Dungeon Crawler

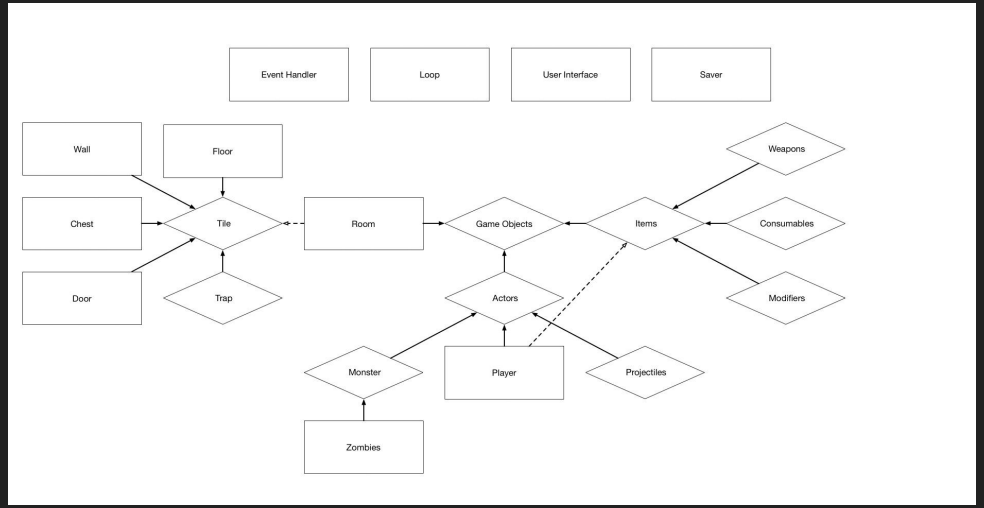
Charles Emerson
Christian Fraser
Thatcher Lane
Bremner Nickisch

Concept

- Roguelike dungeon crawler
- Procedurally generated
- Text based such as ASCII or even something like Zork
- Items

Structure

- Event Handler
- Game Objects
  - Parent Class, everything derived From here



- Level Class
  - Evolution of room class
- Aggregation Based vs Inheritance

Graphics

- Tilemap which contains different 32x32 textures are loaded onto the screen to create a dynamic map.
- By putting everything into fixed size tiles, it was easy to check for collision and boundaries for the map
- Sprites for the player and enemy are drawn on top of the tile map and interact with each other and the map itself
- Sprites had their own sprite maps for each direction they can face. Each texture is assigned to each direction and drawn depending on the movement.
- Level::draw draws every entity in the map from iterating over a vector that contains all entities

Features

- Menu
  - Save, Load, and Exit from Game
- saveGame()
  - Calls level.toString()
- loadGame()
  - Reads in from save.txt
  - Same order as printed by save function
  - Sets object variables and loads game

toString( )

```
std::string Level::toString()
{
    std::stringstream sStream;
    sStream << m_Items.size() << '\n';

    if (m_Items.size() != 0)
    {
        for (auto it = m_Items.begin(); it != m_Items.end() - 1; ++it)
            sStream << it->toString() << ' ';

        sStream << m_Items.back().toString();
    }

    sStream << '\n';
    sStream << m_Enemies.size() << '\n';

    if (m_Enemies.size() != 0)
    {
        for (auto it = m_Enemies.begin(); it != m_Enemies.end() - 1; ++it)
            sStream << it->toString() << ' ';
        sStream << m_Enemies.back().toString();
    }

    sStream << '\n';
    sStream << m_Player.toString() << '\n';
    sStream << m_stairs.toString() << '\n';
    sStream << m_map.toString() << '\n';
    sStream << m_sizeOfTileMap.x << ' ' << m_sizeOfTileMap.y << '\n';
    return sStream.str();
}
```

loadGame( ) Sample

```
//read in enemy values
file >> enemySizeString;
int enemySizeInt = atoi(enemySizeString.c_str());

//resize enemy vector
level.m_Enemies.resize(enemySizeInt);

for (int i = 0; i < enemySizeInt; ++i)
{

    //read in enemy attributes as strings
    file >> tempEnemyXCoorString;
    file >> tempEnemyYCoorString;
    file >> tempEnemyHealthString;
    file >> tempEnemyDirectionFacingString;

    //Convert string enemy attributes to int values
    int tempEnemyXCoorInt = atoi(tempEnemyXCoorString.c_str());
    int tempEnemyYCoorInt = atoi(tempEnemyYCoorString.c_str());
    int tempEnemyHealthInt = atoi(tempEnemyHealthString.c_str());
    int tempEnemyDirectionFacingInt = atoi(tempEnemyDirectionFacingString.c_str());

    // Sets game attributes for e_Enemies[i]
    level.m_Enemies[i].setGameCoordinates(sf::Vector2i{ tempEnemyXCoorInt, tempEnemyYCoorInt });
    level.m_Enemies[i].setHealth(tempEnemyHealthInt);
    level.m_Enemies[i].setDirectionFacing(intToEnum(tempEnemyDirectionFacingInt));

    // Actually change the drawing coordinates of the Sprite to match the GameCoordinates
    level.m_Enemies[i].getObject().setPosition(sf::Vector2f{ 1.0f * tempEnemyXCoorInt * level.
        m_tileSize.x , 1.0f * tempEnemyYCoorInt * level.m_tileSize.y });
}
```

Possible Improvements

- More Items
  - Different weapons
  - Consumables
  - Modifiers
- Enemies
  - Smarter AI that chases players
  - Generate several enemies per floor
- Map Generation
  - Verify path to door

Play the game?