Grinch Fringe

Authors: Dhruv Lohani, Ishwar Suriyaprakash, Arindam Kulkarni
Revision: 5/24/22

Introduction: (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. All blueprints must be collected. 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, taking away from the Officer's health. While retrieving the blueprints, you have to evade the traps and the Grinch to exit the maze alive. For example, the Grinch happens to emit ionizing radiation, so a Geiger counter will help you determine your proximity to the Grinch. Also, a teleporter will be given, which transports the officer to one of the blueprints. Teleporters can only be used once. Additionally, the user will have a path finder tool that helps them navigate to the closest blueprint.

Instructions: (Arindam)

At the start of the game, we have a menu, which displays 3 options of randomly-generated mazes to pick from. In gameplay, the user can press W, A, S, and/or D to translate the officer and help it move. W key would make it go up, A would make it go left, S down, and D right. The user also has the option of using the arrow keys to navigate. The Mouse changes the line of sight - it would change the direction in which the officer's flashlight points towards, and would allow you to look in different directions. To help the user pick up blueprints in the maze, the screen will tell the user that a blueprint is nearby and that he/she should press E to pick it up. Also, the user can press T to use one of the officer's teleporters and P to use the officer's path finding tool. Furthermore, to accelerate the officer, the user can press the spacebar, although this should be used conservatively because acceleration incurs a cost on the officer's health.

Features List (THE ONLY SECTION THAT CANNOT CHANGE LATER): (Dhruv)

- The program has multiple options of maze layouts for the user to pick from.
- The program can randomly generate mazes so that the user doesn't play in the same maze multiple times
- The program displays a 2D maze with 2D graphics.
- The program displays an info bar that contains details on the officer's current state.
- The program displays blueprints around the maze that need to be collected by the officer.

- 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 and kills the officer if in close proximity.
- The maze has traps rigged around the maze that are not immediately visible to the officer.
- 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 program can give path recommendations for beginner players - a shortest path to the closest blueprint is drawn.
- The program has background music during gameplay.
- As the officer gets close to the grinch, the background color of the screen turns more red.
- The program imposes a time limit of 2 minutes on the user to play the game. If the user does not complete the game's objectives on time, the user loses.
- The program allows the user to:
    - Press WASD keys to move the officer
    - Press E to pick up blueprints - this will be said to the user in the game
    - Press T to use a teleporter
    - Press P to draw a path to the closest blueprint
    - Press ESC to pause / resume the game
    - Press the spacebar to accelerate the officer (but this incurs a cost on its health)

Class List: (Dhruv)

Main
DrawingSurface
abstract ScreenObject - superclass of all objects displayed on the screen

- Actor *extends ScreenObject* - represents a character in the game
- HauntedMaze *extends ScreenObject* - houses all the physical elements of the game
- InfoBar *extends ScreenObject* - displays information on the officer in the game

Actor - main characters/players in the game

- Officer *extends* Actor - represents the protagonist of the game
- Grinch *extends* Actor - represents the antagonist of the game

Screen - for displaying different things at different times

- GameScreen - the screen where the game is played
- OptionScreen - the screen in which the game starts and options are displayed
- EndScreen - the screen in which the game ends and displays the result of the user's gameplay
- InstructionScreen - the screen that displays the instructions for the game to the user

abstract Item - for traps and or resources that the player can use

- Trap *extends Item* - an item that causes damage to an officer's health
- Blueprint *extends Item* - the item that has to be collected by the officer
- GeigerCounter *extends Item* - the item that helps the officer measure the radiation level
- Teleporter *extends Item* - the item that helps the officer teleport to a blueprint

- PathFinder *extends Item* - the item that displays a path to the closest blueprint for the officer

ScreenObject - represents a generic object on the screen
MazeData - stores data on the structure of the maze
gridP - represents grid points in the maze
AudioPlayer - used to play audio in other classes

Credits: (all)

Ishwar:

Week 1: 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 (with most fields and methods) and rudimentary method calls, which made the task of coding the project modular.

Week 2: I completed the GameScreen, HauntedMaze, and Officer classes, and I have also contributed to other classes like Screen and Actor. I implemented most of the HauntedMaze class, which contains and manages an Officer, a Grinch, and a group of items (either Blueprints or Traps). In HauntedMaze, I implemented the draw() method, which draws all the characters and items, and I also included lighting effects, which only reveal parts of the maze that are illuminated by the officer's flashlight. I also coded the whole Officer class, which has the ability to move and pick up Blueprints from the maze. I coded most of the GameScreen class, in which I added support for user interaction with the game (through the keyboard and mouse) and programmed the Officer's movements in response to these inputs. I completed the GeigerCounter class that reports the level of radiation emitted by the Grinch.

Week 3: In GameScreen, I added a time-limit feature which restricts the time that the user can complete the game's goals, and I allowed the user to pause/resume the game using the escape button. For the Grinch class, I found an image to represent the Grinch, and I coded the Grinch's ability to move in the direction of the Officer. To give the Officer an advantage in the game, I included a feature where the user can press the spacebar to accelerate the officer, but this action incurs a cost on its health. Also, I made image handling in the project more organized, and I helped add functionality for collisions between actors in the maze and maze walls. I organized the drawing of InfoBar by positioning relevant screen elements together.

Week 4: In the final stages of the project, I wrote code that turns the background of the window more red as the Officer comes closer to the Grinch, and I made contributions to the Teleporter class. I also created most of the slides for our presentation.

At each checkpoint of the project, I built the javadocs and the jar for our game.

Arindam:

For the first week, 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 completed a significant amount of the readme, such as the full instructions section, classes section, and stretch features. Finally, I worked on the UML diagram with Ishwar and Dhruv.

For Week 2, I completed the options screen class from start to finish, ensuring that a clickable button was made that took the user from the options screen to the game screen. I coded everything that involved switching screens and such. This included drawing methods such as MousePressed. I also completed the info bar class. This involved setting up its constructors, fields, and then also drawing it on the grid and printing appropriate info such as the health, status, and blueprints collected of the officer. Finally, I coded the item and blueprint classes, making them pop up on the screen waiting to be collected.

For Week 3, I set up blueprint pins that showed up in the maze that the officer had to pick up. This involved setting it up such that when the officer came close, it would be able to click E to pick it up. The blueprint would then show up in the Infobar, with an icon and identifier that represented it. In addition, I created hearts! These hearts represent the health of the officer, and the infobar would show the health of the officer as a number which was printed, and perhaps more aesthetically presented as hearts (full/half/empty). I then also coded resize methods in DrawingSurface that allowed the whole program to resize. I added an end screen, which differed based on a winning/losing result. If the user won the game, it showed a winning screen, and if the user lost, it showed a losing screen. Both screens included a replay button that took it back to the option screen. While initially, the option screen would once again lead to the end screen, I tweaked with game screen and option screen classes, and the timer methods, such that timer and the whole game screen was always reset, ‘restarting’ the game. Finally, I vastly improved the option screen and end screen aesthetics, adding images instead of rectangles for buttons, and background images, as well as a custom made logo for our game!

In the final days leading up to our project, I made the full Teleporter class from scratch that involves a teleporting device to move the officer to a blueprint. This teleporter class shows up as an image in the infobar, and can be used once transporting the officer directly to the nearest blueprint to be picked up. Then I coded nearly the entire music / sound effects class, resulting in a spooky soundtrack in the background of the game with sound effects when the grinch gets too close and such. Finally, I made the instructions class which detailed all the instructions on how to play the game. I also created buttons for different types of mazes from the starter screen. I finished off by creating the jar file for our project, and doing javadocs for any remaining public fields or methods.

Dhruv:

Week One: 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(the must-have, want to have, and stretch features) and the class list on the Readme. I also helped organize the initial code structure for all the classes yesterday on Zoom with Ishwar and Arindam.

Week Two I completed the entire MazeData class. This included working on creating the logic to random perfect mazes, as well as reading in mazes from files. The random logic of mazes requires

an exhaustive backtracking depth first search, someone that took a good bit of time. I am very proud of this feature. I also created a grid point class which represents points in the maze. I also completed the Grinch class, which represents the Grinch in the game. The grinch is an actor that is represented in the maze. It follows the actor around, and I attempted/started this logic. I picked some possible images for grinch. I wrote up Javadocs comments for most of the classes. I also contributed to the screen and actor classes. This included setting up variables, creating constructors, and creating some methods. This also meant finalizing how all the classes work with each other.

Week Three:

In the third week I mostly focused on the maze collisions. This mostly meant figuring out how to ensure the actor would not be able to walk through a maze. In order to complete this, I had to write an entire new class called Rectangle. I had to create all the methods in Rectangle such as isPointInside and intersects. I had to create new fields and methods that represented walls in the game. I had to brainstorm edge cases such as when the actor touches two corners. All this was very challenging, and was the hardest part of the project so far. I also coded the Trap class this week. This included adding a trap to the maze, and different types of traps. Some types of traps affect health, while others affect the speed. I had to create logic so it randomly spawned around the maze. I also had to make it so the traps are only visible when seen with the flashlight(sometimes) unlike the blueprints. The blueprints are always visible. Lastly I thought about how I could add encryption into our game, as a want to have feature. I ended up dropping this idea due to the complex nature of it. One final small thing I did was help Arindam design the looks of the different screens so they look somewhat pretty in nature. In the end he chose the final design of the screen, but I provided my drawings of what I think the screen should look like.

Week four:

In week four I mostly worked on the shortest path feature. This is a feature where if the user presses ‘p’ it displays the shortest path from the user to the nearest blueprint. This was quite a complex feature once again, and I had to use/research numerous shortest path algorithms. Eventually I had to code two different breadth first search algorithms in order to find the shortest path from the user to the nearest blueprint. Furthermore, I had to draw this path which took a considerable amount of effort. I helped Arindam set up the audio classes, and helped him pick out music samples for the audio class. I also wrote/revised a significant portion of the javadocs for most classes during this final week.

Sources (links here)

- Grinch image: https://stock.adobe.com/hu/search/images?k=hacker+mask
- Officer image: https://www.istockphoto.com/vector/police-officer-badge-icon-vector-illustration-gm1191176219-337941031
- Losing screen image: https://cdn2.vectorstock.com/i/1000x1000/57/36/you-lose-rubber-stamp-vector-17695736.jpg
- Arindam: add your stuff