CPSC 416 Project 2 Proposal

Jerome Rasky, Madeleine Chercover, Raunak Kumar, Vaastav Anand

A Distributed Game

We're interested in building a distributed game for our term project. We feel that this area is interesting because it introduces real-time constraints into our distributed system. Namely, a game requires low latency. Whereas the blockchain can afford to take ten minutes to confirm transactions, players might be upset if it takes ten minutes to move at all in a game.

There is prior art in the area, and some professional games such as Destiny are based on distributed systems. Clock synchronization is likely to be a topic of interest in our system, which means that we can apply some of the topics we've learned in class. We'll have to do more research on how best to match peers, which might involve a central server like in project 1, or some kind of distributed way to bootstrap into the network.

In order to keep track of game state across the network, we might make use of conflict-free replicated data types, such as a vector clock. We'll have to work hard to ensure low latency, which will mean using networking tricks and whatever else is otherwise available to improve latency.

Game Mechanics

We'll build a game not unlike Tanks, where players can move around and shoot at each other. The last player standing wins. Initially, our game will be very simple, without any health or stage hazards or anything. We'll grow our game mechanics as time goes on, adding things like stage hazards, status effects, and alternative goals. With this approach, we can grow the complexity to an almost arbitrary degree by adding features, but we can also scale back our ambitions if we run out of time.

Stats

We'll keep track of player stats, such as a k/d ratio, using conflict-free replicated data types. This will provide a dimension of distributed systems design that is less latency-bound than the regular game mechanics.

How we'll use the cloud

We'll have a centralized server for peer discovery, which will be hosted on Azure. We also plan to have a "backup" client on Azure, so that the game state persists even if no clients are online.

Technology stack

We'll use go for nearly everything. We'll use a simple 2D game engine, so that we don't have to work too hard on graphics. We'll use a games library to facilitate that development, but otherwise we'll stick to relatively standard go.

SWOT Analysis

Strengths

- Team members have worked with each other on the previous assignments of the course.
- Team members are diligent and punctual.

- All members are good at researching as well as solving potential issues.

Weaknesses

- There are very limited resources available for building a low-latency distributed game.
- None of the members have any prior experience with making multiplayer online games.

Opportunities

- We can make the game as complex as we would like given the time limitations that would allow us to expand on the distriuted system we are aiming to make.
- Conflict-Free Replicated Data Types is a very new concept for all of the members and would require a lot of work for a proper implementation.

Threats

- Commitment to exams or assignments from other courses may interfere with progress.
- None of us have any experience with GoVector, ShiViz or Dinv, all of which we are aiming to incorporate in our project.

Resources

https://www.cs.ubc.ca/~gberseth/projects/ArmGame/ARM%20Game%20With%20Distributed%20States%20-%20Glen%20Berseth,%20Ravjot%20%20%20%20%20%20Singh.pdf

http://www.it.uom.gr/teaching/distrubutedSite/dsIdaLiu/lecture/lect11-12.frm.pdf

https://www.microsoft.com/en-us/research/uploads/prod/2016/12/Time-Clocks-and-the-Ordering-of-Events-in-a-Distributed pdf

https://en.wikipedia.org/wiki/Berkeley_algorithm

http://pmg.csail.mit.edu/papers/osdi99.pdf

https://en.wikipedia.org/wiki/Conflict-free_replicated_data_type