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----------

[Rocket Simulation]

Authors: Ryan Wagner, Rohan Gupta, Krish Jhurani
Revision: 4/13/2022

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 program allows users to build/design a rocket and then run simulation flights to see how good it is. The point of the program is to make a rocket simulation with the purpose of testing how different interchangeable parts work together. We wrote our program to make an easy way for people to learn how different types of parts of a rocket work, and how effective different combinations of those parts are. There are not many rules that the user has to follow other than they have to fully complete the rocket before they test it. They either pass or fail the flight mission. The goal is to try and combine different versions of parts of a rocket, and make it so that it completes the set task at hand. People who would want to use our program are teachers, and people who like problem-solving, aerospace engineering, and/or just rockets in general. The primary features of our program are that you drag and drop parts of a rocket to make a model rocket, simulate a rocket flight, have levels with increasing difficulty, data and statistics of the rocket in both flights, and on the ground. There will also be on-screen instructions that explain how the program is intended to be used. The rocket building and testing will be displayed on different screens.

Instructions:
[Explain how to use the program. This needs to be **specific**:

At the beginning of the program, the home screen will be displayed. There will be a couple of levels, but users will start at level 1. This is the most basic level, where users will be prompted with instructions on how the program works, and the different components of the rockets. The first level will be simple, and the next level will increase in difficulty as the user moves on. In each level, the user will be given certain components to build a rocket, and as the level increases, more and more components will be given. After the rocket is built, there will be a test to see how efficient it is, with the tests increasing in difficulty as more and more levels pass by. Levels are completed by reaching the success zone (this varies depending on what the objective of the mission is). During the construction of the rocket, parts will be given, which can be dragged and placed onto a given pedestal. During the test, the velocity of the rocket will be controlled by the arrow keys. The direction the rocket is going is controlled by the right and left arrow keys. A distance measure showing how close the rocket is to the planet/moon will be displayed at the top of the screen, telling you when to slow down and when to speed up.

Which keyboard keys will do what?
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?]

<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)]

- Drag and drop parts of a rocket (engine, materials, fuel) from the left sidebar onto a model rocket, where the parts will "click" into place. The sidebar will display the currently selected specification of the part, and the user can change which specification is selected. For example, only one material will be shown on the sidebar such as steel, but the user can change which material is displayed through a menu, changing the displayed material to something else such as carbon composite

- A simulation of the rocket's flight following basic physics principles. The rocket will be controlled by the user through the arrow keys, but its flight will still act in a realistic manner, including exploding during flight or successfully completing its mission depending on how well the rocket is designed.

- Multiple levels / level select, with each level increasing in difficulty. Level 1 will be as basic as possible, so the user can get a feel for the program. The next few levels will slowly try to introduce new parts and types of engines, fuels, materials, etc. Each level will be a mission of sorts, such as trying to reach orbit, or trying to orbit the moon.
- A data / statistics display on the launch screen that will show in real time data and telemetry of the rocket. This would include things such as altitude, speed, atmospheric pressure, etc. Another display would be shown on the build screen, displaying "ground" metrics such as the weight, cost, efficiency, and probability of success

- On screen instructions that explain how the program is intended to be used. The instruction's primary purpose would be to make sure that the user correctly uses the program, by informing them that they need to pay attention to the data presented to them, in order to make sense of what is happening and why.

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.]

- Custom Keybinds. Rather than using just the arrow keys, the user can change and select their own custom keybinds for the flight of the rocket. This will be done through a settings menu (if we get around to implementing one)
- Second stage implementation / customization. Allowing users to customize and play with the second stage of the rocket, just as they do with the first stage. For example, the user would be able to drag and drop different engines onto the second stage. Stage separation events would also be included in the flight to further immerse the user in a realistic experience

- More levels. More levels / missions for the user to complete. These levels could expand on / previous levels, or they could offer entirely new missions. New missions would entail orbiting / reaching further away planets, or successful orbit insertion of payload (if payload system is implemented)
- Payload implementation. Implementation of a payload and faring system. This would allow the user to customize their fairing size / payload capacity, which would add additional complexity to the design of the rocket, as extra weight and cost must be factored in.
- Settings menu for the user to play around the settings of the program. This would include things such as keybinds, colorblind settings, etc.)

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.]

- Different types of rockets to test fly (ex: water rockets, missiles, etc) and different types of the parts used to build them. Have goals for these as well.
- Better graphics (3D rocket) to make the simulation more realistic.
- Implementation of the reusability of the rocket which would include both landing / reusing of the booster / first stage, as well as a landing system for missions which require the user to land on a planet such as the moon or mars. Reusability could stretch the payload fairings as well, and ways to reuse them.

Class List:

DrawingSurface class: Same as drawing surface in GamePhysics. Would draw the “Home Screen”

Main class: contains the main method that runs the program

ScreenSwitcher: switches between the scenes

Sidebar: contains the sidebar data

Screen class: Superclass of all screens

- **LevelSelect Class**: extends Screen, displays home screen/menu

- **BuildScreen Class**: extends Screen, but superclass for other build screens displays build menu/screen. (could be an interface also)
    - Different levels could be different screens, or one screen could draw different “levels”
    - **Build1 class**: extends buildScreen class
    - **Build2 class**: extends buildScreen class
    - Additional levels here

- **LaunchScreen Class**: extends Screen, but superclass for other launch screens displays launch menu/screen. (could be an interface also)

    - **Launch1 class**: extends launchScreen class
    - **Launch2 class**: extends launchScreen class
    - Additional levels here

- **LevelSelect Class**: extends Screen, displays level select menu/screen

- **Instructions:** is a screen that contains the instructions

Rocket class: Represents the rocket. The state of the rocket is also represented as a field here, and the state of the rocket refers to if it is flying, or it has blown up. If still flying / not blown up → draw the rocket. Else (rocket has blown up) → draw an explosion

Rocket HAS-A Data
Rocket HAS-A Engine
Rocket HAS-A Material
Rocket HAS-A Fuel

Data class: extends rocket, stores and calculates all data

Engine class: represents engines, extends rocket class

Material class: represents materials, extends rocket class

Fuel class: represents fuels, extends rocket class

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*.
  Ryan:, LevelSelect, LaunchScreen,Build1, Build2, BuildScreen, Launch1, Launch2
  Krish: Launch1, Launch2, Meteor,
  Rohan: Sidebar, DataDisplay, Data Engine, Fuel, Material, Rocket
  Build1, Build2, BuildScreen
- Give credit to all outside resources used. This includes downloaded images or sounds, external java libraries, parent/tutor/student coding help, etc.]
  -Drag & Drop demo
  - processing widgets demos
  - g4p library
  -moving screen demo
  -images from google