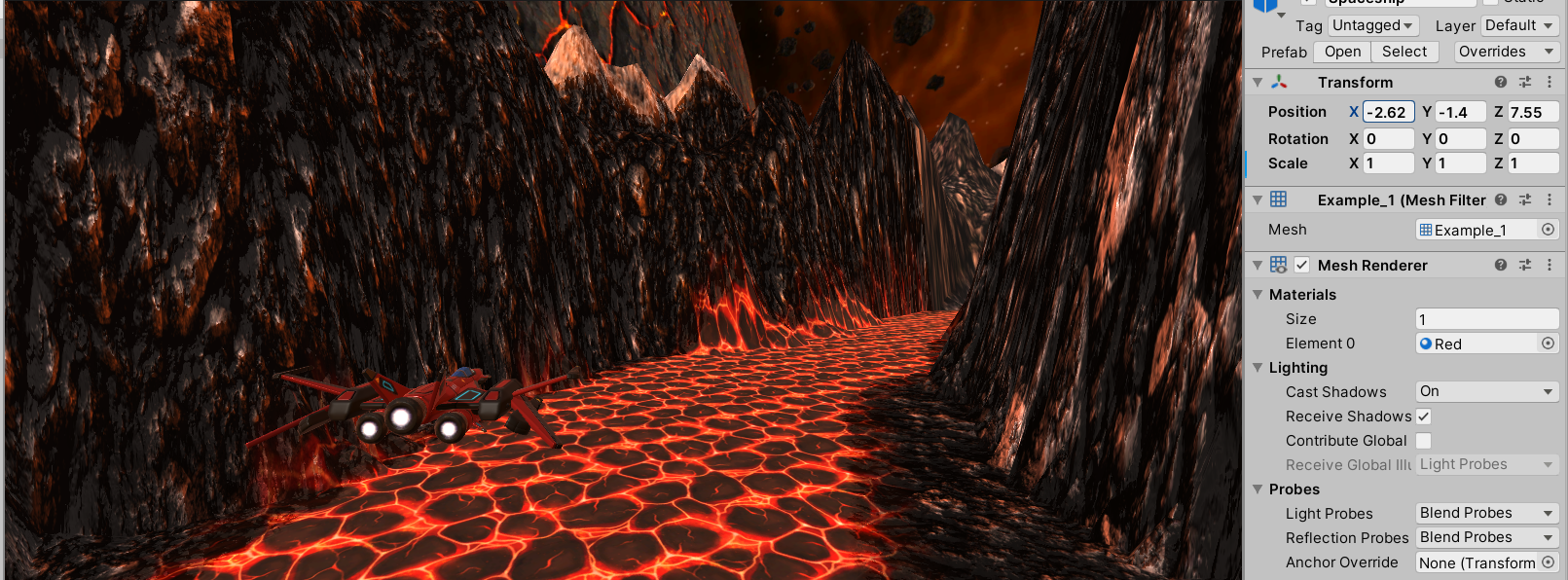
**Restricting Movement**

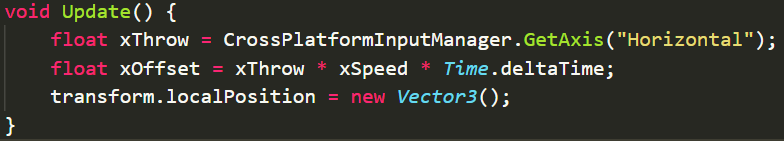
**Objective**: In this video we are going to restrict the x range of movement of our ship, so that it can get to the edges of the screen without overshooting the edges. Also, we're also going to make the Rocket ship move.

1. Rename **xOffsetThisFrame** to **xOffset.** So once we've gotten an x offset, how are we gonna move this ship?
2. In Unity go into the Player ship inspector window and lets simulate moving the ship by manually changing the Position. And that is the **local position** of the ship relative to the camera.

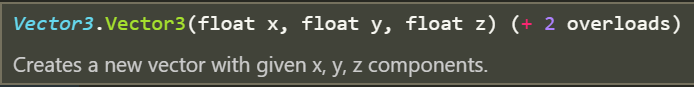




1. In the update method lets type **transform.localPosition** at the end of the update method, So the transform's local position is the thing you see in the inspector there. And we want to set its local position to something new, to a new Vector3.



1. Now Vector3 takes 3 parameters.



Now the only position we want to change is the x parameter. We want the y and z position of the ship to stay they same. Specifically we want the x parameter to be whatever the current offset is based on the A/D keys.



1. Go to Unity and let’s test out our changes. Press play click on the game window and try to move the ship.

Now that ship should be trying to move but keeps staying at its original position. Also it might also look kind of buggy. Why is it glitching??

1. Because we don't want to set the local position **just** to the offset. Because it's going to snap to the offset and then immediately come back to the same place. What we want to do, is we want to work out, before we do that, a new xPosition. Lets create a new variable **after the xOffset** initialization and lets call it **rawNewXPos.**



the raw new exposition without any conditioning is the current x position plus that offset. And then we provide that as part of the new vector that we make.



1. Back to unity to test our change!

Now the only problem we have is that the Rocket ship can fly off the screen. The slope of this line is the speed. So at first, the position is creeping up slowly and then the position goes up faster and faster and faster, until the speed gets to a maximum. This dotted line is where the speed is at a maximum. And then we have a straight line, then we want it to hit a limit.



Limit

How are we gonna make it hit the limit so it doesn't go off the side of the screen?

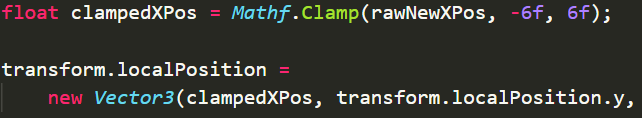
Well, we have a function called Mathf.Clamp, very useful. It restricts the output of the method, this expression evaluates to something that is restricted. It's easier and clearer than using a set of loaded if statements. And it's very simple to use, this is the syntax.

pos = Mathf.Clamp(rawPos, -10f, 10f);

If we have some raw position, and we want to clamp it between -10 and plus 10, then this is the syntax. And what happens to position is all the time it's between -10 and 10, position just takes the value of raw position. But the moment raw position goes under -10 or above +10, this value gets clamped to -10 or +10 respectively. It just restricts the range.

**Challenge:** Your ships movement needs to be constrained to the screen. Ideally just use one number ± from the center.

1. lets add this line right before the transform.localPosition at the end. Lets create a new variable and called it **clampedXPos**



1. Lets create a serializedField for this range. At the top under our xSpeed variable lets do this



and use our new variable



1. While we’re at it lets rename our **rawNewXPos** to **rawXPos** since “New” didn’t really make sense anymore.

