UNC PEMBROKE | MAYNOR HONORS COLLEGE

Janus

Senior Project

In partial fulfillment of the requirements for
The Esther G. Maynor Honors College
University of North Carolina at Pembroke

By

Ian Schmude
Art
May 17, 2021

_____	_____
Ian Schmude Honors College Scholar	Date
_____	_____
Adam Walls Faculty Mentor	Date
_____	_____
Joshua Kalin Busman, Ph.D. Senior Project Coordinator	Date

Acknowledgements

I'd like to thank both Dr. Busman and Professor Walls for putting up with my constant absent-mindedness and inability to remember what is due when, as well as for being encouraging and supportive throughout the semester – without which I undoubtedly would not have gotten as far as I did.

Well worth mentioning are the communities of users around the software used to make this project, especially Blender and Godot. Their numerous and well-made tutorials and documentation gave me an excellent foundation for implementing my designs.

Abstract

This paper describes the process of designing and creating a video game as my honors thesis project. Using tools such as Blender or TrenchBroom, I learned how to create 3D models, animations, textures, and levels. These assets were compiled into a playable demo using the Godot game engine. Ultimately, this project did not reach the levels of polish that I had hoped for, but nevertheless provided an opportunity to gain hands-on experience with game development.

Janus

For my honors thesis project, I decided that I would design a video game. As an art and computer science student, creating a game for my thesis seemed like a natural choice for the culmination of my higher education. It's a career that I've been thinking about since middle school, and I have been toying with a number of different game ideas since then. Until now, I had little experience with actually implementing a game, instead experimenting with some 2D game making tools. For this project, I settled on a first-person horror game similar to early 3D first-person shooters such as *Quake* or *Half-Life.* This would at the very least give me an introduction to the design process for a much bigger type of game than what I had done before.

Preproduction is the most exciting stage of any game project I've attempted and the one where I get carried away with concepts. This is where I created concept art for the environment and characters I wanted to include, and wrote about the theme, came up with ideas for enemies' behavior, gameplay, and the player's abilities and equipment. Additionally, I listed the assets I would need to procure and a timeline in which to do so. Most of the ideas I came up with I knew I would be unable to implement, so I defined a minimum viable product: a player character, at least one enemy NPC, and a game world to explore.

As I moved on to actually implementing my project, my first task was choosing what engine to work with. A game engine is the software that runs a game built on it – it renders the graphics, interprets the user's inputs, performs physics calculations, etc. Development tools – such as a level editor, software or features for designing and decorating a game world – are often included. There are a number of game engines available for use by both independent and big-budget game developers, such as Unity or Unreal Engine. By using an engine complete with a suite of development tools created by another team, developers save a significant amount of time and effort, since they don't have to program their own tools like a physics engine or a level-editor. I had no budget whatsoever, so I needed an engine that was free to use. My prior experience with game development involved 2D games that used a simple visual scripting workflow – an abstraction of "coding" that was rather limited. I would need an engine that either used this workflow or had a great deal of user-made tutorials or official documentation in order to learn the engine in a reasonable amount of time. Luckily, I found both with Godot. It's open source, which means not only is it totally free, I can modify it, sell games I make with it, or use it however I'd like. It has bountiful first-party documentation as well as numerous tutorials made by the community.

Godot uses nodes and scenes to create game objects or levels. As the Godot manual puts it, nodes are like the ingredients in a recipe – individual building blocks

that can be combined to create brand new things. For example, if I wanted to create an object with physics properties (it will be affected by gravity and collide with other physics objects) I would create a RigidBody node. I would need to give it a 3D shape and a way to collide with other object instances, so I would combine it with a MeshInstance and CollisionShape. All these nodes are loaded in a scene – essentially a container for nodes. Scenes can be loaded in other scenes, too! I could save my physics object as a scene, then load that scene into my level. This is the essential workflow of Godot.

Next, I needed to learn how to actually build a game. Creating a few nodes or displaying "Hello world" is one thing, creating something interesting to interact with is another. Godot's first-person shooter tutorial includes instructions on creating a first-person game complete with movement, weapons, and … well, that's about all I stuck with it for. Once had I used the tutorial to create a player object and learn how animations were handled, I dove headfirst into level design. The Godot manual mentioned TrenchBroom and Qodot – a level editor tool originally built for idSoft's *Quake* and a plugin that would allow TrenchBroom maps to be imported into Godot. I thought this was a brilliant idea, since I knew I wanted my game to look and feel reminiscent of *Quake*. What I disregarded was that this is intended as a method for *prototyping* level designs, not final builds. TrenchBroom is simple and easy to use – I felt fairly proficient after only a few hours playing around with it. When it came time to import what I created into Godot is where I ran into problems. I frequently crashed Godot attempting to import the maps, and if I discovered any errors with the level geometry, I would have to reimport everything. After much trial and error, I had a pretty neat level for the player to explore. The only problem was there was nothing to do with it.

This project made use of custom art assets as well as free assets found online. I am not an accomplished texture artist, nor did I have the time to learn to create my own, so I found a very nice collection of textures designed for TrenchBroom released by the artist Makkon. I did want to model my own creatures and a few other props, which I did using Blender, another open-source tool for 3D modeling and animation. Normally, game development should start by implementing the systems necessary for all the features you'd like. If I wanted my players to pick up objects, fire weapons, open doors, I should first start by designing and coding those features *before* creating the art assets for them. Being an artist before anything else, I did that a bit backwards. I had modeled and textured some props and enemies before I had any way for the player to interact with them. This turned out to be the most challenging part of the project.

The gameplay loop involves the player exploring, looking for keys that would unlock new areas of the level, and either avoiding or fighting some AI monsters. The concept for the enemies that the player would encounter involved a level of

interactivity that I was unable to implement. At first I wanted the player to be able to use either stealth, direct combat, or some kind of non-lethal combat, and then change how the enemies would react to the player based on their approach. As it turned out, just implementing a functional (albeit rather unintelligent) AI was difficult and time consuming enough.

There were two primary goals of this project, and depending on which you might consider more important, this version of *Janus* is either a resounding success or a mediocre failure. The original concept for this game was far more ambitious than my skills or time allowed for, and the final product is frankly boring and not very interesting. However, as an introduction to solo game development, it was an invaluable lesson. I learned about nearly every aspect of the technical side of game development, and faced the realities of attempting such a project on my own with little prior experience.

Works Referenced

Textures:

David Kelvin, "Q.wad", https://www.wad-archive.com/wad/4cd033967cb9435a1dee4742d3dc8c7002c88ff3

Textures.com, "DoorsMetalDouble0006" https://www.textures.com/download/DoorsMetalDouble0006/8133

Ben "Makkon" Hale, "Makkon1.wad", https://www.artstation.com/makkon

Ben "Makkon" Hale, "Makkon2.wad", https://www.artstation.com/makkon

Ben "Makkon" Hale, "Makkon3.wad", https://www.artstation.com/makkon

Tutorials:

Code with Tom, "Simple FPS Enemy AI in Godot - Make an FPS in Godot Part 4". Youtube, Apr 5, 2020. https://youtu.be/OSYehj6oa3U

Contributors to the Godot community, "Godot Docs – 3.3 branch", https://docs.godotengine.org/en/stable/about/index.html

Fixmox, "Godot 3D Door Tutorial" Youtube, Aug 7, 2019. https://youtu.be/6ESV14iLy84

Garbaj, "3D AI Pathfinding". Youtube, Aug 19, 2020. https://youtu.be/YFgrpp1fpOI

Garbaj, "Enemy AI: Aiming And Shooting - Godot Tutorial AI Series Pt 2". Youtube, Jun 10, 2020. https://youtu.be/wFyQLwWL0q4

GDQuest, "Global Illumination and the GIProbe in Godot". Jan 29, 2019 https://youtu.be/lPngD4uzWVc?list=LL

Grant Abbitt, "Rigging a Low Poly Person". Youtube, Nov 14, 2019. https://youtu.be/srpOeu9UUBU

Grant Abbitt, "BASICS OF ANIMATION | Blender 2.8 - Part 3 - Bones & Armature". Youtube, Apr 13, 2019. https://youtu.be/IAiTYaiZmY0

Software:

Shifty, Qodot v1.6.4. https://github.com/Shfty/qodot-plugin

Blender Foundation, Blender 2.92. https://www.blender.org/

Juan Linietsky, Ariel Manzur, Godot Engine contributors. Godot Engine. 2007-2021. https://godotengine.org/

Serif, Affinity Photo 1.9.2. https://affinity.serif.com/en-us/photo/

Kristian Duske, TrenchBroom, https://trenchbroom.github.io/index.html