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----------

Authors: Aaditya Raj, Ishaan Singh, Rajiv Venkatesh
Revision: 4/27/22

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 program is a 2 player game, with a side-view similar to mario. In this game, there is a battlefield, which consists of walls that serve as borders and platforms to jump on. Players start from their home bases, one in the bottom-left corner and the other in the bottom-right. Players have the ability to navigate the maze and jump up onto platforms to eventually reach the flag. They each have a paint gun, with which they can shoot paint onto the map (walls, platforms, etc.) or shoot the other player and eliminate them. If there is already paint in the spot, nothing will change. The paint on the ground gives the player the ability to move 4 times faster if they 'swim' in it by passing through. When the player shoots their opponent, they lose health, and if they are eliminated, they respawn back at their home base after a few seconds of cooldown time. At the top-center of the map there is a flag, which also serves as a stronger paint gun. The goal is to take the flag and go back to your base, which is how you win. When someone takes the flag, they can use it to shoot paint with a bigger stroke width and damage their opponent more; however, it shoots less frequently than a regular gun. The player holding the flag is also two times slower compared to their regular movement, and they lose the ability to 'swim' in their paint. When someone dies holding the flag, the flag will remain in the death location. In addition, there is a time limit, and the user has the ability to influence the amount of time for the game based on difficulty selection, along with the choice of map. At the end of the time limit, the win is awarded to the player who has the most points. Points are a combination of paint percentage on the map and whether the flag has been collected. A flag collection would be either 50 or 0 points, while paint points are just your paint percentage. So if you painted 10% of the map and got the flag, 50 (collected flag) +10 (paint) = 60, so you would get 60 points. Your opponent would get 0 (ceded flag) + 90 (paint) = 90. This means that a player that has collected the flag will not necessarily win if they have a much smaller paint percentage than their opponent.

People who would love our program are those who enjoy games such as Splatoon or Capture The Flag. This game merges both in 2D, so the game feel and mechanics are altered completely. Gamers who play Halo 5, Call of Duty or Fortnite are also likely to be fans of this game, as they already play games in which aiming at the right target with the gun is the most important goal.

One primary feature of this program is the ability to shoot paint at the ground to 'swim' in it for faster speed or to eliminate the opponent. Walls and platforms are another major feature that enhance the user experience by making the game more difficult and interesting. Flags are another crucial feature. This object serves a dual purpose – the player has to capture it and bring it back to their base to win, but can also use it as a stronger paint gun. Lastly, home bases are useful for the respawning of the player.

The reason all three of us were interested in making this program is because we all love multiplayer games where we play against each other and we all like the idea of capturing the flag, and the idea of a territory battle with paint.

Beta Release Instructions:

1. **To test multiplayer functionality:** Use the arrow keys to control the player who starts on the right and the WASD keys to control the one on the left. As of right now, only the left player can shoot due to the limitations of a single computer. We hope to fix this by the final release with true networked multiplayer.
2. **Map with obstacles:** When you start the game, you'll see a couple of platforms that the avatars can jump on. This adds an interesting element to the game. We would appreciate it if you tested the boundaries in odd ways, to make sure our code to prevent the characters from going through the platforms actually works.
3. **Running and "swimming" through paint:** To test this feature, shoot paint at the platform nearest to the player on the left. Use WASD to walk on the paint on that platform, and you should notice an increase in speed.
4. **Paint Shooting:** Clicking around the map results in paint being shot out of the gun. Notice that you are not able to spam, because the gun reloads for 1 second after every five shots. If you hit the player on the right three times, they will respawn. We hope to extend this with networking so that the game is complete.
5. **Flag:** This is probably the only feature that is represented "in some way," but does not actually work. If a character collides with the flag, that character will pick it up. The specific mechanics of the Flag afterwards are a little odd, and we would appreciate a short list of all the problems you see.
6. **Home bases:** The home bases are initialized in the Avatar's constructor. When you shoot the player on the right three times, you will notice that it respawns back at its home base. We pan to implement the code that would end the game if the flag is brought to a home base once the Flag code works properly.

Final release:

Main Menu (options):

1) Create a Game
    a) Choose time limit for the game and number of rounds
    b) Given a randomly generated game code
    c) Wait for opponent to join the waiting room
        i) Simple loading screen
    d) Once second player joins via network, a matchup preview will appear
        i) Who is which color paint
2) Join game
    a) Enter game code and join

Gameplay:

1) W to jump

2) A & D to move left and right
3) Mouse to shoot in a specific direction
4) Shift to toggle 'swim' through paint

<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)]

- Multiplayer function
    - Arrow keys for one player, WASD for the other
- Map with platforms, barrels on platforms, wall border
    - Restricts both players' movement
    - Cannot swim over barrels, must jump over
- Running and 'swimming' through paint
    - Regular running - walking on non-paint territory or your color
    - Swimming in your paint faster than regular walking
    - If you walk on opponent paint, you get 'dizzy': slow down and gradually lose health
- Paint-shooting
    - Try to cover the platforms, walls, barrels with your color paint
        - Determines win if no one collects the flag by the end of time limit
    - Three hits to kill opponent
    - Automatically reloads (time-based)
    - Reload the paint gun faster by 'swimming' through your color
- Flag
    - Goal is to reach the flag at the center and bring it back to your home base.
    - Flag lets you shoot bigger paint strokes, but less frequent shooting, and slower movement
        - 3 hits to kill opponent
- Home bases
    - Base where you start and where you respawn after dying

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.]

- Multiplayer function with networking, two different devices
- Choosing different paint guns with different reload speeds, paint stroke width, paint stroke length, etc. before the match
- Each player starts with a few paint bombs: items that they can throw and they will burst with paint covering a certain radius

    - Paint burst does not pass through walls/platforms
    - Select how much each player starts with when creating a game
- Deploying small towers that shoot paint around itself anonymously in a specific pattern, can damage the opponent, prevents the opponent from passing through and it has less health.
    - # of towers per player would be set when creating game
- Choose out of a few maps to play when creating a game
- Leaderboard
    - The top 10 users are displayed from the server and it shows the max points they received in a game.

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.]

- One stretch feature we could implement is 3D graphics. This would require a number of extraneous classes, which we felt would take up valuable time that could be spent improving the user experience.
- Another of these features is the first person perspective. As of right now, we do not know where to start with this, and although it would be a very cool addition to our project, we are fine with the perspective we have now, which will also make debugging easier.
- Finally, we hope to be able to incorporate more than two players in the game, but this would require a larger map and likely more complex networking.

<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.]

- Main
    - Runnable class that creates drawing surface
    - Uses DrawingSurface
- DrawingSurface
    - The drawing surface on which the game is displayed
- Various Screen classes
    - Each screen that the user sees
- MainMenu (extends Screen)
    - Beginning screen of the game
    - Where the player is introduced to the came
- GameRoom (extends Screen)
    - Where the player can customize and create a new game for another person to join
        - Set a time limit, map, etc.
- Game (extends Screen)
    - Displays the actual game
    - Tracks score

    - Displays the remaining time for the game.
    - Shows the player's total health.
- Platform
    - The platforms that each player can jump on to reach the flag
    - Has a PaintBlock border to represent the parts that have been painted
- Flag (extends PaintGun)
    - Is a paintgun, but there is only one in the center of the map
    - If one user is able to capture the flag and return it to their home base, the game ends and that player is heavily favored to win
- Avatar (extends Sprite)
    - Represents the player on the map
    - Health, speed, color
    - Jump, move, throw bomb, shoot paint
- PaintGun (extends Sprite)
    - int ammo, reloadTime, velocity
    - PaintBlock stroke
    - shoot(int x, int y)
- PaintBomb (extends Sprite)
    - A bomb that the player can throw
    - Has a certain radius of explosion, releases many PaintBlocks when exploding
- PaintBlock (extends Rectangle)
    - Is a square, represents a 'block' of paint used to paint the map

Credits:
[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*.
- Give credit to all outside resources used. This includes downloaded images or sounds, external java libraries, parent/tutor/student coding help, etc.]

Background

Rajiv

- PaintGun & Flag
- Set-up (Main, DrawingSurface, Sprite import)
- Server networking
- Images & other resources

Aaditya

- PaintBlock, Platform classes
- PaintGun

- Avatar
- Game
- Server networking

Ishaan

- Avatar class
- Game class
- Other Screen classes
- Server networking
- Platform