ECE 493 Final Report

Arun Woosaree

April 5, 2021

1 Problem domain

Multiplayer games or something

2 Existing Solutions

A quick google search reveals that there are a plethora of online projects which already exist that allows one to play a multiplayer game online, with balls and paddles.

3 Our Solution

We chose a client server model over a peer-to-peer model

We use WebSockets for quick, bi-directional communication between the game server and clients

We decided to use Typescript as the main language for writing our frontend and backend code in. This allowed us to define common classes and interfaces that were shared between the frontend and backend code. This helped to standardize the code base, since the backend and frontend are use the same interfaces, we avoided problems where the data passed between the client and server could be mismatched.

Furthermore, the Typescript language can be compiled to Javascript, which runs in modern browsers, so a user of our application does not have to install any new programs on their computer. Typescript gives us the advantages of having a type system, so some errors like referencing a null value can

be avoided by statically analyzing the code before it is run. The type system also allowed us to standardize the classes and interfaces that the frontend and backend depended on.

For our frontend client library, we chose to use svelte as a framework for the website. There is no particular reason we chose Svelte over some other javascript framework like React, other than it is a newer framework and seemed interesting to use. We did find that using Svelte was kind of nice, since a lot of the code is in the same style as vanilla HTML and CSS, whereas with something like react has a little bit more of a learning curve with its JSX syntax and subtle differences. With svelte, we also converted a vanilla HTML and CSS prototype with minimal effort, whereas it would have taken longer to do with React, and we would have likely needed to rewrite more of our initial prototype.

For our backend, we chose to use Deno. We could have used Node.JS instead, but we chose deno because it seemed cooler. Plus, it has Typescript support out of the box, and claims to be more secure. Honestly, our lives would have been easier if we went with Express, because Deno is not widely adopted yet, and the community is not as large, however, I like where Deno is going, and it was fun to work with. I'd probably still use it in a future project.

We added powerups to add something different to the base pong-like experience, to help vary things. We think this helps us distinguish ourselves from the competition

4 Potential Impact on Society and the Environment

To be honest, I don't see our project significantly changing someone else's life for better or worse. It's a simple, fun game that friends could get together and play for a bit, but also it is a simple game with not much depth

Hosting our project online as a website accessible at polypong.ca undoubtedly has an environmental impact. Our frontend is hosted using Cloudflare Pages, a service offered for free by Cloudflare. Our domain is also registered with Cloudflare, and we are using their DNS services. Because this is an on-demand service, this means that the servers used to run our frontend code does not always need to be active, if users are not using the

website. These resources can be used by other users of the cloudflare network when our demand is low. Furthermore, Cloudflare appears to be a company which is conscious about their impact on the environment. For example, in 2019, they purchased Renewable Energy Certificates to match their electricity use for all of their data centres and offices around the world. https://blog.cloudflare.com/the-climate-and-cloudflare/

Our backend code is hosted on Cybera, a local nonprofit organization in Alberta. Unfortunately, due to how we desinged the project, the backend server must run constantly to be ready for a client to connect to it. Also, this code does not automatically scale back when the demand is low. Fortunately, it is not running on dedicated resources, as the CPU and memory resources are shared, so other users of the service can use the resources when we are not.

In my limited searching, I did not find any information about efforts Cybera is making to lessen their impact on the environment, however, we did find that Cybera is using data science to support green tech solutions https://www.cybera.ca/cybera-uses-data-science-to-support-green-tech-solutions/

5 My Role

For this project, I found myself acting as a senior developer of sorts. In the sense that I got to make decisions about which technologies we were using, and I found my teammates asking for advice on best practices and such, because I have previous experience with Javascript frameworks like React and Node.JS. I got to make the big design decisions, and get the ball rolling, so to speak, Things like creating initial boilerplate code that we could all build on later. I worked mainly on the backend and database side of things. I also touched a fair bit of the frontend code, mainly hooking up core functionality and making sure that the frontend can communicate properly with the server. I also worked entirely on the authentication system, database functionality, and getting our project deployed so that we can play the game on polypong.ca. The deployment involved writing Dockerfiles to make sure our program can be easily deployed on most platforms. The authentication system invoved

Michael mainly focused on the frontend, UI design, making my initial "functional" prototypes look pretty. He came up with the initial UI prototype for the frontend, a large portion of which we have tweaked and kept

in the final product. He also worked on getting the game to render on the HTML canvas, and did geometry work, collision detection, input handling, and designing the game loop. I pair programmed with Michael a lot over the course of the project, getting things between the frontend and backend in sync, and making sure they communicate with each other. Together, we got a basic game working, one without synchronization. "Basic game" meaning that we got multiple players to connect to the server, get their paddles to move on each others' screens, and a ball moving. However, at that point, there was no synchronization, and although we could see the ball moving in the same direction on all screens, they were in different positions because of the lack of synchronization

Josh mainly worked on the synchronization algorithm. We also pair programmed for a bit, like when we got the paddles to move simultaneously. He also implemented the remaining power ups

6 History of Changes

```
* c368319 bind:this for canvas
* c35b6f8 Code cleanup, adding comments
* cb34633 Update getPaddleY() for up to 7 players (need to finish 8-12 later)
* 55f9aa7 Add attribution
* 708e3ee Path Trace powerup done, better initial ball speed
* 9849a65 Distracting background working, stops when player who applied it is elim
* d8bed85 this was confusing the compiler I guess
*   42495dc Merge branch 'master' of https://github.com/PolyPong/PolyPong
|\
| * 97409a6 one space
| * 1866e33 cleanup
| * d831fe9 formatting
* | e3b97e1 2 Player wall collision detection working
|/
*   2635ddd Merge branch 'master' of https://github.com/PolyPong/PolyPong
|\
| * e5a8ccb signup works now
| * 7c088c8 oops
| * 6e5ccce oops
| * ac7d5c2 signup theoretically works
```

```
| * 475f2fe missing }
| * a0a7a2c redirect user if logged in but account does not exist
| * 81cc3fd handle loginwithredirect callback
* | 6074ad7 5 Powerups implemented (bigger, smaller and 3 invisibles)! Need testin
|/
*   69fe1c4 Merge branch 'master' of https://github.com/PolyPong/PolyPong
|\
| * 2b9e162 copy, share link works now
| * 344cc95 oops
| * a94864f create get username endpoint
* | 53f0a36 Added list of powerups on each client
|/
* 70804d4 Update images, Lobby now waits for everybody and has power up selection
* 081f246 global and local leaderboard now using rest methods, just need to get us
* e8e23a8 setskin works from the frontend now (in the sense that it saves the new
* 1b40a09 oops
* ad0b811 we're using rest to handle this stateless data instead of passing it ove
* c11e716 set skin rest endpoint
* 72a2fb8 remove unused code in server.ts
* 7306fc0 local leaderboard rest endpoint
* 6009ccc global leaderboard rest endpoint
* ade41a6 getavailableskins rest endpoint
* 759dd24 use auth0client.isauthenticated
*   8f9836d Merge branch 'master' of https://github.com/ECE493Capstone/project
|\
| * bb8bda8 login with redirect also, getTokenSilently() returns a proper token no
| * 9463ba2 implement getxp endpoint using traditional rest for simplicity
* | 5e54f66 smoother paddle movement
|/
| * 4821c69 Work in progress to synchronize emails and usernames across client and
|/
* a956424 Game background colour is now the same as rest of background
* 5628aca Fix background color
*   ab60467 Merge branch 'master' of https://github.com/PolyPong/PolyPong
|\
| * 5c0a06c leaderboard functionality
| * 4c4bf61 leaderboard also theoretically works now
| * 0e8cf83 add logic for getting skins and setting it (with authentication!) (unt
```

```
| * c65fecf fix some linting and allow port to be specified as an environment vari
| * 26fff32 add leaderboard functions for dbhelper, and tests for all of dbhelper
| * 6c3827e server can run locally again
| * d6916f4 let's see if this is faster
| * 14b0ff6 oops
| * 3812267 ooh let's try this
| * 3177e7a pls
| * bd9b095 ohhh it might be a CORS thing
* | 0a6155f When we finish one game, a new game is started!
|/
* a7d7ffa Improve the animation
* 9c827df Update getPaddleY() to work better, add Game Over text animation
* 1ec3a21 try this port on cloudflare
* 09dd68a let's see if this works
* 32ff7d0 try binding to 0.0.0.0
* dec0d48 oops I spoke too soon
* 272adda aaand theoretically we should be live
* ec610f9 it works
* e8d3dc4 rename it so it works on cybera (because I'm already running another ins
* 9ccb72f this should work
* 54788c8 create dockerfile
*   a852a09 Merge branch 'master' of https://github.com/PolyPong/PolyPong
|\
| * 2917325 oops
| * acd4dc4 logic for levelling up and setting skins and getting available skins
| * fa4b545 add some pretty colours
* | cfae28a Gameplay working and smooth, graphicaly glitches ironed out, Client ca
|/
*   a25d53e Merge branch 'master' of https://github.com/PolyPong/PolyPong
|\
| * 260a729 undo rotation
* | 4bf30cc Collision detection working across multiple clients, graphical glitche
|/
*   6940d71 Merge branch 'master' of https://github.com/PolyPong/PolyPong
|\
| * 42f197a don't send websocket buffer
* | 74d6856 Working on getting ball consistent
|/
```

```
* 1d95da8 Collision detection is poor but working
* 00757ec send unrotated dy dx for ball
* 8f4370a skip sending websocket buffer in the update lol
* 035bb48 red paddle should move now on other clients
* 8e626d9 Collision detection now works client-side! Ball is also drawn correctly
* 5cf8b12 idk :/
* ac0a03b oops
* 0e12240 ball should be in sync now
* b8ae6e0 ball moves in sync until there is a collision also collisions are broker
* 3814bc4 refactor
* 61f7b36 refactored game object
* dfee8ff Include interface from PPC (removes error message)
* 3e67b26 Login now checks db, we let users set their username on first login
* 0dc17aa we can see other players' paddles moving now!!!
* a3da283 Log In on Home page now prints user email to console if authenticated
* 07f5092 Fix Auth0 bugs
* 9c3dd00 Update db.ts to import from PP-Common
* aa6ce39 auth0Client now in store.ts
* 4d4227e Change default port to 3000
* ec001d6 attempt to start game
* a395ced moved client side lobby websocket stuff to global scope in store.ts
* ccdf142 why did I change these types lol
* 370549f delete old stuff
* 895b5d6 format tsconfig
* 7a7f748 idk
* 05fdd37 Revert "install in preinstall?" mistakes were made...
* fbd103e install in preinstall?
* b22b713 frontend can build now
* 603ebdd if I have to do another UML diagram I'll wanna fucking kms
* a0ca534 render diagrams
* 65c19d3 Create fr26.puml
* 1b90018 Create fr25.puml
* 0ce9d9f Update fr16.puml
* 82f5fab Create fr24.puml
* 3dcd30e Create fr23.puml
* a16f102 Create fr22.puml
* 74b20bb Create fr21.puml
* 95b4032 Create fr20.puml
```

```
* b47b915 Create fr19.puml
* 38a674c Create fr18.puml
* fed2552 Create fr17.puml
* 3402647 Create fr16.puml
* 747b23f Create fr15.puml
* 6b0ac6b Create fr14.puml
* 1b2907e Create fr13.puml
* 61dd277 Create fr12.puml
* aa1e2b8 we have a basic database
* 254af7d Update fr27.puml
* 2649182 Update fr28.puml
* e18eca7 changes
* d339d94 delete fr12 for now
* 86248f7 oops forgot to commit this one
* 5ba66dd add docker-compose
* 1f4c0b0 hopefully last time updating fr1sequence
* 758aaa0 Update fr1.puml
* f780ed8 Update fr6.puml
* 53a72bb final fr28 hopefully
* db93bfa final fr27 hopefully
* 56449d8 final fr28 hopefully
* 92291c5 final fr10 hopefully
* 69de270 final fr9 hopefully
* 36c7fa2 final fr8 hopefully
* 96d460d final fr7 hopefully
* b056895 final fr6 hopefully
* 8caa1aa final fr5 hopefully
* ab61aa3 final fr4 hopefully
* 818bbb0 final fr3 hopefully
* 7e35a5f final fr2 hopefully
* 1d17042 script to generate power up sequence diagrams when we figure out how the
* 91b3235 tweaks
* 0875ad3 remove need for jwt to view leaderboard
* c6a266b play game does not involve lobby
* e0930b3 check for 3+ users fr3
* cbeddbd fr12sequence
* c21c408 fr28sequence
* 7fb4f70 fr27sequence
```

```
* 2879334 fr10sequence
* 7360541 fr9sequence
* 4e54651 fr8sequence
* 51c2cc5 fr7sequence
* 72d270e fr6sequence
* 49b9cd4 fr5sequence
* 9d5c1ef fr4sequence
* 7aad86c fr3sequence
* fe90914 fr3sequence
* dd09d87 update fr2sequence
* 33bcadc update fr1sequence
* 8ac287e fr2sequence
* ff37e84 fr1sequence
* c6c7837 the websocket is now accessible from anywhere in the application
* 04c6bb7 use the types I defined for transmitting messages
* de4b815 add joinsuccesspayload to payload.ts
* c9a8575 fixed the enter lobby id bug. now uses the textbox input
* e250620 here's what the communication protocol might look like
* f9e9273 hey we can now create a lobby, and join it with a unique id!!!
* b180a2f add uuid
* 0d70162 package-lock go brrrrrrrrrrrrr
* 95a3d19 document how to install private package
* f2eb902 create polypong-common package
* c6d248a v0.0.1
* 0a64f51 move Game types to npm package
* 51b2d4e Player number done; paddles rotate accordingly
* 0d60aee Ball now scales based on client's window size
* c5d9a42 Added a ball and basic collision detection
* 7cc2cc0 Cleaned up code, paddles have colors!
* ffad707 Paddles are now bounded!
* 0dfdaf9 Bottom paddle moves! No boundary checking yet
* dccca91 Initial paddles are drawn
* 0fb8f2c Adding classes but not yet working
* d81ecac Shapes are drawn again
* 6f9fadb work breakdown structure
* 345723b idk what I changed but it works beautifully now. If the user is already
* e07fb58 getting somewhere
* 34625da hm
```

```
* 61928d8 add auth0 client library again
* d158d9c oh boy lots of changes
* 5c62216 add auth0 client library
* ccea9cc frontend mockup but now in Svelte still kind of ugly because we're not r
* 739fbe9 actually let's try svelte@next
* f51de3c add lockfile to gitignore for now
* 3573a86 use typescript
* 0acc958 init svelte
* 4369d8e move frontend mockup
* bd0191d copy over stuff from main branch
* 7b293ed signin works
* d38e4b4 signin works
* 026077b Create README.md
* 61a93f6 Delete .DS_Store
* 279df4a Delete .DS_Store
* c3a08ef Delete .DS_Store
* 58569af Delete .DS_Store
* 1d00abd Delete .DS_Store
* 1aab6ff Delete .DS_Store
* fe524da Add gitignore
* 72beaa3 Initial mockups
```

7 Summary of Defects

oh boy....