|  |  |
| --- | --- |
| Program: | GDP1 |
| Course: | INFO-6023 “Game Algorithms & Gems” Project #1, Winter 2023 |
| Professors: | Michael Feeney |
| Project # 1: | Escaping from “threaded” beholders! |
| Weight: | An equal portion of your 60% “project” mark  (i.e. all projects are equally weighted) |
| Due Date: | Saturday, March 4th at 11:59 PM |

Description and Purpose

You are to make an OpenGL application (for visual output) that simulates many beholder monsters wandering through a massive maze (that is too large to display at one time).

**You will submit:**

* Your amazing **Visual Studio solution** (Windows “win32” API, 64-bit in **both** Debug and Release mode).
* A **short video highlighting your code** (more details later in the document)

Restrictions:

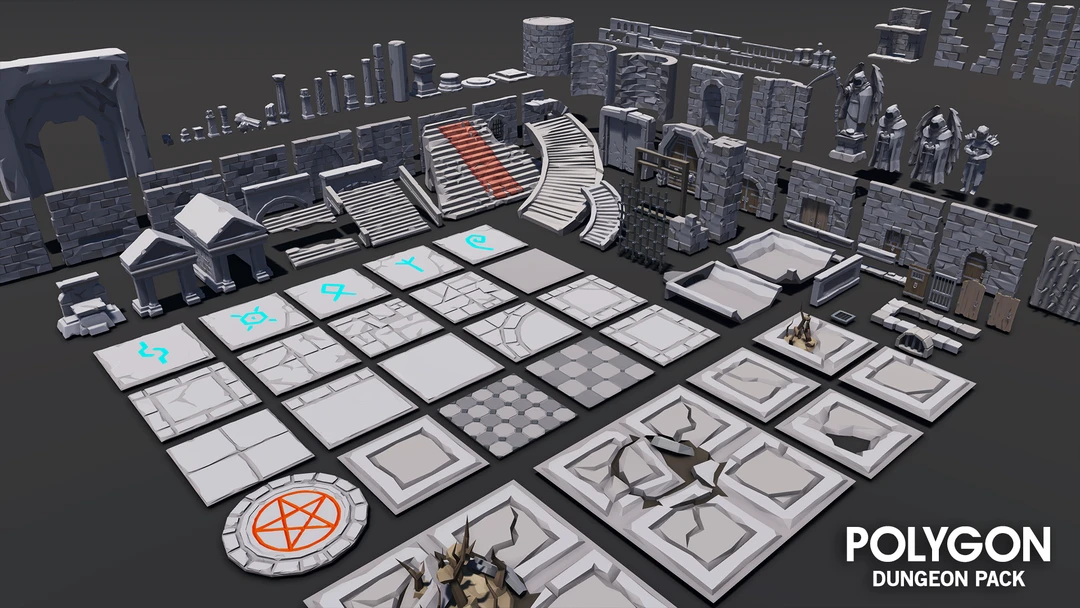
* **If the solution does not build (and run), I will not mark it** (so you will receive zero on questions that can't be built and/or won't run). When I say "run", I'm not speaking about some, random, unforeseen bug, but rather something that you should have obviously dealt with, like memory exceptions, etc.
* No “**auto**” or the **boost library** (or any other 3rd party library we haven’t used or you haven’t cleared with me first).   
  If you use either, I will not mark your submission and you will receive a mark of zero.  
  No exceptions.

Details:

You are to create one or more “ruined buildings” made up of the parts from the Synty Studios “POLYGON - Dungeon Realms” (<https://syntystore.com/products/polygon-dungeon-pack>) and “POLYGON - Pirate Pack” (<https://syntystore.com/products/polygon-pirate-pack>).

Many of the “environment” assets can be combined like LEGO bricks into any shape you’d like. They are all of similar size. In the picture below, you can see there’s a bunch of different “floor” models as well as a number of “wall” models.

These are in the “3D\_Models” folder.



**Some notes about the models:**

* They are aligned in one corner of the model. This allows you to “snap” them together more easily (MeshLab “Render”, then “Show Axis” will show this.)
* The textures that correspond to the model are shown in the ply file, for example, the SM\_Env\_Floor\_Grate\_01.ply file has the following:  
    
   comment TextureFile Dungeons\_2\_Texture\_01\_A.png   
    
  This means that you need the Dungeons\_2\_Texture\_01\_A.png file as a texture.
* I’m pretty sure I’ve converted them all to an xyz+normal+rgba+uv format, but there’s 100s of files in there; if I haven’t, then you should be expected to convert and/or load them yourself (or ask me nicely…).
* Since this is isn’t a “graphics” project, you’re not getting any marks of now gorgeous this is – just if I can clearly see what’s going on. So a single “directional” light is OK, don’t use wireframe unless it clarifies something (like maybe where the beholders are when blocked by walls, etc.).
* The beholder model is the same as the Fall 2022 final exam. They have textures, but the also have the textures “baked” into the vertex colours (note the dungeon models *don’t* have textures baked into the vertex colours).   
  It’s facing along the negative (-ve) Z axis. The “cone” is a separate model, and if you draw it at the same location and orientation, it will align with the large eye of the beholder.   
  (You don’t have to use the cone if you don’t want to, but it does clearly show where the beholder is looking – maybe make this wireframe or semi-transparent or something?).

Diagram

Description automatically generated

The “maze maker” code is something I found online a while ago, by Jaden Peterson. Thank you, Jaden Peterson!

Here’s the original link: <https://codereview.stackexchange.com/questions/135443/c-maze-generator>

The cMazeMaker\_W2023 class is his code, placed inside a class, along with some additional code specific for this project.

What I’m saying is that you are free to use his original code if you’d like, but the class might be much easier for you.

What the class will do is generate a maze (I know, shocking!) and if you make a *really* large maze (you *will* be making a very large maze), it will also allow you to isolate (“see”) only a smaller portion of that maze.

Your mission, should you choose to accept it…:

1. (5 marks) Using the cMazeMaker\_W2023 (or equivalent), make a HUGE maze that’s at between 1,000 and 10,000 on each side (so that’s a square maze with the width and height being in that range).   
     
   Keep in mind that the number of “cells” or “rooms is the square of that (100x100 🡺 10,000 “cells”, 1,000x1,000 🡺 1 million “cells”, etc.), so if you pick a *really* large number, it’ll take a *long* time to run *and* use a lot of memory.  
     
   Here's the times I got on my machine (an i7 laptop, running on one core):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Memory used** | | **Time** | |  |
| Size (n x n) | Cells/Rooms | Debug | Release | Debug | Release |  |
| 10 | 100 |  |  | 1.08 | 0.15 | ms |
| 100 | 10,000 | 3,564 K | 976 K | 113.71 | 4.12 | ms |
| 1,000 | 1,000,000 | 295 M | 53 M | 12.49 | 0.43 | seconds |
| 5,000 | 25,000,000 | - | 1.34 G | - | 14.65 | seconds |
| 10,000 | 100,000,000 | - | 5.1 G | - | 52.40 | seconds |
| 100,000 | 10,000,000,000 | - | 500 Gbytes??? 🤷‍♂️ | - | 90??? 🤷‍♂️ | Minutes |

(Note that I never ran the ones with the “-“ or the last row – those are guesses/estimates based on the general trend)

You can run this and save it to a file, or run it when the program starts, whatever you think is easier and/or faster.

Note you are *not* being marked for the speed of this part or the size of the maze (as long at it’s at least 1,000x1,000).

1. (200 marks) Using the “POLYGON Dungeon Realms” components, make floor and walls of your maze.   
     
   Since the whole maze is far too large to display, you are only to display a portion of the maze “close” to your camera. This is up to you, but likely something like a 20x20 to 50x50 section of the entire maze. GetMazeAreaSquare() or GetMazeAreaRound() will return a portion of the maze centred around a particular grid cell/room of the maze.   
     
   From this information, display a portion of the maze using the 3D models. This process should be multi-threaded in that the updating/replacement of the list of objects should be produced on another thread, and “consumed” (drawn) by the drawing code.   
     
   Feel free to use the maze code (or any other code) from class.   
     
   Your code should be *significantly* different from the in class demo code both in how the 3D models are chosen and how the threading (and possibly buffering, locking, etc.) is accomplished.
2. (200 marks) Draw a number of beholders scattered in the maze. You’ll need enough that one could move around the maze and come across one fairly regularly. In other words, you shouldn’t have to move around for a few seconds without seeing some. I could guess you’d need 100s of them at least.   
     
   These beholders will move through the maze at random. Specifically, they should go down hallways until they come to the end, the “randomly” pick another direction. This includes “dead ends” where the beholder would only have the “choice” to turn around and go the other way.   
     
   The beholders shouldn’t change direction in the “middle” of the hallway, though. In other words, if they start going down a long hallway with no branches, the stay in that direction *and* if they come to a hallway with a *single* turn, then will continue to “move forward”, taking the only turn. One way to implement this is to simply not allow the beholder to “turn around” unless it comes to a “dead end” (can’t go forward, left, or right from it’s perspective).
3. (200 marks) Place the behaviour of each beholder on its own thread. This will mean that they would have to share the maze information (and in the next question, the location of each other).   
     
   NOTE: All subsequent beholder “behaviour” needs to be executed on separate threads.
4. (200 marks) If two beholders “see” each other, they will approach to within one “tile” (the size of the 3D models) and “attack”, where one of them “dies”.   
     
   When a beholder “dies”, it needs to physically shrink, change colour in some way, and fall on its “back”.   
   Think like it was burned up to a charred remains or something like that. They need to stay on screen, though!   
     
   Once they are “dead”, they are ignored by all the other beholders. If you make the “dead” beholders small enough, and the “alive” beholders are floating high enough over the ground, then they can just float over the dead ones.   
   (But if they just pass through, that’s OK, too)  
     
   Beholders can “see” another beholder if they are directly in front of them and not blocked by any walls.   
     
   It’s possible that one beholder “sees” another, but the other beholder is *not* seen. This can happen if a beholder turns into a hallway where the “seen” beholder is heading away from it, or passing by it from the side. In this case, the one beholder will pursue the other beholder (as above), but the other beholder (the one that doesn’t know it’s been seen) just continues to wander around. This “seen” (but oblivious) beholder might turn around and see the other one, or might be chased around the maze for a long time.
5. (200) Handle multiple beholder interactions, specifically where beholders can only attack one beholder at a time.   
   This means that:  
   * If there are multiple beholders, it will only pick one to pursue and attack If it “kills” one beholder, it goes back to wandering and may not see any other beholders – like if it happens to be facing away from one.
   * If a beholder is chasing another beholder and is interrupted by another beholder, it should attack the “closest” one (that’s up to you, but I’m looking for something reasonable here). If it ends up killing this new target, it will “forget” about the original beholder it was pursuing.
   * If two (or more) beholders kill another beholder, they all go back to wandering about. Now, they will almost certainly “see” one of the other beholders right away…
   * Basically, the beholders can only “focus” on chasing and/or killing one beholder at a time. Once they are trying to kill a beholder, they don’t really care which beholder it was, just that it’s a “one chase = one kill” sort of thing.