Beer

A Graphical Representation of Beverages in Unity

Norris Chan
University of California Santa Cruz
nchan6@ucsc.edu

Ryan Santiago
University of California Santa Cruz
ryasanti@ucsc.edu

Jessie Aniguid
University of California Santa Cruz
janiguid@ucsc.edu

Yash Dua
University of California Santa Cruz
ydua@ucsc.edu

ABSTRACT

This Unity executable models a glass of beer on a table, with a light coming from above. The physical elements we modeled were fizz, foam, and glass, among several others.

INTRODUCTION

This project set out to simulate the behavior of contained beverage in three dimensional virtual space. While the various settings could be altered to simulate most other beverages and liquids, the project focused on creating an accurate representation of a beer in a large glass stein. This includes a liquid taking on the shape of its container, its tendency to slosh around when the container is moved, and the caustic rays that occur when light shines through its glass container. The project was then divided into lower level parts: the liquid, the container, and caustics. The liquid was divided further into smaller components: foam, fizz, and movement.

1.1 **Liquid**

The liquid is achieved using a shader that can adjust how much of an object is rendered to the screen based on a uniform called "_FillAmount." The fill amount uniform is used to adjust the y component of the affected object's world position. The modified world position is then passed to the fragment shader in an attribute called fillEdge, which determines how much of the object to color and what to render transparent and colorless. Because it affects only the y component of the world position, even if the object is rotated, the invisible part of the mesh will always be at

the top of the object, giving the partially rendered mesh the appearance of being filled with liquid.

1.1.1 Foam

A foam line can be added to the top of the liquid by using a uniform to devote a certain amount of what is rendered to a color output different from the rest of the liquid. In the case of the below image, the liquid is the product of a color and a refracted background image while the foam is the product of a passed in texture multiplied by a tint color.

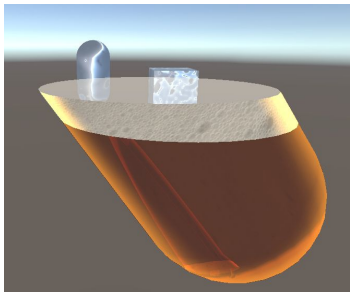

Figure 1.1.1a The liquid's top always faces up regardless of the rotation on the object.

1.1.2 Wobble

The liquid appearance is further reinforced by adding wobble to the top of what is rendered. The wobble is created using a script that measures the translational and rotational by subtracting the rotation and position from the previous frame by the position and rotation on the current frame and dividing the resulting differences by deltaTime. A certain amount of wobble is then applied across the object's the X and Z axes based on the previously calculated velocities. Wobble values in the X and Z directions are multiplied by sin(pulse*Time). These values are passed into the shader program as the uniforms "WobbleX" and "WobbleZ" and are used to modify the y component of the world position, causing the top of the mesh to fluctuate up and down until no velocity is recorded and reaches an equilibrium.

1.1.3 Bubbles and Fizz

These were handled using a particle system that emitted tiny bubbles from inside the liquid mesh. The bubbles traveled upward from the bottom of the glass, and if they were to touch the sides of the container, they would stop moving as if to simulate bubbles sticking to the sides of a glass. If the bubbles reach the foamy area of the liquid, they would stop moving and disappear after a short while.

2.1 **Caustics**

The math behind replicating and understanding caustics are fairly complex and computationally heavy. While we understand the methods well now, it still isn't very simple to get realistic caustics, especially not for unique objects. With the HD Render Pipeline, we would have been able to replicate realistic caustics for our glass, but the shader we used for the glass was not compatible with the HD Render Pipeline. Unfortunately, we were unable to work around this and could not implement our initial idea for the caustics. There are simpler methods for replicating caustics, but they are much more suited for replicating caustics that may appear at the bottom of a pool, for example. Without access to the HDRP method, we had to use this method to implement caustics. First, we used a caustics generator program to generate 16 .bmp files with similar images of caustic rays. These .bmp files were attached to a projector object positioned

above the glass. Using a script, the bmp files were repeatedly cycled through, creating something akin to the caustics on the bottom of the floor of a pool. However, this look is not quite what we intended for this project, so it can be toggled on and off. There is a sphere that uses one of these .bmp files as it's texture placed in the shadow of the beer bottle, which is visually closer to the actual caustics the beer might have, but still not what it could be with the HDRP.

3.1 **Motion Blur**

In the below paragraph, it is explained how motion blur is implemented in Unity. To add a blur effect, follow these steps:

 In a Unity shader, create a simple vertex and fragment shader
 Write a function that returns a float4 and call it Box() that takes in a texture, the uv's, and the texture texel size
 In Box(), add together the 8 neighbouring pixels at that uv coordinate
 Divide this value by 9 and return it
 Call Box() inside the fragment function making sure to pass in the main texture, the uv, and the texel size of the main texture and return it
 In a separate C# script, create a public reference to a Material and name it "BlurMat". In the editor, attach the previously made shader onto this reference.
 Create an OnRenderImage() that takes in two RenderTextures: "src" and "dst"
 Create a temporary RenderTexture temp by using RenderTexture.GetTemporary and passing the height and width of "src".
 Use Graphics.Blit and pass in "src" and "rt".
 Write a for-loop that runs "Iteration" times. Inside the for-loop, instantiate another temporary RenderTexture rt2 that also takes in the height and width of "src"
 Use Graphics.Blit and pass in "rt" and "rt2" and "BlurMat".
 Release rt using RenderTexture.ReleaseTemporary and set rt to rt2
 Outside the for loop, use Graphics.Blit and pass in "rt" and "dst".
 Release rt using RenderTexture.ReleaseTemporary

Below are steps to control the intensity value of the blur in Unity. To alter the intensity value of the blur effect in Unity, follow these steps:

 In Start() Instantiate float values "mouseVelocity", "lastMousePosition" and "changeInMousePosition" and set them to 0.
 In Update(), set "changeInMousePosition" to Input.GetMousePosition minus "lastMousePosition".
 Create an if-else cascade. If "changeInMousePosition" is 0, decrement "mouseVelocity" by x amount.
 If "changeInMousePosition" is low, set "mouseVelocity" ~= 1000. Otherwise, set "mouseVelocity" ~= 1500.
 Set "Iterations" to "mouseVelocity"/300.
 Set "lastMousePosition" to Input.GetMousePosition.

4.1 **Scene Dressing**

4.1.1 Skybox

First, create a 6 sided skybox material and find an image for the skybox. Then found an image to use as your skybox. I used gimp to break up the picture into 6 different sides of the picture to place into the skybox to create the scene: front, back, left, right, up, and down.

4.1.2 Table

Used a 3D model of a trapezoidal table found online and placed into the scene. Created a texture for the table and placed a wood texture into it.

4.1.3 Glass Mug

Used a 3D model of a beer mug that was found online. Created a glass material using the standard shader from unity. Placed glass material onto the beer mug object. Then changed the smoothness and metallic settings to create a glass like appearance to the mug object.

REFERENCES

[1] mug object: https://www.turbosquid.com/3d-models/beer-mugs-s-free/311470

[2] Creating glass material: https://www.youtube.com/watch?v=k4HVe9WpQwU

[3] table object: https://www.turbosquid.com/3d-models/trapez-table-3d-model-1172716

[4] coaster object: https://www.cgtrader.com/free-3d-print-models/house/kitchen-dining/saucer-for-mug