Diptera Transcendence

Technical Write-up

Contents

[Outline 2](#_Toc434770968)

[Creating player 3](#_Toc434770969)

[Camera Movement 5](#_Toc434770970)

[Lanes 7](#_Toc434770971)

[Creating Webs 8](#_Toc434770972)

[Spawning Webs 9](#_Toc434770973)

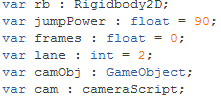
[Player – Web collisions 10](#_Toc434770974)

# Outline

Diptera Transcendence is a game where you play as s fly whose dream is to fly higher than any living thing has ever flown before. The player will mash enter to fly higher and faster, while pressing Q and E to switch between three lanes to dodge obstacles. Your score is determined by how fast you make it to outer space.

# Creating player

1. In assets, create the following five folders: Images, Sounds, Animations, Prefabs and Scripts (This is entirely optional, but will help keep things tidy).
2. Import the fly sprite.
3. Create a 2D sprite game object called Fly.
4. Set sprite type to multiple, and slice sprite automatically.
5. Assign the fly sprite to the sprite-renderer.
6. Add a Rigidbody2D and a BoxCollider2D to the fly.
7. Click on the Main Camera game object. Add a new script called cameraScript ( we do not do anything with this yet, we only need it because we reference it in the Fly object)
8. Create a new script called playerScript. Open it in Monodevelop.
9. Create the following variables (some of these won’t be used until later sections).



rb is the Rigidbody2D of the fly. We need a reference to it to apply forces.

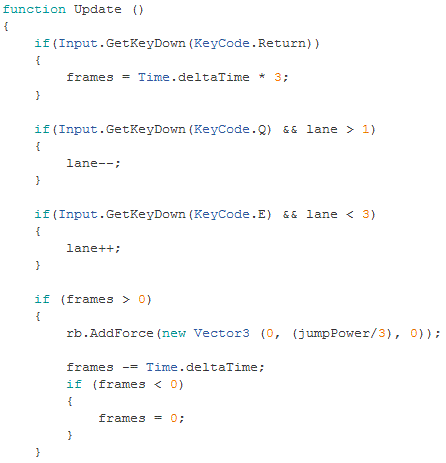
jumpPower is the force we will apply to the fly when the player presses enter.

frames is a variable used to smooth the application of jumpPower.

lane determines which of the three lanes the fly is in.

camObj and cam are used to calculate certain positions.

1. In the Start() function, assign rb, camObj and cam as shown below. C:\Users\user\Documents\College Things 2\Game Dev\Diptera Transcendence\Screenshots\screenshot2.gif
2. In the Update() function, check if the user pressed enter. If so, set frames to Time.deltaTime \* 3.
3. Check if the user pressed Q, and they are in a lane higher than 1. If so, decrement lane.
4. Check if the user pressed E, and they are in a lane lower than 3. If so, increment lane.
5. Check if frames is greater than 0. If so, apply a force of (0, jumpPower/3, 0) to the fly, reduce frames by Time.deltaTime. If frames is less than 0, set frames to 0.

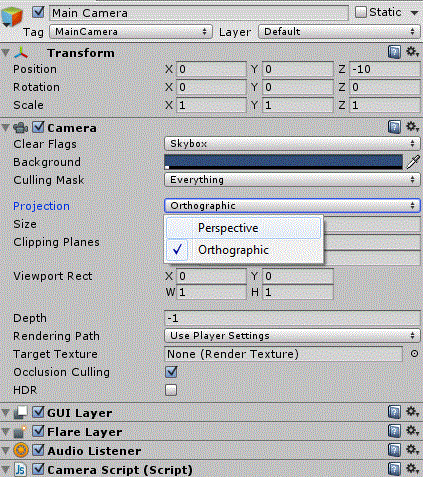


By using frames and jumpPower as we did above, we apply the jumpPower over three frames rather than in a single frame. This lends to smoother visuals in game.

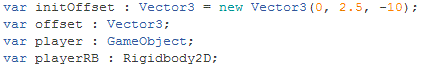
We need to add more features before we can actually use the lane variables, so at the moment there is no visible effect if the player presses Q or E. If they mash enter, they will fly off-screen. This issue will be addressed in the next section.

# Camera Movement

1. Click on the Main Camera in the hierarchy. Make sure that projection is set to perspective. This will allow the camera to show farther away images as smaller.



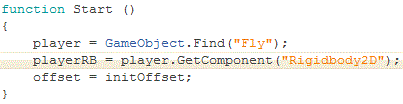
1. Open cameraScript.
2. Create the following variables.



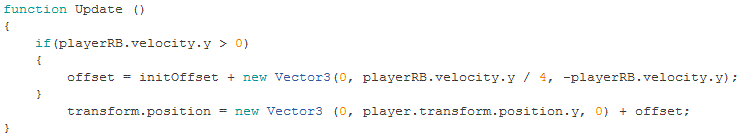
initOffset is as the base position for the camera. Assuming that Fly is at 0, 0, 0 This will make it stay about ¾ down the screen.

offset is a Vector3 we will use to move the camera.

1. In the Start() function, assign the player and playerRB variables. Also, set offset to initOffset.



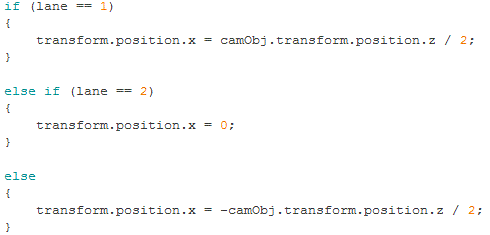
1. In the Update() function, check if playerRB.velocity.y is greater than 0. If so, set offset as shown below.
2. Set transform.position as shown below.



This code adjusts the cameras position relative to the players speed and position. The y and z values of offset are changed as the player speeds up, and move the camera up the y-axis, and out the z-axis. Effectually, the camera zooms out, and keeps the player relatively low in the view if the player is travelling upwards. After this calculation, the position is adjusted by this offset, and also by how high the player currently is, so that the camera does not get left behind.

# Lanes

1. Now that we have our camera set up, we return back to playerScript. At the end of the Update() function, check which lane the player is and adjust the transform.position.x as shown below.



We have to set the position relative to the camera, because if we don’t there will be very little difference between the three lanes at high speed with the camera zoomed out. This code should mean that the player will always be ¼ , ½ , or ¾ , across the screen, regardless of how far out the camera is zoomed.

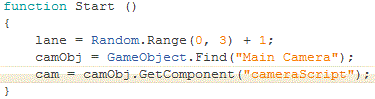
You should probably now test the game to make sure the camera follows the player, zooms out correctly, and that the lanes are working. If you are having trouble making the fly speed up, increase jumpPower in playerScript.

# Creating Webs

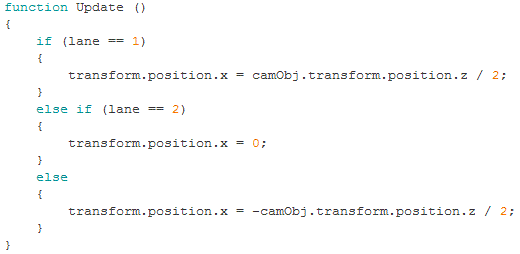
1. Import the web sprite.
2. Create a 2D sprite game object called Web, and assign the web sprite to the sprite renderer.
3. Add a Rigidbody2D and a BoxCollider2D.
4. Add a new script called webScript.
5. Open webScript in monodevelop.
6. Create the variables seen below.

C:\Users\user\Documents\College Things 2\Game Dev\Diptera Transcendence\Screenshots\screenshot9.gif

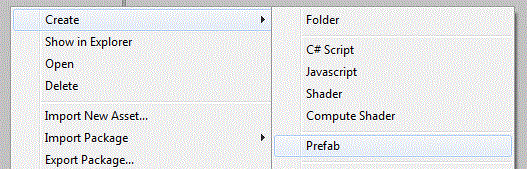
1. In the Start() function, assign camObj and cam as done previously. Initialise lane to a random number between 1 and 3.



1. In Update(), we check the lane and adjust position as done in playerScript.



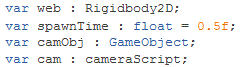
1. Now navigate to the Prefabs folder and create a new prefab.



1. Drag the Web to this new prefab. You can now delete it from the scene.

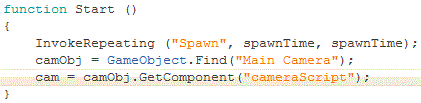
# Spawning Webs

1. Create a new empty game object called Spawner.
2. Add a new script called spawnerScript. Open it in monodevelop.
3. Create the following variables.

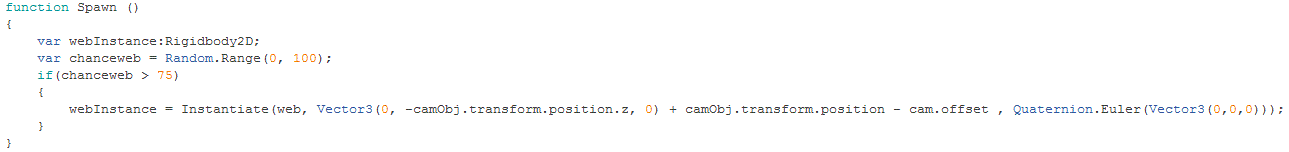


spawnTime is used for an InvokeRepeating call. It is not necessary, but allows us to adjust the rate of spawning from unity instead of needing to open the script.

1. In the Start() function we must once again assign the camObj and cam variables. We also call InvokeRepeating(“spawn”, spawnTime, spawnTime).



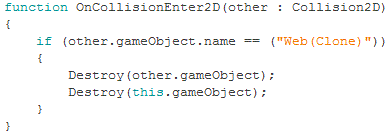
1. Declare a Spawn() function.
2. Declare the variable webInstance of type Rigidbody2D. Declare the variable chanceWeb and set it to a random number between 0 and 100.
3. Check if the variable is above a certain number (the higher the number, the lower the chance of a web). If it is greater, instantiate a webInstance. It’s position should be: Vector3(0, -camObj.transform.position.z, 0) + camObj.transform.position – cam.offset.



This code essentially makes the webs spawn at the top of the screen by making their y coordinate relative to how far the camera has zoomed out, adding the camera’s current position, and finally subtracting the camera’s offset to place it at the same depth as the Fly.

# Player – Web collisions

1. Open the playerScript.
2. Create the OnCollisionEnter2D(other : collision2D) function.
3. Check if the name of the other collision is “Web(Clone)”. If so, destroy the web, then the player.



We will come back later, and make the game over a tad more dramatic, but this will suffice for now. Test the game again. Make sure you can dodge the webs, and that the webs kill you if you hit them.