ITAS 188 Final Project Proposal

Nile Olenick

Idea

The main idea was to create an arcade of multiple, small and very simple games. Pong, asteroid, you know, those type.   
Ideally, the app would consist of 4ish main activities dedicated to the games. One for a home screen, and 3 or so dedicated to the games.   
The grand idea of having a few small games turned into a larger game that consisted of 27 classes. This is a simple, straight Java game with no special requirements, permissions, or internet connection.

It starts with the game loop, which is the engine of the game. It is responsible for the rendering of the game to the screen, all game progresses, and updating all states of all objects in the game world.   
Essentially, it is responsible for updating and rendering all states of the objects in the game between each frame. This is important to know, because the frames per second must be above 24 for quality smooth movement, but not too high where it will slow or even crash the game.  
This will include:

* Import *of Activity, Bundle, Window, and WindowManager.*
* *While loops* which will be to update and render the game using a *try/catch block*; pause the game loop to not exceed the target updates per second by using an *if statement and a try/catch*; skip frames to keep up with the *target UPS* with a while loop, and calculate the average *UPS* and Frames Per Second using an if statement.
* Current time, using *System.currentTimeMillis();*
* A “synchronized” block *synchronized around the (surfaceHolder)* and encapsulating the update and draw methods. This is to prohibit multiple threads from pulling the *surfaceHolder* at the same time.

There will also be:

* Lots of variables and lots of constructors
* *On touch events* in the *Player class* and *Game class.* This is to provide *ACTION\_DOWN*, *ACTION\_POINTER\_DOWN*, spell casting, joystick movement, *ACTION\_UP*, and *ACTION\_POINTER\_*UP events.
* *Math.pow()* methods to calculate the joystick *isPressed()* Boolean.
* *Math.sqrt()* method for the *setActuator()* method in joystick and other things.
* An array list for the enemies (see game class and enemy class) (similar set up for spells).
* *Math.random*() for position of enemy spawn (similar set up for spells).
* .*remove* to delete enemies if they collide with the player (similar set up for spells).
* *If() statements* to register if the player has died, based on their health bar and displaying a “game over” banner in the even that the player does die.
* Use of the *Bitmap* *import* to allow the computer to understand what the image we are passing in is.
* An *enum* list to track the state of the player (*STARTED\_MOVING, NOT\_MOVING, IS\_MOVING*), within the *PlayerState. java* class.
* *Try/catch* within the *PlayerState.java* class to know which state the player is, with a break to let the program know there could only be one state at a time.
* *If() statements* to make the program ask if the player is moving in the x or y axis (*velocityX, velocityY)* within the PlayerState.java class.
* A *Sprite[] array* to return the different number of sprites for the player, which was a total of 3, within the *SpriteSheet.java* class.
* A similar *case/state/break* method within the *Animator.java* class, to aid with the sprite movement through controlling the *updatesBeforeTheNextFrame*.
* A *double array[][]* for the layout of the map (x and y directions) in the *TileMap.java* class
* A nested for loop to place a tile down in both the x and y axis(*iRow and iCol*) until the total number of tiles (60) is reached(this number is clarified within the *SpriteSheet.java class*). This is found within the *TileMap.java class.*
* An *initializeLayout method with a int[][] array*, which physically draws out the map, and what the tiles within them will be drawn as (0 - water,1-lava,2-ground,3-grass,4-tree).

**Android Features**

As far as I’m aware, this app requires so android phone features.

**What Google Documentation or Tutorials were Helpful?**

The tutorial that I followed during this process is the one found here:

<https://www.youtube.com/watch?v=GIc2GRCUI8s&list=PL2EfDMM6n_LYJdzaOQ5jZZ3Dj5L4tbAuM&index=1>

This tutorial was amazing, to say the least. Because It was on YouTube, I needed to type everything myself. I listened to everything he said and took many notes. I learned so much.

**Sample Display Screen**

Wasn’t this done in person?

**Timeline / and tasks**

I didn’t use a timeline, and if I wrote one, I doubt I would have followed it. It’s sad, but that’s how my brain is.

**Problems I might Run into**

In coding, there are no problems. Computers are black and white; right or wrong, and are human designed for human use. If there is a problem, it is strictly to do with the loose nut behind the keyboard.

So, the only problems I thought I might run into were mostly to do with my own weaknesses.

I’m a slow worker, and take more time than your average bear to comprehend these types of things as my brain is more skewed towards creative processes.

I never seem to be able to find the information online as quickly as my classmates, and end up spending hours finding the answer to a problem that should have only taken 30 minutes max.

Fat-fingering the keyboard.

Miss-hearing the explanations in the tutorial and not knowing how to express that appropriately I the program.