Computer Graphics CS 523
Project Milestone: Bharata: An AOE like Isometric strategy game

Name: Mihir Kulkarni NetID: mak575

Intended Objective:

The goal of the project is to develop a 2D Isometric Strategy game using OpenGL from the ground up. The goal of the project is to focus on low-level graphics rendering techniques.

The objective of the project has shifted from just a 2D game, as described in the project proposal to an Isometric styled game. The primary reason for this change was purely aesthetic and as a tip to a childhood favorite game of mine, Age of Empires II.

Technical Objectives as outlined in the Project Proposal, with current progress:

- Implement a clickable map-based level selection screen – This is yet to be implemented but is primarily a UI component. I suspect that it **would be an extension of the logic already employed for the clickable Isometric Grid.**
- Develop a 2D game grid with a tile-based rendering using OpenGL – **This has been successfully implemented** and is being worked on further to render specific textures based on the value of the grid cell.
- Implement Unit selection, movement and combat mechanics – **Movement has been implemented to a good degree using A* search algorithm** which is using the cell that is clicked as the destination for a "player cell" to transition to. Researching implementations of open source games/projects like 0AD and Open Empire to understand unit selection and combat implementations.
- Improve visual quality by adding textures, lighting and shading – Have not gotten to this stage yet.

Changes in the choice of Tech Stack:

Previously, I had mentioned the use of GLFW library for window creation and OpenGL contexts. I have decided to instead use Simple DirectMedia Layer II (SDL2) as it is considered a better alternative for game development.

This is because SDL2 is a full-fledged multimedia library that will not only allow me to create windows and handle OpenGL contexts, but will also provide me with the API for handling sound, timers, I/O etc. This can be seen with a simple function SDL_Delay that I could very easily use in my MainLoop to provide the feel of movement to my demo. GLFW would be preferred for just window handling and context management, but the project will be more expansive than that.

Functioning:

The project makes use of the following defined modules:

1. src/grid – This module is primarily responsible for generating the randomized grid. This utilizes globals like MAP_HEIGHT & MAP_WIDTH in the *grid_init* function to generate a **traversable randomized grid.** This is done with the help of some helper functions like *blocked_cells*, *dead_end_cells*, *count_open_neighbors*. We also have defined two other functions related to placing elements on the map, which will be utilized later for other objects on the map. The *move_player* function is used for traversal.
2. src/map – This module is driver for the Isometric grid. The only function being defined here is the *getVertexData*. We create a vector of floats that will hold our vertex data. This will be used to populate our VertexBufferObject (VBO) and subsequently, along with the color information – (and later the texture coordinates) - the VertexArrayObject (VAO).

   An isometric grid is a graphical representation designed to simulate three dimensions on a two-dimensional surface. Isometric grids use diamond-shaped tiles that create a perspective effect, giving the appearance of depth and allowing for the visualization of a 3D environment.

   I find the gridWidth and gridHeight in pixels using the defined TILE_HEIGHT and TILE_WIDTH. This is further used to determine the offsetX and offsetY in order to center this grid on the display.

   We then find the center of each cell in the isometric view. The (col-row) moves the tile horizontally while the (col+row) moves it vertically.

3. src/ai – This module implements the movement logic for the player cell. This is done with A* search algorithm and I have implemented this based on a project I had previously built where a bot would find a path to a button in a maze while a stochastic fire spread through the maze.
4. main.cpp
    a. calls *grid_init* to get the generated grid
    b. creates a random initial point for the player and places it on the grid.
    c. *InitializeProgram* creates the OpenGL context and window.
    d. *MainLoop* consists of the main game loop
        i. *Input* polls for SDL events like quitting the window and the clicks from the mouse.
        ii. *PreDraw* – Uses the movementQueue global queue data structure to implement the movement of the player by modifying the grid using call by reference
            1. *VertexSpecification* – Calls the map module to get the vertex data, create VBO, VAO, bind the buffers, create the shader program, reads

the shaders using *readVertexShader* and *readFragmentShader*. We also call the *glm::ortho* function to enable orthographic projection for the isometric view.

iii. The *Draw* Function uses the Vertex Array Object to draw the arrays and render the screen

Current Work:

Currently, I am working on using a sprite sheet as the source of the textures for the Isometric map. I will be using the same logic used for assigning vertices for Isometric coordinates. This way we use the sprite sheet pixels to get the texcoords.

I have limited the scope of the map to only 10*10 for development purposes but have also tested with expanding this, which was done without any problems. I will also be modifying the screen size to be full screen.

Currently, the window closes when I click any point outside of the rendered grid, and the player cell also spawns outside of the 10*10 grid sometimes. This is due to lack of bounds checks which I will be looking into and implementing.

Conclusion:

I have given an overview of the progress made and detailed the working of the current state of the project. I am on track to enable textures and will soon expand the scope to enable larger quantity of objects and more dynamic movement on the screen. The next breakthrough should be the implementation of textures after which I can start working on cleaning the code a little with more functions to enable more entities and multiple unit selection. I will then proceed towards lighting and animation of sprites.