6.3. CSC and CSU Descriptions F

The CSCs are launchers, modifiers, and displays. The launch CSUs are launch_menu, launch_game screen, launch_leaderboard. The modifier CSUs are sense_touch, sense_death, location, submit_name. The display CSUs are display_avatar, display_obstacles, display_form, display_current_score, display_server_scores, display_user_scores.

6.3.1 Class Descriptions F

The following sections describe the details of LineGame's classes.

6.3.1.1 Main Menu class:

The main menu class has buttons that launch the game screen or the leaderboard page. This class is the "backbone" of the application that controls the main navigation through the features.

6.3.1.2 Game Screen class:

The game screen class senses touch and death and displays the current score, avatar, and obstacles. This is the main class that greatly affects the overall user experience, since the game is the main feature of the application. This class receives feedback (user touch) which affects the return of the avatar display. When it senses a collision with the avatar and obstacle display, which signals a "death", this class launches into a death screen. The current score is displayed independently of any of the other displays as it is controlled by time (longer time played = higher score).

6.3.1.3 Avatar class:

The avatar class's main attribute is its location. The location is affected by the touch sensed by the game screen. This class modifies its location based on that feedback.

6.3.1.4 Obstacle class:

Like the avatar class, this class's main attribute is its location. However, instead of relying on user feedback to determine location, this class modifies the locations of the objects based on an algorithm.

6.3.1.5 Death Screen class:

The death screen is a "shortcut" menu that allows the player to start a new game after "dying" without returning to the main menu. It displays the score that the player reached and buttons that give the player of restarting a new game immediately or returning to the main menu.

6.3.1.6 Leaderboard class:

The leaderboard class gives the option to display the top 10 server or user scores.

6.3.1.7 Server class:

The server class holds every users' scores and maintains them in descending order.

6.3.1.8 Screenname class:

The screenname class contains a form that allows the user to set an alpha-numeric 3-character “screenname” to use for their scores.

6.3.2 Detailed Interface Descriptions F

6.3.1.1 Main Menu class:

The main menu interacts with the other “page classes” through a double-sided launch. The main menu can launch the game screen or the leaderboard. The game screen and the leaderboard can lead back to the main menu.

6.3.2.2 Game Screen class:

The game screen class transmits the touch feedback to the avatar class and displays the location that it receives. It displays the current score that is generated by a time-controlled formula. It also displays the locations that it receives from the obstacle class and watches for collisions. Upon sensing a collision, the game screen triggers a launch into the death screen.

6.3.2.3 Avatar class:

The avatar class interacts with solely the game screen, receiving touch feedback and returning a modified location.

6.3.2.4 Obstacle class:

The obstacle class generates its own locations and transmits that data to the game screen.

6.3.2.5 Death Screen class:

The death screen launched by the game screen when a collision is detected. It can relaunch the game screen or launch the main menu.

6.3.2.6 Leaderboard class:

The leaderboard class can either display the top 10 server (everyone) or user scores. It sends the query to get this data from the server.

6.3.2.7 Server class:

The server class receives a query from the leaderboard class and either sends the top 10 server (everyone) or user scores.

6.3.2.8 Screenname class:

The screenname class receives user input and sends that data to the server to set a custom screenname for that user.

6.3.3 Detailed Data Structure Descriptions F

3.6.3.3.1 Obstacles:

The obstacles will be maintained as a double-linked list of nodes. The nodes contain a location tuple. The algorithm for moving these obstacle nodes will alter each node individually to create the increasing difficulty of “faster” obstacles. The list must be doubly-linked in order to prevent the collision from obstacle-to-obstacle.

3.6.3.3.2 Server:

The scores will be stored using mongodb in the format: user_id (pk), screenname, top 10 scores.

6.3.4 Detailed Design Diagrams F

[uploaded separately]

6.4 Database Design and Description

The database will be a mongoDB database which will hold two tables, a user table which holds the user's personal scores and data, and a server table which will hold top scores across the server and have a pointer to the user for more information. The goal of the database is for score logging and allowing users to compare themselves to other to create a simulated competition between users, and to allow the users to track their progress in the game.

6.4.1 Database Design ER Diagram

[uploaded separately]

6.4.2 Database Access

The users will only be able to access the database on one screen of our application. Access will be limited to the score screen where users will be able to change their name, see their top scores, how many times they've played, and their personal high score. There will also be another page where the users can view the top 10 scores on the server, and compare their highscore to the other top users. These call should all be read calls, except for the name update, to the database asking it for information to display on the app. When the user changes their name the database will update their name to the newly input one, changing the username field. The app itself will send scores to the database after a game ends to attempt to update the top scores, but the user will have no control over the data being sent.

6.4.3 Database Security

The database holds no important information, so we hope it will not be the target of any attacks, but we plan for all of the write queries to handled automatically by the application, so the user will not have the ability to try and get into our database. The only possible problem could be the username update, but we plan on having a 3 alphanumeric character limit on the username. This will make security easier as well as serve as a design callback to retro arcade games. A possible solution we are considering testing is

that the username will not update until after the user plays a game, so they cannot continuously send updates to the database in an attempted DDOS style attack.