APCSA Final Project Design Document

1. Intro

Period: 2
Names: Emily Ng & Andrea Wang
Group Name: Crossy Road
Project Title: Crossy Road

2. Description

Describe your project:
We will be replicating the video game "Crossy Road," where the player controls a chicken and navigates it through roads and rivers. The chicken can die from obstacles, like cars and drowning. It can also die if it fails to keep pace with the moving camera. The objective is to gain as many "points" as possible by staying alive and progressing.

List and explain functionalities:

- **move function** - The chicken is controlled by the user's arrow key inputs. Depending on the user input, the chicken's position on the window will change, taking deadly, neutral, or blocking obstacles into consideration.
    - If a player travels in a horizontal direction, the Chicken's drawing "turns" its body to the respective direction of the key pressed
- **addTerrain function -** randomizes the obstacle objects, like rivers, roads, and railroads.
- **makeAvatar function -** in each subclass of Terrain, as well as Chicken. Generates each object's avatar that we developed through shapes in Processing.
- **makeAvatars function -** iterates through the ArrayList background and calls makeAvatar for each object
- **advanceCam function -** iterates through ArrayList background and adds 50 to each object's yPos, shifting it down 50 pixels
- **danger function -** iterates through background, the ArrayList of obstacles. If an obstacle's response is 3, it is a dangerous river, and the obstacle's position is checked in relation to the Chicken's position to see if they overlap. If they overlap, the method returns true, and when the player steps on this river object, it will die (the game will end).
- **onLily function -** similar to the danger function, if an obstacle's response is 1, it is a neutral lilypad, and the obstacle's position is checked in relation to the Chicken's position to see if they overlap. If they overlap, the method returns true, and the player will not die when stepping on the lilypad.
- **autoCrash function -** if a Chicken's current position already overlaps with a car's current position, they "crash" and the Chicken dies.

- **crash function -** if a car's position and Chicken's position overlap while moving, they "crash" and the Chicken dies
- **autoBlock function -** if the response of an object is 2 and its current position overlaps with the Chicken's current position, the player is automatically moved down/away. Hence, the player is blocked by this object, since the Chicken is not immediately killed but cannot move forward into the object.
- **block function -** iterates through background; returns true if xPos and yPos of the player overlap with the current object, based on the direction of movement
- **dieOffScreen function -** if a player stays on the first row of the screen for longer than 5 seconds, they automatically die. This is to encourage the player to keep advancing forward in the world.
- **die function -** game ends when player "dies" which happens as a result of numerous instances, as stated in their respective functionalities above and below
- **autoMove function -** automatically moves the player downwards (by calling the move method in the down direction) so it moves towards the bottom of the screen along with the rest of the Terrain obstacles
- **points function -** counts points based on how long the player survives; The points are displays at the top left of the screen
    - using text() method in draw() and getPoints() method
- **addLilypad function -** adds random number of lilypads to a river (between 4 and 12, inclusive); adds all of these lilypads to the ArrayList lilypads as is an instance variable of the River class
- **roundX function -** returns int that represents the xPos for the closest 50 by 50 square in the 600 by 600 board (12 by 12 squares of side length 50)
- **roundY function -** returns int that represents the yPos for the closest 50 by 50 square in the 600 by 600 board (12 by 12 squares of side length 50)
- **objects:**
    - **camera object** - it will keep track of the window's pixels and will continuously shift the viewer's perspective up. If the chicken is no longer in the moving camera's view, it dies. (Although the real game's camera moves diagonally, for time constraints, we simplified it to vertical movement.)
    - **tree object -** serves as a stationary obstacle for the character
    - **rock object -** serves as a stationary obstacle for the character
    - **road object -** span of ground where cars are the only obstacle
    - **car object -** serves as a moving obstacle, can only be on the road
    - **river object -** span of ground that if the character touches it, the game ends; logs serve as safe zones in rivers
    - **lilypad object** - stationary safe zone in a river, smaller surface area than log, can only be on rivers

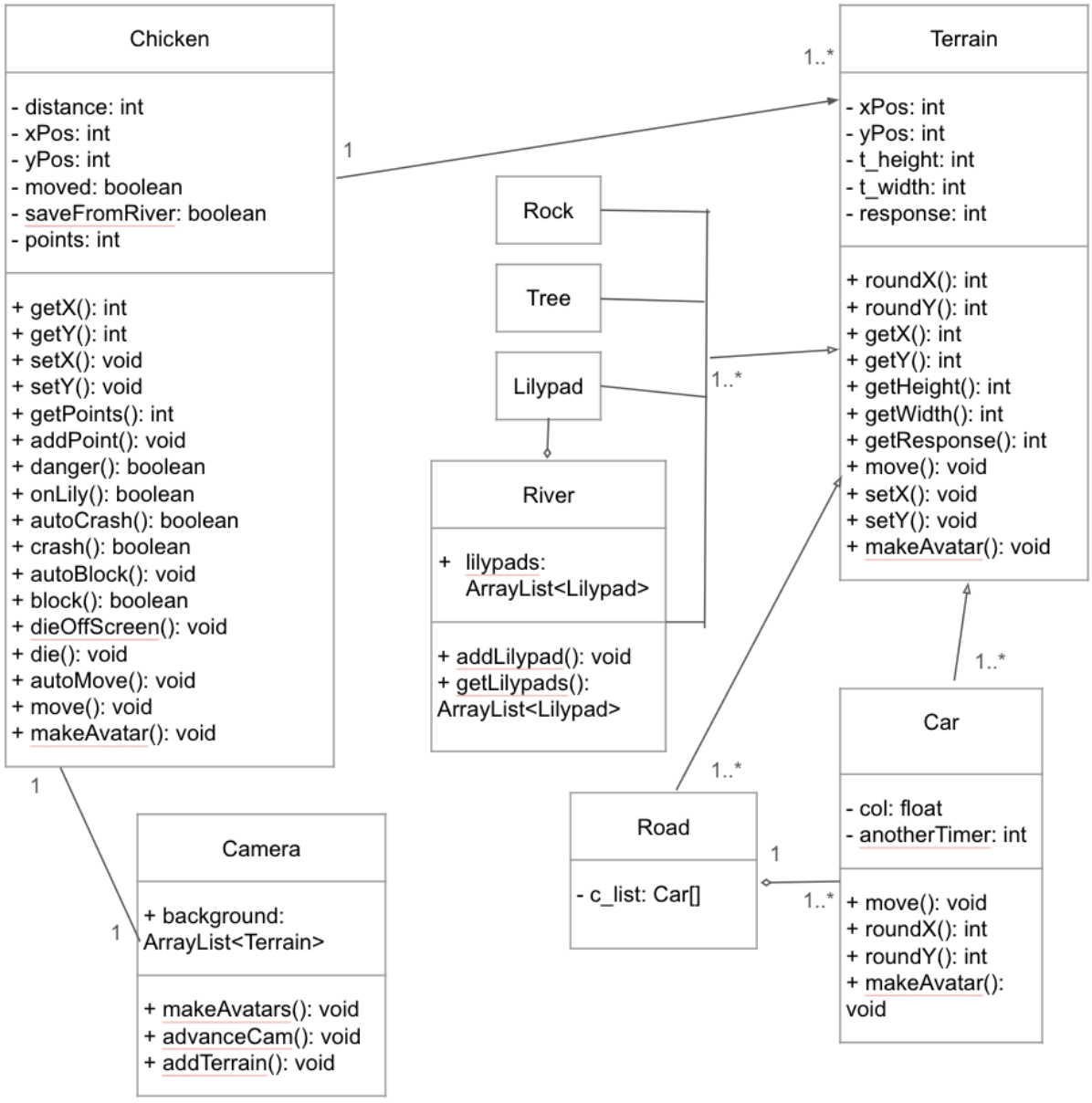
Any libraries needed, indicate the name and how you are using it in your project

N/A

Anything else you would like to add about your project

This game has been simplified to meet the realistic time constraints for this project. These simplifications include a camera movement that advances by 50 pixels as opposed to smooth movements and fewer objects in the background than in the actual game.

3. UML Diagram



4. How does it work?

Explain the steps the user should follow to run and use your project (game, simulator, etc.). For example, if you implemented a game, you should indicate the objective of the game and how to play (keys you need to press or any detail that helps the user figure out what to do when playing your game).

Objective: Survive as long as possible by dodging obstacles and remaining in the game's viewing window.

How to play:

1. At the beginning of the game, the chicken will automatically be spawned. Use the left, right, up, and down arrow keys to move your chicken through the obstacles.
    a. For roads, avoid cars.
    b. For rivers, only step on lilypads. Your chicken will drown if it falls in the river.
    c. Rocks and trees will block your movement, so be sure to go around them.
2. You must continue moving forward. The chicken will automatically move towards the bottom of the game's window. If the chicken falls at the very bottom or outside of the game's view, it dies and the game closes.
3. As time progresses, you will collect points. The points are displayed in the top left corner. Try to get as many points as you can.
4. Good luck and stay alive!

The following section should be added only for the final document:

5. Log

Please describe how each group member participated in the implementation of the project. You may indicate the functionalities implemented by each collaborator.

Design document planning - both
UML diagram - both
Terrain, river, lilypad, tree - Andrea
Chicken, car, road, rock, tree - Emily
Game - both
Camera - Andrea
Response (how the obstacle affects the player - deadly, blocking, or neutral) - Emily
Points - Emily