Phase 2 Report

Upon completing Phase One of our game development project, we had a solid grasp of the core ideas and concepts for the game. However, we faced a daunting challenge when it came to implementing the game in Java. None of us had any prior experience in the language, which presented a significant hurdle in bringing our vision to life. When Phase Two was released, we immediately got together as a team to review the new requirements and plan our strategy moving forward. We began by creating the appropriate Maven files and setting up a GitLab folder to serve as a collaborative workspace for our coding efforts.

To ensure everyone was on the same page and making progress, we established a regular meeting schedule every two to three days. During these meetings, each member took turns giving a brief update on their individual progress, including any challenges or issues they encountered. We then merged our individual code branches into the master branches and worked together to resolve any conflicts that arose. One of the most important parts of these meetings was discussing the bugs we encountered and how they affected the game's adherence to the requirements. We would divide responsibilities and assign tasks for addressing these issues, ensuring that everyone had a clear understanding of what needed to be done before the next meeting.

We also used these meetings to discuss broader aspects of the game, such as its design and gameplay mechanics. By working collaboratively and sharing ideas, we were able to create a cohesive vision for the game that reflected the input of the entire team. Despite the challenges posed by our lack of experience in Java, our regular meetings and collaborative approach enabled us to make steady progress toward our goal. Our meetings typically lasted between one and two hours, and they proved to be an essential component of our successful game development project. To tackle the challenge of implementing the game in Java, the first step we took was to conduct thorough research on the internet, which lasted for about two to three days. We scoured the web for tips and resources that could help us overcome the obstacles we faced. Once we had gathered sufficient information, we came together as a team and discussed our findings. This ensured that we were all on the same page and had a clear understanding of what we needed to do to move forward. One of our biggest initial challenges was designing the game board.

When it came to developing the game, we had to carefully consider which libraries to use to ensure that we had the best tools at our disposal. After researching various options, we decided to use a combination of the Abstract Window Toolkit (AWT) and Java Swing. The AWT library provided us with a solid foundation, as it was both lightweight and easy to understand. However, we found that the true powerhouse for developing the game was the Java Swing library. This library allowed us to implement the game's graphical user interface (GUI) with ease, thanks to its built-in features and functions.

One of the key advantages of using Java Swing was that it offered a wide range of pre-built functions for event handling, layout, frame management, and other features. This made the development process much more straightforward and streamlined, compared to other libraries such as Java FX. Overall, the decision to use Java Swing for the game's development was a wise choice, as it allowed us to create a polished and well-designed game interface that was both intuitive and user-friendly. We were able to leverage the library's built-in functionality to create a visually stunning game that exceeded our expectations. With four members on our team, we decided to divide the work between us, with two members working on creating the UI pixel art for the game board while the other two focused on implementing the class diagrams into the master branch initially.

By splitting the work in this way, we were able to make efficient progress toward our goal. The UI designers were able to create a visually appealing game board that met our requirements, while the implementation team was able to effectively translate the class diagrams into code that worked seamlessly with the UI. Our division of labor was a key factor in our success, and we were able to work collaboratively to ensure that everyone was making progress toward our shared objective. Overall, our research and division of labor helped us to overcome our initial obstacles and lay the foundation for a successful game development project.

In order to create a visually appealing game board, we needed to find high-quality pixel art resources that we could use for free. We searched the internet for animal and house tiles that we could use and after combining multiple free-use resources, we were able to gather all the necessary tiles we needed to complete the game. These included the four sides of the mouse, the dumb and smart cat, the floor, wall, outer border, cheese, mouse trap, steak, and more. We made sure to keep all the resources 16 by 16 pixels, but there was a couple of pixel art that we could only find in the form of 32 by 32 pixels. To ensure consistency, we decided to scale them down or scale up all the other resources to match the size of the game. Additionally, all the pixel art we used had no background, so we were able to easily apply them to the game board.

However, we encountered issues with objects such as tables, chairs, candles, and other items that had some black borders since they had no background. To overcome this, we edited the pixels in an image editor by first taking the floor tile and keeping the object on the top layer of it. After adding the new pixel art to the master branch, the game board was fully equipped with all the necessary objects. Initially, our map was simple, so we decided to design a second version of the map where there were more objects spread out on the map. This made the game more challenging and engaging for the user, as it would be harder for them to win the game.

Overall, the process of creating the game board involved a lot of research, creativity, and problem-solving. We were able to effectively gather resources, manipulate them to fit our needs, and design a map that was both functional and visually appealing. Once we had completed the game board, it was a significant milestone that we had overcome. With the game's base complete, the rest of the implementation process went much more smoothly. We were able to divide up the work efficiently among the team members, based on our individual strengths and skill sets. This allowed us to work more effectively and ensure that everything was progressing smoothly.

In terms of the division of roles, Deep Bhimani was in charge of the UI part of the game which included gathering the game tiles, creating the title, instructions, and win and lose screens. Katherine Lee was a backend coder who made sure the player successfully interacts with respective things and that the game is fully designed based on the requirement. Beibei Tang was in charge of creating the AI model for the smart cat who chases the mouse. He was also in charge of the dumb cats and making sure they work. Wendi Xiao helped in all aspects, also part of the backend coder but was mainly in charge of making sure that the objects are correctly working including the cheese, door, steak, and mouse trap. He also worked on displaying the time and the scoreboard.

As we progressed with the implementation, we faced a few challenges and obstacles along the way, but we were able to overcome them through teamwork and communication. We held regular meetings every two to three days where each person would explain what they had done, and we would merge all our individual branches into a master together and resolve any conflicts. We also discussed any issues or questions that anyone had with their responsibility for the game, and we would work together to solve any bugs or glitches in the game. This approach allowed us to stay on top of everything and ensure that the game was working seamlessly and according to the requirements. Overall, the process of implementing the game was a great learning experience for all of us. We were able to learn new skills, overcome challenges, and work effectively as a team to create a game that we were all proud of.

Compared to our initial design from the class diagram, we didn't have the game panel class in a runnable class but we decided to change it and added it into a runnable class. This is because it will allow characters to keep updating when the program is running which will update utilities, objects and etc which will allow the game to run correctly and smoothly according to the plan. Moving on to the initial design of the project based on the used cases, we have made 3 main changes. First, we have removed the dog character which was supposed to scare away the cat helping the mouse. We decided to remove it because we realized we are overcomplicating the game before our original game even runs efficiently. Second, we decided to change the number of cats from 1 smart cat and 3 dumb cats to just 2 smart cats. We also changed from one mouse trap to two mouse traps. We decided to make these changes to make the game more challenging for the user. Lastly, we initially thought that we would just randomly play the steak (bonus points) around the map, however, after reviewing the requirements we realized we had to randomly display it around the map at different times. After implementing all the changes, our game has become significantly better.

To enhance the quality of our code, we used many factory methods to simplify the code and make it more readable. Additionally, we utilized inheritance from the object class to create custom objects and the entity class to create special functional characters, making the code more efficient and organized. During the development of the game, we faced several challenges. The first obstacle we encountered was creating the game board and getting started with the implementation. However, in the middle of the development process, we ran into a major bug where the mouse would get stuck in the walls and move improperly. Another significant challenge was implementing AI for the smart cat algorithm. To do so, we had to create a linked list to store parent nodes in its own children node and move on to the next node. Once we reached the endpoint, we had to backtrack and create many conditions for the cat to make appropriate choices and move accordingly. One of the most challenging bugs we had to fix was ensuring that the mouse moves only one block at a time and that other characters move only when the mouse moves. We realized that this was a requirement and that the characters were moving independently, causing the game to malfunction.

Toward the end, we encountered a bug with the win or lose page where the game would display the losing page when the mouse caught the steak, and the replay button would not allow the user to go back into the gameplay state. We fixed this issue by identifying the root cause and making appropriate adjustments. As we were granted a few extra days, we took advantage of the opportunity to further improve our game. Specifically, we decided to enrich the gameplay experience by implementing several key features. Firstly, we enhanced the text box to provide users with more detailed feedback, such as notifying

them when a door is unlocked. Secondly, we added the option for users to select from two different levels, each with a unique map. Additionally, we incorporated a new button that enables users to easily return to the home page, creating a seamless and immersive gaming loop.

To elevate the overall quality of the game, we divided the map into four quadrants, with a variable that stores a value from one to four. Whenever an object spawns on the map, the game now randomly selects one of the four quadrants and then spawns the object randomly within that quadrant. This effectively resolves the issue of objects clumping together, resulting in a more balanced and enjoyable gameplay experience. Despite these challenges, we were able to tackle them as a team by utilizing each individual's strengths and working collaboratively. In the end, we were able to create a successful game that met our expectations and was enjoyable for users to play.