CS21 Project Proposal

March 5, 2022

Team Name:

The name of our team is Mark's Sharks.

Team Members:

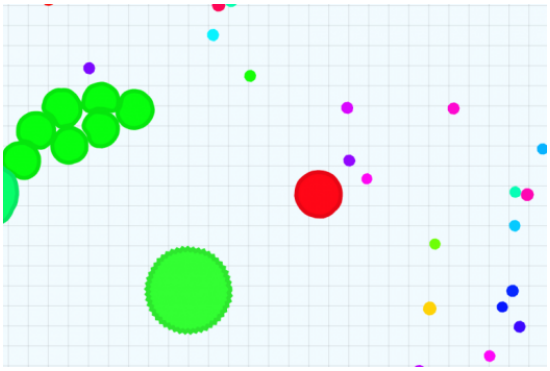
Our team consists of four members — Ankur Dahal, Ellis Brown, Jackson Parsells, and Rujen Amatya.

Project Description:

"SharkIO"

Using Python3, we will be creating an online game with similar game mechanics to the popular online games "agar.io" and "snake.io", which are online, browser-based games which allow for 1-20 people to play concurrently. Our game involves moving a blob/snake around, eating food to grow in size, and eating other players to grow in size.

A screenshot from the online games that are giving us inspiration for SharkIO can be found here.



A player will control a single blob, and can move around in 2-dimensions on the game board. There will be obstacles that the player should avoid in order to not lose their mass.

Minimum Deliverable:

Our minimum deliverable will be a single player game with randomly moving “players” that will start up when the player joins. We’ll use a concurrency based python backend with each thread representing a player. To consume another player, you must be currently larger than that player; to gain initial mass, you must eat randomly scattered bits. We’ll also introduce stage hazards that will damage you if you are larger than it. In our minimum deliverable, we plan to focus on just getting the game working, so animation will not be present in any place not explicitly necessary, and the players will stay the same size despite how much they’ve consumed (score will be written on the player’s avatar though and will be updated as the game plays on).

Expected technologies to be utilized:

- Python backend (possibly with Django or Flask)
  - The python backend is what enables concurrency
- HTML + CSS + Typescript frontend (possibly with react and next)
  - Web based interface for user to play

Maximum Deliverable:

Our maximum deliverable/goal would be to implement the multiplayer mode, with up to 20 players, and our in class demonstration would allow every student in the class to play concurrently. We are aiming to add animations and graphics to show the scores of individual players, and also have a global leaderboard to display the current leader in the game room. We plan on allowing multiple rooms of the lobby to exist that are independent with their own players. If time permits, we plan on allowing for spawning processes across multiple hosting web servers as well.

First Steps:

The first steps would be to focus on the backend and the game logic, which would mean setting up the server using Flask or Django. We would make choices about the concurrent models in our game, and the local / shared data they contain. To do this, we would sketch out the architecture of our web application and create a model of the data flow for all entities involved.

Potential Problems and Questions:

- Using web sockets for multiplayer mode
- Python GIL for scaling into players above ~10
- Shared gameplay for more than 1 user
- Animations and rendering graphics
- Any unforeseen technical issues because of using a production server to host the web app (shared state among multiple processes, etc.)