Hacker Man Requirements Document

**Overview:**

Scroller game in which a character navigates obstacles by jumping and ducking to try to reach the end of the level. The camera should be centered on the main character so that the game “scrolls”.

**General game architecture:**

**Menus:**

A general menu class will be required. The game will open to a splash screen (simply a title page with some artwork showcasing the game) that then goes to the main menu as soon as a key or mouse button is pressed. The main menu will have the options to

1. Play Game (executes the game Engine class)
2. Help (rules/tips for playing, controls)
3. Credits (screen with credits)
4. Whatever else designer wishes to include

After play game, the Engine class will take over to play the actual game. A summary of its architecture is as follows:

**class Engine**{

private:

GameObjectManager; //container for all GameObjects

Renderwindow

Gamestate//a type enum that tracks the game state for the loop (ie playing, paused, exiting)

gameLooop()//main game loop, controlled by gameState and a switch statement

showPauseMenu()//function to show pause, derive from menu class

public:

startGame(): coordinates the private functions //ex: sets up objects and puts in container before //running game loop

Static screen width//we must decide on the window dimensions which will be stored here

Static screen height

};

**Game Objects**

\*\*\*IMPORTANT\*\*\* all visible game objects must derive from the GameObject class. This includes

1. Characters
2. Obstacles

The gameObject class already has most of the functions you would want for a GameObject. I suggest studying this thoroughly to save you time. Some points about GameObject class:

1. Each object is repsponsible for updating itself (ie setting its new position or changing its color when an event happens). THIS IS ALL DONE IN THE UPDATE FUNCTION WHICH IS PURE VIRTUAL AND MUST BE OVERRIDEN. The update function takes in arguments for the elapsed time since the last frame update(in seconds), the event that took place, and the container containing all of the games objects. This allows you to update the object based on things like velocity, input from the user, and the positions of other objects.
2. Updating position using the time since last update: say you have an object that is moving at a given velocity in pixels for second. You would place the velocity multiplied by the time since last update in the move function, since v\*t = distance. The move function is a member function of the sprite, Andy used it in class.
3. Use the getBoundingRect() for collision detection. The function returns the sprites bounding box as a sf::Rect. The Rect has a useful intersects member function to tell you when the position of two Rects intersect each other.
4. The heart of the class is the sprite: the size of the object is automatically the size of the image that is loaded to it (I believe you can change the scale with a sprite member function if you wish the change the size of the sprite at runtime)
5. To access the sprite in the derived class, use getSprite() to get a ref to the sprite. This gives accesss to the sprite and all of its SFML functions.
6. When accessing other objects to compare positions, you must downcast the pointer to the object received from the container.

**Projects**

Projects listed in order of general priority. Please do not place your name next to one unless you are wholeheartedly working on it. Indent and add a subproject underneath the project if only working a small portion of it (ex: you are working on the audio for just one object, not an overall audio system). Please include the current status of the project. Also, I put everyone’s names next to some project from memory doesn’t mean youre stuck with it we can still change things. Communicate!

1. Artwork; Daniel; not started
   1. I feel like this is probably where most of the work is, maybe everyone should pitch in
   2. Screen size; I decided on height 768, width 1024 just to get the ball moving
2. GameObjectManger; Reid; functional
   1. GameObject; Reid; functional
   2. Note: may want to change organization of container to simplify collision detection with character
3. Engine; Reid; not started
   1. I know this was going to be Daniel’s domain, but right now I feel like this is too tied to the Gameobject manager that they should be made by the same person, let me know if this bothers you or you still want to do it
   2. Making the game scroll
4. Menu class; Andrew; not started
   1. Splashscreen
   2. Main menu
   3. Pause menu
   4. Credits
   5. Help/controls page
5. Main character; Daniel; not started
   1. Jump height
   2. Duck height
   3. Lives?
6. Obstacles; Patrick; not started
   1. Firewall, maybe something to jump over?
      1. Maybe different walls to go over and under
   2. Bug, probably something that just moves back and forth and must jump over
   3. Bug/aerial obstacle at characters head height that must be ducked under
   4. Any others?
7. End boss level
8. Networking
9. Audio