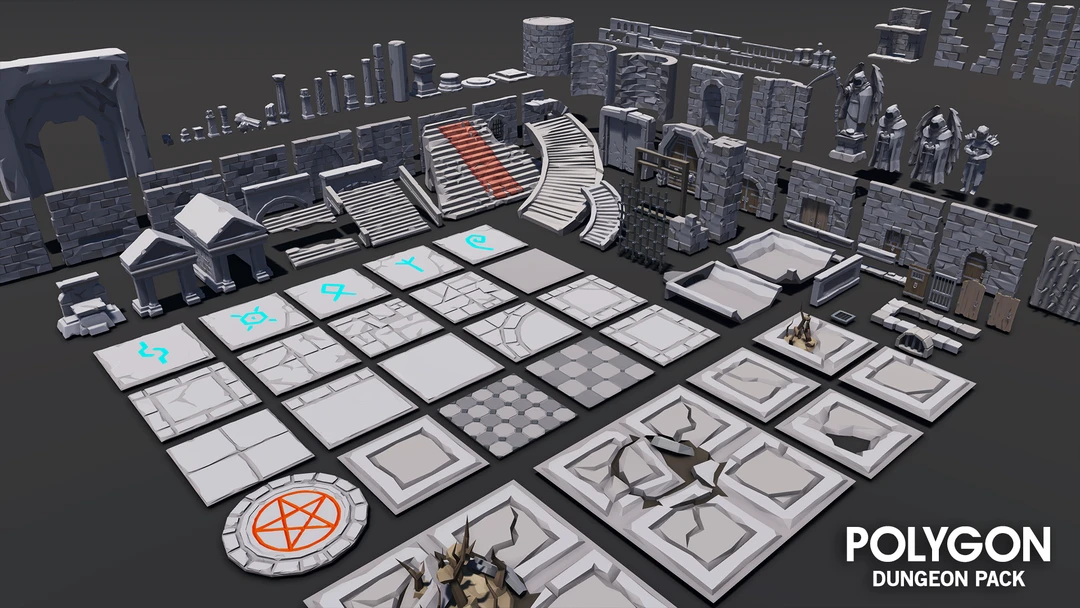
|  |  |
| --- | --- |
| Program: | CPA3 |
| Course: | INFO3111 – C++ Graphics – Summer 2024 |
| Professors: | Michael Feeney |
| Project # 1: | Basic 3D scene of loaded, transformed models |
| Weight: | 15% of your final mark |
| Due Date: | Thursday, June 6th, 2024, before the start of class (@12:00 PM)  (Note: This is due *after* the mid-term on Tuesday, June 4th)) |

Description and Purpose

You are to create a 64 bit Win32 (Windows 7/8/10) + OpenGL 4.x application that displays a “sensible” static (or dynamic) 3D scene using external models that have been translated in the scene using transformations, and either coloured or lit using a basic GLSL shader. You will have to be able to interactively manipulate (i.e. “move”) the camera to view the scene.



**You are going to make a “dungeon” or “labyrinth” using modular models from the Synty Studios “POLYGON - Dungeon Realms” (**[**https://syntystore.com/products/polygon-dungeon-pack**](https://syntystore.com/products/polygon-dungeon-pack)**).**

The Models and textures:

The models are in the “Dungeon\_models” folder under “Projects” and “Project\_1” in the class github repository. These assets can be combined like “LEGO” bricks into any layout you’d like. They are all proportional sizes. In the picture (on page 1), you can see there’s a bunch of different “floor” models as well as many “wall” models. You can make something like this:

A room with furniture and lights

Description automatically generated

**Regarding colours and textures:**

These models are textured, *but we haven’t looked at textures, yet* – but there are “texture coordinates” in the model files that you’ll have to deal with when you load them.

If you place the texture that’s listed beside the “comment” (it’s almost always: “comment TextureFile Dungeons\_2\_Texture\_01\_A.png”) in the same folder as the model, you can get a sense of what the models would look like fully textured.

For this project, ***you only need to assign a solid colour to the models***.

Dungeon layout and “decorations”:

To get the layout of your dungeon, run the **INFO3111Project1\_S2024\_DungeonMaker.exe** program.   
This will generate five (5) variations of dungeons and you can pick the one that you’d like.

**Note that you need to submit your log file**, too.  
  
(this log file is the same output as on screen and is just a text file).

**Some notes about the INFO3111Project1\_S2024\_DungeonMaker.exe program:**

* This isn’t some work of amazing art, so there’s no error checking in it because, seriously?
* Now, if you enter correct info, it will work, but if you enter stupid nonsense, then it likely won’t.
* It will generate a file called INFO-3111\_log\_XXX.log, where “XXX” is a **number that’s generated based on your** student number (but it’s not your student number – it’s to catch sneaky cheaters).   
  This is the same output that you see on the screen.

**You MUST include these log files with your submission.**

**If you do NOT include these log files, your exam will NOT be marked (and you will get a mark of zero).**

If you screw up the files, then regenerate them – the values should be the same and is based on your student number.

**As an example, assume that your log output on the right 🡪**

XXXXXXXXXXX.XXXXXXX

X.....X.X...X.....X

XXX.X.X....XX.XXXXX

X...X.X...........X

X.XX.......XXXX.X.X

X.......X.........X

X.......X.........X

X.................X

X.XX....X.........X

X.........X.....X.X

X...XXX...XX....X.X

X.........X.......X

XX..X...........XXX

X...X.............X

X...X...........X.X

X.X.X...........X.X

X.X.X......XXXXXX.X

X.................X

XXXXXXX.XXXXXXXXXXX

Note: you will use *your* ***own***dungeon layout **\*NOT\*** this one.

The “X”s mark a “wall” and the “.” indicate just a floor.

You can make the “walls” by combining four (4) wall models to create a “pillar”. If they are beside each other, then will blend into a complete wall.  
(if you want something fancier, see the “bonus” questions later.)

Note: If you are competent in blender/Maya/3DsMax, etc. you might be tempted to add all the models in there, combine them to a single model and load that, but I would *strongly recommend* that you *don’t* do that, as the models for the mid-term and subsequent checkpoints, project 2, and final will assume that most (or all) models are separate items in the scene that can be manipulated independently.

Details:

1. (50 marks) Create a floor and outside wall.   
     
   Note: You are *not* making a “roof” to this dungeon, just that walls and floors.   
   (We can pretend there *is* a roof, but we can magically look through it, or something like that…)

* The “X” are walls or pillars and the “.” are open areas (i.e. they are just floor)
* Note that the wall and floor models are not the same thickness. In other words, the “X” characters around the outside are much thinner (they are not the 4-sided pillars). You are making this outer wall.
* In this question:
  + - Make a large open space with at least six (6) variation of the “floor” models (in the “Floors” folder).
    - Place walls around the outside using at least six (6) variations of the “wall” models.
* Note that there’s one or two entrances to the room (a break in the “X”s around the outside. In the example, you can see this at the bottom and top.
* Add enough evenly lit lighting to see everything, like the “dark” ambient lighting.   
  This should be “bright enough” but not “too bright” – it’s an underground dudgeon, right?
* Apply appropriate colours to the walls and floor (I’m assuming some grey stone colour?).

1. (100 marks) Fill in the rest of the walls inside the room/dungeon.

* The “X” inside can represent walls or pillars/columns.
* If there’s a few “X”s in a row, then it’s a wall.
* If there’s an “X” by itself (i.e. not touching another “X”), then it’s a pillar or column.
  + Note that if an “X” touches another “X” *at a diagonal* then it’s *not* connected.  
    i.e. it’s not part of the wall or it’s an adjacent pillar.   
    There aren’t any “diagonal” walls.
* Walls need to be placed right on the floor (or slightly penetrating if that looks better).
* Pillars/Columns are made of four (4) wall models in a square, the size of a floor model, making a square shape, with the “backs” of the wall models facing towards the inside.
* Colour the wall models a sensible colour.

1. (150 marks) Place several lit “torches” in the scene:

* Choose one type of “torch” (light) model in the “Torches” folder.
* Place **eight (8)** of these torch models **spread out evenly in the scene**, attached to the walls.   
  The idea is that they are torches with fire or candles, attached near the top of the wall.   
    
  (If they are all clumped close together or some bizarre size or sitting on the floor or in a line along one wall, or some other strange situation, then I’m going to assume those torches *aren’t* there.)
* Place small, bright **red-orange** (i.e. *not* white: candles don’t give off “white” light) lights where the fire/candle of the torches would be.   
  These should light up the area nearby (like a small candle or flame would).
* Make the torches “flicker” by slightly perturbing the linear attenuation each.  
     
  You do this by first setting up the “regular” (or non-flickering) attenuation you’d like.   
    
  Then, for each frame, adding a small random number to this “base” attenuation of the light and passing *that* (i.e. total) to the shader. Don’t update the original “regular” value, though. The idea is that you’re *passing* this slightly adjusted value each frame, not saving it.   
    
  Make sure this small random number is negative as well, so when you are “adding” this small value, the torch can get dimmer as well (adding this small negative number makes it slightly dimmer, while adding the small positive number makes it slightly brighter).

1. (100 marks): Place the “Add the following props to your dungeon:” items “randomly” through the dungeon.   
   * “Randomly” means they “look” random enough and aren’t, say, dumped in the same place in a big pile, in a line, or some other lazy/strange placement.
   * Some items *can* be together if that “makes sense”, like a mug, candle, and barrel beside a dead body/skeleton “makes sense”. Three (3) mugs beside themselves with nothing else there? That seems a little odd – like how did that happen?
   * Note that you must choose *different* items where possible. For instance, if you need to place three (3) skeletons, then they must be *different* skeleton models.
   * If you *must* repeat items, that’s OK. Like if need to place ten (10) bottles, you only have seven (7) models to choose from, and two of them have candles in them…
   * Some items might require another (additional) items: like there might be a “mug” lying on the floor, but it’s unlikely that it’s sitting upright. It’s more likely that it’s lying on its side (like it fell) or on top of a barrel (if its upright – i.e. someone was drinking out of it).
   * Similarly, the candelabra/candle – it’s very unlikely an upright candelabra is just sitting in the middle of the floor somewhere, right?
   * If you place everything in one big pile or in one “room”/location, then I reserve to right to give you almost no marks for this item.
2. (50 marks): Place several crystals in the scene.   
   * Place at least **five (5)** of the “crystal” models (in the “Crystals” folder) throughout the scene. These should look like they have randomly been scattered or dropped or were just naturally there. Like they shouldn’t be dropped together or in a line, etc.
   * Make the crystals *different* bright colours (like gems or crystals you’d see in a child’s cartoon).
   * Make the crystals very “shiny” (i.e. have a very high specular component).
   * Like question 4, if they are all placed in one spot, I might give you almost no marks for this.
3. (150 marks): Place a few plants that are growing in “holes” in the ceiling.   
     
   Pretend that the roof of the dungeon has some holes that have caved in.  
   * Place three (3) “groupings” of plants in the room.
   * Each “grouping” consists of a **single tree** with a **few smaller plants** around them.
   * These “groupings” of plants should be scattered around the dungeon.
   * To mimic the “holes in the ceiling” by placing brighter whiter/yellow lights “shining down” (i.e. above) these three (3) plant groupings.
   * These lights should only light up the plants (and “nearby”) – like a hole in the ceiling would.   
     Like it shouldn’t light up the entire dungeon.
4. (20 marks) Place the “SM\_Env\_Camp\_Well\_01” model somewhere appropriate in the dungeon.
5. (60 marks) Using the keyboard, set up variations of camera locations:

* Pressing the “1” key will move the camera above the model, looking down at the entire dungeon, like the angle of the camera in the page 2 image.
* Pressing the “2” key will move the camera to look at one of the groupings of plants, as if you were inside the dungeon, about eye height.
* Pressing the “3” key should place the camera inside the dungeon, looking at one of the unfortunate skeletons.
* Pressing the “4” key should place the camera inside the dungeon, looking at the largest crystal you have, showing off its bright colour and shininess (high specular value).

Note that the regular camera movement controls should still work after these buttons have been pressed.   
Like you aren’t “locking” the camera into this location, just placing/moving the camera there.

**BONUSES (5% overall each):**

* Instead of using “pillars” (4 wall models) for the inside walls, just use two sided walls (thin walls).
* By pressing a button, mimic the *gradual* change of a day-night cycle by *gradually* change the light that is coming in from the holes in the ceiling. This light is coming from the sun, so it would *gradually* get brighter, then gradually dimmer until it’s gone (and you only see the candle/torch light), then it’s dark (nighttime) for a while, then it’s dawn and the cycle repeats.
* Alter your project so that you can pass in a different dungeon file (the “X and .” file).
* Add an option to prevent the camera from going through the “walls” so you can mimic moving first-person-style through the dungeon.   
    
  There are two options here:
  + Classic old-school dungeon crawler style where you can only turn 90 degrees at a time and move one “tile” at a time: <https://www.youtube.com/watch?v=GQgHSX2tof4>
  + (*Another* +5%) Modern, where you can go forward and turn at any angle.   
      
    To do either of these, you’ll have to update both the eye position and the target position.
* If you can think of another interesting thing that could be a bonus, let me know.

You will submit:

* **Your entire solution** (PLEASE remove the “extra” files from it, making it smaller), and compress it.
* **A video demonstrating your application.** This can be using OBS or zoom or FRAPs or even your cell phone camera but must *clearly* show:  
  + You launching the application from visual studio.
  + CLEARLY showing some identifying information (your name and student number) and the date.
  + What you are doing (buttons being pressed, etc.) as it is happening.

Additional requirements:

* While you may freely “borrow” mine code, your code should be “sufficiently different” from mine in terms of the output on screen. See the "plagiarism" test, later in this document, for more details.
* Further, you *cannot* simply use an existing game engine (or part of a game engine), even if it's a "from source" engine (i.e. you have the entire source) to complete this assignment; it should be either completely new of significantly modified. This includes, but is *not* limited to: Unity, Unreal, Cry, Anarchy, XNA, Cocos, Ogre, the framework from the OpenGL text, etc.
* This also includes chatGPT (or equivalent) AI code generation.  
    
  **Related to plagiarism/cheating and not doing much work:**   
  + If you simply submitted the in-class code, then you invested zero time, so you did no work, so you get a mark of zero (in that case, it is a clear case plagiarism/cheating, and an academic offence would also be submitted).
  + If you took the in-class code and made some trivial changes - like replaced the teapot model with the rabbit model, slightly repositioned them, and maybe changed their colour - you might not have technically "cheated", but you did essentially no work: "How long would it take me - your instructor - to make those changes?" If it's something that would take 10 minutes, you won't get many parks for that.
  + It must be something that a random "typical" person would say "yes, that's significantly different" in order to "be different".

Here’s the thing: I know the code *\*I\** wrote and if I’m suspicious that you had didn’t really write (in other words: understand) the code you submitted, I’ll simply ask you about it face-to-face. Like why you did this or that, how you might do it differently, ask how you might have changed stuff. If I get the sense that just “got this code somewhere” and don’t understand what it’s doing and/or why it’s there, I’ll assume you did (get it from “somewhere” – so you didn’t write it yourself - and file an academic offence.

* If you code does not even compile, I will not mark it. Since it can't run, you would get a mark of zero.
* If you code does not build (i.e. linker error) and run (i.e. no crazy run-time crash that is unexpected), I may investigate this further, but only if there is some simple problem and/or *very* slight and/or *very* obvious (and easy to fix) configuration error or last minute typo. (I might do something minor like try to unzip the libraries files that github stripped out or something along those lines, but I should be able to download it, build it, and run it, without incident.)
* It **MUST** build and run in **RELEASE** mode (64 bits).
* Warning level 3 is fine. In fact, I’ll almost certainly completely ignore any “warnings”. You can’t even build most common libraries on level 4, so don’t even bother.
* **No “boost” library or “auto” keyword. Use of these will give you a mark of zero, no exceptions.**

Project Corrections

If any corrections or changes are necessary they will be posted to the course web site and you will be notified of any changes in class. It is your responsibility to check the site periodically for changes to the project. Additional resources relating to the project may also be posted.

80/8-year old “squinty eye” plagiarism test:

I have very little tolerance for plagiarism, but some students might be unclear about what it is.

Basically, it’s submitting somebody else’s work as your own.

There is sometimes some confusion over this because you could argue nothing is actually “unique” (see: <http://everythingisaremix.info/> for a fascinating overview of this).

The whole point of assignments/tests/projects in this course (or any course, really) is to try to see if you are actually able to ***do*** the coding that’s asked of you. In other words: How competent are you? Handing me someone else’s code and/or making a trivial change isn’t good enough.

Also, it’s illegal:

* <http://www.plagiarism.org/ask-the-experts/faq/>
* <http://definitions.uslegal.com/p/plagiarism/>
* <http://en.wikipedia.org/wiki/Plagiarism>
* <https://www.legalzoom.com/articles/plagiarism-what-is-it-exactly>

In other words, I’m not going to be drawn into a giant debate over how “different” your code is from mine or anyone else’s, if any sensible person (including me) would conclude that the code/application is pretty much the same thing, then it is. It is up to my discretion to decide this.

* While you may freely “borrow” mine (or anyone other) code ***but*** your code should be “sufficiently” different from mine (you might want to replace the word “sufficiently” with “significantly”).
* In other words, you *cannot* simply use an existing game engine (or part of a game engine) to complete this assignment; it should be either completely new of **significantly** modified.
* How will I determine this?
  + If I showed your application and/or your source code to either my pragmatic 80-year-old mother, or a typical 8-year-old, or even some random person walking down the hallway (i.e. a non-expert), and they looked at it, tilted their heads, squinted their eyes, and said “you know, they look the same,” then they ***are*** the same.
  + Another test would: How much time it would take for a "competent programmer" (me, for example) to make the changes you are submitting? The point here is that I don’t “care” if you tell me “But it took me *weeks* to make the changes!” Fine, but if I can make those same changes in 10 minutes, then not a lot of work has been done (certainly **not** sufficient work for someone who is trying to convince me they are proficient).