NORTHERN ILLINOIS UNIVERSITY

Random Generation and VR Environments: Creating Personalized and Unique User Experiences

A Capstone Submitted to the

University Honors Program

In Partial Fulfillment of the

Requirements of the Baccalaureate Degree

With Honors

Department Of

Computer Science

By

Noah Miller

DeKalb, Illinois

May 11th, 2019

University Honors Program Capstone
Approval Page

Random Generation and VR Environments: Creating Personalized and Unique User Experiences

Student Name: **Noah Miller**

Faculty Supervisor: **Dr. Michael Papka**

Department of: **Computer Science**

Date of Approval: **May 4th, 2019**

Check if any of the following apply, and please tell us where and how it was published:

□ Capstone has been published (Journal/Outlet):

□ Capstone has been submitted for publication (Journal/Outlet):

□ Capstone has been presented (Conference):

□ Capstone has been submitted for presentation (Conference):

Completed Honors Capstone projects may be used for student reference purposes, both electronically and in the Honors Capstone Library (CLB 110).

If you would like to opt and not have this student's completed capstone used for reference purposes, please initial here: _____ (Facility Supervisor)

Honors Capstone Abstract

"Random Generation and VR Environments: Creating Unique and Personalized Experiences" is split into two components. The video showcases a live demonstration of the VR project, and an in-depth view of the integrated development environment used to assemble the project. The paper attached to the project serves as a reflection of the experience of designing for a virtual reality environment. It documents the struggles and roadblocks present throughout development, and the growth the experience brings to a developer. It is highly advised that this report is read after the video component of the project is viewed. The project was developed with Epic Games Unreal Engine version 4.21. A combination of the Unreal Engine's built-in scripting tool Blueprint and the C++ programming language were used to assemble the project. Development required extensive use of the Unreal Engine documentation available at Epic Games official website. The project was developed using the standard HTC Vive model in the Data, Devices, and Interaction Laboratory (DDI Lab). The project is also cross-compatible with the PlayStation VR headset.

The video presentation component of this project clarifies the development of this application. It is recommended to view that before reading this follow-up. I choose the Unreal Engine as my primary tool for the project due to its compatibility with the Vive VR Headset, and my previous exposure to the engine. It wasn't until about midway through the production of the project that I had realized the extensive optimization work the project required. Like other engines in its vein, Unreal Engine is heavily reliant on frame time calculations. These calculations vary from general logic, such as basic mathematical expressions, to more complex calculations such as how polygons render in a scene. The second proved to be the more problematic of the duo because an extensive amount of polygons in the scene was incredibly likely to occur due to the random nature of the project. A lower framerate is generally acceptable in most environments, but virtual reality is different in this regard. We need virtual reality to maintain a higher framerate performance, with our application roughly updating at least every 16 milliseconds at minimal. The reason we need the higher framerate is that even a slight performance drop can completely disconnect the user from the virtual reality experience.

This problem above wound up consisting throughout development. Unreal Engine utilizes a specific rendering pipeline, which is unmodifiable without extensive engine rework. Engine rewrites completely fell outside of the general project scope that was set. The engine handles frame time rendering during runtime, generating polygons on an actor-by-actor basis, for each actor visible in the current frame. An actor with a visible component utilizes Unreal Engine's static mesh component. In a more general project, performance is heavily regulatable due to the static nature of most projects. However, due to this project’s dynamic nature, the solution was not as simple. A simple solution was to limit the generation's scope, but as a result, sacrifice the original vision of the project. Another solution was to design for specific hardware,

but this felt like it limited the reach of the project. Neither solution was ideal for the project, so I decided to see if I could design an algorithm to mitigate the issue. A solution I wound up designing, used a priority-based system to push spawn points to certain portions of the map. Future actors spawned would be forbidden from spawning near the previously spawned actor. However, this algorithm failed to improve framerate performance on smaller maps with a large actor amount. Small maps have the issue of most actors being visible in a single frame. As a result, the issue was unresolvable in these smaller environments.

Outside of general performance hurdles in the project's development, another aspect which was more difficult than I originally assumed was the actual modeling of the static meshes for actors. Modeling a static mesh for each actor is a multi-phase process which can easily require several hours of work for a singular model. Being a novice in creating models for software, this was primarily uncharted territory in my academic career. Unreal engine requires the developer to design a static mesh in a separate modeling program and to set up an individual lighting map for each mesh. Without a lighting map, the static mesh will be lit inaccurately in the randomly generated time of day present in my project. As a result, I had to dedicate an increased amount of time learning Blender to model all the custom actors present in the project.

To conclude this report, I would like to express a minor reflection on the project. This project was my first formal introduction to developing in a virtual reality space, as well as my first time using virtual reality. As an individual with reoccurring motion sickness symptoms in many environments, I was slightly worried that it would prevent me from fully enjoying the virtual reality experience. However, I had a minimal amount of ailment when using the HTC Vive. Furthermore, I found that a good majority of my skill set obtained from my previous courses at the university proved useful during the development of the project. Unreal Engine is

built around the C++ programming language, and its visual scripting component is heavily based around that language as well. Northern Illinois University regularly uses the language throughout several courses, so I had already felt well versed in the language at this point. Resulting in a stark contrast with my knowledge before university where previously I had used Unreal Engine, but I had minimal exposure to C++ at that point. So as a result, this project felt like an opportunity to reflect on and utilize the skills obtained over my tenure at the university. Overall, the project proved to be an engaging and thought-provoking project which allowed me to escape the traditional terminal-based assignments in our coursework, and instead develop a project in a robust three-dimensional space.