Summary (Design)

Table of Contents

Summary

In our design documentation, we have created a blueprint that integrates structural planning, user interface (UI) design, data management, and strategic software patterns to create a robust and user-friendly software application. The foundation of our design is laid out in the class diagrams, which describe the intricate relationships between classes, detailing their attributes, methods, and visibility. This ensures a coherent structure that aligns with our software's core functionalities and objectives.

For the UI of the game, our design is visualized through detailed mockups created in Balsamiq, emphasizing easy navigation and seamless user experience. These mockups serve as a visual guide, showcasing the thoughtful arrangement of interactive elements and the logical flow between screens. The aim is to make the application not only functional but also engaging and accessible to users. For data storage, we chose the CSV format for its simplicity and efficiency. This decision highlights our commitment to reliable data management and interoperability, using CSV's straightforward tabular nature to easily handle the data within Java. Our architectural and design approach is further enhanced by the selection of software patterns, chosen for their ability to optimize performance and maintainability. The Model-View-Controller (MVC) pattern ensures a clean separation between logic, presentation, and control, promoting modularity. Lazy Loading enhances the application's efficiency by delaying object initialization until necessary. The Flyweight pattern optimizes memory usage by sharing common objects, ideal for scenarios with numerous similar objects. The Bridge pattern separates the abstraction from its implementation, allowing flexibility across different platforms. Lastly, the Command pattern encapsulates requests as objects, offering the flexibility to manage operations, undo actions, and more.

Together, these elements form a cohesive design strategy that not only meets the current project specifications but also lays a flexible foundation for future enhancements. This integrated approach, combining a solid structural blueprint with a user-centric UI, efficient data management, and strategic software patterns, highlights our team's commitment to delivering a high-quality, maintainable, and performant application.

Middle school students of today have already lived a life with technology all around them. The internet has been present in their life since they were born, they were likely introduced to it and other software as soon as they were born. Many students today use software in school, using apps like Google Drive to store their school work. Very few assignments are even written by hand these days, students have to know how to use a computer and the software on it from a young age. Even after school, they'll rewind by scrolling through TikTok, watching a video on YouTube, or perhaps playing Minecraft with their friends from school. These examples illustrate how large of a role software will play in these students' life, it will be constantly present from the time they're born. If software is going to be such an important part of these students' lives, then they must learn the foundational skills needed to give them an understanding of how software is created. Coding is already becoming a part of the basic curriculum in public schools across Canada. Having a strong foundation of the skills that software engineers use every day when designing software can only be beneficial as children grow up in a world increasingly centred around tech. Software isn't going anywhere, as current middle school students graduate school and enter the professional world they do so in a world that is beginning to see artificial intelligence play a large role. The emergence of AI is only going to increase humanity's reliance on software, more job fields are going to require a base understanding of software as it becomes a larger part of all fields.

Our game is a cartesian puzzle-solving game. One where middle school and secondary school students will be challenged to put their logic and math skills to the test to solve a series of mazes. In doing so, they will be taught the basics of programming and have to put these newfound skills to the test to complete our game. The skills they learn in our game will equip the students with skills that will help them navigate a world that will only continue to see tech become an even more integral part of our lives.

Terms, Notations, Acronyms

Terms	Meaning
GUI	Graphical user interface.
Screen	A self-contained GUI area that has a unique set of buttons/ text boxes.
Menu	An overlay that can be displayed on top of any screen.
Popup	A menu that opens in a new window.
Stage	A self-contained puzzle that has a set of starting conditions, available action blocks, and objectives. Completion of earlier stages is required to access later stages.
Level	Level of difficulty. Several stages may be used to teach & test the same concept at different levels of difficulty.
Action Block	A rectangular GUI element can be chained together to make the character perform a series of actions to complete the objective of a stage.
Action Chain	An ordered list of action blocks that is executed from left to right by the gameplay logic.
AI	Artificial Intelligence.
CLI	Command line interface.
ASCII	American Standard Code for Information Interchange, which is a character encoding standard.