**Update 0**

My name is John, and my friends name is Chris. We have wanted to make a 2D RPG style game for some time and we finally decided to start work on this game.

Our plan is to develop a Roguelike RPG style game. However we want to introduce a story, and a world that the player can interact with, much like Dwarf Fortress and more recently the Caves of Qud. Bear in mind our aim in the project is not to make money, so we don’t see them as competition. My aim in this project is to learn how to develop a game without the use of pre-existing engines and Chris’s aim is to improve his artwork.

We also decided to post our progress here for a few reasons. First, we would really appreciate any and all feedback on our progress from other people. Second, the motivation helps. If the project kicks off, and you guys like it, the desire to not let you guys down will really push us further.

Ok enough introductions, time to talk business.

As any standard roguelike, the game will take the form of a grand turn based game. In essence the game loop consists of 4 steps: (Eat, Sleep, Rave, Repeat… Joking!)

1. Draw the world.
2. Wait for user input
3. Process the inputs effect on the player.
4. Update any AI enemies / NPCs present / Scene changes etc
5. Rinse and Repeat.

Now personally, for me I realise this project will get very complicated programming wise. To make things a little easier on myself, I will be using SFML to help me. (I really don’t feel like trying to program OpenGL/DirectX manually - maybe in the future.)

In terms of my approach to the programming side of this project, I aim to take baby steps, and develop this piece by piece.

I will start with absolutely essential backend bits. A basic GameLoop, and Scene Manager to start. The job of the Scene Manager is to simply see where the game is in its running cycle, and show the appropriate scene accordingly. This will be followed by a base class for Scenes, and some opening credits for the game window giving credit to DragonDePlatino and DawnBringer (explained below in the art section), SFML, ourselves and any other special mentions that come along the way.

I will also be updating my GitHub Project every time I do some work on this game, as well as a follow up blog posts on this forum. If you wish to see the source Code, please take a look at the below GitHub link:

https://github.com/Tarolion/Project-Roguelike

In terms of art, Chris and I are thinking 2D Pixel art graphics would fit this very well. Personally, I hope to see a nice vibrant range of colours. However, while Chris works on art for the game, I found a really lovely set of tiles made by DragonDePlatino of OpenGameArt.org and of course DawnBringer. It’s called Dawnlike, and can be found here:

http://opengameart.org/content/dawnlike-16x16-universal-rogue-like-tileset-v181

Absolutely lovely I’m sure you guys would agree. So there’s our placeholder graphics.

**Update 1**

So here it is, end of day one. Not much to really show for it as I’ve only worked for maybe 2/3 hours on this so far….

I wrote a little INI file parser ready to start reading settings files galore. Will need to later modify the class to allow saving of changes, I did include the method ready, but not programmed yet.

I also wrote a Game class. Now this game class will basically be the heart and soul of the game. It contains the game loop, as well as a reference to the window pointer. Now I’m not sure how useful this will be at present, as I intend to pass the window Reference through a SceneManager to all objects that draw stuff to the screen, but at least it is there if I need it.

The final thing I added was the Settings class. This class is pure static; it will simply hold some very useful variables that I’d like to be global, i.e I can access from anywhere. Now I know that won’t happen… Never does, unless I’m doing something wrong. (Probably am to be fair.) Right now it simply holds information regarding the window’s Width, and Height, as well as the Tile Size referring to the resolution of an individual tile, and whether or not the window is in Windowed mode, or fullscreen.

The basic introduction of SFML to the project right now simply includes a small window that appears when the game opens. Doesn’t do much but it does read the window’s resolution from the settings.ini file in the resources folder. A great achievement for myself let me tell you. However although it uses the width and height, it does absolutely nothing with the Windowed Property at present. Also fun fact, if the settings.ini file doesn’t contain a numerical value for Width/Height/TileSize, the game will crash out due to an std::stoi invalid parameter error. The same is true if the field is empty. I can fix this really easy with a quick error check before running the stoi function.

Anyway TODO list for tomorrow is:

1. Perform an Error check in Settings.cpp before applying settings values to the class.
2. Develop a Save Ini File Method for the File Parser.
3. If the settings.ini file cannot be loaded in Main.cpp. Generate it using the save method.
4. Program Camera class
5. Program Scene Manager Class
6. Program Scene Base Class

I think that should be enough for tomorrow. Just in case I am able to achieve this and still have time left over, a good stretch goal would be a Credits Scene class ready to tell the world the names of everyone who helped make the project what it is…. (A big black box in the middle of your screen… Great!)

**Update 2**

What can I say…? It’s been a long day. While suffering through a mixture of sleepiness and nagging parents, I have soldiered on with another update.

I achieved all but the stretch goal from yesterday. I decided on some nomenclature for the code. From now on, if a player causes an update manually, then it will be known as the Scene’s Processing an Event. I basically poll for an event, and pass it onto the scene in a “Process” function This will be the guy responsible for calling the AI updates later on. Scenes also Update once per frame, I thought I’d use this for Animations etc. I figured since we’re using 2D graphics instead of ASCII, since we want it to look pretty later, we’ll want moving parts that don’t necessarily wait for a player to move. Finally We have the Draw function that runs each frame. This does what it says on the tin, and a little more.

The Camera class doesn’t do much. It just kind of makes it easy for us to centre the view of the player on a particular target position. It really doesn’t do much at all, I figure later I’ll store a position and scale value there, and when drawing the world I can use these variables on the camera to adjust the player view accordingly. I am toying with the idea of maybe giving the Camera class access to a “Map” pointer, so that I can perform some bit blitting for the maps on the camera class. But I’m not sure whether to do that on the camera, or on the Scene. Gut instinct tells me its easier on the Scene class. But we’ll cross that later. We are not there yet, that’s probably tomorrow or the day after.

The SceneManager is pretty straight forward. It holds a vector of SceneBase’s, of course Specific scenes inherit from the base, so I’ll add them to this vector without any trouble. This class doesn’t really do much apart from this. I suppose arguably it acts as a shuttle for the Process, Update and Draw calls from the Game Loop through to the Scenes themselves. I don’t really see this class doing much more.

Finally the SceneBase, this really acts as the equivalent of a C# interface I suppose, its really not going to do much except provide functions to scenes that inherit from it later. It’ll start to become very useful tomorrow as I start the Credits Scenes.

Okay so TODO List for tomorrow:

1. Opening Credits Scene (Which means finally some visuals! Yay!)
2. Texture Atlas System
   1. This “System” will load the games texture atlas at the start of the game.
   2. Then Break the atlas down into TileSize pieces.
   3. Store an enum that will allow me to pick textures with a name instead of a number.
   4. I will include a class in here too that creates a sf::Texture\* for a requested Tile. That should make it easy for me to grab textures for actors, walls etc.

If I achieve that, my stretch goal will be a LevelScene class. This class should be interesting because it has the task of loading 3 files that describe the Level that is being loaded. Once it had loaded the Level, it then needs to break the Level data down into Chunks, which I will then Generate a Texture for when drawing the Level Later.

Normally I would look at doing this after programming a Main Menu, and Settings Menu. But here, I want to make the Main Menu and Settings Menu’s levels of their own, where the player controls their character and moves them into door’s to select options etc. which of course I cannot do until I have a Level loading class ready.

Anyway, for now, its sleep ready for another big day tomorrow.