REPORT

Overall Approach to Implementing the Game

The overall approach to implementing the game is structured across three key milestones, each focusing on specific aspects crucial for the development process.

Milestone 1: This initial milestone focuses on learning Java fundamentals essential for the game development while also in initiating the game's foundation. It involves gaining proficiency in Java programming, with a focus on drawing and animation techniques to lay the groundwork for the game's visual aspects. Dividing the work among team members allows for a focused approach to mastering these fundamental skills.

Milestone 2: Building upon the groundwork laid in Milestone 1, Milestone 2 focuses on implementing core gameplay mechanics and features. Key tasks include enabling player movement, enemy movement, and object collision detection. Additionally, implementing entity interactions and collision detection enhances the game's complexity and immersion. Integrating game states and user interface (UI) screens cater to seamless transitions between different game states.

Milestone 3: The final milestone revolves around testing and refining the game's logic and adding essential features to complete the gaming experience. This includes implementing point systems, victory conditions, and reward collection mechanisms. Ensuring a polished user interface and incorporating feedback loops for player progression enhances the overall gameplay experience.

Adjustments and Modifications to the Initial Design of the Project

The adjustments and refinements made to the initial project design reflect an ongoing learning journey geared towards improving efficiency and streamlining implementation. One notable change we implemented was expanding the design to include additional features like rendering, enriching the game's visual appeal and enhancing player immersion. This extension aligns with our team's goal of crafting a polished and captivating gaming experience.

Another significant modification involved removing the position attribute from the Cell class. This decision arose from the realization that simpler methods exist for handling positioning within the game without the need for aggregation.

By eliminating the position attribute, we simplified the overall design structure, reducing unnecessary complexity and potential confusion. Moreover, this adjustment was justified by the understanding that the position of cells on the board isn't directly utilized within the game's mechanics, making it redundant to the core functionality.

In addition to these adjustments, we removed the concept of "max score" from the scoreboard, opting instead to count "meat" as it provided more meaningful information about

the state of the game. The removal of "max score" was driven by the realization that it didn't offer sufficient insight to the player regarding their progress.

Furthermore, we decided not to use the board controller and board view, opting instead to merge rendering into different entities and game states. This decision allowed for a more cohesive integration of rendering with various game elements and states.

Moreover, we modified the game to manage different states rather than simply playing the game. By introducing different states that run the logic for the game when the game is in that state, we were able to handle the game's logic more effectively and maintain a modular and scalable architecture.

To ensure a uniform way of interacting with points, most entities that interact with the player now implement an interface called "interactable." This provides consistency in how points are handled, avoiding the need for specific punish and reward functions.

Finally, we added an AI package to assist with enemy movement, enhancing the game's complexity and providing more dynamic gameplay experiences. These adjustments and modifications collectively represent a proactive approach to refining the project design, prioritizing simplicity, efficiency, and clarity in implementation, while also adapting to evolving requirements.

Management process of this Phase and the Division of Roles and Responsibilities

Throughout this phase, the team implemented a structured management approach to streamline coordination and monitor progress effectively. We convened three times a week to review our advancements, tackle any obstacles encountered, and strategize our next steps. Additionally, to uphold version control and foster collaboration, we established a protocol requiring team members to announce each code push to GitHub on Discord. This practice served to keep everyone informed about the latest developments and encouraged team members to synchronize changes with their local repositories promptly.

Communication emerged as a cornerstone in project management, with Discord and Zoom serving as primary platforms for discussions. These tools facilitated real-time collaboration, allowing team members to seek clarification, provide updates, and exchange ideas seamlessly. Furthermore, our regular meetings served as invaluable checkpoints to monitor progress, identify any bottlenecks, and adjust our priorities as necessary, ensuring that the project stayed on course.

The approach to implementing the game follows a structured progression divided into three key milestones. In Milestone 2, the primary focus was on implementing the game itself. This involved delving into Java's essential functionalities while concentrating on pivotal aspects such as drawing and animation techniques crucial for game design.

Each team member assumes distinct responsibilities tailored to their strengths and interests. Pardeep spearheads the development of the game engine, game loop, game state management and leveraging abstract classes and entities to streamline the process, while also leading game camera and level construction. Tom focuses on UI, housing, and state management, ensuring a seamless and intuitive user experience. Will takes charge of knight, knight movement, UI components such as the timer, and collaborates with others on reward mechanisms and play state logic. Manya undertakes the creation of enemy entities, including the Goblin, Goblin movement, Trees, Meat, and Dynamite classes, crucial for adding depth and challenge to gameplay. She also handles scoreboard management, AI movement, play state logic, coin, dynamite, and the comprehensive report, ensuring thorough documentation and reflection on the team's journey and achievements.

With this collaborative approach and a focus on transparent communication and clear division of roles, our team remains poised to lay the groundwork for an engaging and visually captivating gaming experience, marked by meticulous attention to detail across all game elements.

External Libraries

Currently, our project stands completed without any integration of external libraries. This decision underscores our commitment to simplicity and adherence to our learning goals.

Enhance the Quality of Code

In our efforts to improve the quality of our code, we've taken several important steps. First off, we've made sure to give our variables, methods, and classes clear and easy-to-understand names. This helps everyone on the team follow along with what's going on in the code. We've also used comments a lot, including simple ones called JavaDoc comments, to explain complicated parts of our code and how different methods work.

We've been careful to stick to the same style and format throughout our code, following the best practices in the industry. This makes our code easier to read and work with, which is important for keeping it in good shape over time. Another thing we've done is breaking our code into smaller, reusable parts whenever possible. This not only makes it easier for us to collaborate but also saves time in the long run.

To help everyone understand how different parts of our code fit together, we've used diagrams called UML class diagrams. These show the relationships between different classes and how they depend on each other. We've also been using a tool called Git to keep track of changes in our code and make sure everyone's work is coordinated.

Lastly, we've been regularly reviewing each other's code and writing detailed documentation. This helps us catch mistakes early on and makes it easier for new team members to get up to speed. Overall, these efforts have created a more collaborative and accessible environment for everyone involved.

Challenges

Navigating the intricacies of Java and game development posed a significant particularly due to the demanding schedules and assignments from other classes. Many team members found themselves grappling with implementation of Java concepts for the first time, requiring extra dedication to grasp the practice of fundamentals. This learning curve was exacerbated by the competing demands of coursework, making it challenging to allocate sufficient time to the project. Striking a balance between academic responsibilities and project commitments proved to be a daunting task, slowing down our progress and causing occasional frustration.

Despite these obstacles, our team persevered, demonstrating resilience and determination in the face of adversity. Through collaborative efforts and effective time management, we managed to make steady strides forward. This experience underscored the importance of prioritization, adaptability, and perseverance, strengthening our resolve and deepening our understanding of Java and game development principles.