Albert Chang
Andrew Chen
Joanna Clark
Steven Tran
Valeria Vikhliantseva

ICS 168 Design Document - Frogger

1. Our choice of game was Frogger. We thought that it would be a fun challenge compared to Bomberman or Tetris (which already have multiplayer game modes); also, by making the game multiplayer, players would be forced to make more risks (as the game will be similar to a race), especially if they are behind, adding a new layer of depth to the classic game.
2. The player can move left, right, up, and down, one tile for every press of the arrow keys. The goal is to get to the goal without running into cars or falling into the water. The game objects besides the frog character are cars, which move in a straight line with fixed velocity from one side of the screen to the other, and wood, which is slower than the cars and can only spawn in water tiles (but besides that moves very similarly to cars); the main difference is that cars kill the player, while wood objects must be walked over to cross the water without falling in. There are also car and wood spawners that randomly creates an object at one of the spawn points, but must first check whether a certain amount of time has passed from the previous spawn.
3. At the time, some ideas to make the game multiplayer on each client would be to transform it into a sort of race game. Players would attempt to reach the end of the screen the fastest and avoid the obstacles the game creates (while the camera follows them), and in addition there would be a tongue mechanic that allows players to stun others for a short period of time (by pressing the W, A, S, or D key in the direction the player wants to shoot the tongue, at the cost of being forced to stop their own movement). There may also be power-ups, such as a speed up or an ability to use the tongue without stopping. At the moment, we are thinking about tracking player information with a username/password system with information held on a file where the server runs.
4. We plan to use client-server architecture, where the players connect from separate end hosts and the server runs from an independent host. The clients would be rendering all of the game, while sending to the server messages of the location of the frog and whether or not the frog is stunned, or is trying to stun another player. The server will be authoritative and run the whole game by taking care of the spawning of obstacles, goals, etc.
5. We will be using the Transport Layer API for unity in C# for the networking portion of the code. This API is variable enough to allow for us to be flexible about how we handle

the connections, while also not abstracting too much of the Unity networking functions to allow us to grasp the idea of coding the networking system.