Jay Reyes
Tej Singh Pooni
Pranav Sawhney
Bob Lu

Group 3 - Report

Phase 2

Overall approach to implementing the game:

For this Phase, our approach to implementing the game from scratch. We started off by thinking of the components and the parts that our game will consist of. We wrote everything on a piece of paper from a player's perspective. Our game shows the player a home screen which we call as Start Menu. The Start Menu consists of two buttons - Play now and Exit Game. It also displays the title of the game **Maze Game 9000.** The player can either click on the **Play Now** button if he wishes to play the game or the **Exit Game** button if he wants to quit. The game transitions to a new screen called Game Screen if he clicks on Play Now. The player will find a 2D layout of our Game board which has all the components of the game like **Punishments** - Moving Enemy, Bomb, **Rewards**- Coins and Bonus Reward, **Barriers** - Brick Walls, **Start Point** denoted as a blue star and **End Point** denoted as a red star. The game is surrounded by walls on all the four sides. If our player comes in contact with a moving enemy the fail state of the game is evoked where a window pops up with the message that you have failed the game. From here you have 2 options, either quit the game or play again which redirects the player to the start menu. If the player reaches the end point a congratulatory screen pops up with the option to quit the game and play again.

The aim of the player is to travel from the Start Point to End Point escaping all the barriers and enemies while collecting the maximum amount of rewards that come across his path. Each coin is valued 5 points (Changed the values of points compared to Phase 1) and every coin collected adds up to his score. If the player comes in contact with the Bonus reward- Green gems his score bumps up by 15 points. The time it takes him to move is calculated in seconds for each tick of the game and is displayed in the left panel alongside the score. It increases until the player finishes the level or dies.

If the player comes in contact with the stationary enemy - the bomb his score lowers down by 50 points. If the score is negative the player will lose. The player can move up, down, left and right.

State and justify the adjustments and modifications to the initial design of the project (shown in class diagrams and use cases from Phase 1)

From the initial phase of the project, we made notable adjustments and modifications. Looking back at our UML class diagram, we had a main abstract class, Placeable, along with 4 abstract subclasses which each hard were also abstract followed by 9 concrete classes. These classes included Placeables such as Tiles, Entities**,** and Items. For our concrete classes, we had a solid wall, start and end cells, player, enemy, the game board, the regular and bonus rewards, and as well as our punishment. Initially we had 8 use cases such as, moving, pausing the game, reaching the end, touching a moving enemy, touching a punishment, collecting required/bonus treasures and choosing a level.

The modifications we made from this initial design included omitting a formal score changer as initially suggested by our UML class diagram as we can just have a conditional within one of our other classes to see when we reach the end to decide whether the game is ready to transition to the end state. The bulk of the modifications from phase 1 came with our use cases as mentioned above there were 8 use cases but we omitted a few of them. The cases we didn't consider in this phase were the player pausing and choosing a level. The reason as to why we chose this approach was that pausing the game can in some sense remove that improvisation for the player if one pauses and looks at the map for an extended time period the player can devise a new escape strategy to avoid our enemies. Choosing a level was omitted due to the fact that our map generation can be easily tweaked and in addition to that we didn’t want our game to have multiple levels as it would be repetitive as the difficulty is relatively the same.

Jay Reyes
Tej Singh Pooni
Pranav Sawhney
Bob Lu

We decided to change the value of the rewards and punishment compared to phase 1. Regular rewards are now 5 points earlier they were worth 10 points. Punishments will decrease your score by 25 points when before, they'd reduce it by 50 points.

Explain the management process of this phase and the division of roles and responsibilities

We divided the work equally for this phase. Everyone held each other together throughout the building phase of the game. Our team believes in the approach that we win together and we lose together. So all decisions were taken by everyone's consent. It is very hard to say that we worked separately because all the divided tasks have input from everyone.

Jay and Bob did most of the work related to the game screen like implementing the code for making the enemy move, adding coins, making the bonus reward appear randomly in between the game. Whereas, Pranav and Tej were responsible for creating a menu system which covered all transition states of the game (between start menu, win menu, and lose menu) as well as the timer. Additionally, they were responsible for the report for this phase.

We all worked as a team and had no issues working together. Everyone agreed at all points. Our meetings were held on an everyday basis and the time depended on the work that we had to finish on that particular day. We all adjusted to the time differences we had as some members of our team are in different time zones. Somehow, we made it work.

List external libraries you used, for instance for the GUI and briefly justify the reason(s) for choosing the libraries

We used AWT and Swing to do all the rendering and GUI for our game. The reason we chose them was because they are native to the Java Development Kit. This means that they are popular, so there would be lots of documentation and help in their usage. Also, they were far easier to use than compared to, say, LibGDX or JavaFX. This meant we were comfortable with using them. Most of the members of our group were comfortable using it as it is easy to work with in comparison to other

interfaces. It offers a wide range of standard components. The components are lightweight and don’t rely on peers.

Describe the measures you took to enhance the quality of your code

We tried to stick to the UML class diagram we made for phase 1 since we already had an idea of what the class structure of our code would look like. The UML diagram didn’t include packages, so we added some for phase 2 to add encapsulation to our code.

To make our code easily understandable by the reader (whether it be the marker or a team member), we all decided to add comments for all the sections (Java Classes). This helped us understand what other team members wanted to convey through their code. We also decided on creating different branches on our git repository when we added new features. This lets us pull each other's code and see what we were working on. Later on, we merged stuff between our branches.

Discuss the biggest challenges you faced during this phase.

The first challenge was deciding which library we should use to implement the game. When we looked up “java game libraries", we believed complex ones like LibGDX or JLWGL were the ways to go, but then they looked too complex to use. So we just stuck with Swing and AWT.

The second challenge was that it was hard to implement the pathfinding algorithm for the moving enemy since we didn’t know where to start. We tried the methods that we learned in other courses, namely depth-first search and breadth-first search, but they were fruitless. Ultimately, we used the algorithm named A* which worked nicely with our grid-based gameplay. Unfortunately, A* didn’t work out with smooth movement (where you can move say, 4 pixels per frame instead of 32), so we had to remove smooth movement, losing a feature we worked hard on to implement.

Jay Reyes
Tej Singh Pooni
Pranav Sawhney
Bob Lu

The third challenge was implementing in-game states - that is, what happens when you win or die. We implemented the JPanels that showed the win screen and die screen, along with their buttons. The buttons let you return to the menu or quit. A problem arose when you returned to the menu. If you returned then started a new level, the win screen/lose screen would stick around. We tried replacing these JPanels with JDialogOptions, but then you'd have to click on its options twice to close them. We found the solution from a Stack Overflow, which said to remove the game logic from the rendering logic. After implementing that, this stick around error was fixed.

A smaller challenge we faced was how you'd have to take window focus away from the game then back onto it to get it working. By changing the Player's movement method to make it use Key Bindings instead of KeyListener (again, recommended by Stack Overflow), that bug was fixed.

We also faced challenges with the transitions between screens. Initially, the menus created new JFrames when you started a new level or returned to the menu. But this was an easy fix by making the menus JPanel instead of JFrame.