CSCI 4448 - Project Part 4: Final Report

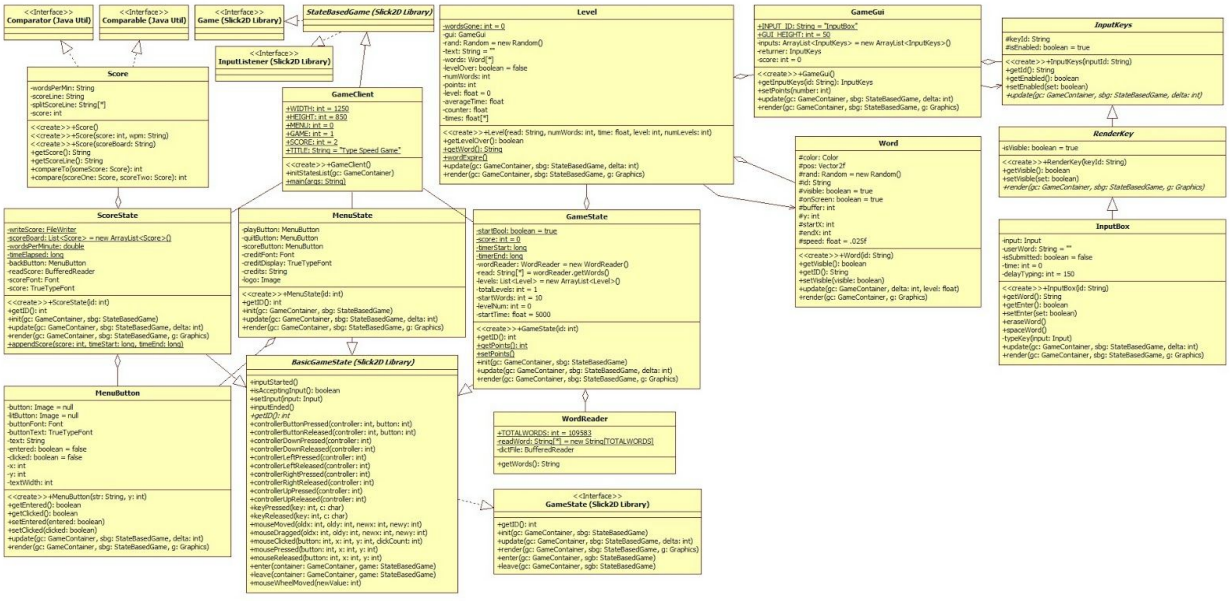
Team: Brady Auen, Nicholas Nocella, Peter Huynh
Title: Type-Speed Game

Question 1:

What features were implemented and a class diagram showing the final set of classes and relationships of the system. (This may have changed from what you originally anticipated from earlier submissions). Discuss what changed in your class diagram and why it changed, or how it helped doing the diagrams first before coding if you did not need to change much.
List the features you implemented (from part 1 and 2 where you listed the features you were planning to implement / requirements you listed) and show the final class diagram, discuss benefits of designing before coding.

Answer:

Final Class Diagram:



Most features and functionality remained the same as to what we planned to implement during the design process of this project. Our team managed to satisfy most of the established requirements that were defined in the project's part 2 document. However, there were some minor changes that were made during our implementation. For example, we had a requirement for difficulty selection within our user requirements,

but changed how difficulty changed within the game. Instead of a difficulty selection button in the main menu, the game progressively gets harder as you play. Each level speeds up and more words will fly past the screen. This allowed for automation and quicker user interaction.

Other than that, we implemented every other requirement that was planned. The game starts on a main menu screen with options to look at the leaderboard scores of the game, play the game itself, and quit the client. The game is divided into states based on which screen the user is viewing, and each state is responsible for different functionality. Our final class diagram remains similar to the original planned diagram, but with some minor additions. There is a separate score class, used as a calculated variable for the scoreboard, and each state is a child of an abstract class implementing an interface, instead of each state directly implementing that interface.

Designing prior to implementation really helped out in the grand scheme of the project. Having the class diagram in particular provided the team with somewhat of a skeletal structure of the whole program, and we referred to it when creating each class. The designing process acted as a guide for this project, and we would've been lost without it.

Question 2:

Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?

Answer:

In our final prototype we used the State design pattern. Other than rendering the graphics on the screen, the main functionality of the game is divided into three basic game states: Menu, Game, and Score. each state represented a screen state within the game, where the client would start within a menu state and can jump to the other two. Within the Slick2D library, there was an abstract class and interface that allowed for a state based java game. Our team created state classes that inherited from the abstract class, and used the primary methods within the given interface.

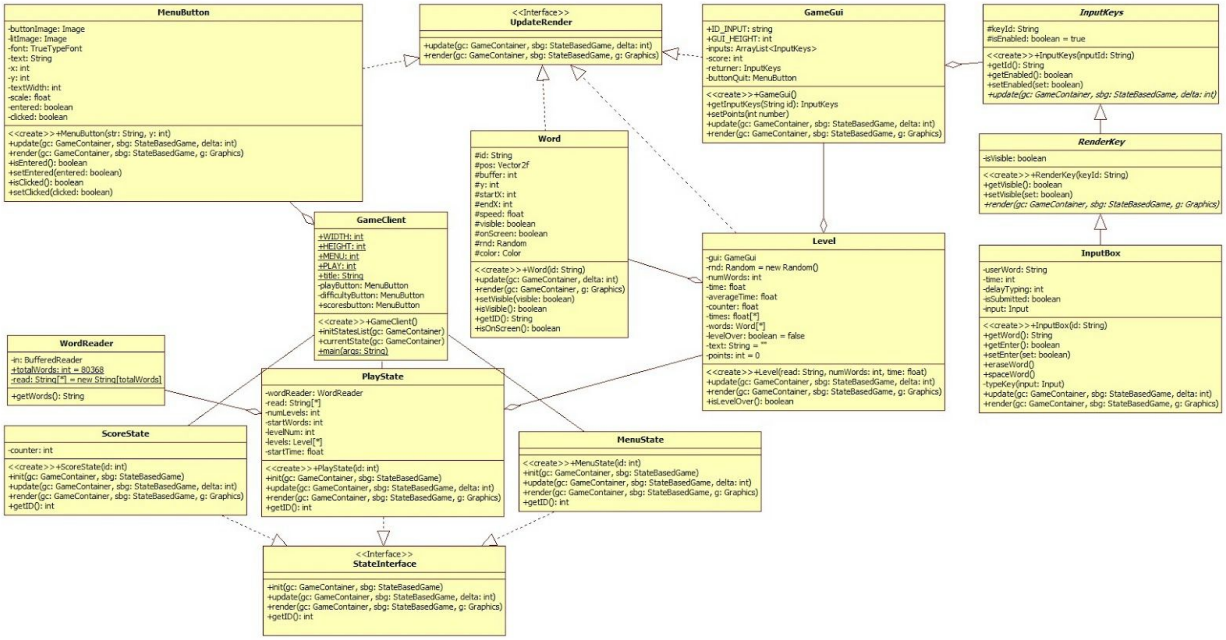
Question 3:

In addition, the report must discuss how the final system changed from the design you presented in Project Part 2. In particular, include the class diagram you submitted for Project Part 2 and use it to compare and contrast with the class diagram that represents the final state of the system.

Compare your Part 2 class diagram and your final class diagram - include part 2 class diagram and point out the necessary changes. You can answer why you didn't realize you needed things during the design and/or how next time you will be able to make a better design after this learning experience.

Answer:

Part 2's Diagram:



Compared to our final class diagram, our original plans for the programming structure were slightly different. Both diagrams are structurally and behaviorally similar, but there were minor organization and additions. For example, the class MenuButton wasn't instantiated in GameClient in the final prototype, but was made into an instance inside two of the three state classes. We also did not implement a StateInterface or UpdateRender interface. Instead, our team used Slick2D's library to provide the state behavior for the program. In our final diagram, we also have a Score class that handles the leaderboard scoring.

Primarily, the reason we didn't implement the two interfaces was because Slick2D provided similar classes for us to use. This helped us by easing the implementation step of the project.

Question 4:

What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

Answer:

We learned that the analysis and design process of this project really did aid us in creating our word typing game. Although, going through the process was long and arduous at times, especially creating diagrams, it allowed us to visualize how our prototype would work before coding. The coding flowed much more easily, and faster rather than the typical way of programming assignment, where almost no analysis or designing is done beforehand. Instead of writing survival code, there was structure and a clear set of self imposed goals that were established. Overall, the process of analyzing and designing each aspect of a programming project helped pave the way for the group project.
