Project Proposal for ITCS 5235 2019

Proposers:

Matthew Mcquaigue, University of North Carolina at Charlotte, {mmcquaig}@uncc.edu

Title:

Ray Trace Rendering Capable of Non-Euclidean Geometries in Webgl

Abstract:

My research began with the fascination of Non-Euclidean Geometry and how it can be used to create unique graphical scenes that can be engaging and unknown. A Non-Euclidean geometry is a 3D shape that follow the first four of Euclid's postulates but could have its own version of the parallel postulate. The parallel postulate states that "If a line segment intersects two straight lines forming two interior angles on the same side that sum to less than two right angles, then the two lines, if extended indefinitely, meet on that side on which the angles sum to less than two right angles." Manipulating this postulate can create interesting and complex shapes not seen in reality. In this project I plan to create a real time ray tracing engine that is capable of non-euclidean geometries.

Introduction:

This WebGL based application will be a basic 3D editor allowing users to render their own unique scenes. Users will have access to a menu of customizable commands/options for editing the scene. Among these options will be a dropdown menu of possible objects to render to the scene. This list will include pre-downloaded .obj files from external websites and basic geometries such a square, sphere, etc. Once a geometry is selected from the list, the user can perform transformations on the object to place it where needed.

Click interactions will be used for the selection of objects in the scene. Rendering and moving a geometry to a position is not finalized. Once an aspect of the scene is clicked, a bounding box will indicate the selected object. With mouse dragging the object can be moved about the scene. Each object will have a list of textures associated with it. When the object is selected, a texture can be specified for the object. Along with the texture, the material of the object can be selected from different choices.

The camera will be manipulated with Euler angles to rotate around the scene for editing. An option will be provided to enter 'FPS' mode to walk around the scene. Walking movement is controlled through 'wasd' keys. These keys will only be active during 'FPS' mode.

Background:

The main aspect of this rendering engine will be ray tracing. Ray tracing is a technique of rendering 3D graphics based on light interactions. This idea provides simple implementation to achieve realistic renderings of mirrors, transparent surfaces, and shadows through the manipulation of casted rays as they reflect and refract around an environment. For each pixel on the view window, a ray is casted into the environment. The ray is followed as it bounces off different objects and reaches a light source. Then color of that pixel/ray is accumulated and the culmination becomes the pixel value.

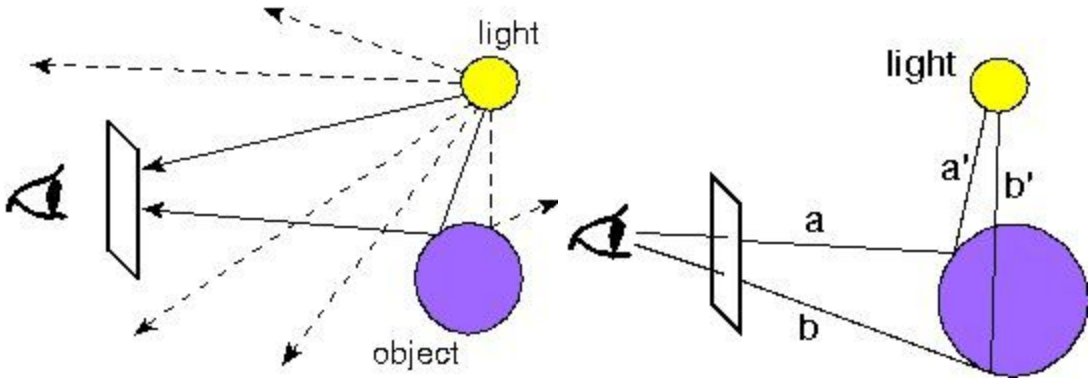

Figure 1. [left] All possible rays/photons coming from the light source and reflecting off object (forward tracing). [Right] Two casted rays from view port. Eliminating rays not reaching camera for efficiency (backward tracing. (Image from [1])

"The only downside of path tracing is the computational complexity required to achieve high visual fidelity. For cinematic quality images, billions of paths may need to be traced. This is because we never compute the actual value of the integral, only its estimate. If too few paths are used, this approximation will be imprecise, sometimes considerably so. Additionally, results can be vastly different even for points that are next to each other, for which one would expect the lighting to be almost the same. We say that such results have high variance."[3] Using this method of rendering, shapes and spaces are highly customizable. Spaces and shapes can be manipulated to form non-Euclidean planes.

Non-Euclidean geometry is similar to regular Euclidean accept for the fifth postulate either replaced or the metric requirement is relaxed. In the case of the parallel postulate being replaced, two forms of non-Euclidean geometry arise: hyperbolic or elliptic geometry. For this project, I will only be changing the aspects of hyperbolic geometry. "Hyperbolic space has many interesting features; some are similar to those of Euclidean geometry but some are quite different. In particular it has a very rich group of isometries, allowing a huge variety of crystallographic symmetry patterns. This makes the geometry both rigid and flexible at the same time. Its properties and symmetries are closely related to tree-like growth patterns and fractals, suggestive of many natural biological objects like ferns and trees."[2] Within

this plane, space and be compressed and stretched where the shortest distance between points are not necessarily a straight line.

Methods:

The first feature in the project will be the implementation of a ray/path tracer and the interaction of rays with aberrations to create hyperbolic geometries. This will be achieved by creating a aberration object that is a textureless and transparent cube. This cube can be placed into the scene the same way as any other object. Rays will be casted to the scene and will check for intersections with objects. Since the aberration will have the same structure of a cube, we can use the same intersection algorithm. Once detected, far plane of the shape that is detected will be stretched or compressed by a scale factor.

The second feature to be added to the project will be small amount of physics to predetermined objects. I plan to have an option set to an object before it is rendered into the scene. If toggled, this object will be spawned into the middle of the scene and be effected by gravity. When the user is editing mode (not FPS mode), the physics object can be selected and picked off the ground, it will fall to the floor upon release. This will require gravitational calculations and collision detection with the floor.

Tools:

I will be using WebGL and javascript for this project. I have thought about using OpenGL but my expertise is not as well versed. WebGL has different API's and libraries to use but I will be sticking to raw WebGL for this implementation. I will also use the ATOM text editor and Windows/Linux based on the machine I will be using at the time. I will try to use a modular approach to shading. This implies building the shader or concatenating the strings based on the current object being rendered. This method allows for faster running execution and easier to read shaders.

References

[1]P. Rademacher, "Ray Tracing: Graphics for the Masses", *Cs.unc.edu*. [Online]. Available: https://www.cs.unc.edu/~rademach/xroads-RT/RTarticle.html. [Accessed: 16- Mar- 2019].

[2]C. Series, *Hyperbolic Geometry*, 2nd ed. 2008, p. 40.

[3]T. Akenine-Möller, E. Haines and N. Hoffman, *Real-Time Rendering, Fourth Edition*, 4th ed. Boca Raton Florida: CRC Press.

[4]J. Cannon, W. Floyd, R. Kenyon and W. Parry, *Hyperbolic Geometry*, 1st ed. MSRI Publications, 1997.

[5]"Hyperbolic Geometry, Section 5", *Pi.math.cornell.edu*, 2019. [Online]. Available: http://pi.math.cornell.edu/~mec/Winter2009/Mihai/section5.html. [Accessed: 18- Mar- 2019].