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--------------------

Grinch Fringe

Authors: Dhruv Lohani, Ishwar Suriyaprakash, Arindam Kulkarni

Revision: 5/6/22

<u>Introduction</u>: (Ishwar)

To entertain yourself, imagine that you are an ordinary police officer in a world ravaged by hunger and disease. You are tasked with finding certain blueprints for a machine that will help accelerate food growth and production. To achieve this goal, you are sent to a maze, within which the blueprints are stored in rooms, and you have to navigate the maze to retrieve those blueprints. However, the maze is occupied by an elusive Grinch, who is very antagonistic to intruders and is also very possessive of its blueprints. The Grinch has rigged the maze with invisible traps that injure intruders, and the Grinch also has the power to kill intruders in close proximity. While retrieving the blueprints, you have to evade the traps and the Grinch to exit the maze alive. To help, you will be given several kits and tools to help you identify the Grinch and its traps. For example, the Grinch happens to emit ionizing radiation, so a Geiger counter will help you determine your proximity to the Grinch. Additionally, the Grinch may have evil collaborators in the maze who help it plant more traps and attack intruders, and, to communicate, they send encrypted messages between each other. You can intercept and decrypt those messages to find out more about their plans.

<u>Instructions</u>: (Arindam)

We use W, A, S, and D keys to translate the police officer and help it move. W key would make it go up, A would make it go left, S down, and D right. The Mouse changes the line of sight - it would change your point of view, and would allow you to look in different directions, which is a key aspect of any interactive game. You can click the mouse to attack or get the Grinch. We have a menu, which allows for better sight and reference. These menus could be a bird's eye view. In addition, resources will be required for the round trip through the maze. This would require the police officer to pick up inventory, another aspect of the game. Inventory could include things such as a flashlight.

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

Must-have Features:
- The program displays a 2D maze with 2D graphics
- The maze has a stationary Grinch that kills the officer if in close proximity
- The maze has traps rigged around the maze that are not immediately visible to the officer
- The program displays blueprints in rooms around the maze that need to be collected by the officer
- The program has multiple options of maze layouts for the user to pick from

Want-to-have Features:
- The program only shows the parts of the maze that are illuminated by, for example, the officer's flashlight

- The maze has a *moving* Grinch that tries to attack the officer
- The maze has other antagonists that help the Grinch attack the intruders and plant more traps
- The program has sound effects for events (eg. encountering a trap, Geiger counter beeping, reward sound for when a blueprint is found)
- The game gives path recommendations for beginner players - a shortest Hamiltonian path (a path that visits all destinations - blueprint rooms) can be drawn out for the player before he/she starts the game.
- The game could include the ability for the officer to crack encrypted messages between Grinch and antagonists to find more information about their plans
- To give the police officer an advantage, the officer can have access to teleportation technology but can only teleport between specific parts of the maze.
- The officer can have a UV flashlight to identify the Grinch's footsteps (that is, if the Grinch is moving); these footsteps can help the officer locate the Grinch, which gives the officer an advantage
- Multiple players can play with each other (using networking); players can either play against each other in a free-for-all or play in a team to help each other

Stretch Features:
- The program can have 3D graphics that show the perspective of the officer
- The program can randomly generate mazes so that the user doesn't play in the same maze multiple times
- The program can generate a 'maze' from actual streets of a city such as Amsterdam or Paris. Sort of like a 'chase' scene which meanders through canals and small alleys of the street. It would really make it a fancy project.

<u>Class List</u>: (Dhruv)
[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
DrawingSurface
abstract ScreenObject - superclass of all objects displayed on the screen
Actor/Character - main characters/players in the game
- Officer *extends* Actor
- Grinch *extends* Actor
- (want-to-have) Minion *extends* Grinch

Screen - for displaying different things at different times
- GameScreen
- OptionScreen

abstract Item - for traps and or resources that the player can use
- Trap *extends Item*
- Blueprint *extends Item*
- GeigerCounter *extends Item*

- May have other objects that would extend items as well

MazeData - stores data on the structure of the maze
HauntedMaze - has characters, items, maze walls
Networking class (want-to-have class) - would allow for multiplayer features
Encryption Class (want-to-have class) - would facilitate how the encryption codes works, and how it would be implemented in our game

Credits: (all)

Ishwar:
I contributed ideas in our brainstorm for the capstone project, and we eventually decided to implement one of them in combination with Dhruv's idea (the maze game). I also gave the UML diagram its initial structure, which included almost all of the classes, relationships, and other. I wrote the introduction for the game, which has the context of the game, the main goal that the user should achieve, and the challenges that the user has to overcome in the game. In addition, I helped organize the initial code structure for all the classes.

Arindam:
I worked on creating our final idea with the group, developing the idea of an antagonist in the grinch, and officer. I helped develop the idea of a flashlight as well. I worked on creating and designing our final idea based on the ideas that we as a group collectively came up with. I did about a third of the readme, such as the instructions section, classes section, and stretch features. Finally, I worked on the UML diagram with Ishwar and Dhruv.

Dhruv:
I worked on the original idea stage, and came up with a few final ideas. We incorporated part of my idea, and Ishwar's idea, into our final game. I worked a bit on the UML Diagram, mostly completing what was not done such as class relations and some class variables. I did the features list and the class list on the Readme. Ishwar, Arindam, and I worked on the classes outline yesterday night on Zoom.

To-do:
Ishwar: Screen, GameScreen, Officer, ScreenObject, Main, DrawingSurface
Arindam: OptionScreen, Antagonist, Item, Blueprint
Dhruv: Trap, Maze, Grinch, Encryption

Additional notes/thoughts
What do the traps specifically do to the officers?

- Decrease in health
- Blood loss
- Allergic reactions
- Smokescreen
- Trip wire that closes a pathway

What info do the messages between the grinch and the evil guys have?

- Setting more traps
- Plans to attack intruders

How do we allow the grinch to “punish” the player if the player doesn’t follow the shortest hamiltonian path?