PORTAL II

Authors:

Micael Silva

04/07/2022

Table of Contents

1. First Person Character

1.1. Firing and Spawning Portals

Before we can fire any of the projectiles we have to check that the player is looking at an object that will support those same projectiles when they collide.

So to make this happen we start by drawing a line from the center of the player's camera to his forward vector, if this line collides with an actor we check if the actor has the tag that we specified earlier, in the end we are looking for the PortableWall Tag, if the actor returns this tag we can then fire the desired projectile.

To avoid this check being done every frame we use a timer with a small time interval, giving in the end the same result, but remains more optimized.

If the conditions are met we can then fire the projectile, this projectile will spawn with a default impulse and will check if it hits with an actor that has the PortableWall Tag, if so, it spawns the associated portal. We don't destroy the projectile after spawning the portal because the Portal spawning functions contain a chain of linked behaviors and it will cause a small delay to destroy the portal, as so, we destroy the projectile inside the Portal Spawning function.

1.2. Crosshair

Previously have looked for objects with a certain tag. If this tag was found we call a function in our HUD class that sets the render opacity to the desired one.

2. Portal System

2.1. Portal Self Adjustment

Once we create the portal and receive the information from the projectile, mainly we want to know the location where it collided and the Impact Normal transformed into a Rotator so we can place this portal in the desired location, but before we create the portal we need to replace the old one. Once we replace it we tell you which portal when the player collides, the player should be teleported to that same other portal, so having a Target the player can already know where he is going after colliding with it. Next we check the placement of the newly created portal.

First of all we have to get the vectors associated with the portal we want to adjust and get its limits, these limits are then used to create points where we draw a line to the inside of the object and check if they really collide with something, but for this logic to work we first have to know which object to compare at all its points, to do this we check at the center of the projectile's impact which object it collided with, then we save that object and we are ready to check if at each point the same object exists.

If this object exists in all the points it is not necessary to make adjustments, but if it doesn't exist in the vertical or horizontal points we adjust it according to the point where it doesn't exist. it is important to mention that there is no need to make adjustments from a diagonal point because for a diagonal point to be outside an object, or not find the same object that the center found, because for a diagonal point to be outside then either a horizontal or vertical point will be outside too. so we only check for the really important points, the horizontal and vertical points.

To adjust the position of the portal and the way we do it, we end up using a for loop and for each point we apply a transformation in the desired vector, the problem that followed was to make the adjustments small, and with the same distance. for this to happen when we apply the transformation to the desired vector we multiply it by the current index and divide it by the same giving transformations always with the same distance, whenever we do a transformation we have to re-trace the control lines to know if it is necessary to redo the transformation in the desired direction. Then we repeat this transformation until the object traces the control line and finds in that same point the same object that the center has as reference.

If it then finds the same reference object then we can move on to the next point that needs to be adjusted, if it no longer needs to be updated it moves on to the next, and so on until all the checks are done.

But if it exceeds the maximum number of transformations it means that with the transformations made the portal ended up adjusting more than half of its length, so we destroy the portal to prevent it from having such a large adjustment.

2.2. Traveling Between Portals

To teleport the player we need to know when he is in the portal zone, if he is we start checking if all the conditions are valid for the teleport to happen. If he is not in the portal zone, there is no need to check the conditions.

First we verify that the player is crossing the portal, to do this we create a Plane and we Validate if LastPosition and Point meet their positions with the Plane.

Then we check if the player is at the front of the portal, this condition is validated if the dot product of the created Plane is greater than the point we passed in the function, this point being the player's location relative to the portal. Then, if all the previous conditions are valid, we give permission to the player to be teleported.

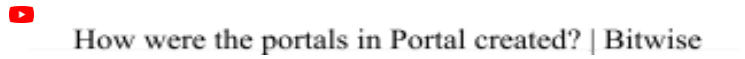
To teleport the player, first we start by saving his speed to later apply after teleporting, we also convert the position and rotation of the player relative to the portal then Set the player Location and Rotation when Teleporting to the portal, after that Adjust the player Control rotation in so he faces the same direction and the same angle that he enters in the portal, after that we can set the speed of the new player and reverse the speed to maintain the flow of the player movement since the rotation of the player will be reversed because he enters the portal with one rotation and after being teleported we reverse that rotation.

2.3. Portal Illusions

To make the portal illusion we must convert the Position and Rotation of the player Camera. As so the SceneCapture location and rotation of that portal's Target must be updated every frame to maintain the illusion of the portal, just like a "window", than we should have a dynamic clipping plane so the SceneCapture can hide everything that is between it and the portal, not rendering it in the portal surface.

3. External Info

3.1. Internet Sources



Why 0.1 Does Not Exist In Floating-Point - Exploring Binary

GlobalVectorConstants::KindaSmallNumber | Unreal Engine Documentation

3.2. Book Sources

“A Tour of C++”, Bjarne Stroustrup, 2013