Nicholas Warfield

Virginia Mushkatblat

Comp 467 - Saturday 8:00AM

21 September 2019

Game Engine Project Proposal

Product Description:

1) Scope, what I am implementing for this project:
    a) File System: Saving and loading assets, both while a user is creating a game and at runtime when a player is playing a game
    b) Game Loop: There are 3 parts to a game loop; processing user input, updating the game state, and rendering the game.
2) What features should the engine have? (Stuff the engine needs to do if I keep working on it)
    a) Render 2D animations
    b) Square grid levels
    c) Drawing is not grid locked
    d) Cutscenes
    e) Import assets (animations, sprites, audio, level data, game object data)
    f) export assets (particularly level & game object data)
    g) file system
    h) render loop
    i) game loop
        i) turn based
        ii) real time (stretch goal)
    j) messaging system

k) user input (keyboard + mouse, controller, remappable)

l) pathfinding (a* probably)

m) AI (npc, enemy characters)

n) effects

   i) shaders

   ii) particle effects

o) ui elements

p) audio

q) random number generation

r) debugging tools (console, log files, cheats, etc)

s) compression

t) camera

3) What features will the engine **not** have?

   a) 3D capabilities (there will be rendering and object layers, but nothing 3D beyond that. No meshes, models, etc)

   b) Physics (there will be user defined rules for moving through the grid, but that's the closest thing to physics. No rigidbodies, collisions, raycasting, etc)

   c) asset creation (no sprite editor, no audio editor)

4) What do other engines do?

   a) Unity: unity is a general purpose engine designed to be a valid and sound choice for developing most games. The two main features of unity are playback editing (being able to change parameters of the game while it is running), and the component pattern (game objects are made up of components like rigidbodies, cameras, and scripts defined by the user).

b) Godot: godot is also a general purpose engine like unity, but it is also free and open source. It does not use a component pattern, and instead uses tree hierarchy to create game objects. I plan to implement my file system using this pattern, so digging through the source code here will be quite helpful for understanding how it works.

c) Game Maker: game maker is a general purpose 2D engine. It can only make 2D games, but it has a physics engine which my project will not have. Because of its limitations, game maker is one of the more easy engines to learn and use.

Tools Selection:

1) C++

a) C++ is the language I am most familiar with and it is quite popular among the game development community, so that is why I am using it. Additionally, the library I want to use is only for C++.

b) C# would be a good alternative, it is harder to write bad code in C# and it is becoming more widely used in the industry.

2) SFML library

a) The SFML library is a free and open source, general purpose multimedia library that utilizes OpenGL. It has 5 modules (audio, graphics, network, system, and window) to handle lower level tasks, like opening and an OpenGL window. I have also worked with SFML in the past, so that should save me some time getting familiar with it.

b) Alternatives would include SDL as another general purpose multimedia library. Or go even more low level and use GLFW, GLEW, and GLM.

3) cmake

a) This is to help me compile my project correctly. The alternative is compiling everything by hand or writing my own makefile.

4) gcc
    a) The C++ compiler I am using. I could also use clang, but I've never used it before.
5) git & github
    a) This is for version control and backups.

Design/Architecture:

1) Game Loop
    a) Part of what I plan to implement for this project. Every game has to use this pattern, otherwise it would not be interactive. It is also some of the most critical code in the program because the game will spend 80% of its time executing the loop.
2) Scene Tree
    a) This is the pattern Godot uses to create game objects. Without this pattern, game objects would have to be programmed into the game engine and compiled along with it.
3) Command
    a) This is a great way to handle user input, it will let the player be able to rebind the controls without having to change any code at run time.

References:

- http://gameprogrammingpatterns.com/type-object.html
- https://www.sfml-dev.org/
- https://www.gameenginebook.com/