AP Computer Science Final Project - README Template

Instructions:
The first step in creating an excellent APCS final project is to write up a README. At this stage, this README file acts as your **project proposal**. Once you've filled in all components, Shelby will read through it and suggest edits. Ultimately, you need a document that adequately describes your project idea and **we must agree on this plan**.

Have one member of your group **make a copy of this Google Doc**. Then, they should share it with all other members **and with Mr. Shelby** so that every group member has edit permissions, and Shelby can add comments on your ideas.

There's a lot of parts of this document that you might not have full answers for yet. Because you haven't written the program yet, it's difficult to think about the **instructions** or **which group members will do which parts**. Even though this is hard to think about, you must have something in these sections that acts as your current plan. However, during the course of the project, you'll **continuously update this document**. This means that you will not be *held* to exactly what you put here - components of this document can change (and it's pretty common!).

There is one exception: the **Features List** section. Once Shelby OKs your README, the Features List section **cannot be modified**. For this reason, it is most important that you get a solid idea of what you want to make and the primary features it will have *now*.

Talk with your group. Consider drawing some pictures of what you think your project might look like. Be precise. When you're ready, fill this out together. Each component in brackets below ([these things]) should be replaced with your ideas. Note that there are several sample READMEs posted on this assignment for you to use as guidance.

----------When README is finalized, remove everything above this line----------

[HEEHAWBrawls]

Authors: Jerry Chu, Julian Ng Thow Hing, Stephen Tan
Revision: 5/6/2022

Introduction:
[In a few paragraphs totaling about ½ page, introduce the high-level concept of your program. What this looks like depends a lot on what type of thing you are making. An introduction for an *application* will look different than one for a *game*. In general, your introduction should address questions like these:

What does your program do?
What problem does it solve? Why did you write it?
What is the story?
What are the rules? What is the goal?
Who would want to use your program?
What are the primary features of your program?]

Our studio HEEHAW brings you a 1v1 arena shooter game with shooting mechanics and power-ups in which two players can shoot bullets at each other in an attempt to take down the opponent. It solves the problem of boredom and a rivalry between two people. Both these problems can be solved with our game to provide a fun activity and to settle a dispute between two people to decide who is just better. You take part in a gun duel against the most skilled fighters of the nation! Take them down in elimination matches and win the duel! Win the match by elimination, king of the hill, or gun game(smg, shotty, sniper, knife). Each match lasts 2-3 minutes. Our game targets epic gamers who want to test their gaming skills in our challenging and fun game. A few primary features we hope to have are a well-designed map with challenging obstacles to challenge each user. We also want to include powerups, unique weapons, and gamemodes to increase user flexibility within the game. Emotes are another feature we hope to implement, especially a HEEHEEHEEHAW emote users can use to taunt their opponents. Our defining features are our world style arena with unique obstacles and powerups that will challenge the user's strategy in defeating their opponents. They will need to strategically avoid their opponent's bullets, while also keeping a lookout for obstacles and powerups they could use to their advantage.

<u>Instructions</u>:
[Explain how to use the program. This needs to be **specific**:
Where will you need to click?
Will you have menus that need to be navigated? What will they look like?
Do actions need to be taken in a certain order?]

The overall procedure of playing the game is as follows: Advance to main menu, select an option from the first screen (ie. Start Game, Instructions), and proceed to gameplay.
When starting up the game, the user will be on the starting screen. There, users will be able to select a game option such as "Start", or "Instructions" by clicking on it. After clicking the "Start" button, users can advance to the game by clicking on the "Confirm Start Game" option, and then they will be at the actual game.
Within the actual game, users will be able to move around using the WASD keys, aim and navigate their player with their mouse, and left click to shoot. Multi-player setup, a "Back" option, and additional features will be added in a later release.

<u>Features List (THE ONLY SECTION THAT CANNOT CHANGE LATER)</u>:
Must-have Features:

[These are features that we agree you will *definitely* have by the project due date. A good final project would have all of these completed. At least 5 are required. Each feature should be fully described (at least a few full sentences for each)]

- Bullet physics, the bullets would travel in the path they are shot and fade when they run into a wall or after they travel a certain distance. The bullets would follow the position of the mouse, different bullet types have different bullet speed. Different weapons have different bullet capacity
- Different types of weapons players could start with at the beginning of the round
    - sniper rifle (5 bullet magazine, long reload time, has large sight range, does high damage (95 damage per shot), shoots with slow, single fire mode, normal base movement speed)
    - submachine gun(25 bullet magazine, medium reload time) has normal sight range, does low damage (19 damage per shot), shoots extremely quick, increased movement speed 1.15)
    - Shotgun (2 shots, short reload time, has short sight range, shoots a shotgun blast of 5 pellets that do middle damage (25 damage per pellet) each outward in like an angle, single shot fire, negative movement speed by 0.9)
    - Knife (one shot one kill, normal sight range, very very close range, knife weapon gives increased movement speed by 1.3)
- Moveable character with 100 health, bullets take away health but some power-ups restore health
- Obstacles that could potentially damage a player's health and challenges their strategy
    - Walls to prevent players from having a straight path
    - Spikes on the ground that could damage players
    - Tar pit that slows player's movement
    - Gas that lowers player's field of vision (fov decreases)
- Respawn feature, when you die you have to wait 10 seconds before you respawn, and you can change your weapon if you want (only when you die)
- Emote feature to taunt opponents - emotes appear over the characters head, displaying animations of an image, will be most efficient after killing opponents and they are viewing your point of view.
- Text UI on the bottom displaying gun type, bullet count, powerup status, etc

Want-to-have Features:

[These are features that you would like to have by the project due date, but you're unsure whether you'll hit all of them. A good final project would have perhaps half of these completed. At least 5 are required. Again, fully describe each.]

- Balanced gameplay with bullets and health regen and powerups complimenting each other well
    - Rate Powerup increases fire rate
    - speed powerup increases movement speed
    - damage powerup increases bullet damage
    - health powerup heals you health, does not increase your max health

    - Balance the bullet damage stats with the powerups so no weapon is too overpowered

- Two people connecting to one game over the internet through LAN
- Cool sprites that have different moving and shooting animations and don't just stay static
- 4 sprites for each player to choose from (amongus sprite, special forces sprite, sprite can sprite, fat guy sprite)
- Graphically pleasing Arena that is more than just boxes and shapes but rather has well drawn meshes and artistically pleasing details
- More than one map for the player to play on, this map would be selected by the player in the menu
- Music and sound effects for gun shooting, kill effects, powerups
- Minimap in the corner of the screen to direct user where the components of the map are (center, respawn point, etc)
- 1v1 gameplay connected over LAN networking between two players that can control their player and shooting
- Usable power-ups to increase bullet fire rate, increase bullet damage (by 1.25%), increase health, increase field of vision, and increase movement speed.
- Different gamemodes like deathmatch, king of the hill, and gun game.
    - Deathmatch - just both players in a brutal brawl for the whole duration, player with most kills wins
    - KOTH - point system, there will be a hill that spawns in 3 different spaces on the map (center of the map, top of the map, bottom of the map). This hill rotates every 30 seconds. By staying inside the bounds of the hill, a player gains points. The objective is to rack up more points than the other players by holding your ground inside the hill and killing them before they kill you. Most points wins
    - Gun game - you start with an smg. When you get a kill, your gun "ranks" up. Smg turns from shotgun to sniper to knife. The first to get a kill with the knife wins the game.
- Scoreboard displaying scores on the top right corner of the screen

Stretch Features:
[These are features that we agree a *fully complete version of this program would have, but that you probably will not have time to implement*. A good final project does not necessarily need to have any of these completed at all. At least 3 are required. Again, fully describe each.]

- 3d graphics for the maps and the players and the guns designed with 3d design apps
- Adding more unique emotes beyond basic emoji reactions such as the clash royale heeheeheehaw
- Team compatibility so like 2v2 or 3v3 game modes
- Single player feature against AI to practice 1v1 or aim

<u>Class List</u>:
[This section lists the Java classes that make up the program and very briefly describes what each represents. It's totally fine to put this section in list format and not to use full sentences.]

MainMenu - Holds an array of screens and manages screen switching (implements ScreenSwitcher), contains the main method

Screen - an interface that represents a certain screen on the GUI

- **MenuScreen -** the first screen that pops up, has two buttons that will lead to the PreGameScreen and Instructions respectively, implements Screen
- **PreGameScreen -** screen that shows up upon clicking the first button, has a button confirming the start of the game that will lead to World, implements Screen
- **Instructions -** screen that shows text that explains controls/other relevant information to the game, implements Screen
- **World -** screen that represents the actual game screen (has a PApplet that draws the game screen), implements Screen

Player - Represents players within the game, player has a weapon

Avatar - a player has an avatar which represents the sprite

Bullet - Represents the projectiles/bullets that a certain type of weapon has

Weapon - Represents the superclass of weapons the game could have

- **SniperRifle -** Subclass of weapon
- **SubmachineGun -** subclass of weapon
- **Shotgun -** subclass of weapon
- **Knife -** subclass of weapon

PowerUp - class that represents a power up

- **HealthPowerUp -** subclass of PowerUp that increases the player's health
- **SpeedPowerUp -** subclass of PowerUp that increases the player's speed
- **DamagePowerUp -** subclass of PowerUp that increases the player's damage
- **RatePowerUp -** subclass of PowerUp that increases the player's fire rate

Obstacle - Object on the arena that obstructs a player's path

- **Wall -** Subclass of Obstacle
- **SpikeTrap -** Subclass of Obstacle, decreases a player's health
- **GasTrap -** Subclass of Obstacle, decreases a player's health
- **TarTrap -** Subclass of Obstacle, decreases a player's health

Tile - Image that represents a tile on the background of an arena

TileManager - Class that manages an array of tiles

<u>Credits</u>:

[Gives credit for project components. This includes both internal credit (your group members) and external credit (other people, websites, libraries). To do this:

- List the group members and describe how each member contributed to the completion of the final program. This could be classes written, art assets created, leadership/organizational skills exercises, or other tasks. Initially, this is *how you plan on splitting the work*.
- Jerry:
  - finished the code for the sprites and added a window with an image that can be moved in all 8 directions
  - World feature completed

    - Designing the map, mugged sprites from the internet
    - Adding walls to the map
    - Implementing collisions between wall tiles and trap tiles (not done)
    - Coding the trap classes and causing them to do stuff to the player if they collide(not done)
    - Coding powerups and causing them to do stuff to the player if they collide (not done)
- Stephen:
    - Worked on the frontend
    - Worked on the MainMenu class (containing the main method)
    - Created the title screen (MainScreen) and other screens that it will lead to (Instructions, PreGameScreen)
    - Connected the main menu to the World feature that Julian and Jerry worked on (clicking on the buttons on the screen will lead to World)
    - Incorporated updates from code to UML diagram
- Julian:
    - Player + avatar/character design
    - Created sprites & working on implementing collision detection
    - Created the Sniper, Shotgun, and Submachine guns, and bullets
    - Coding the the Player HUD(displays health, ammo, weapon type)
- Everyone:
    - Lan network with bullets and movement being transferred real time
    - Arena with obstacles that do stuff and powerup, arena design
    - Features of the actual game

- Give credit to all outside resources used. This includes downloaded images or sounds, external java libraries, parent/tutor/student coding help, etc.]
    - https://www.youtube.com/playlist?list=PL_QPQmz5C6WUF-pOQDsbsKbaBZqXj4qSq
    - http://studio.processingtogether.com/sp/pad/export/ro.9bY07S95k2C2a/latest
    - https://www.123rf.com/photo_120986151_pixel-art-style-set-of-different-16x16-texture-pattern-sprites-stone-wood-brick-dirt-metal-8-bit-gam.html
    - Looked at Mr. Shelby's GamePhysicsDemo to learn how to implement screen switching.