Paintball Shooter Simulator

Final Project Report by
Johnny Mak 40002140
Dylan Seto 40001991
Helen Tam 27845537

For Dr. Tiberiu Popa
COMP 477 - Animations for Computer Games

December 4, 2017
Concordia University

Description

We are very inspired with the idea of making something similar to the game Splatoon, a game where players must shoot blobs to paint the entire playing field. Therefore, we want to make a project similar to it where the user can move the camera in first-person view and shoot onto the terrain. Upon impact of the blobs, they will explode and change the color of the affected terrain. To do so, we will implement fluids that will naturally help us animate the impact and explosions of the projectiles. Our project will be set in a sandbox area where the user can freely move, look, and shoot around. Additionally, there will be obstacles like small walls where the user can also paint over. For this project only, we will not be implementing any blasters that the user will see on their screen. The liquid projectile will be shot out via direction of the camera. In addition, the user will have the option to toggle between day and night. During the day, the paint will have a goo-like texture to it but during the night, the paint will start to glow.

Motivation

We want to demonstrate fluid physics and are also very inspired by the game Splatoon. We believe that coding fluid physics is much more interesting in comparison to making physics for rigid objects due to the fact that we never worked with liquid objects before and want to learn how. The goal of this project is to demonstrate fluid physics in a subtle and interesting way. The idea of self-creating a colorful world also sounds interesting as there are no boundaries in what we can make.

Objectives

1) Day & Night Time Lighting
2) Set proper fluid impact parameters to animate small paint explosion
3) Implement physics for paint projectiles
4) Implement collision physics between terrain and paint
5) Vary force of blaster shot based on mouse hold time
6) Add basic lighting
7) Enable movement in a sandbox
8) Allow choices between different paint colors
9) Implement a skybox
10) Implement a frame rate counter

The project was split as follows:

Dylan: Physics (SPH, collision detection, forces)
Helen: Lighting and coloring
Johnny: Backend (Buffers, base code)

Difficulties Encountered and Fixes

During the course of our project, we've encountered difficulties ranging from basic coding with OpenGL to implementing fluid simulation. Here is a list of obstacles we've faced:

Lighting and Color

- **Computing vertices' normals**
  Each vertex's normal is computed based on the two vectors that share that vertex. Because the scene objects' vertices are created one at a time and the triangles' indices are not incremental, computing the normals was a hard task.
- **Implementing flat shading**
  Due to possibly wrong normals, the scene's colors were blended into another. This created a blur effect that was not pleasant to see. To fix this, we've passed our normals and object color into the shader with the flat interpolation.
- **Adding bloom effect**
  The bloom effect requires blurring the scene and render it as a texture. This caused a lot of complications as we need to buffer the data dynamically. We resolved the issue by implementing a faint glow onto the paint.
- **Painting the scene when a paint ball collides with an object**
  This difficulty came about when we needed to find a way to store where the paint ball hits, and with which color. This data was to be passed into the shader in order for it to replace the hit object's color with the paint color, but couldn't find a way to do it.

Fluid Particles

- **Simulating fluid particles with physics**
  Because of the complexity of simulating fluid particles, applying physics to them required a lot of time and research in SPH. In addition, with so many particles to render, we had to optimize its number for the best possible performance.
- **Computing correct density and pressure values for the particles**
  Due to the large amount of particles that we simulate, density and pressure values for each particle needs to be computed relative to their positioning, as well as their neighbouring particles' properties. At times, the values passed were correct, but their outputted values weren't.
- **Computing the force that shoots the fluid**
  A force needs to be applied in order for the paint ball to shoot. For this, the main difficulty that came with it was to apply a force to each particle, so that some can splatter, or fly off the center of the liquid. That is, the liquid should have proper physics. This force is used to give each particle a velocity and is part of simulating fluid particles with physics.
- **Working with shaders**
  Shaders have a maximum uniform array size, so passing particle data was difficult. The way around this was to write the data into its texture's RGB values.

General Coding with OpenGL

- **Passing data to the buffers**

Although minor, this caused a problem in drawing the scene objects properly when data wasn't passed to the buffers correctly. One of the mistakes that led to this was indicating the wrong strides to our buffers.

- **Updating data in the buffers**
  Some data must be computed in the shader, and returned back to the source code for further handling. To do this, we had to find ways to retrieve data from our shaders. Transform feedback was a solution, but it required time to implement it correctly.
- **Integrating individual codes into one**
  This is a more general problem that we couldn't have avoided. Doing individual tasks didn't cause any problems, but putting all individual parts together was difficult due to code dependency and merge conflicts.
- **Optimizing performance and code efficiency**
  Trying to optimize our code was also a challenge as it needed to be changed and, at times, frequently. When some components were modified, dependent parts wouldn't function properly so it was sometimes hard to fix.

External Libraries

- OpenGL 3.3 (GLFW)
- SOIL: Simple OpenGL Image Library
- GLM: OpenGL Mathematics

Resources

[1] https://www.youtube.com/watch?v=qN4w5D2tzME
[2] http://prideout.net/blog/?p=58
[3] https://www.youtube.com/watch?v=LyoSSoYyfVU
[4] http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/
[5] https://learnopengl.com/#!Lighting/Basic-Lighting
[6] https://open.gl/feedback
[7] https://nccastaff.bournemouth.ac.uk/jmacey/MastersProjects/MSc16/13/thesis.pdf
[8] https://nccastaff.bournemouth.ac.uk/jmacey/MastersProjects/MSc16/15/thesis.pdf
[9] https://www.opengl.org/discussion_boards/showthread.php/198661-Is-there-a-better-way-to-send-data-to-a-shader