



# Software Development Portfolio

---

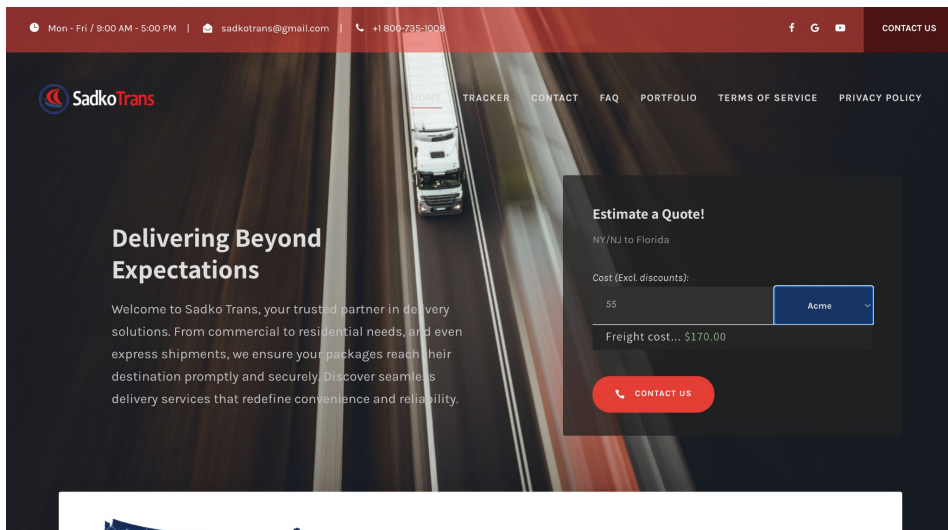
---

---

---

**Aleksey Panas**



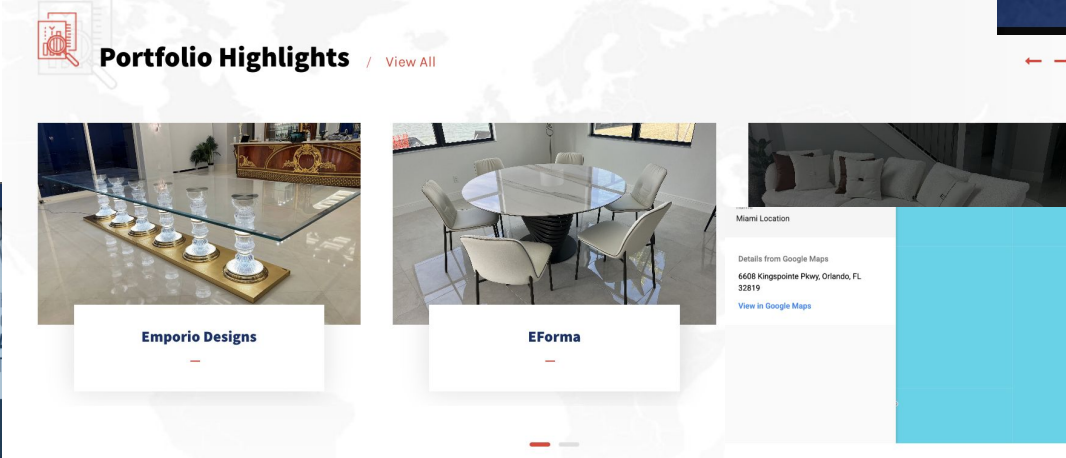
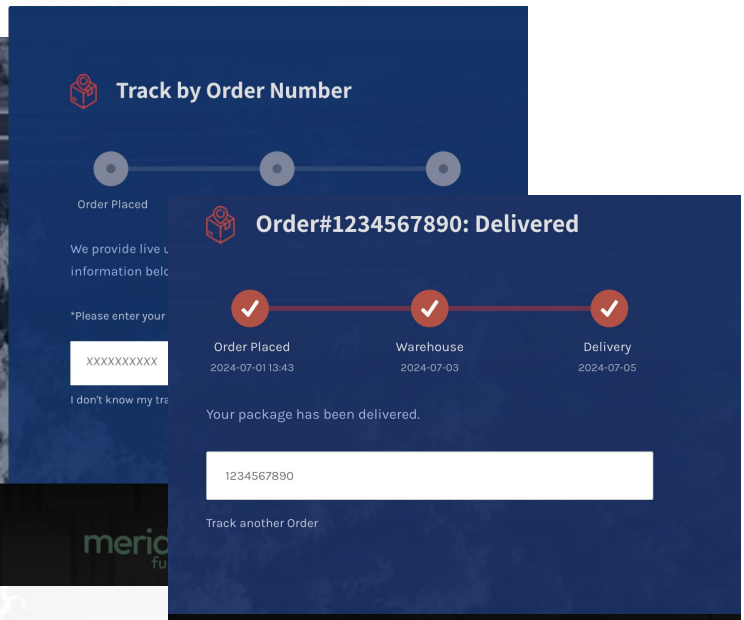


About Four month **summer full time job** at the SadkoTrans furniture delivery logistics company as **software engineer**, warehouse assistant, and box truck driver

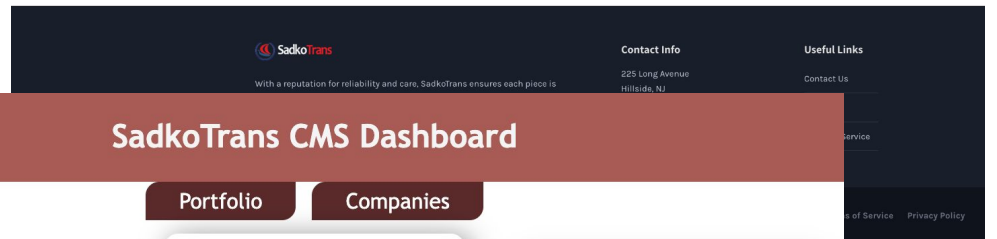
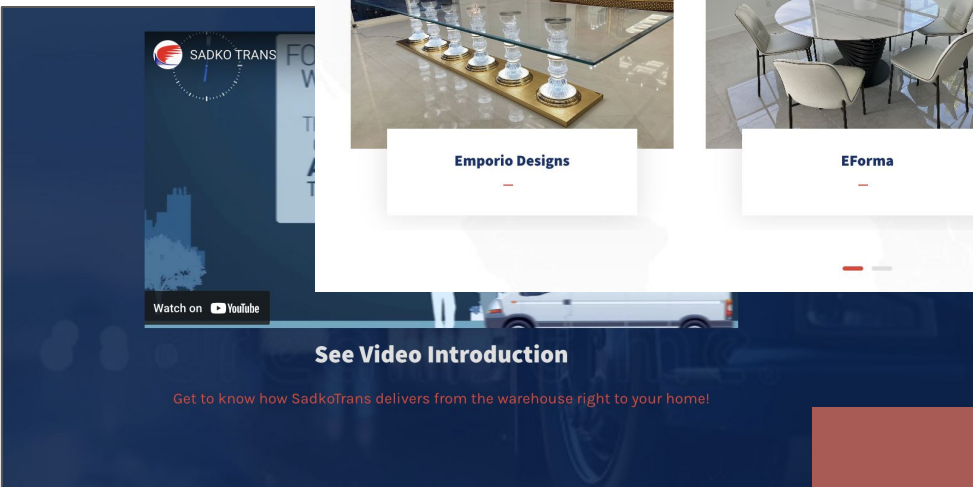
## Technologies



- ReactJS
- NanoJSX
- JSX/TSX
- Bun
- JS/HTML/CSS
- ExpressJS
- Eleventy
- Nunjucks
- NodeJS
- Websockets
- NginX
- AWS
- Web File Upload
- Shopify

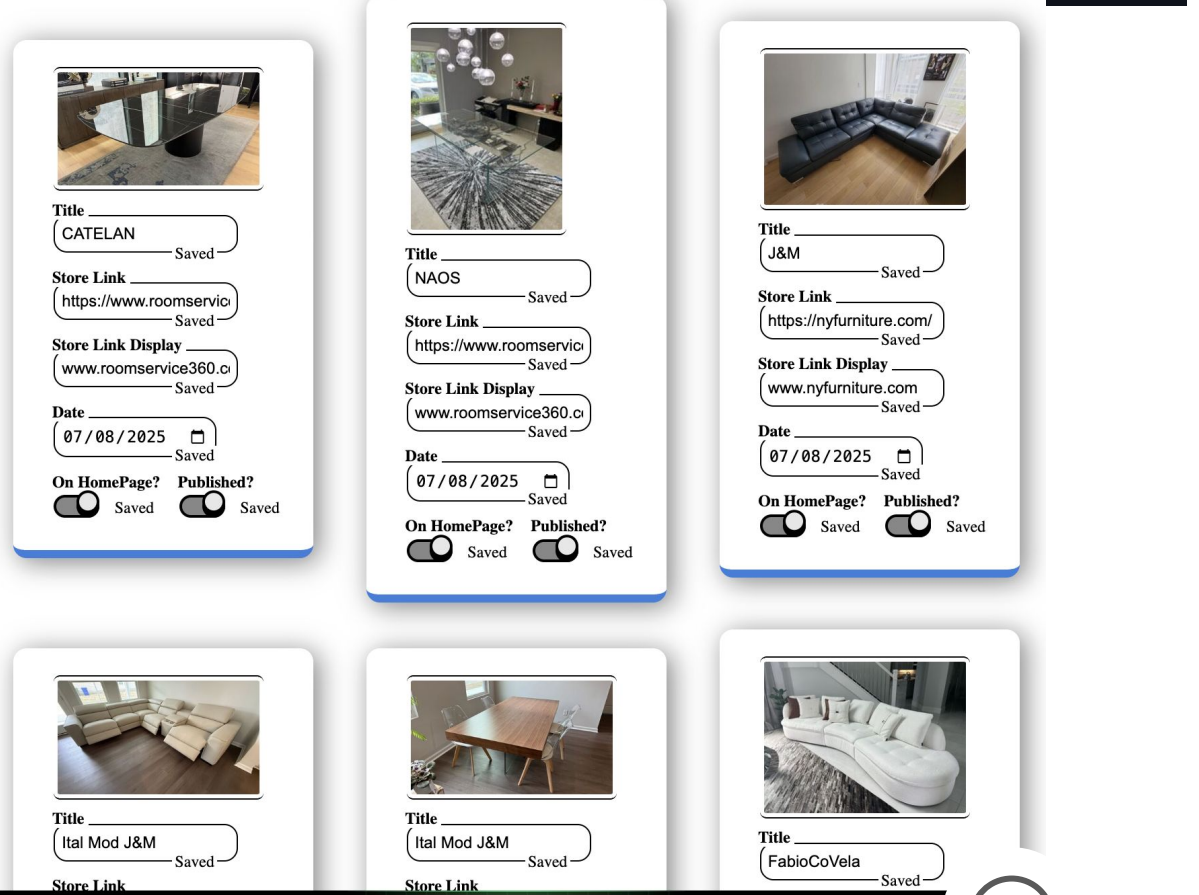


**UPON REQUEST**



## Contribution

- Built [sadjotrans.us](https://sadjotrans.us), a client-facing site using **Eleventy** for modularity and easy extensibility.
- Developed an **admin CMS** in **NanoJSX + ExpressJS** with a portfolio **file upload** pipeline optimized for delivery drivers, using **caching** and lightweight **NanoJSX** rendering for **lightning-fast** functionality & loading even **under weak internet conditions**.
- Assisted with **peak-hour logistics management** and **truck driving**, gaining hands-on insights that informed **software design** to improve company efficiency.



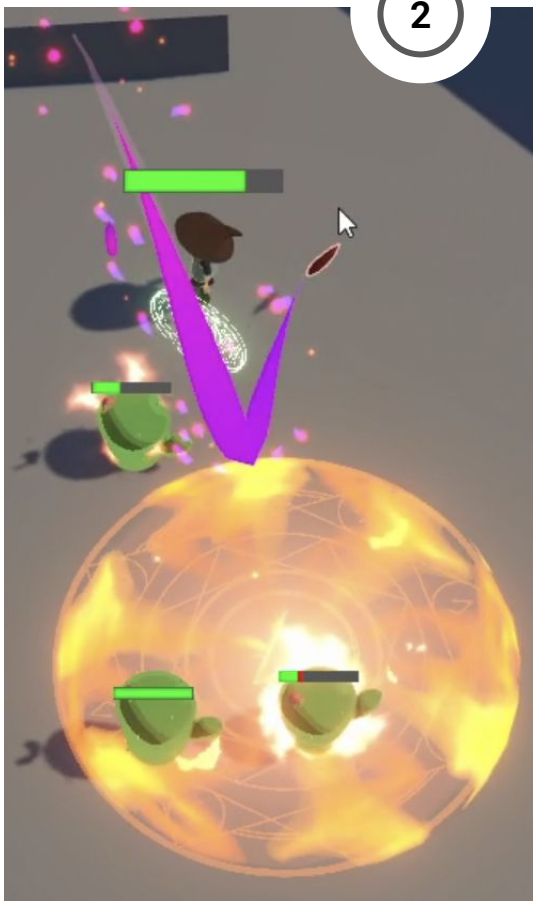
**BUILD SITE** \*

Backend Admin Content Management System (CMS)



About

Magenggaard is a 3D magic fighting PvP/PvE game where spells are cast by drawing unique symbols on the screen



2



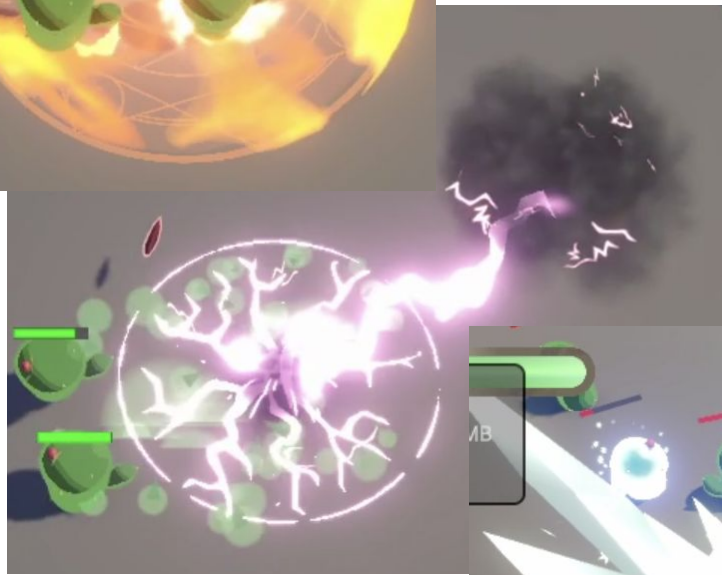
Technologies

- Unity
- C#
- Netcode
- Blender
- Python
- Photoshop
- CSG (Mesh booleans)
- Gesture Recognition
- Git, CI/CD, Agile, Clickup



1

[badlek.itch.io/magenggaard](https://badlek.itch.io/magenggaard)

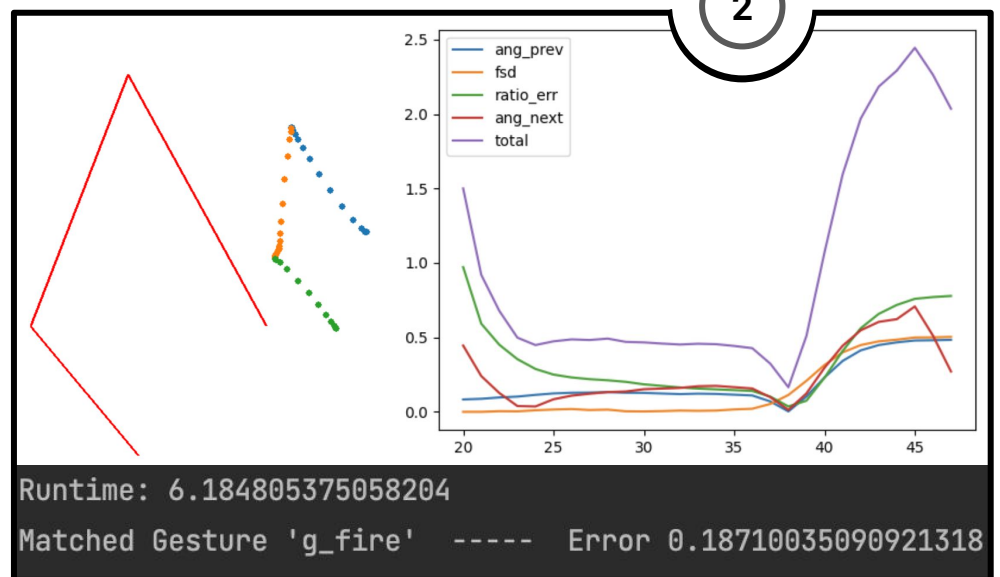


2

Contribution

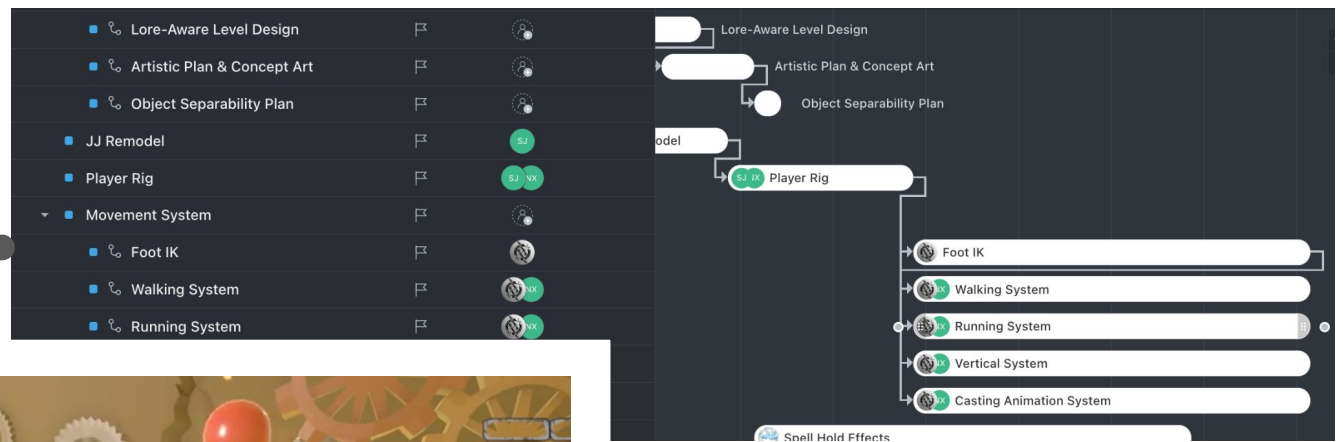


- Conceptualized the core game mechanic of gesture-based spellcasting
- 250+ commits including...
  - [1] A custom in-house mesh union algorithm for freezing water (C#)
  - [2] A custom in-house gesture recognition algorithm using function minimization techniques (C#)
  - 3D spellbook UI with pages that turn by mouse dragging using inverse kinematics (C#)
  - System to pick up arbitrary items off the ground (C#)
  - Netcode for multiplayer "Demifox" bossfight where one player controls the boss! (C#)
  - 3D modelling of entire second "Ruins" level in Blender
- Managed our team of 10 artists, programmers, and sound designers, implementing Agile through ClickUp
- Handled team admin tasks, coordinating share distribution and setting development timelines to prepare for upcoming Kickstarter launch and Steam release



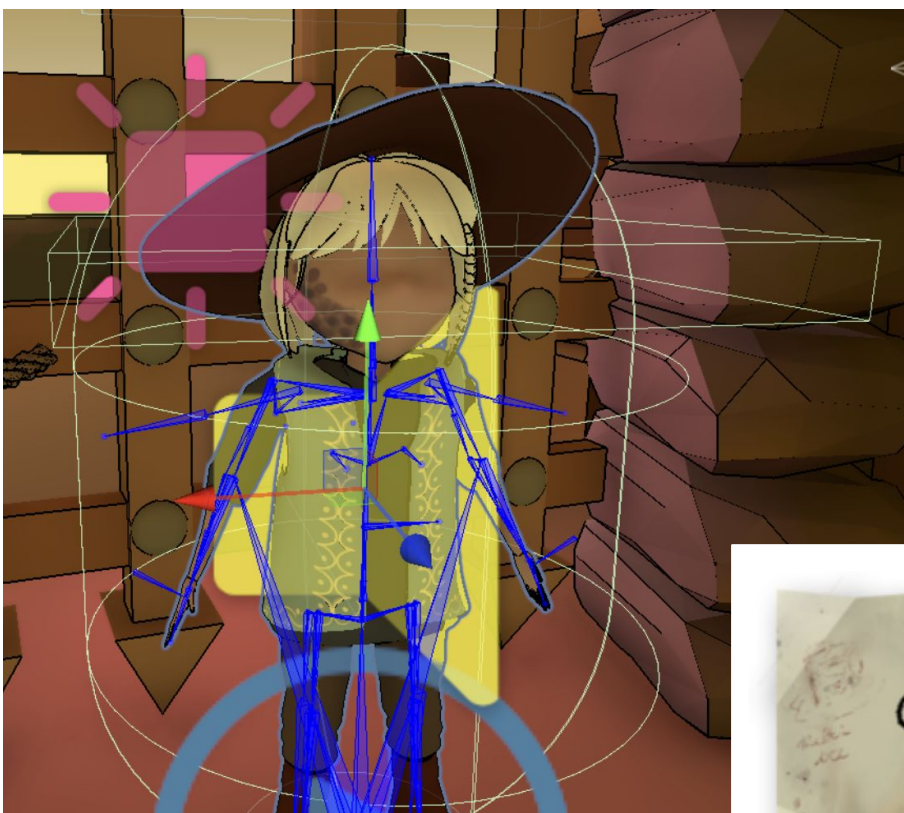


Agile project management using Clickup



Designed and modelled puzzle in second "Ruins" level

Modelled "boiler room" in second "Ruins" level



Implemented inverse kinematics and animations in code on our rigged main character

**What's Ahead**

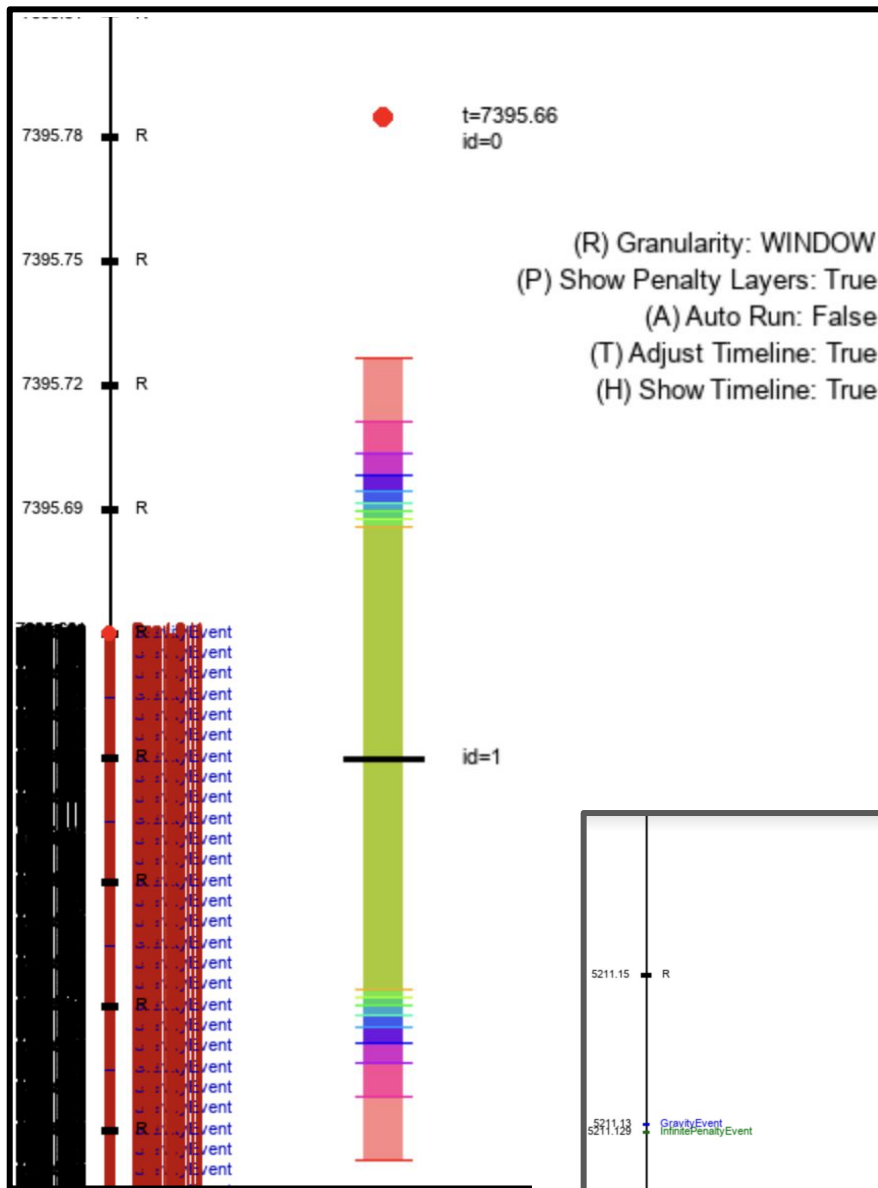
While Magenggaard's current PvE experience was great for a proof of concept, we found our PvP multiplayer bossfight very fun!

We are currently working to cut scope and turn Magenggaard into a well-polished competitive PvP experience, aiming for a Kickstarter launch and Steam Release in the first half of 2025



Drew the UI page assets that can be picked up throughout the game to learn new spells





## About

Working with supervision on replicating and innovating on an algorithm in the subfield of contact mechanics in physics simulations.

The goal is to implement this algorithm on the GPU using CUDA and leverage its high parallelizable nature

## Contribution

Read, understood, and implemented Speculative Parallel Asynchronous Contact Mechanics ([SPACM](#)) in **python**

Wrote an extensive visualization and logging system

leveraging **tensorboard**, **numpy**, **matplotlib**, and **pygame**

**Debugged** complex **energy behavior** issues in the simulation



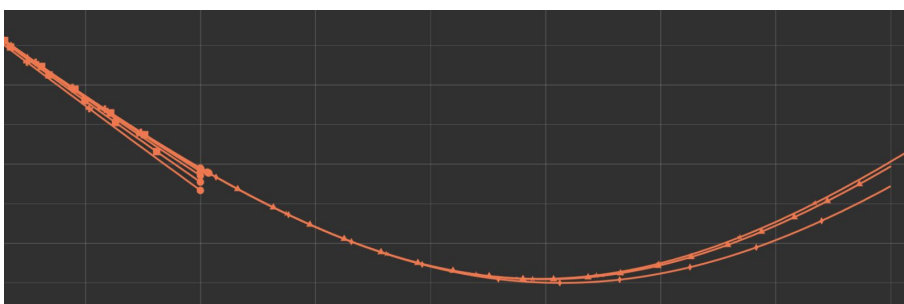
## Technologies

- Speculative Parallel Asynchronous Contact Mechanics
- Tensorboard
- Python
- Pygame
- Numpy
- Matplotlib

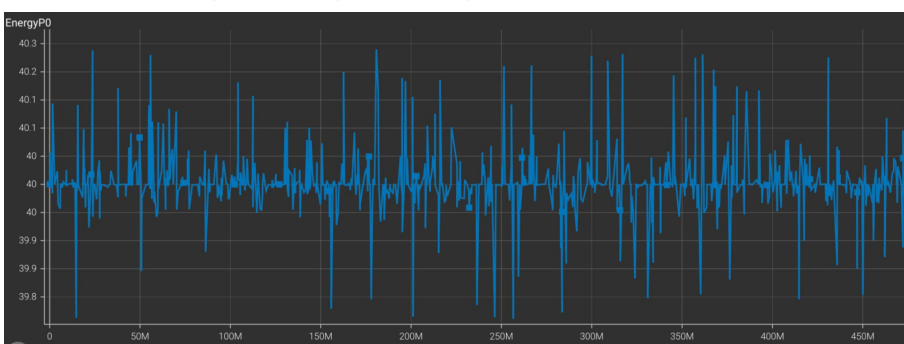


## What's Ahead

Currently working on implementing SPACM on the GPU using **CUDA** through **warp**

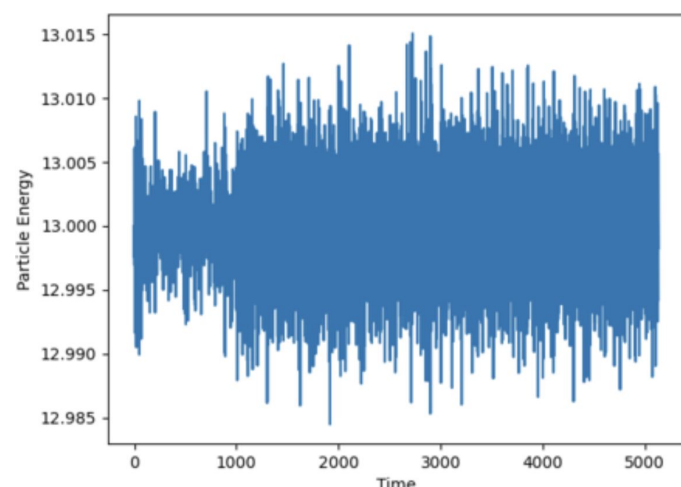


Tensorboard particle position plot with rollback visualization

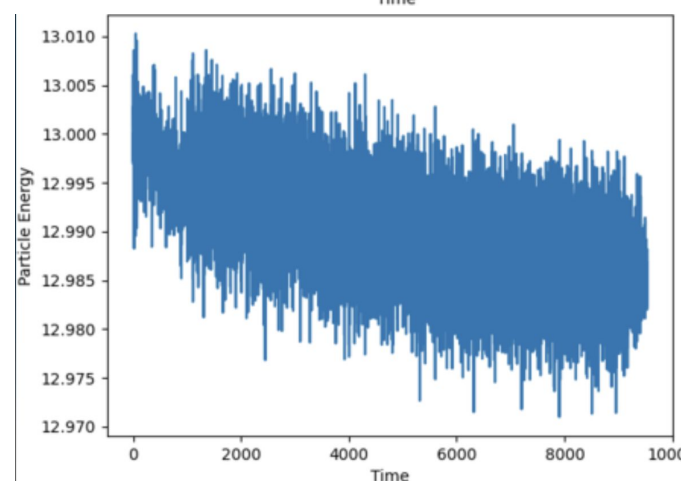


Tensorboard particle energy plot

Pygame visualizer for 1D sim: colorful layers are penalty regions which apply collision-preventing repelling forces via repeating events as shown in the timeline view on the left



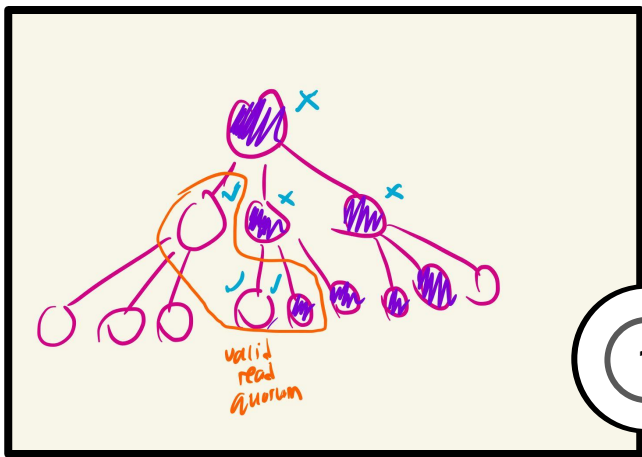
Good energy behavior (after fix)



Bad energy behavior (before fix)

Simulation logs in binary used to debug bad energy behavior

```
ROLLBACK
t=0b111011101000111101011100001010 ev=0b0 x=0b10000000
t=0b111100001000111101011100001010 ev=0b0 x=0b10000000
t=0b111100011101011100001010001111 ev=0b0 x=0b10000000
t=0b111100101000111101011100001010 ev=0b0 x=0b10000000
t=0b111100110011001100110011001101 ev=0b0 x=0b10000000
t=0b111100111101011100001010001111 ev=-0b10000001000000
ROLLBACK
t=0b111011101000111101011100001010 ev=0b0 x=0b10000000
t=0b111100001000111101011100001010 ev=0b0 x=0b10000000
```



## About

Primary work as Research Assistant involves developing novel algorithms in the strong consistency field of distributed networks. Latest work in particular involves optimization of local reads and attempted generalization from Paxos to other classes of algorithms such as Leaderless.

Contact my supervising Professor Eyal de Lara at [delara@cs.toronto.edu](mailto:delara@cs.toronto.edu) if a reference letter is needed

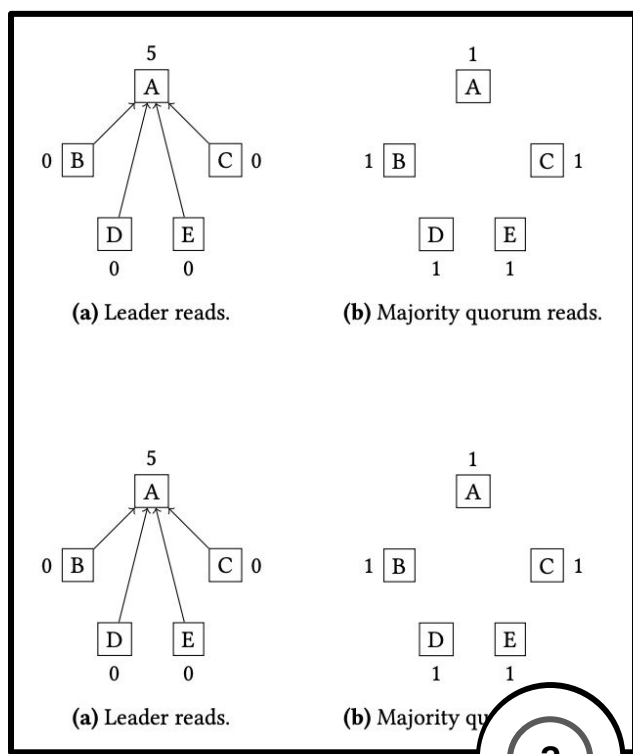
## Contribution

With mentorship from a PhD student...

- Contributed over **50%** of the content to a [completed research paper](#).
- Worked on brainstorming and discovering over 10 minor and major innovations in the strong consistency field, particularly for local reads
- Implemented the algorithms to test them experimentally
- Formally analyzed the innovations to ensure their correctness and efficiency compared to other algorithms
- Presented my discoveries to colleagues and Professors in the lab
- Read over **50** papers in the area

```

1 procedure read(o):
2   cntnr := cntnr + 1 and index := 0
3   R := closest_read_quorum()
4   if R is only the current process then
5     index := MaxP
6   else
7     /* cntnr on lines 7 and 9 is the same as line 2 */
8     send <READ, cntnr> to all processes in R
9     TI := ∅
10    upon receiving <R_ACK, cntnr, T', MaxP'>:
11      index := max(index, MaxP')
12      foreach process p in the system do
13        if (p, *) ∈ T' then TI := TI ∪ {p}
14    wait for |TI| ≥ ⌈(n+1)/2⌉
15    /* Executes the read request o against the local
16       replica once all writes up to and including index
17       have been applied and return the result */
18    return Read(index, o).
    
```

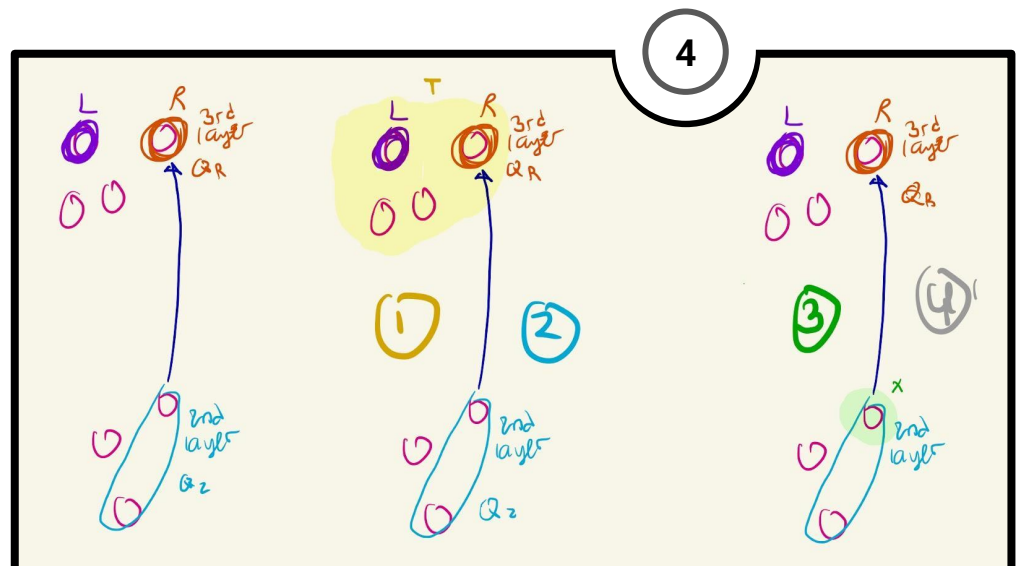


## Technologies

- Strong Consistency Algorithms (MultiPaxos , Raft , Leaderless )
- Local Reads Algorithms (Paxos Quorum Leases , Paxos Quorum Reads )
- Clock Sync Algorithms , Lease Algorithms
- Various properties/metrics of distributed systems: Fault Tolerance , Read/Write latency , Throughput , Recovery Time , etc
- Reconfiguration Algorithms: Matchmaker Paxos , Vertical Paxos , SMART
- Linearizability
- Quorum Intersection Algorithms: Crumbling Walls , Tree Quorums
- Many other algorithms in the area

## Captions

1. Tree Quorums sketch for self-verification
2. Small lemma from algorithm proofs
3. Lease diagram from local reads work
4. Hierarchy scenario comparisons from local reads work





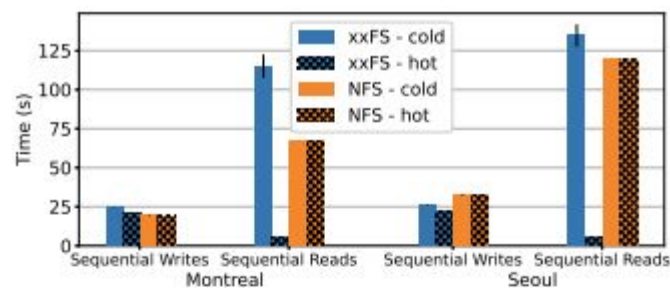


Fig. 7: FIO benchmark. Average total time to complete a FIO benchmark of sequential writes or reads at both the Montreal and Seoul edge nodes. xxFS performs better than NFS during hot runs due to local edge caching whereas NFS flushes local cache on file closes.

4) *Summary:* The case studies demonstrate the advantages of xxFS in hierarchical edge deployments. They highlight the benefits of the hierarchical design, on-demand replication, localized data access, and remote notifications.

#### B. FIO Benchmark

We next compare the performance of NFS and xxFS using the FIO benchmark [2]. A FIO execution is performed by creating a new directory at the root node in California, which is pre-populated with five files totaling approximately 1GB in size. Next, a trace of sequential reads or writes is executed twice back-to-back at either the Montreal or Seoul edge node. We refer to the first and second executions of the trace as *cold* and *hot*, respectively. The traces are generated using synchronous, sequential IO, a Zipf distribution of accesses, and roughly 200,000 operations are performed, totaling 1GB of IO. Furthermore, each file remains open throughout the duration of the benchmark. We record the total time to execute the benchmark across 20 runs and plot the average of the last ten along with 99% confidence intervals in Figure 7.

The results show that xxFS performs almost identically to NFS in the cold and hot runs of the sequential writes trace when ran at the Montreal edge node. Furthermore, when the same trace is ran at the Seoul edge node xxFS performs better than NFS, indicating that xxFS handles better the increase in network latency. In the sequential reads case xxFS outperforms NFS during the hot execution. This can be attributed to the fact that all file data is already replicated at the local edge node from the previous cold run. Furthermore, NFS does not see such performance improvements from the cold to hot run due to the fact that, once a NFS file handle is closed, all cached data is flushed back to the server.

## About

xxFS is a **distributed file system** that optimizes for **edge** applications developed in the **University of Toronto Systems & Networks Group**. The system prototype is built in **Java** on top of **Cassandra** using a custom **FUSE** implementation

## Contribution

As **Research Assistant**, conducted an experiment using a standard file system benchmark, generated a plot, and wrote the corresponding section in the paper (*final result shown on the left*)

In the process, discovered several bugs in the system, **3** of which were **mission critical**, deducing their root cause, and **fixing 2** of them in code.

*e.g one of the bugs involved an obscure case of a file being shown in full from the `cat` command but not when indexed sequentially via `vim`. The cause was an indexing error when file seeking.*

## Technologies



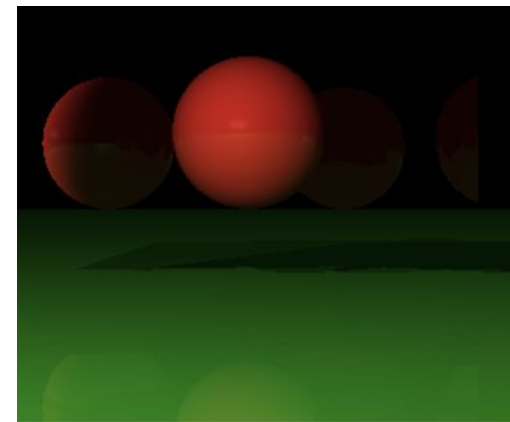
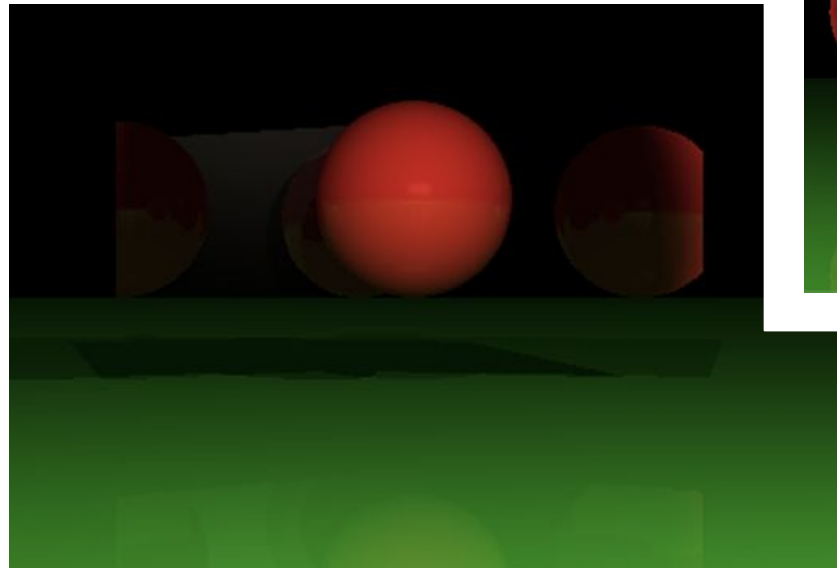
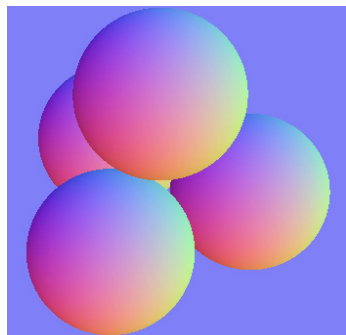
- Used **Python Fabric library** to automate experiment execution allowing us to perform over **100** repetitions
- **AWS** used to create an edge-like setup with over **30** machines
- **FIO Benchmark Tool** used to perform the standard benchmark
- **Traffic Control** on **Ubuntu Linux** used to manipulate **network tables** and simulate latencies for various experiment setups
- **Docker** used to deploy xxFS across the machines
- Deployed **NFS** across the machines to compare against xxFS





Contact my supervising Professor Eyal de Lara at [delara@cs.toronto.edu](mailto:delara@cs.toronto.edu) if a reference letter is needed

## About

The computer graphics course involved 8 major independent C++ projects. Here are some highlights.



- Technologies**
- C++
  - GLSL
  - Ray Tracing
  - Physics-based animation
  - Rigid Skinning
  - Linear Algebra
  - Calculus (Gradient descent)
- 
- 



1

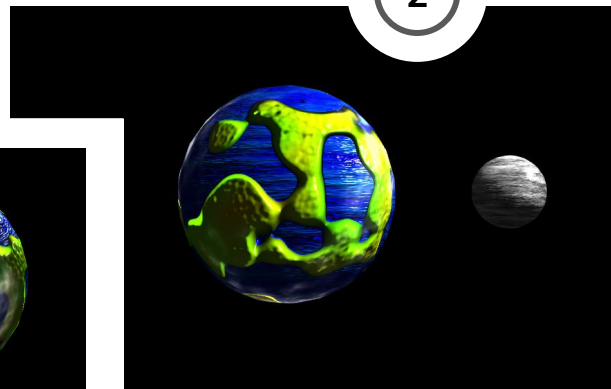
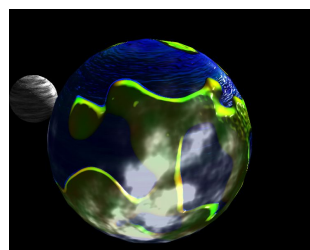
### 1: Ray Tracing

A ray tracer built in C++ capable of rendering scenes containing planes, spheres, and triangle meshes implementing shadows, reflections, and specular lighting

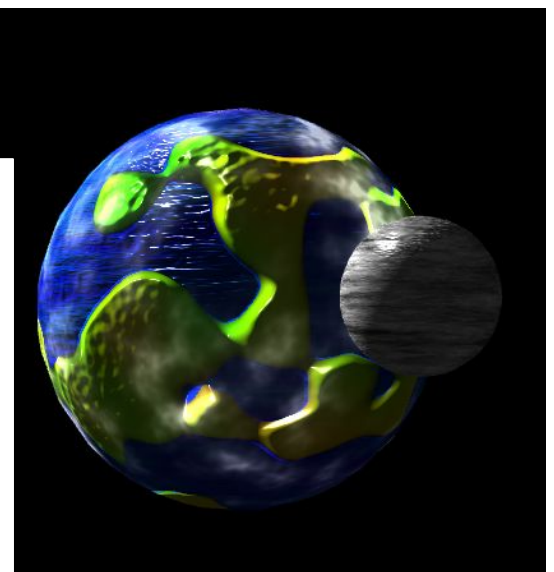
### 2: Shader Scene

An animated planet scene generated entirely with shader code.

Involves vertex shaders, tessellation, and fragment shaders



2



### 3: Surface Normals

Implementation of per-face normals, per-vertex normals, and per-corner normals for different shading effects

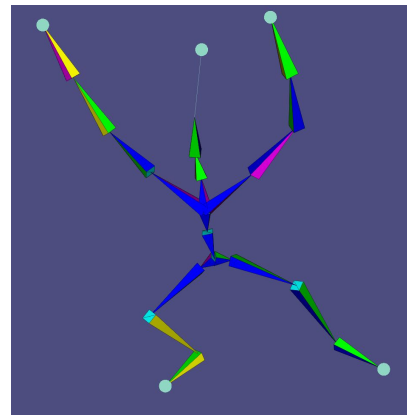
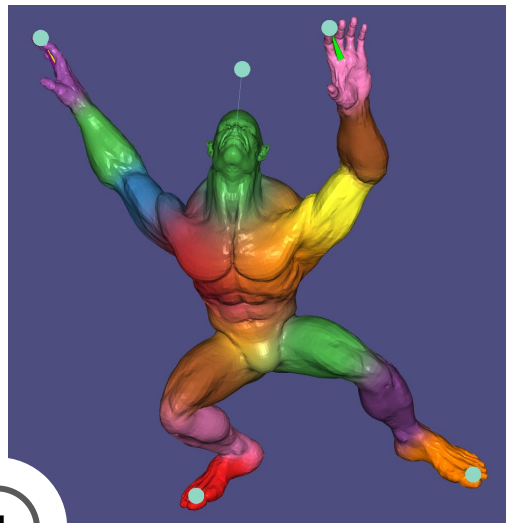


3





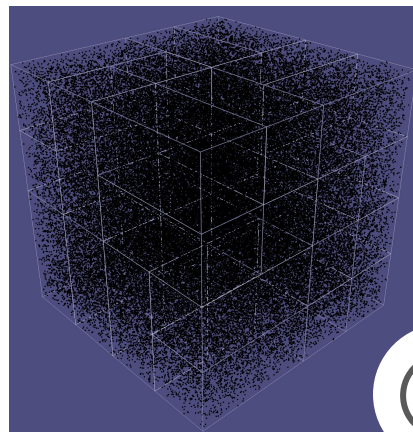
4



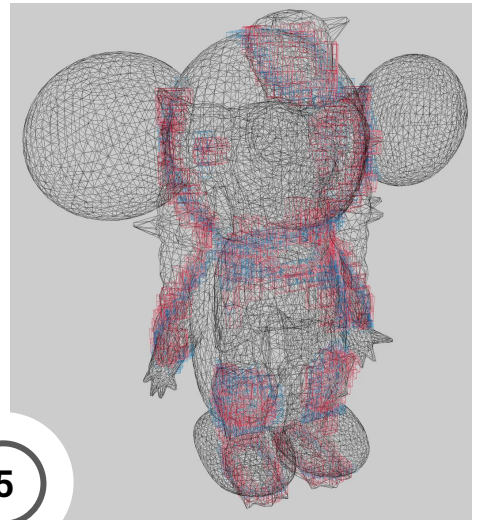
#### 4: Rigid Skinning

C++ implementation of the rigid skinning method which involves bone matrix transformations and mesh deformation to the bones.

The images show inverse kinematics achieved via gradient descent to deform the mesh via target endpoint positions.

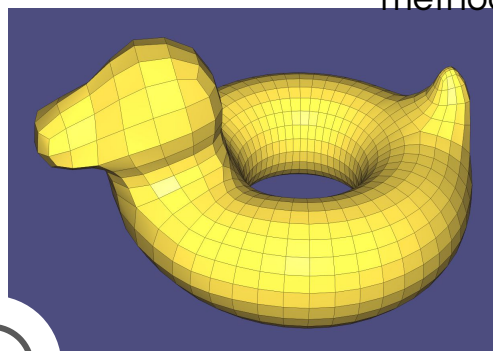
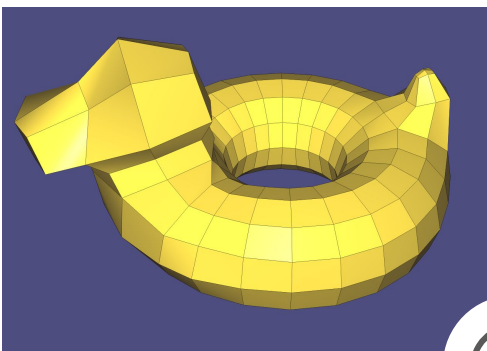


5

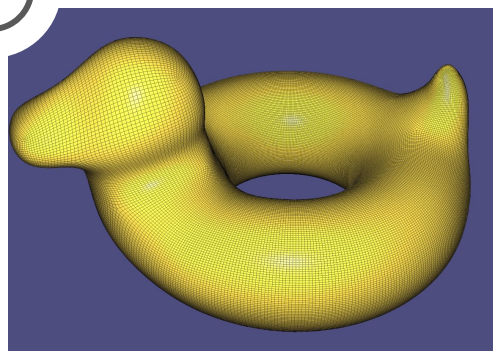
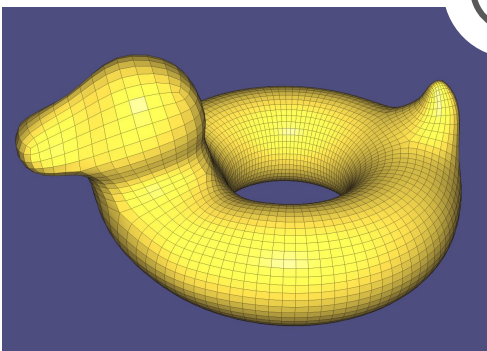


#### 5: Bounding Volume Hierarchies

Hierarchical structure to organize scene for optimizing intersection and distance queries to  $O(\log(n))$  time. Additionally implemented Devillier's triangle intersection method



6

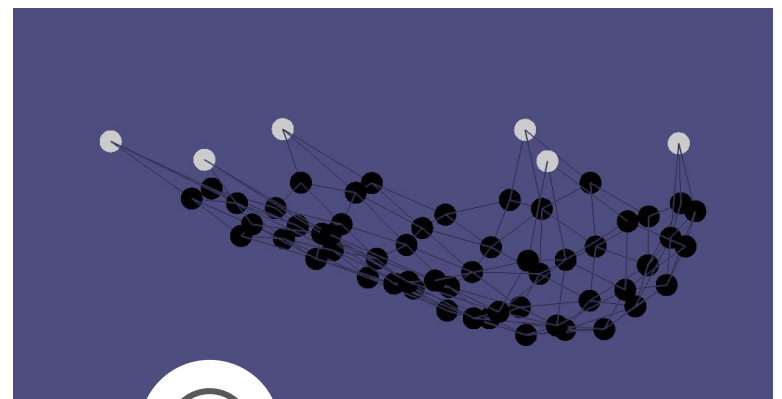


#### 6: Catmull Clark Subdivision

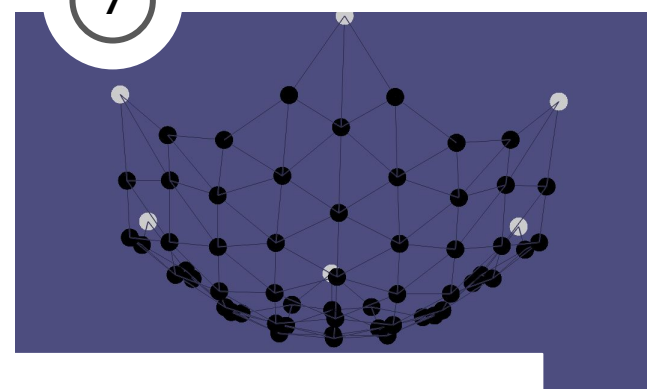
Implementation of the Catmull Clark subdivision scheme for increasing the number of polygons in a mesh

#### 7: Physics-based Animation

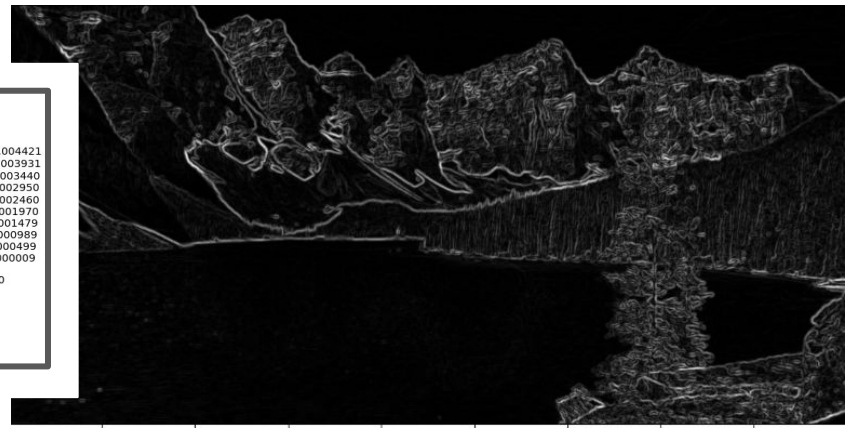
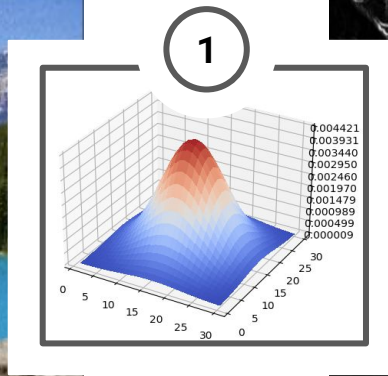
Implemented [Fast Simulation of Mass-Spring Systems](#) paper to simulate cloth-like objects



7







```
Sequential(
  (0): Conv2d(3, 16, kernel_size=[3, 3], stride=[1, 1], padding=[1, 1], bias=True)
  (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU(inplace=True)
  (3): Conv2d(16, 16, kernel_size=[3, 3], stride=[1, 1], padding=[1, 1], bias=True)
  (4): ReLU(inplace=True)
  (5): MaxPool2d(kernel_size=[2, 2], stride=[2, 2], padding=[0, 0], dilation=[1, 1], ceil_mode=False)
  (6): Conv2d(16, 8, kernel_size=[3, 3], stride=[1, 1], padding=[1, 1], bias=True)
  (7): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (8): ReLU(inplace=True)
  (9): Conv2d(8, 8, kernel_size=[3, 3], stride=[1, 1], padding=[1, 1], bias=True)
  (10): ReLU(inplace=True)
  (11): AdaptiveMaxPool2d(output_size=[1, 1], min_value=-1000, max_value=1000)
  (12): ReLU(inplace=True)
  (13): Dropout(p=0.5, inplace=True)
  (14): Flatten(start_dim=1, end_dim=-1)
  (15): Linear(in_features=64, out_features=1000, bias=True)
  (16): ReLU(inplace=True)
  (17): Dropout(p=0.5, inplace=True)
  (18): Linear(in_features=1000, out_features=1000, bias=True)
  (19): Softmax(dim=1)
)
```

## About

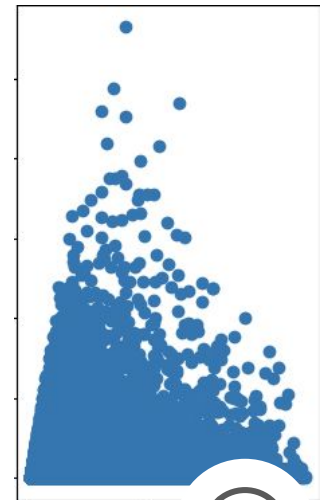
The computer vision course involved four major independent assignments. Here are some highlights.

### 1: Convolution & Edge Detection

Manually implemented convolution with numpy and used it to compute gradient magnitude of images for detecting edges

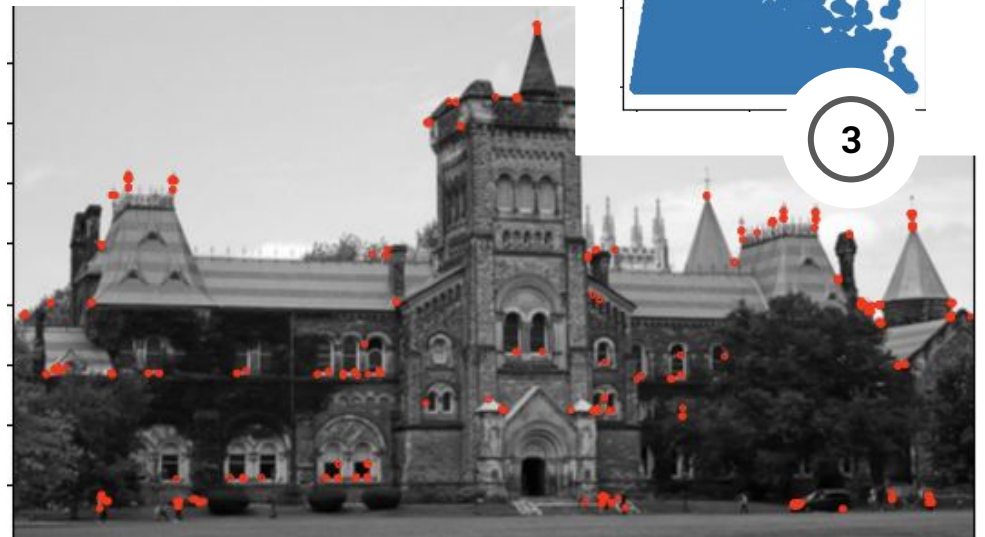
## Technologies

- PyTorch
- Conv Nets (DL)
- OpenCV / Numpy
- Computer Vision:
  - SIFT, RANSAC, Homographies, Stereo (depth recovery), etc



```
Epoch 8
100% |██████████| 7/7 [00:03<00:00, 2.07it/s]
[1.7910284996032715, 1.7399219274520874, 1.7826]
Train Accuracy: 0.41904761904761906
Test Accuracy: (0.3395061728395062, 55.0, 162)
-----
Epoch 9
100% |██████████| 7/7 [00:03<00:00, 1.97it/s]
[1.6891350746154785, 1.6749614477157593, 1.6843]
Train Accuracy: 0.4452380952380952
Test Accuracy: (0.36419753086419754, 59.0, 162)
-----
```

2



### 3: Histogram of Gradients

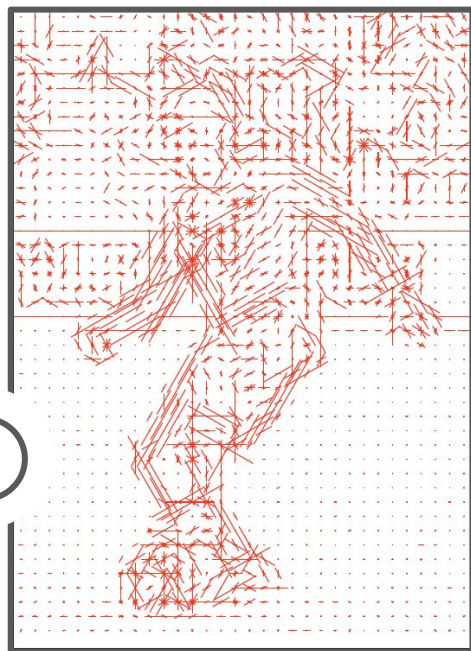
Implemented an algorithm to compute the histogram of gradients for an image: a process commonly used in old-school object detection

### 2: Deep Learning

Designed and trained a conv net in Pytorch to distinguish dog breeds. Also applied transfer learning on ResNet

### 3: Corner Detection

Detecting corners via the eigenvalues of the second moment matrix (scatterplot for threshold-setting)

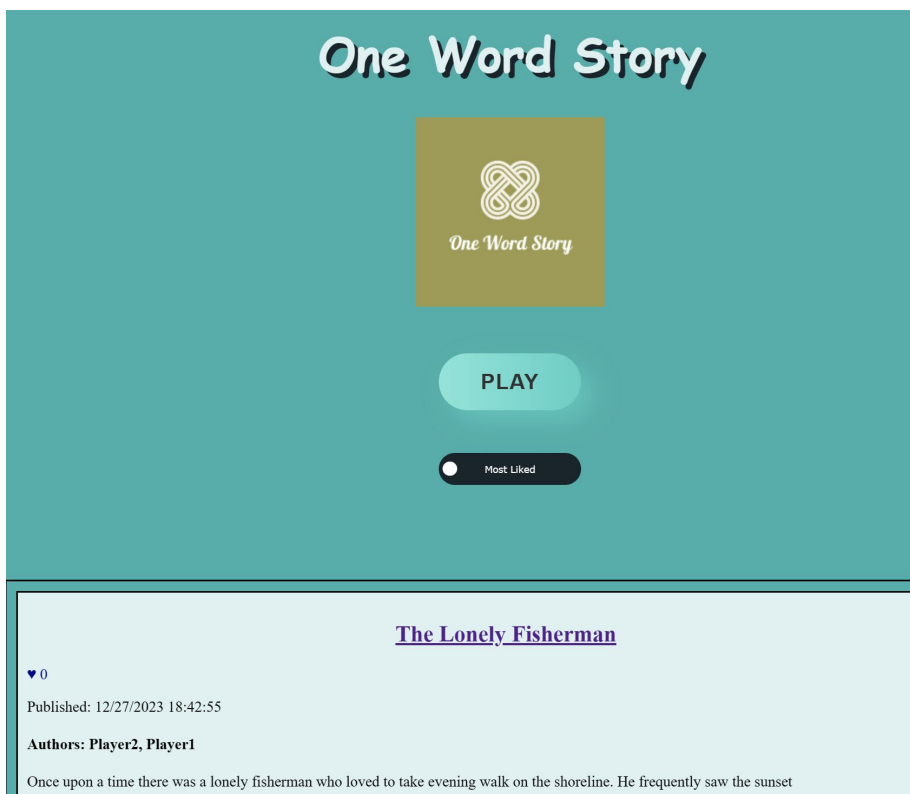


### 4: Homographies, RANSAC, SIFT

Tasked with replacing an arbitrary planar object (selected via 4 points) throughout a video, applied SIFT and RANSAC to determine a homography transformation within each frame







## Purpose

"One Word Story" is a web-based game inspired by the classic word game where players build a story one word at a time.

Developed by an eight-member team for a **Software Design** course, the platform allows players to join a match in an io-game fashion and automatically displays completed stories on its homepage after each game session.

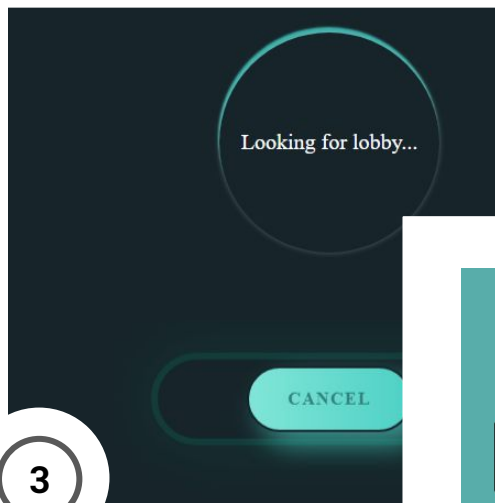
1

2



## Technologies

- Java Spring Boot
- AWS
- PostgreSQL
- Websockets
- Multi-threading
- ReactJS
- HTML/CSS/JS
- JQuery
- Clean Architecture
- SOLID Principles
- Git / Github
- CI/CD
- Regex



3



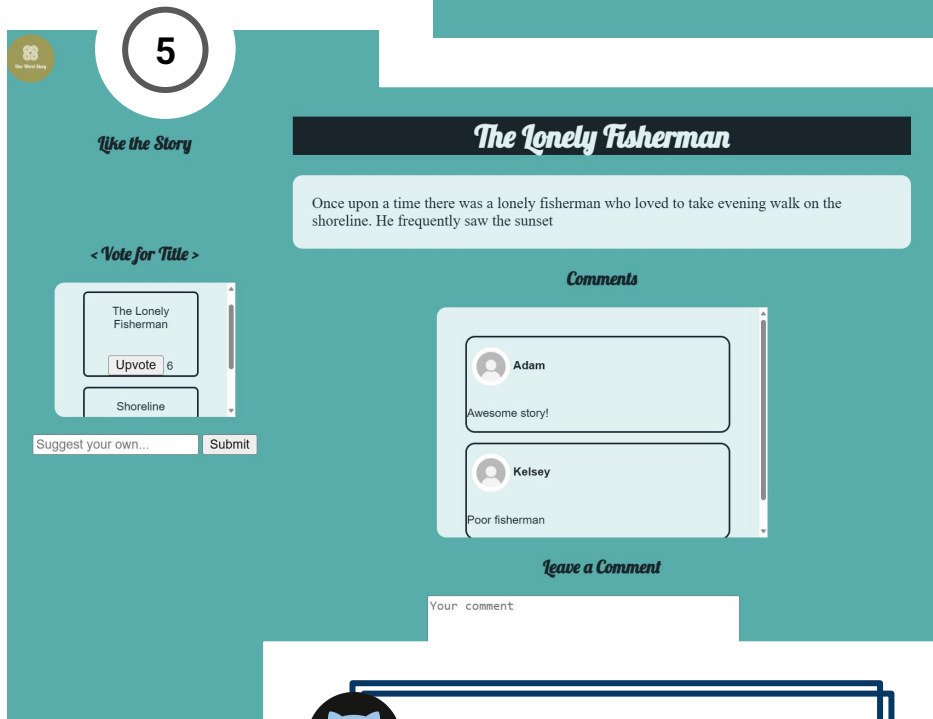
4

## Contribution

Created the vast majority of the backend, including the **websocket** code for the core game, created the **frontend** for the [4], [6] game page, and took care of domain setup and deployment via **AWS EC2**, **RDS**, and **Route53**



6



5

## Captions

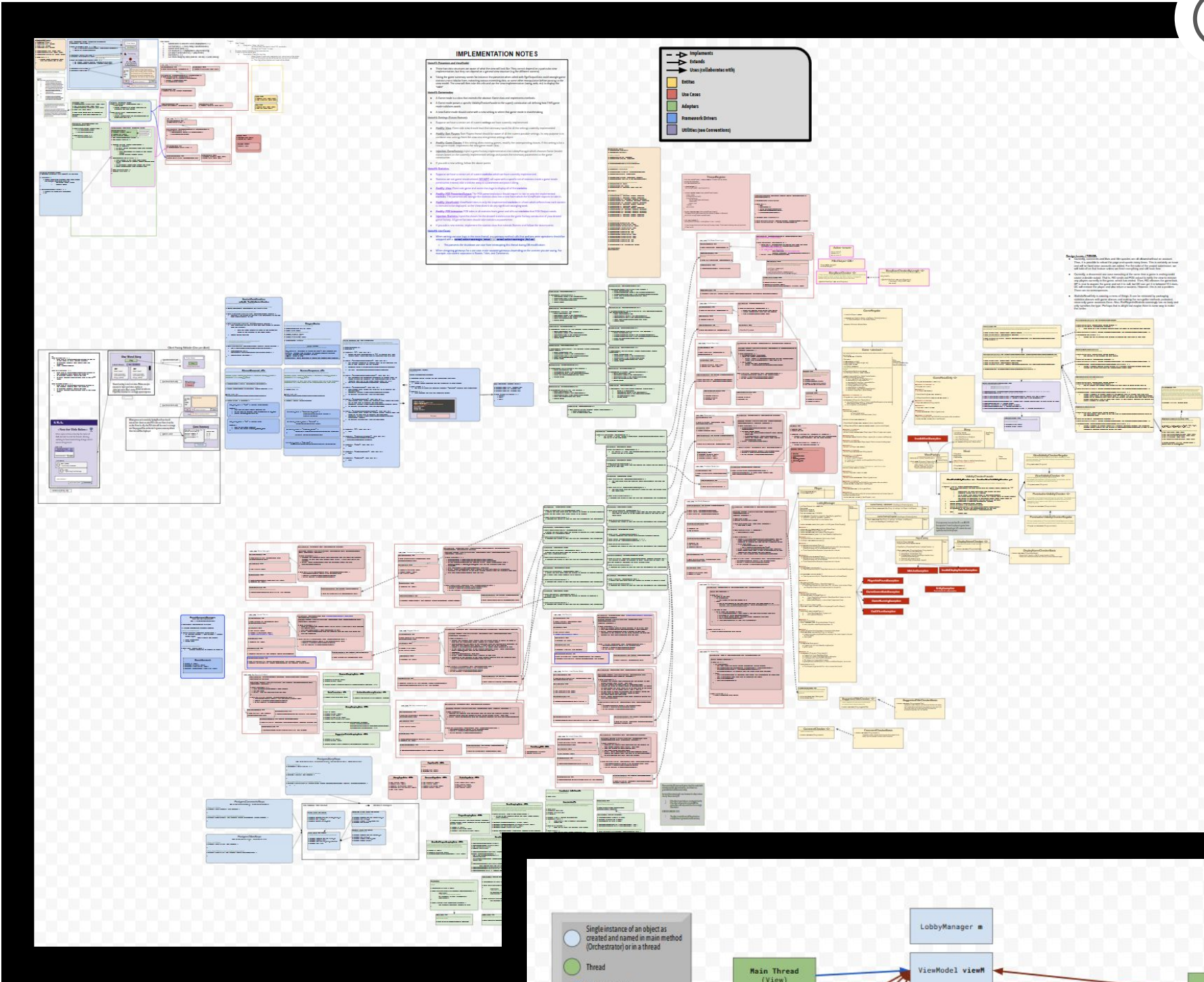
1. Homepage with completed stories on showcase
2. Post-game summary shows fun statistics for each player from the finished game
3. Loading screen
4. In-game UI
5. Page for a specific story, allowing comments and title voting
6. Regex-based input parsing

## What's Ahead

**Ongoing work** to port the game to **ReactJS**. Implementing an account system for private lobbies. Adding new game modes for variety



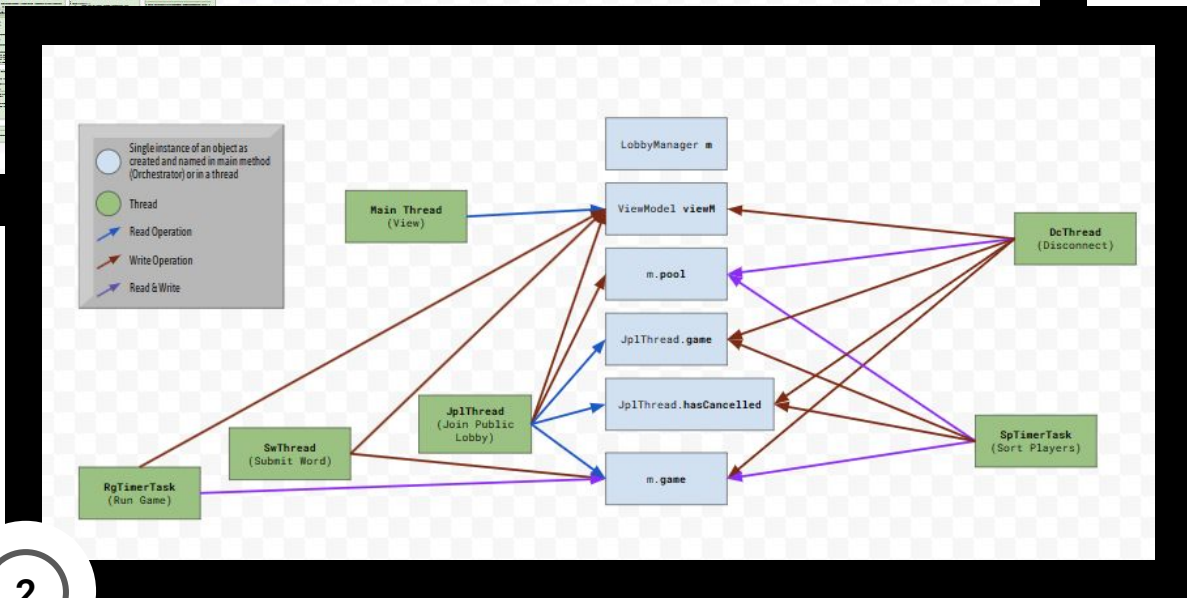
1



Captions

1. **Design diagram** used by team software architects to design new features in high-level pseudocode and organize classes into **clean architecture layers**

2



2. **Thread** chart mapping Use Cases threads to core objects they access for reads and writes. This was used extensively to resolve or prevent **deadlocks**

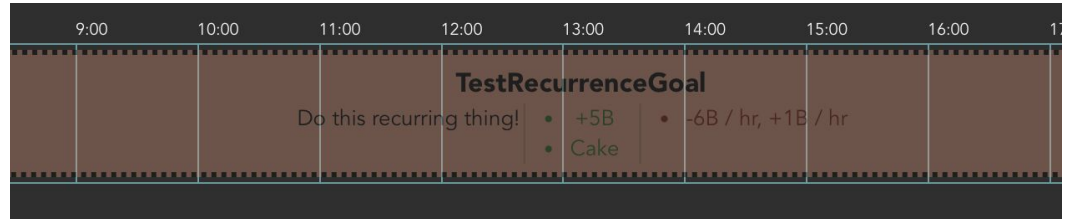


# Goal-Setting App

05/2024 - Present

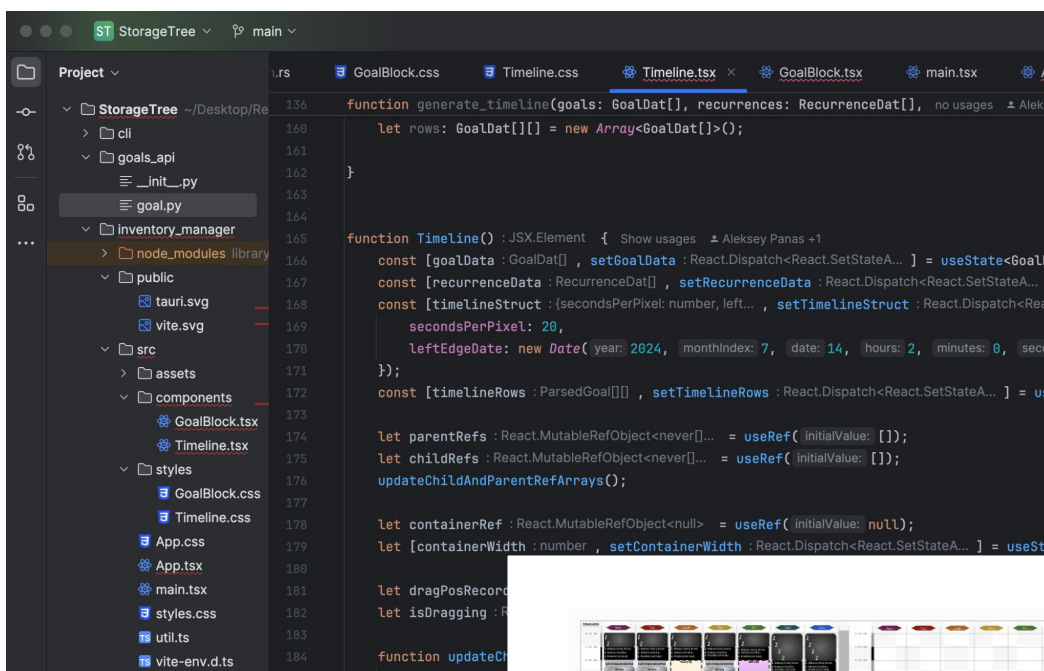
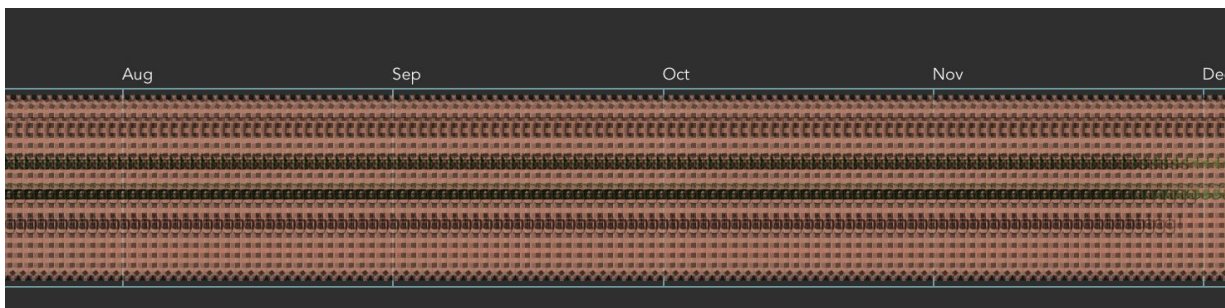
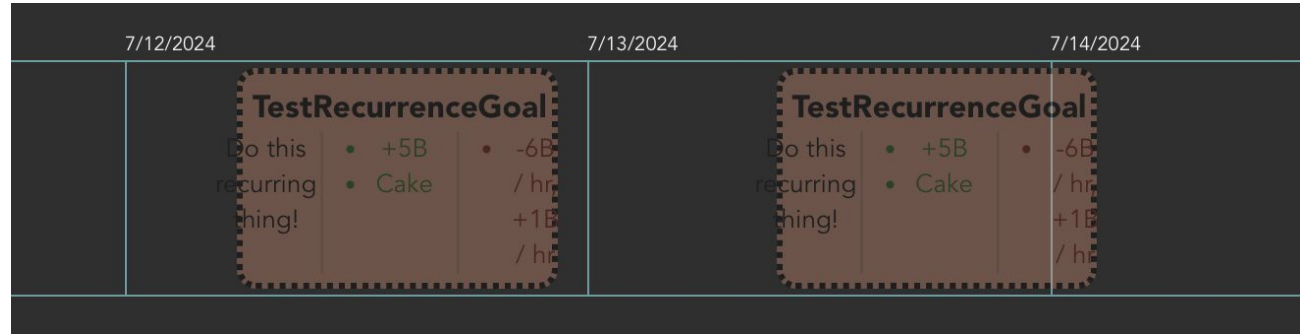
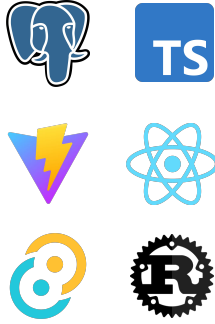
## About

No matter how many goal-setting apps I tried over the years, nothing seemed to give me the exact features I needed, so finally I decided I'd write my own solution.



## Technologies

- Tauri
- ReactJS
- Vite
- TS/TSX/CSS
- Rust
- Postgres



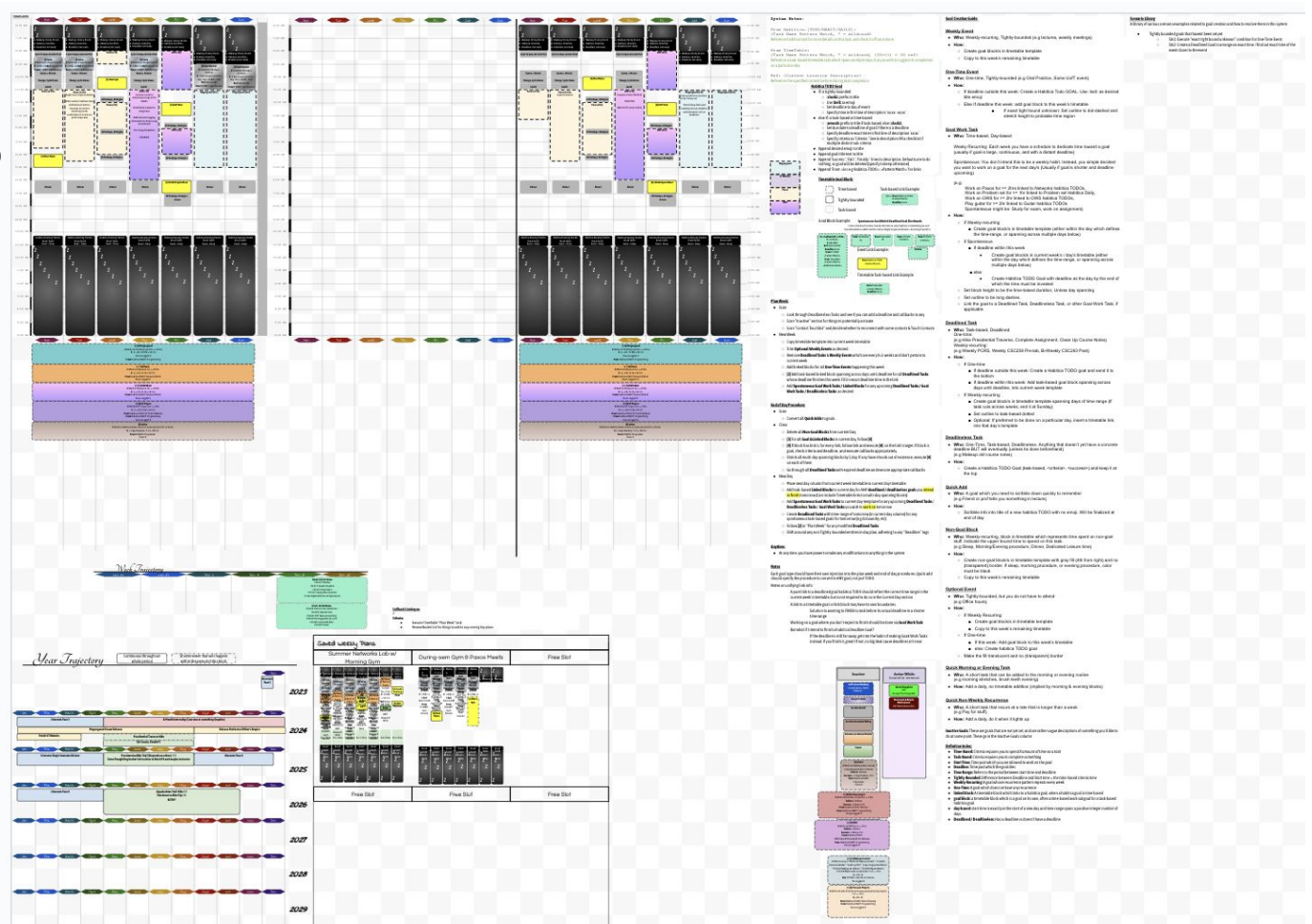
## Contribution

As part of this solo project developed a full-stack web & desktop application using **Tauri**:

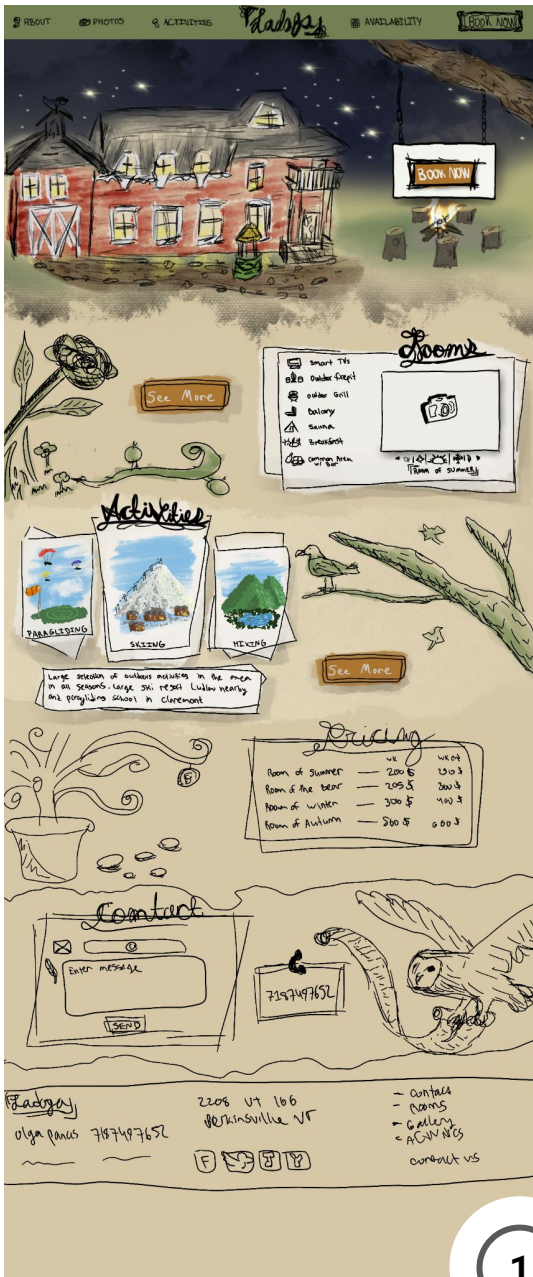
- Wrote an extensive **API backend** in **Rust** allowing the creation, modification, deletion, and resolution of goals
- Designed an in-house zoomable timeline view in the frontend using **ReactJS**
- Deployed a **PostgreSQL** database using **AWS RDS** connecting it to the backend in **Rust**

This document is my current goal-setting solution. It involves manually performing many automatable-tasks such as executing callbacks, recursively checking goal dependencies, and more.

After years of adding to this doc, it now serves as "pseudocode" for the app





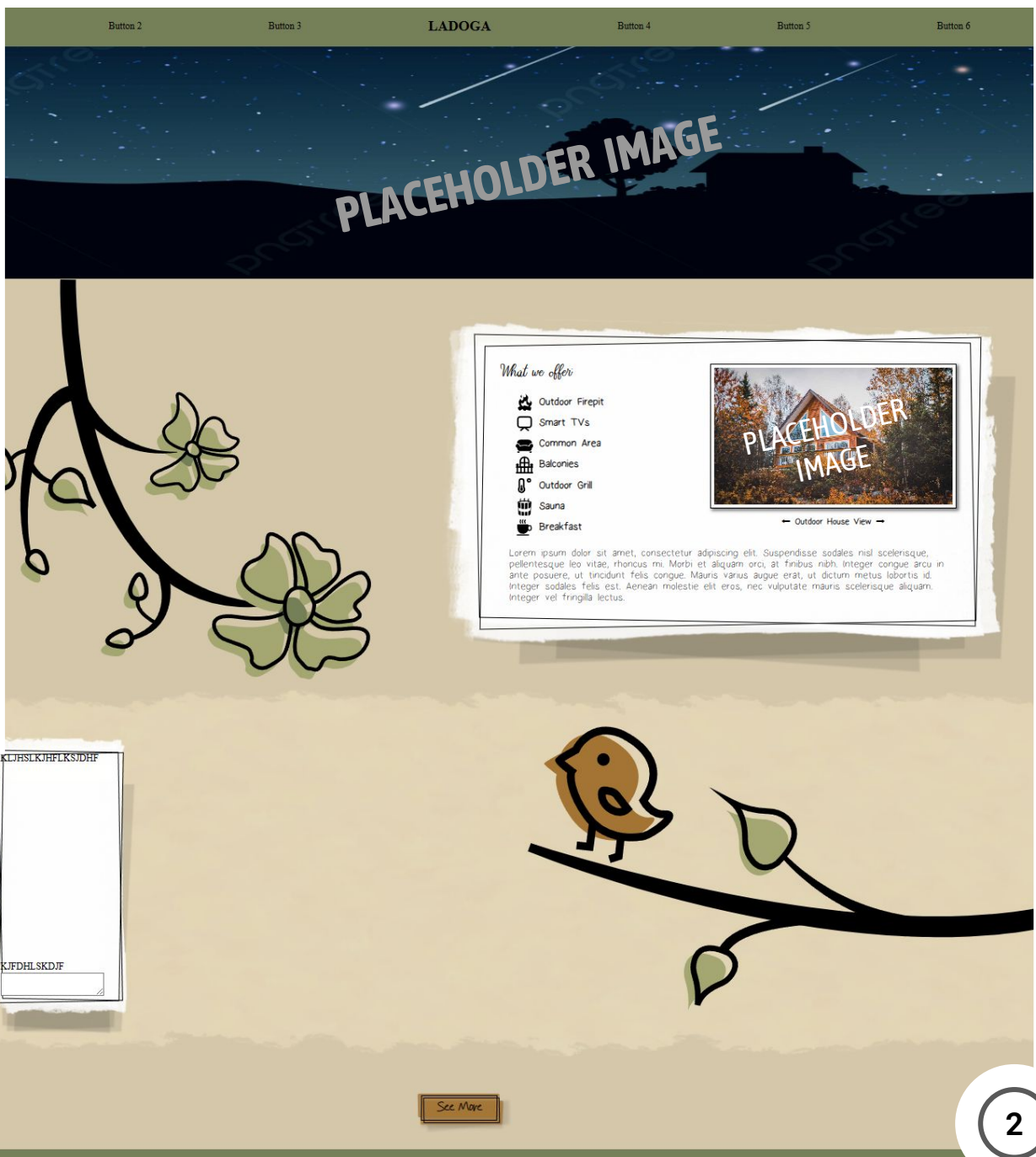


1

## Purpose

This is a two-person **client project** for a **bed & breakfast business** in New England, USA. The project's first priority is a customer-facing visually-appealing frontend to view information about the B&B, book rooms, see local activities, or contact the owners.

An additional **React-based admin dashboard** will allow management of bookings, cancellations, payments, and customer correspondences. To move off of AirBnB, the platform also includes **Stripe** payments

















2

## Captions

1. Design mockup made in Procreate
2. Custom component showcase

## Technologies

- **Java Spring Boot** backend 
- **Gradle** with customized **Kotlin** script to compile the multi-project build 
- **HTML / Typescript / SASS** for customer-facing website   
- **ReactJS** for admin dashboard 
- **Stripe** for payment processing 
- **AWS EC2, RDS, Route53** for hosting 
- **Git/Github** for version control and project management 
- **ViteJS** for frontend tooling 
- **PostgreSQL** as database choice 
- **Procreate** and **Photoshop** for mockups and frontend design  
- **SVG** vector art for animating frontend visuals 

## Contribution

Configured the **Gradle Kotlin** script to build multiple **React** and non-React frontend projects with a **Spring Boot** backend stack, as well as to execute **unit tests** and automatically pull production changes on the server

Designed the website frontend mockup and working towards implementing it in **HTML/SASS/TS** and **React** components. Additionally working on the backend for the customer facing website, admin dashboard, and **Stripe** integration.

Managing project tasks using Agile with a partner, delegating SVG work and 30% of the frontend

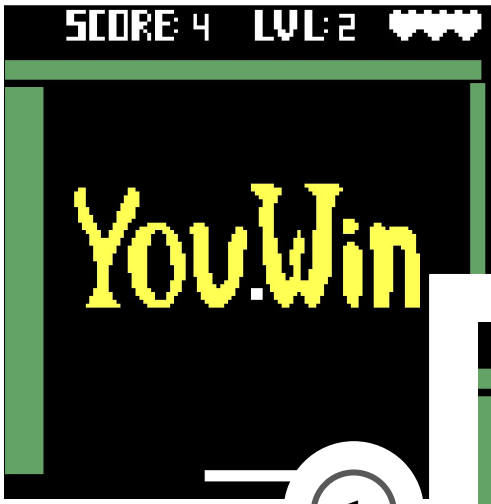


# MIPS Assembly Breakout

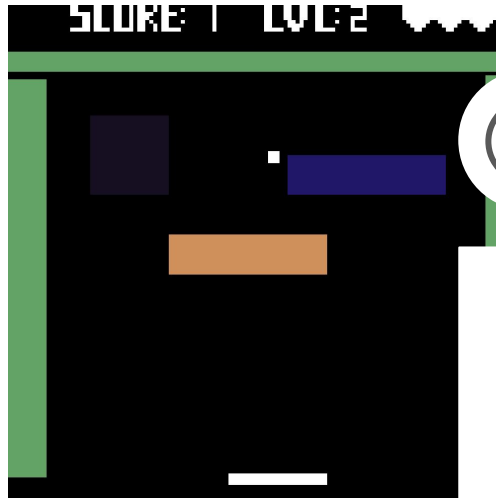
01/2023 - 05/2023

## Purpose

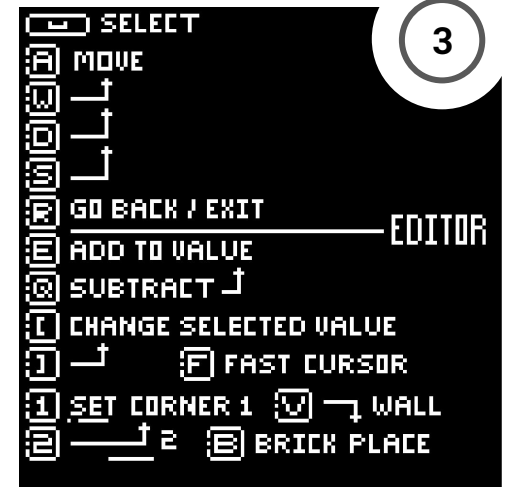
Solo project for a **Computer Organization Course** written entirely in **MIPS Assembly**. This submission goes **beyond course expectations** implementing an entire custom level editor, a menu, saving to disk, and detailed bitmaps designed with the help of a supplementary **Python** script



1



2



3

## Technologies

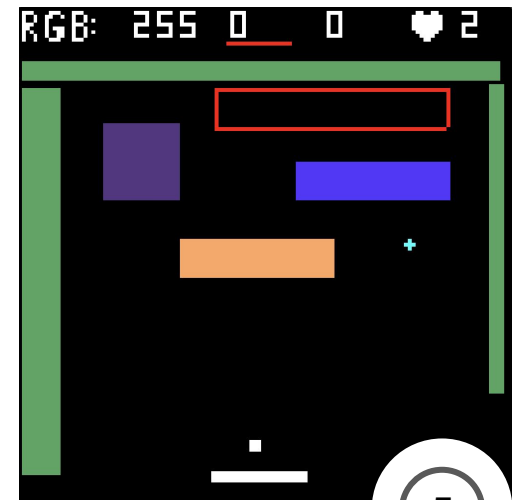
- MIPS Assembly
- Python / Pygame
- Bitmaps



4



6



5



7



## Captions

1. Win Screen
2. Gameplay
3. Help Screen
4. Game Over Screen
5. Level Editor

6. Main Menu
7. Level Selection Screen
8. Function to draw an arbitrary multi-digit base 10 number on screen (e.g for score or lives display)
9. Function to convert two points to a rectangle
10. Struct planning for game objects



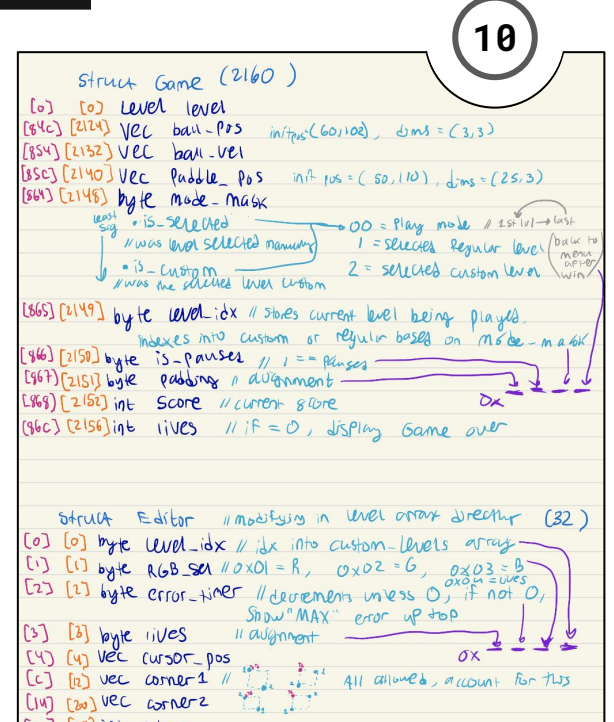
```

5312 # Given value, draw base 10 integer (supports mul
5313 # Precondition: val has <= 9 digits
5314 fn_draw_int: # (Vec top_left, int val) -> void
5315     # t0 x, t1 y, t2 val = addr, val
5316     addi $sp, $sp, 4
5317     lw $t2, 0($sp)
5318     addi $sp, $sp, 4
5319     lw $t1, 0($sp)
5320     addi $sp, $sp, 4
5321     lw $t0, 0($sp)
5322
5323     # if (val == 0)
5324     bne $t2, 0, ENDIF_fn_draw_int_0
5325     IF_fn_draw_int_0:
5326         # draw_bitmap(Vec(x,y), &BMP_ZERO)
5327         sw $ra, 0($sp)
5328         subi $sp, $sp, 4
5329         sw $t0, 0($sp)
5330         subi $sp, $sp, 4
5331         sw $t1, 0($sp)
5332         subi $sp, $sp, 4
5333         la $t9, BMP_ZERO
5334         sw $t9, 0($sp)
5335         subi $sp, $sp, 4
    
```

```

6388 # Given two points, compute the top left and width height of the corresponding rect
6389 fn_corners_to_rect: # (Vec corner1, Vec corner2) -> Vec t1, Vec w_h
6390     addi $sp, $sp, 4
6391     lw $t3, 0($sp) # t3 = corner2.y
6392     addi $sp, $sp, 4
6393     lw $t2, 0($sp) # t2 = corner2.x
6394     addi $sp, $sp, 4
6395     lw $t1, 0($sp) # t1 = corner1.y
6396     addi $sp, $sp, 4
6397     lw $t0, 0($sp) # t0 = corner1.x
6398
6399     # t4 = min x, t5 = max x
6400     bge $t0, $t2, ELSE_corners_to_rect_1
6401     IF_corners_to_rect_1: # if (t0 < t2)
6402         # t4 = t0, t5 = t2
6403         addi $t4, $t0, 0
6404         addi $t5, $t2, 0
6405
6406         j ENDIF_corners_to_rect_1
6407     ELSE_corners_to_rect_1: # else: // t0 >= t2
    
```

9



8



```
Alek@Badlek ~/steamcmd/ark/ShooterGame/Binaries/Linux/rcon_script $ ./start_rcon_client.sh
[Server 10.0.0.159:27020] Connected to 10.0.0.159:27020 successfully!
[Server 10.0.0.159:27020] Reception thread started.
[Server 10.0.0.159:27020] Password sent.
[Server 10.0.0.159:27020, ID=50, TYP=2]
Enter 'i' to input a command, 'q' to quit, or 's' to shutdown a cluster
i
Enter an rcon command: say hi
[Server 10.0.0.159:27020] Sent command 'say hi'
[Server 10.0.0.159:27020, ID=69, TYP=0] Server received, But no response!!

[Server 10.0.0.159:27020, ID=0, TYP=0] Keep Alive
i
Enter an rcon command: broadcast hello world
[Server 10.0.0.159:27020, ID=0, TYP=0] Keep Alive
[Server 10.0.0.159:27020] Sent command 'broadcast hello world'
[Server 10.0.0.159:27020, ID=70, TYP=0] Server received, But no response!!

[Server 10.0.0.159:27020, ID=0, TYP=0] Keep Alive
i
Enter an rcon command: saveworld
[Server 10.0.0.159:27020, ID=0, TYP=0] Keep Alive
[Server 10.0.0.159:27020] Sent command 'saveworld'
[Server 10.0.0.159:27020, ID=71, TYP=0] World Saved

[Server 10.0.0.159:27020, ID=0, TYP=0] Keep Alive
[Server 10.0.0.159:27020, ID=0, TYP=0] Keep Alive
i
Enter an rcon command: doExit
[Server 10.0.0.159:27020] Sent command 'doExit'
[Server 10.0.0.159:27020, ID=72, TYP=0] Exiting...

[Server 10.0.0.159:27020] Disconnected Successfully.
[Server 10.0.0.159:27020] Connection closed on other end, stopping reception..
[Server 10.0.0.159:27020] Reception thread closed.
q
Quit command issued.
Socket closed.
Alek@Badlek ~/steamcmd/ark/ShooterGame/Binaries/Linux/rcon_script $
```



```
Config file generated:
SAVED_FOLDER_SEARCH_DEPTH=4
MAX_BACKUP=3
SAVED_FOLDER_PATH=../../../../ark/ShooterGame/Saved
JSON_KEY_FILE_PATH=serverbackup-395501-d87a69dc68f5.json

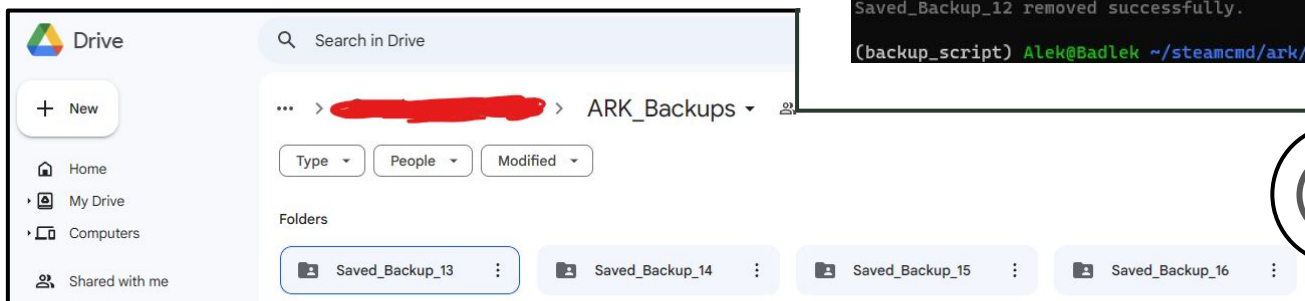
Looking for root cloud directory...
Root cloud directory found: Name: ARK_Backups, ID: 1qcgBTI70uf0hGr_6pE4eQ9M51-hxeoKB

Looking for existing backups...
Backups found:
Saved_Backup_15
Saved_Backup_14
Saved_Backup_13
Saved_Backup_12

Uploading folder...
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_02.11.2023_20.13.05.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_26.12.2022_03.28.25.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/1228672982.arktribe
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_20.12.2023_01.18.21.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/76561198409587803.arkprofile
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_21.12.2023_01.03.23.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_20.12.2023_03.28.21.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_31.07.2022_02.51.10.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_29.07.2022_02.10.24.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_28.10.2023_19.46.33.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_02.11.2023_03.20.17.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/SavedArks/TheIsland_17.12.2023_22.06.38.ark
File Uploaded: ../../../../../../ark/ShooterGame/Saved/AllowedCheatSteamIDs.txt
Folder uploaded successfully with name Saved_Backup_16

Checking to delete old backups...
Removing Saved_Backup_12...
Saved_Backup_12 removed successfully.

(backup_script) Alek@Badlek ~/steamcmd/ark/ShooterGame/Binaries/Linux/backup_script $
```



## Purpose

Two scripts were developed to streamline the management of an ARK: Survival Evolved server on a local machine:

- **RCON Script:** Using Python with TCP sockets, this script implements Valve's RCON protocol from scratch, enabling remote execution of server commands outside of the game
- **Google Drive Backup Script:** This Python script automatically detects the game save folder within the server directory and subsequently uploads it to a Google Drive folder for backup

The ARK server was configured on a custom **Gentoo Linux** installation where the two scripts are now used on a regular basis

## Captions

1. **RCON Script** in action
2. Execution of the **Google Drive backup script**
3. Shows latest backup appearing in Drive

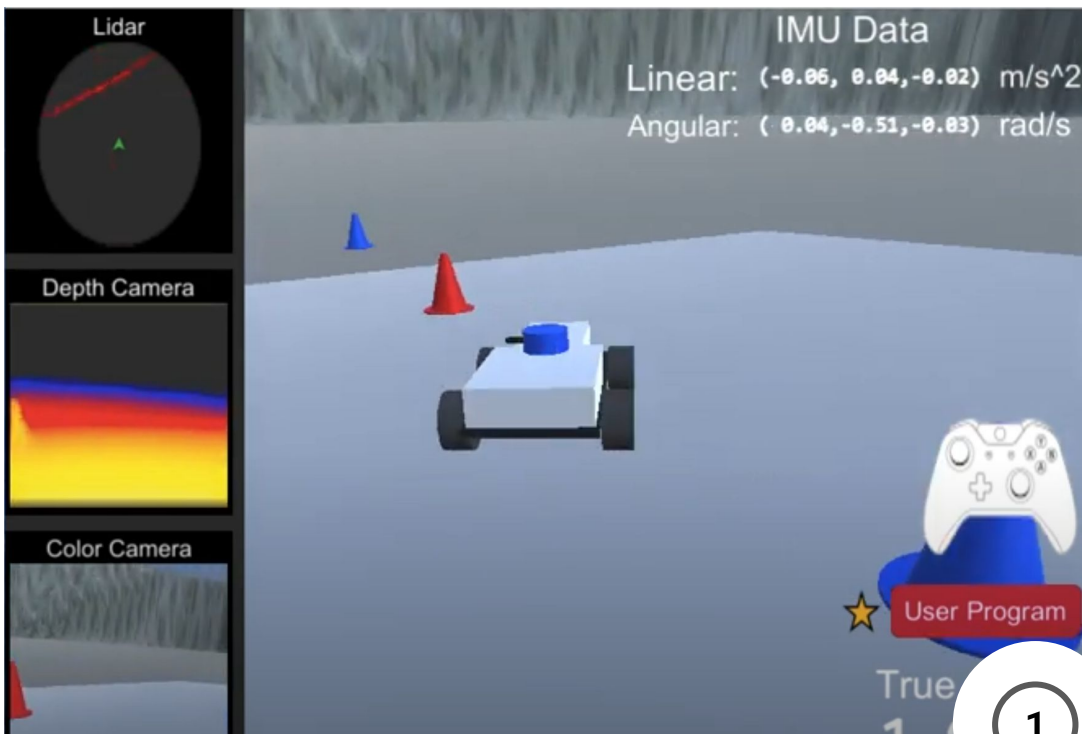
## Technologies



- Python
- Gentoo Linux
- TCP Socket Programming
- RCON Protocol
- Google API

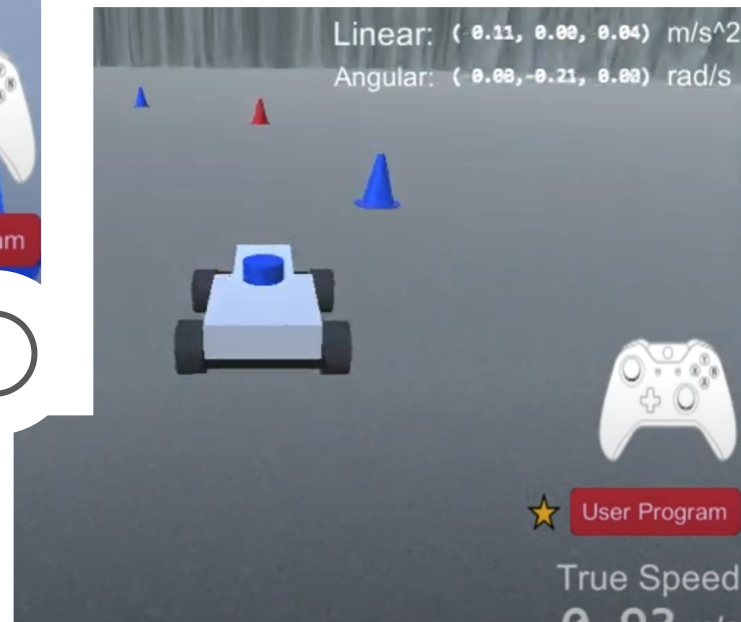


# MIT Beaverworks: RACECAR Course 07/2020 - 08/2020



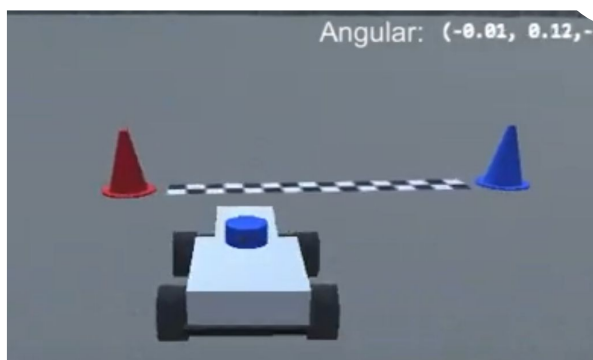
## Purpose

Completed a month-long course at MIT Beaverworks progressing from the foundations of OpenCV to writing autonomous code for a car to navigate an obstacle course consisting of lane following, cone slaloms, ledge avoidance, and more

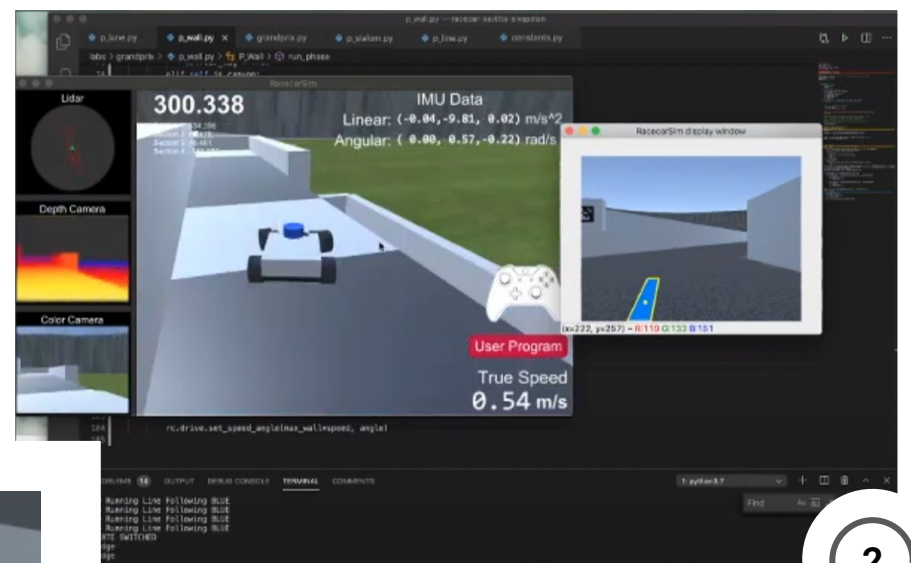


## Technologies

- OpenCV
- Numpy
- Depth Camera
- LiDAR
- SLAM
- Ubuntu
- Jetson NANO
- Sensor Fusion
- Unity
- Python
- AR Codes



- [Cone Slalom](#)
- [Group Presentation Video](#)
- [Time Trial 1st Place](#)
- [Final Grand Prix Course Run](#)

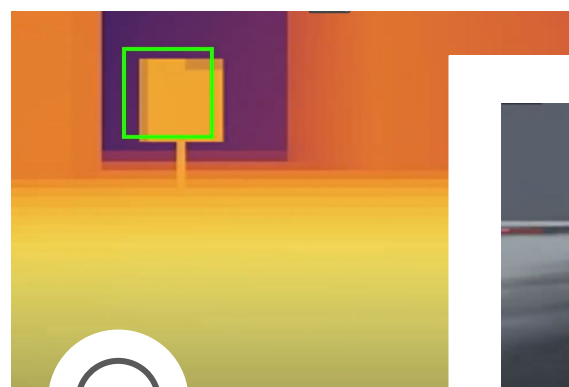


## Captions

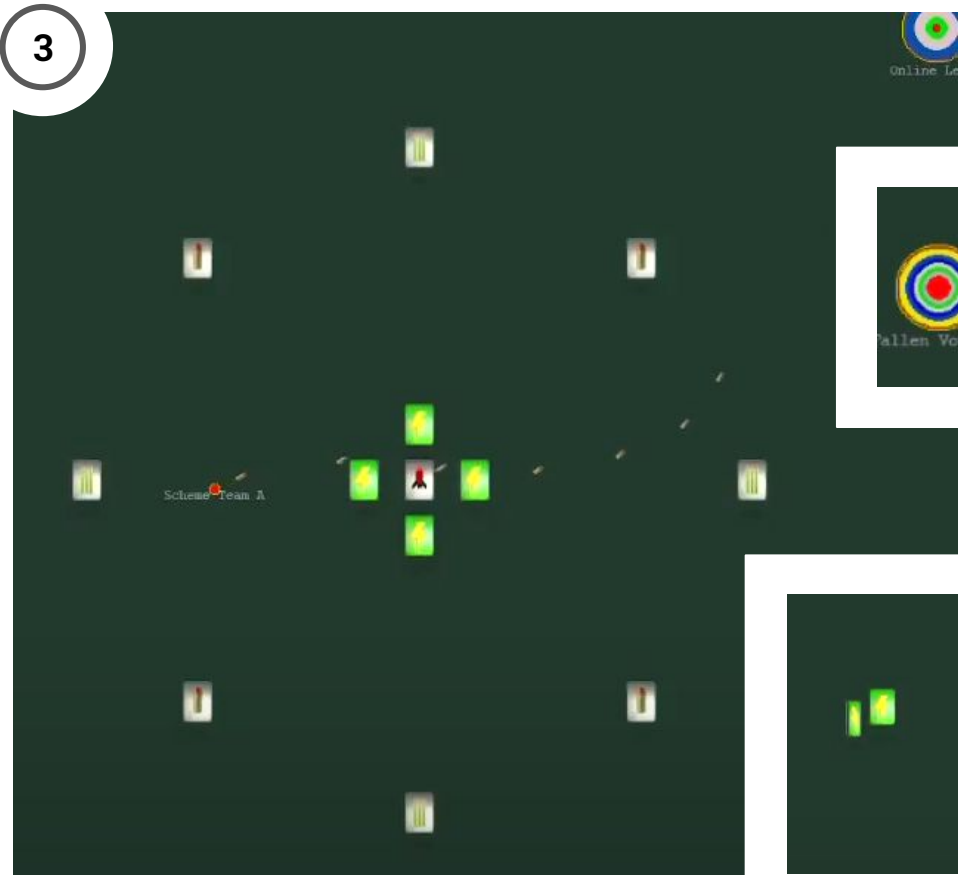
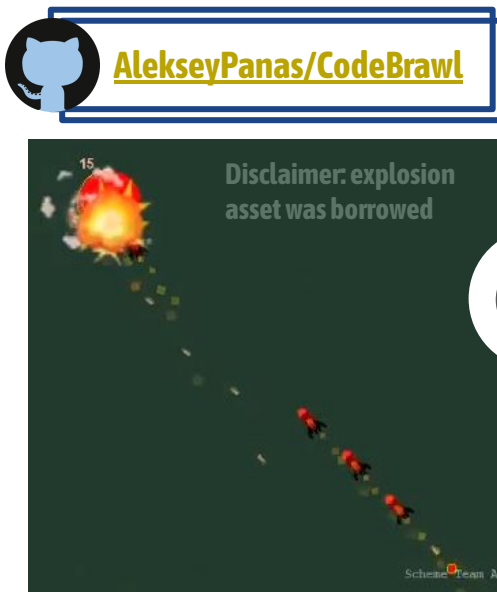
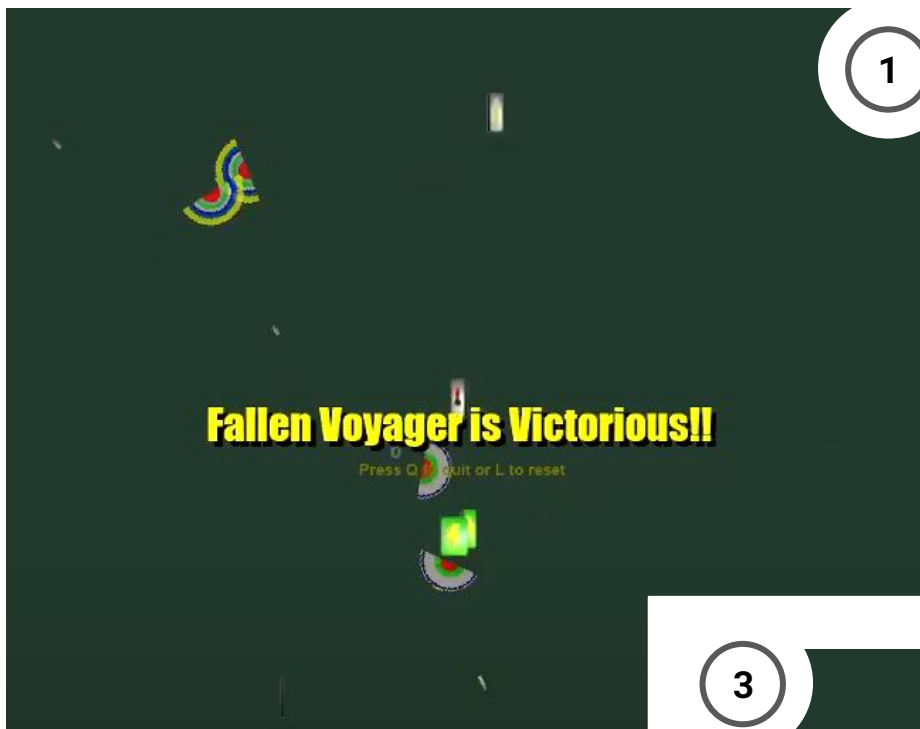
1. Cone Slalom
2. Ledge avoidance and wall following
3. City designed in Unity for navigation testing
4. Depth camera closest AR tag detection
5. Line follower and AR tag detection

## Course Assignments Involved...

- Wrote a line and lane follower with color switching based on AR tags
- Wrote a cone slaloming algorithm
- Wrote SLAM code to navigate a wall obstacle course and keep track of location
- Incorporated data from a color camera, depth camera, LiDAR, accelerometer, and gyro in the autonomous navigation code applying sensor fusion techniques
- Deployed code to real robotic race car via Ubuntu on Jetson NANO
- Created the video for and contributed to our group's final presentation
- Wrote code for an autonomous time trial course winning 1st place among all students



AlekseyPanas/BWSI-Labs  
apanas9246/GRANDPRIX



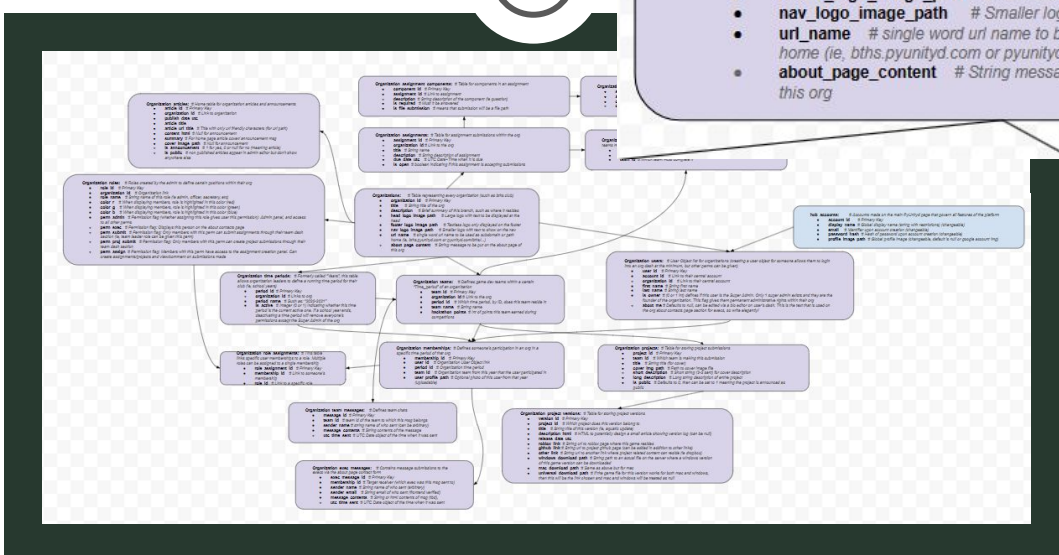
## Purpose

Codebrawl was written to host a final coding competition in the **Brooklyn Tech PyUnityd Game Dev club**




The game is multiplayer over **TCP** and is played by implementing an AI client script. The server then calls the AI scripts of each participating player and displays the visuals of the ongoing battle.

The **gameplay** involves customizing player stats pre-game, moving to acquire energy, shooting, close-range swordplay, and missiles.

-  [Competition Round 1](#)
-  [Competition Round 2](#)
-  [Competition Round 3](#)



## Technologies

- Python / Pygame  
- TCP Socket Programming
- Photoshop to design assets 

## Contribution

Developed the entire game start to finish in Python. Brought on two others to assist in writing additional clients in Scala, C#, and Java

## What's Ahead

Ongoing work to port this game to a webapp

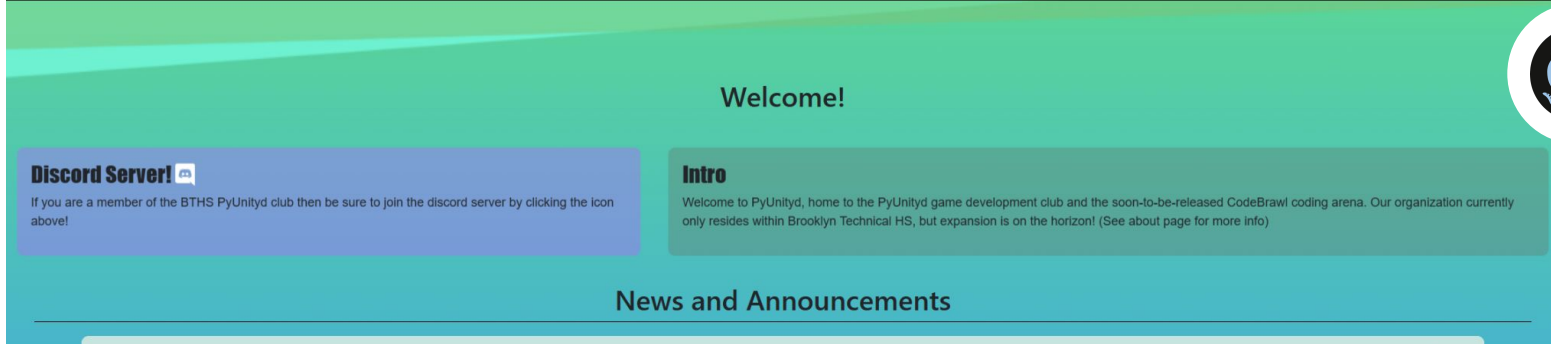
## Captions

1. Victory screen with custom splitting and fading animations on defeat
2. Missiles being fired
3. Starting arena appearance and bullet firing
4. Damage number indicators, sword, and bullets
5. Sword blocking mechanic
6. Database map for ongoing work to port project to web
7. Lobby screen

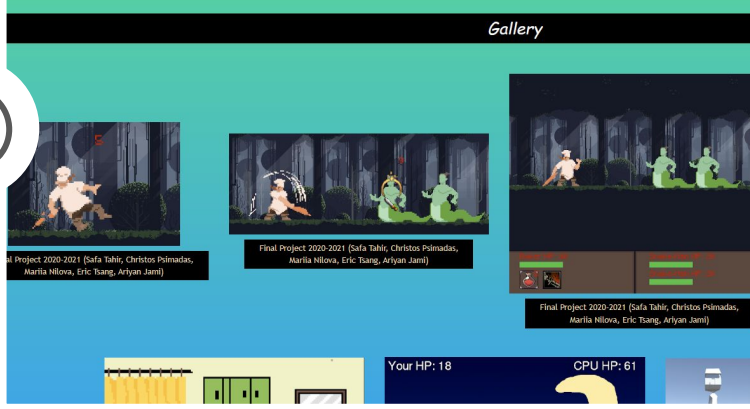
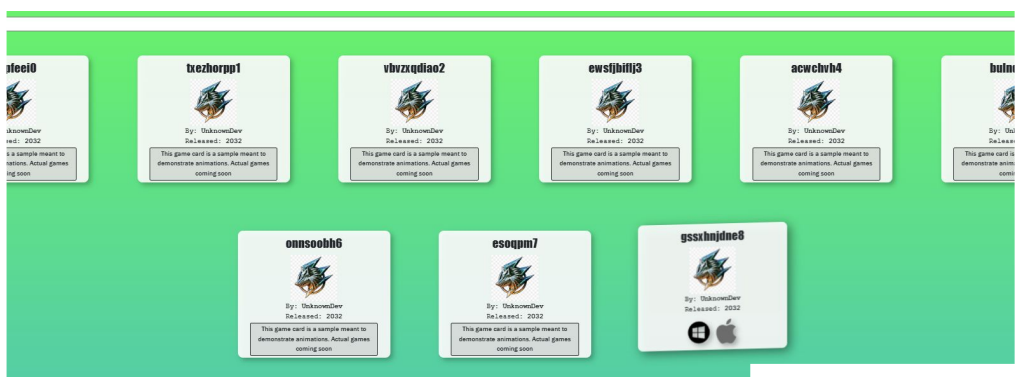
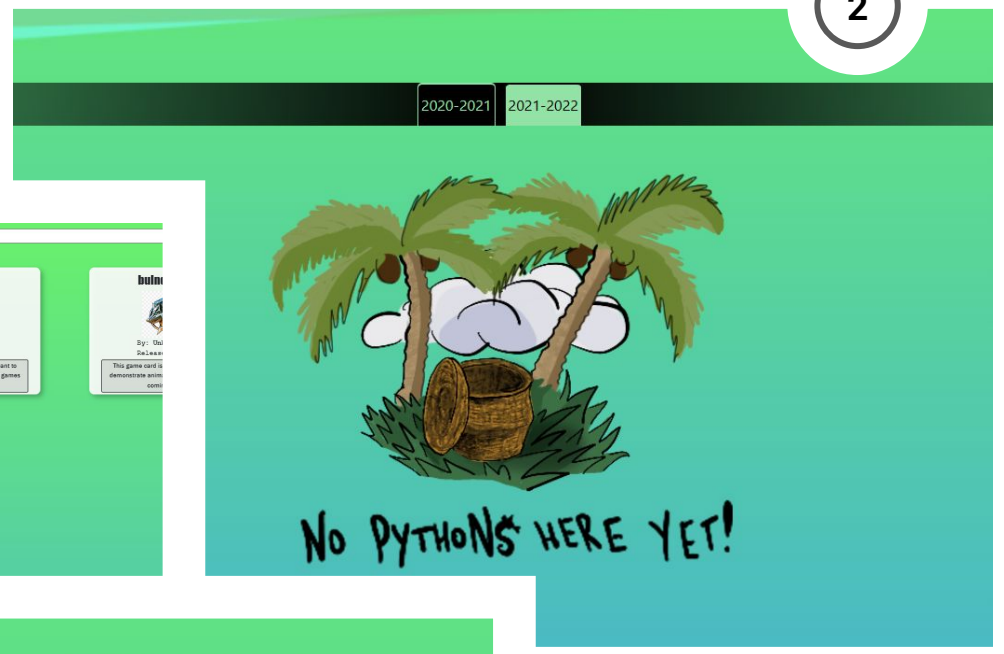




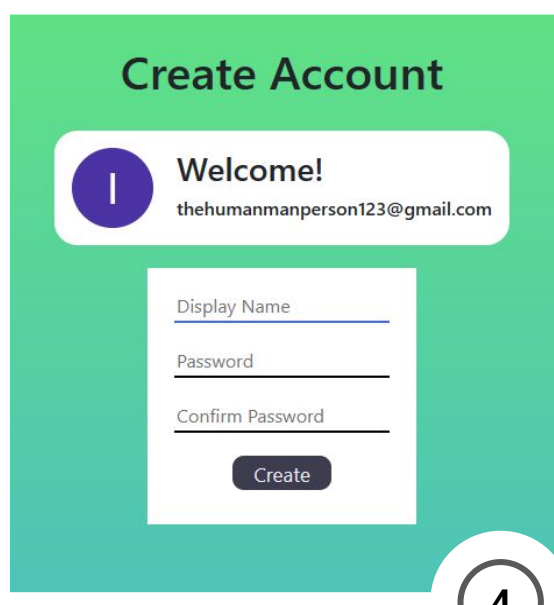
1



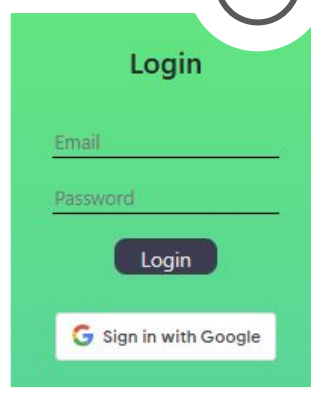
2



3



4



### Technologies

- NodeJS
- ExpressJS
- PostgreSQL
- OAuth
- AWS EC2, RDS, Route53
- TLS (SSL)
- Git / Github
- NginX
- HTML / CSS / JS
- Async / Await
- Ubuntu
- Photoshop / Procreate

### Purpose

Official website for the Brooklyn Tech PyUnityd Club created to display club membership, host competition results, and showcase club projects.

### Contribution

As Founder and President of the club, initiated the decision to create an organization website and built the solo project from scratch

### Captions

1. Home page
2. Art created in **Procreate**
3. Project Page with gallery of club members' creations
4. Account creation system with google login option

### What's Ahead

After graduating High School, plans were put in action to turn this into a public game development competition platform. That is still an **ongoing** development



1

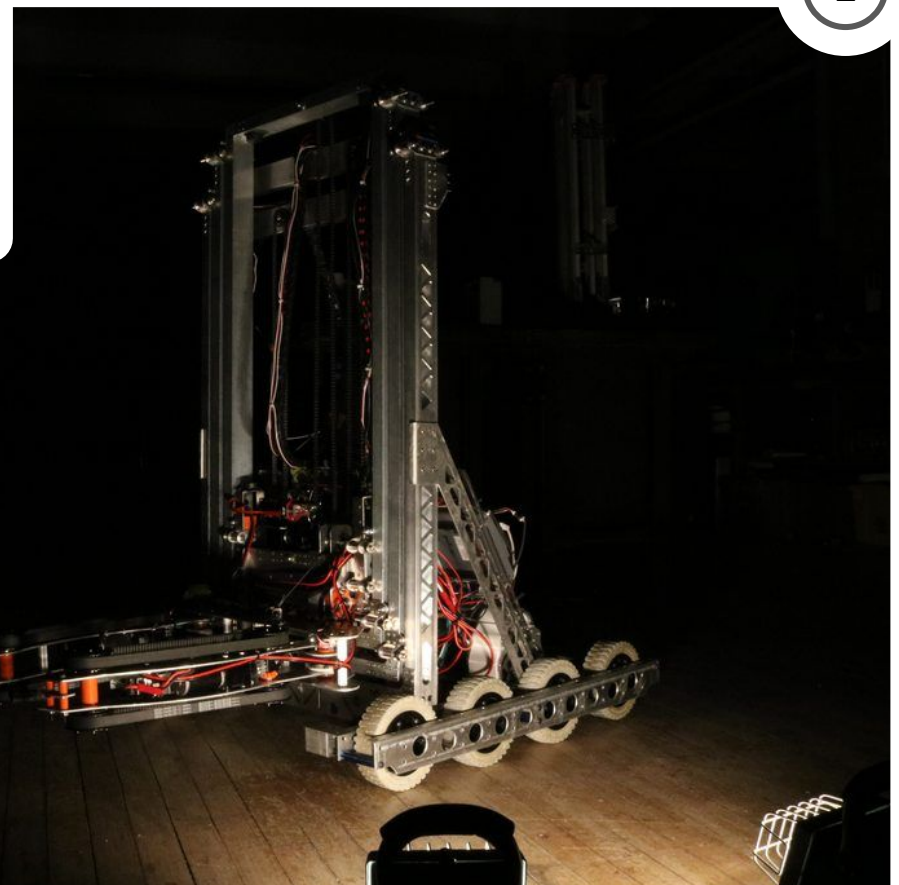
## Purpose

Brooklyn Technical High School's Robotics Team (334 TechKnights) participates in an annual FIRST Robotics competition.

Each year, a new game is revealed comprising both autonomous and human-operated phases.

Within a six-week build season, the team designs, constructs, and programs a robot from the ground up to meet the game's challenges.

2



## Captions


1. Fun side project: a ride-able "seat bot" coded in **C++** and powered by Arduino
2. Team334's robot for the 2019 competition
3. 2015 Robot used frequently for testing code (LED ring for aiding in computer vision tasks)

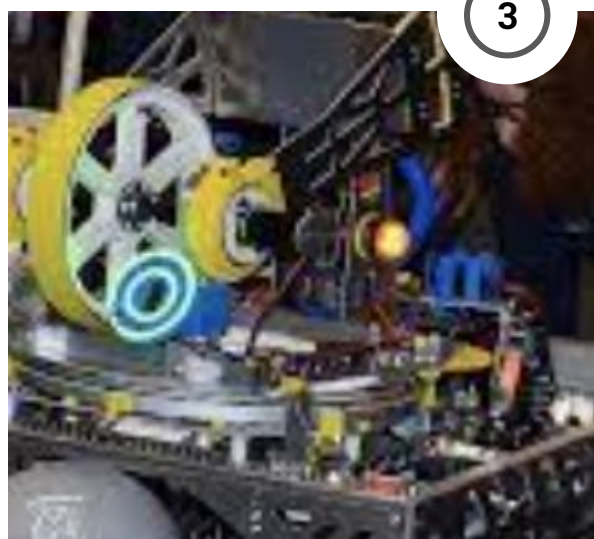
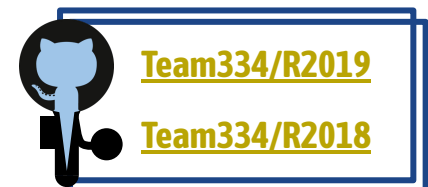
## Contribution

Contributed code for four years, writing computer vision in **OpenCV** with **Python** and the remaining autonomous and teleop code in **Java**.

In latter two years, acted as programmer lead implementing an **Agile** workflow.

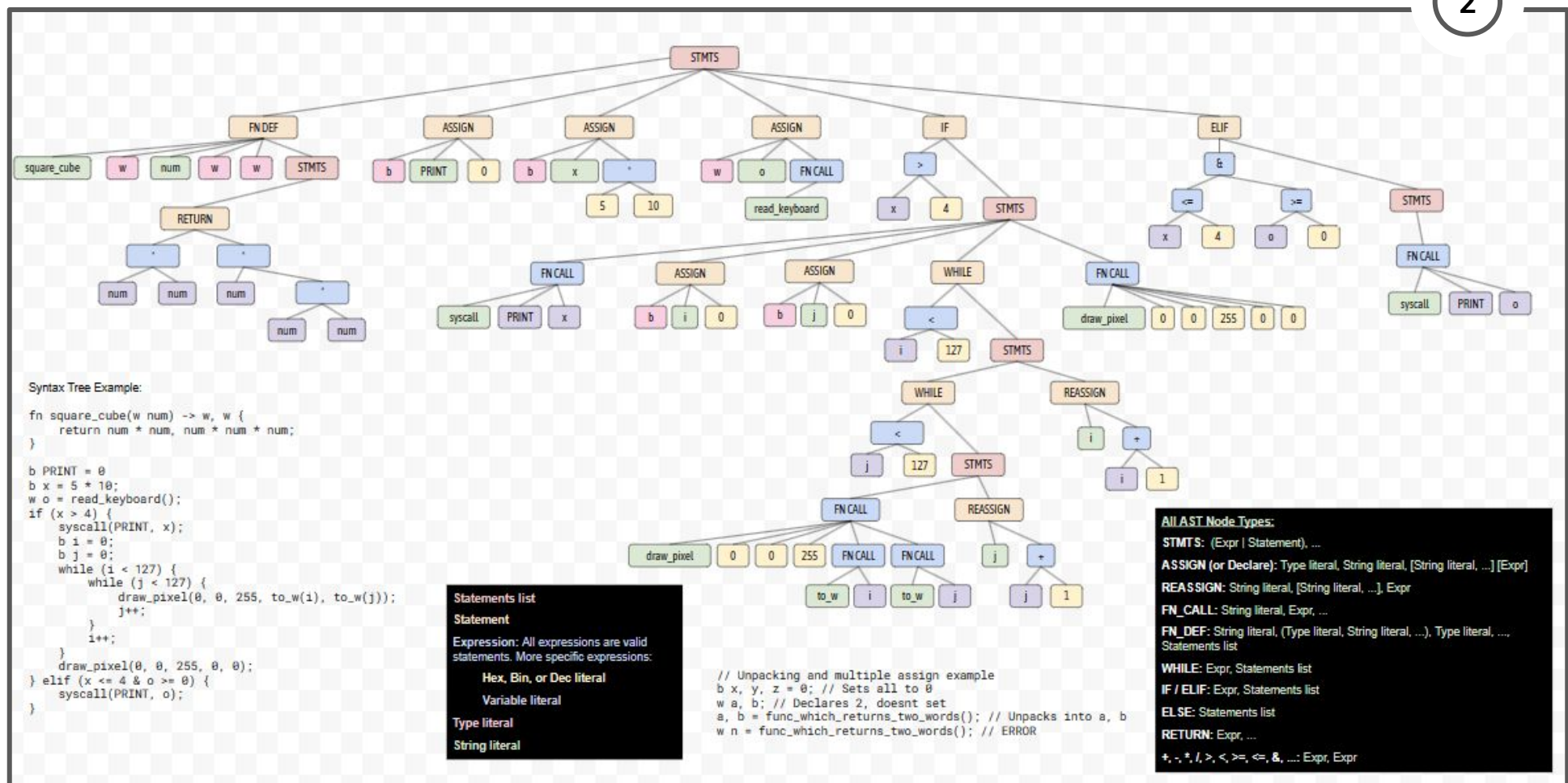
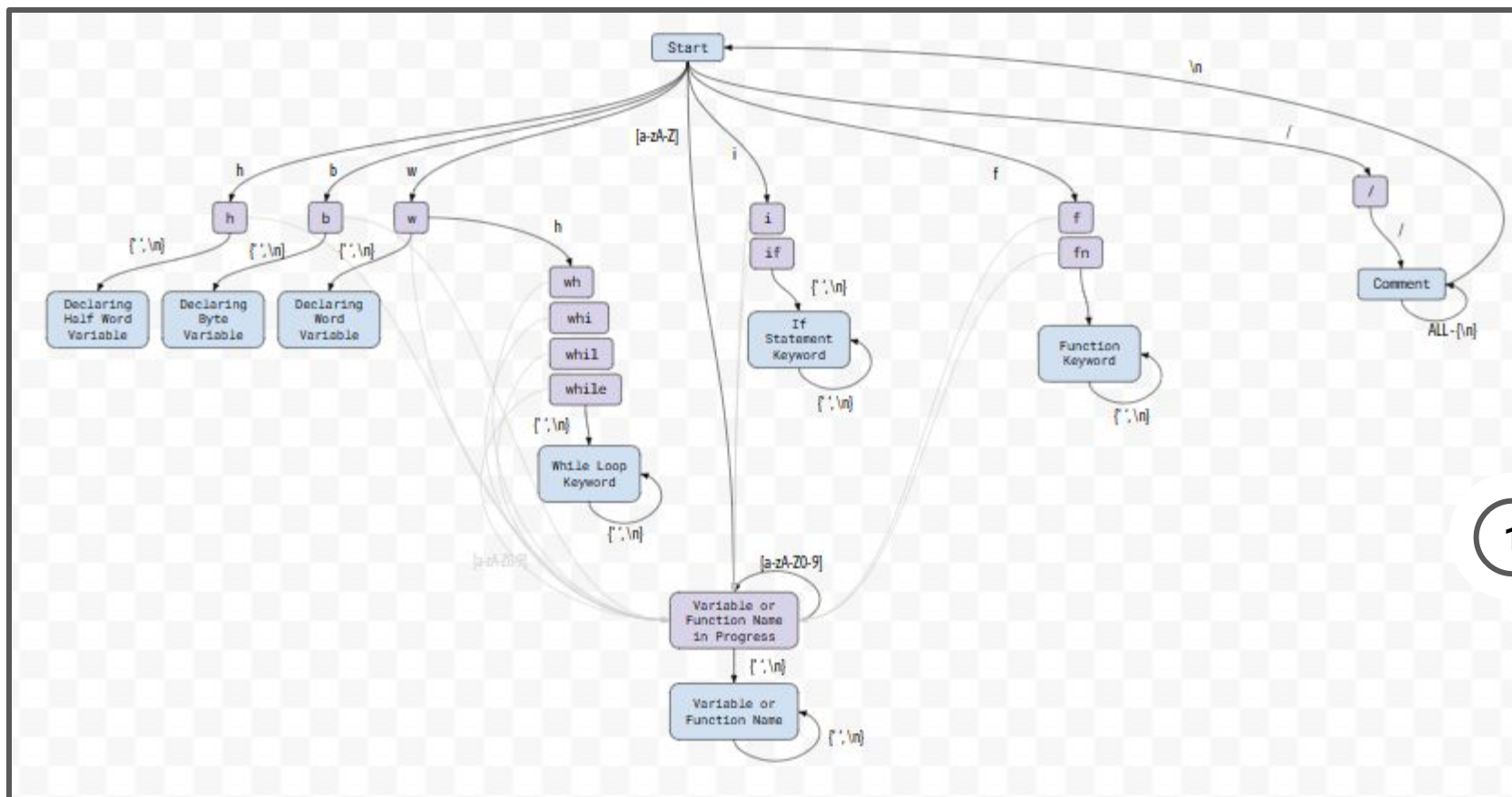
## Technologies

- OpenCV 
- Numpy 
- Java 
- PID Loops
- Jetson NANO 
- Arduino 
- C++ 
- Ubuntu 
- Robotic hardware & Sensors
  - LiDAR, Gyro, Ultrasonic, Camera, Accelerometer, Potentiometer, Pneumatic systems, Motors
- Python 
- ROS 
- Git/Github 
- Agile (Shortcut)



3





## Purpose

A personal solo project conceived after finishing Computer Organization course out of curiosity to write own person language. The project is **ongoing** in its early stages with a completed grammar parser and 10% completed compiler. The language is low level including only byte, word, and halfword datatypes, array functionality, loops, and if statements.

## Technologies

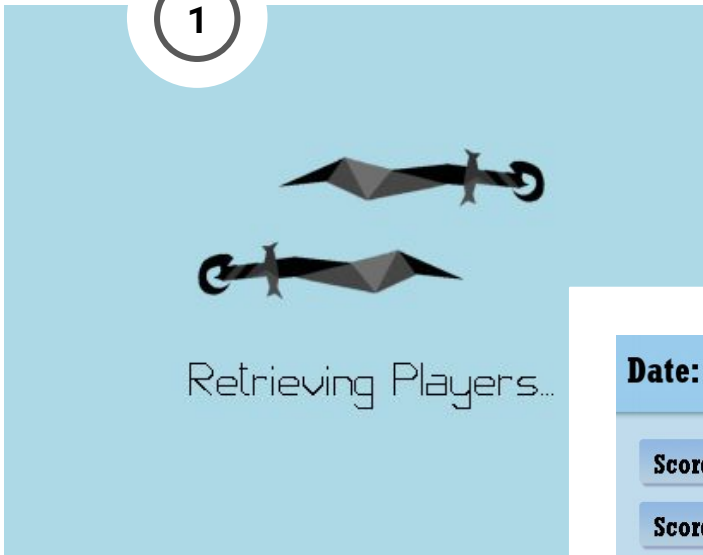
- C
- Abstract Syntax Trees
- MIPS Assembly



## Captions

1. DFSA diagram initially used when brainstorming how to parse grammar
2. Syntax tree when brainstorming language syntax and components

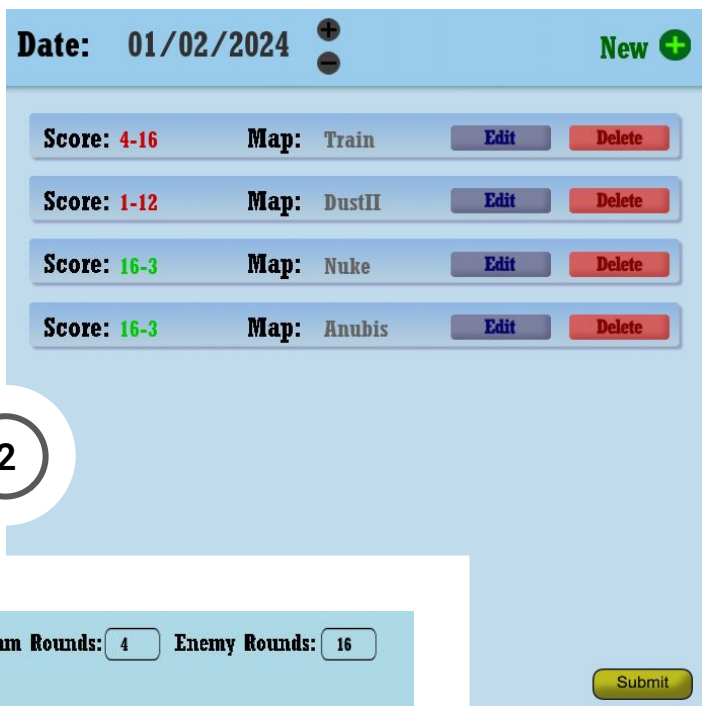
1



### Purpose

Solo-built app created after launching a semi-casual Counter Strike team to track games played and per-player statistics to gauge team performance and analyze areas of improvement

2



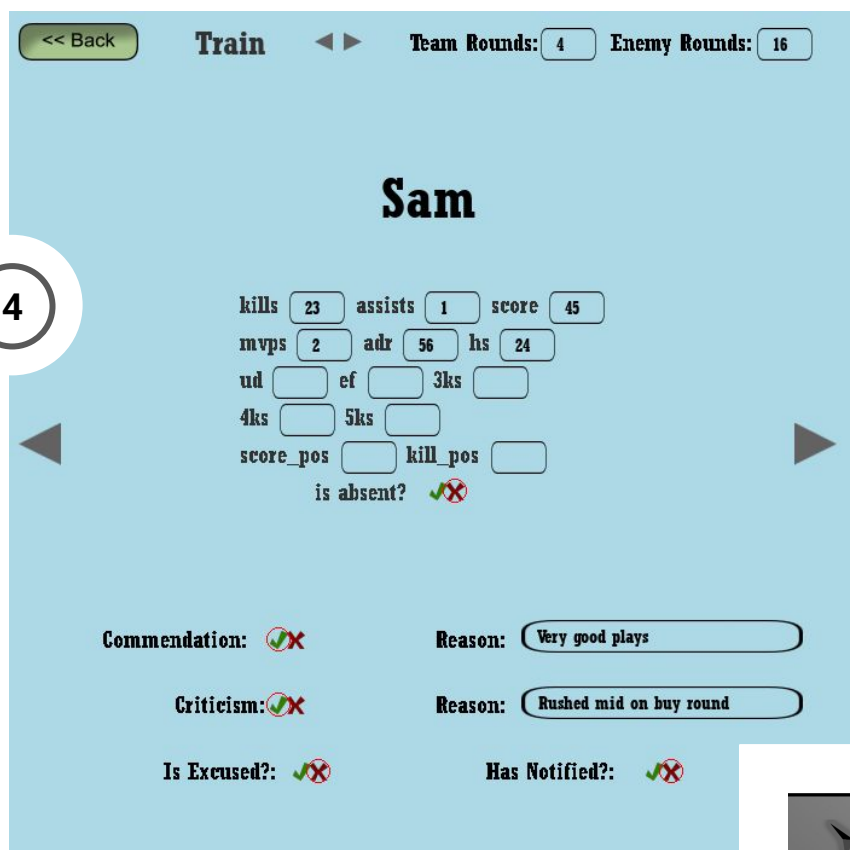
### Captions

1. Animated loading screen with custom-made team logo
2. Interface for adding played games
3. Errors when connections fail
4. Interface for inputting game statistics
5. Web frontend for viewing team information (work in progress)

3



4



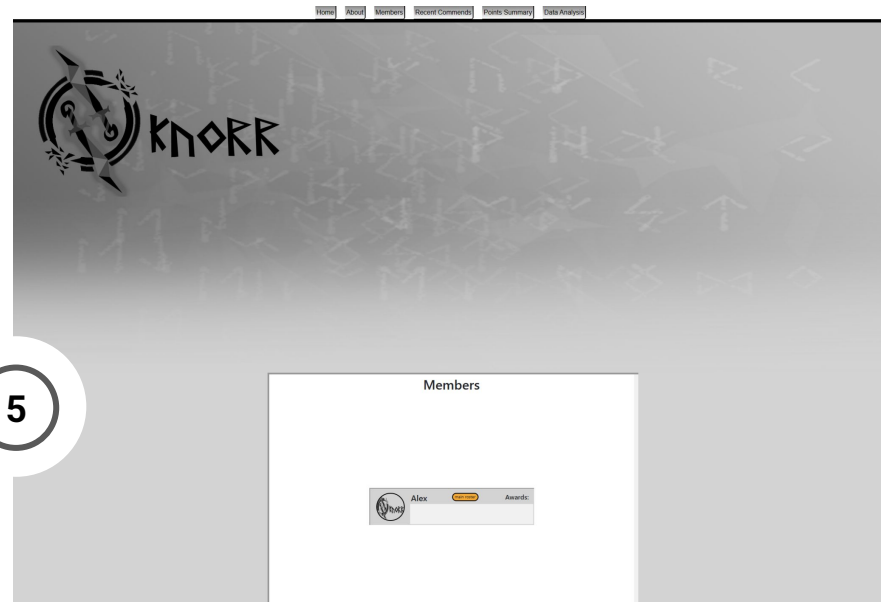
### What's Ahead

Although the team was dissolved after first two years of Covid, **ongoing** work is being done to implement automatic tracking with a steam bot pivoting towards an open-source release

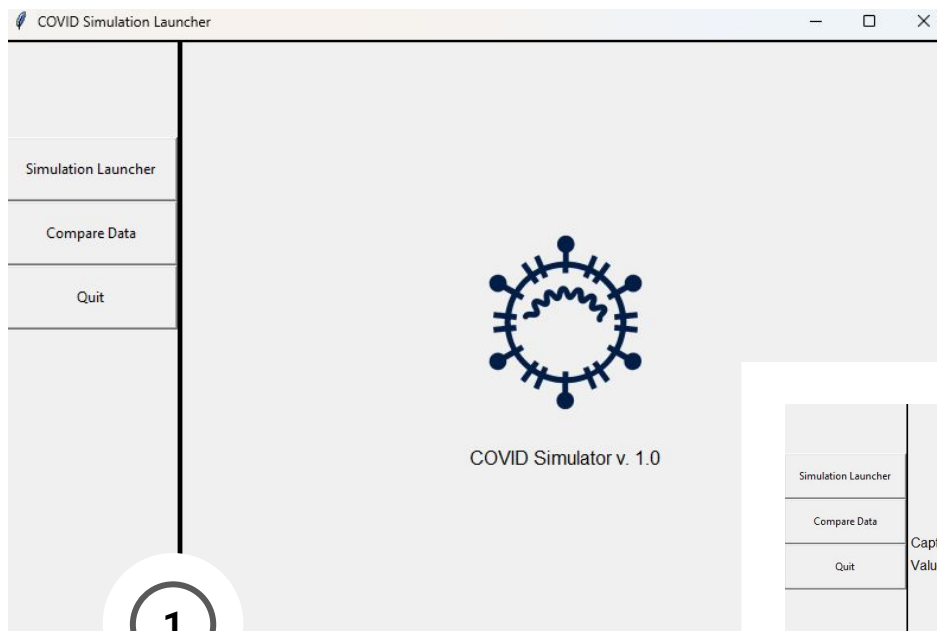
### Technologies

- NodeJS / ExpressJS
- PostgreSQL
- Python / Pygame
- HTML / CSS / JS

5



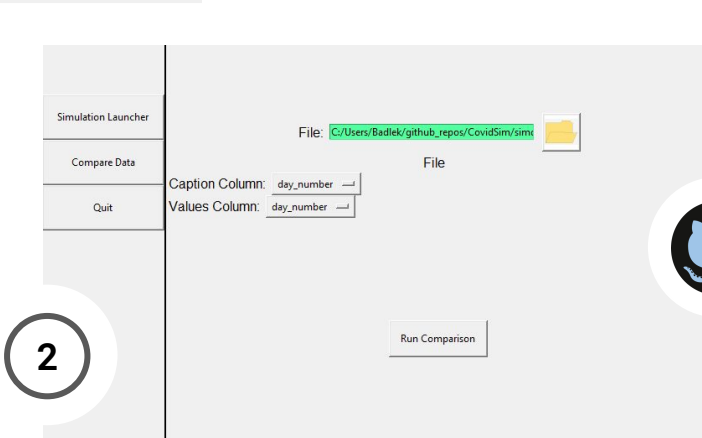




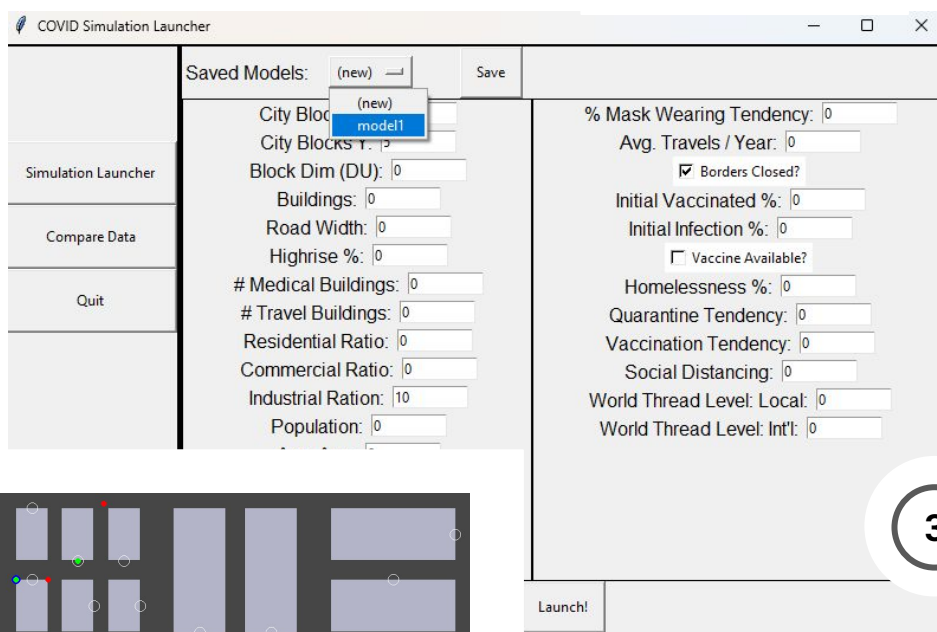
1

## Purpose

Four-person first year University computer science project with a theme around Covid. The app runs an infection simulation for an arbitrary city defined with customizable parameters.



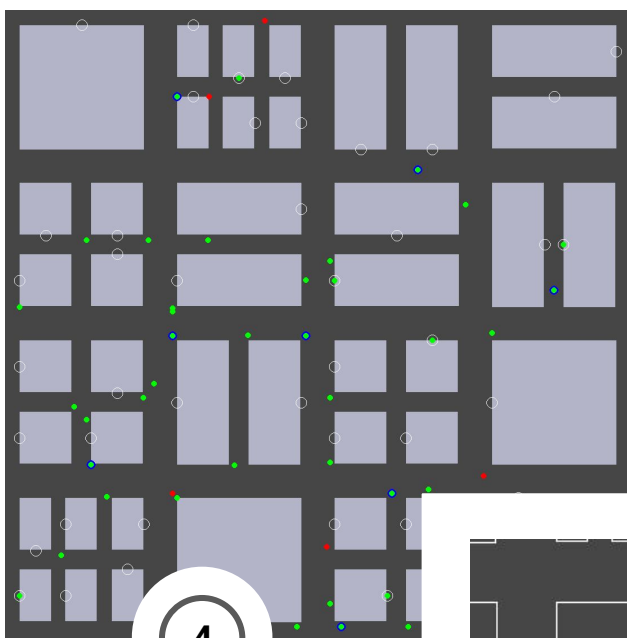
2



3

## Captions

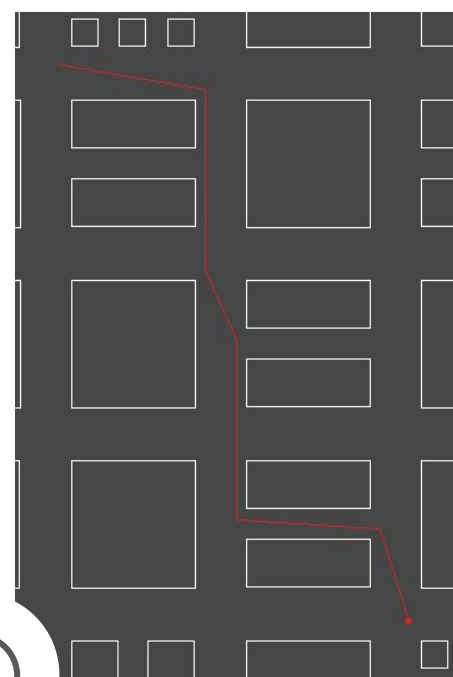
1. Tkinter launcher GUI
2. Load CSV files to view statistics via Pandas
3. Simulation configuration interface
4. Simulation view, dots are people with red as infected and blue as vaccinated
5. Custom-built path finding algorithm



4

## Technologies

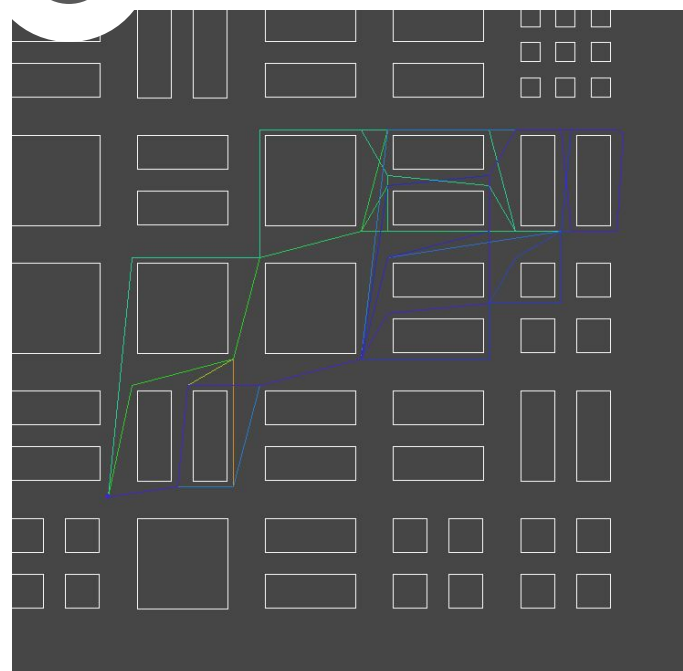
- Python
- Pygame
- Tkinter
- Pandas
- Numpy



5

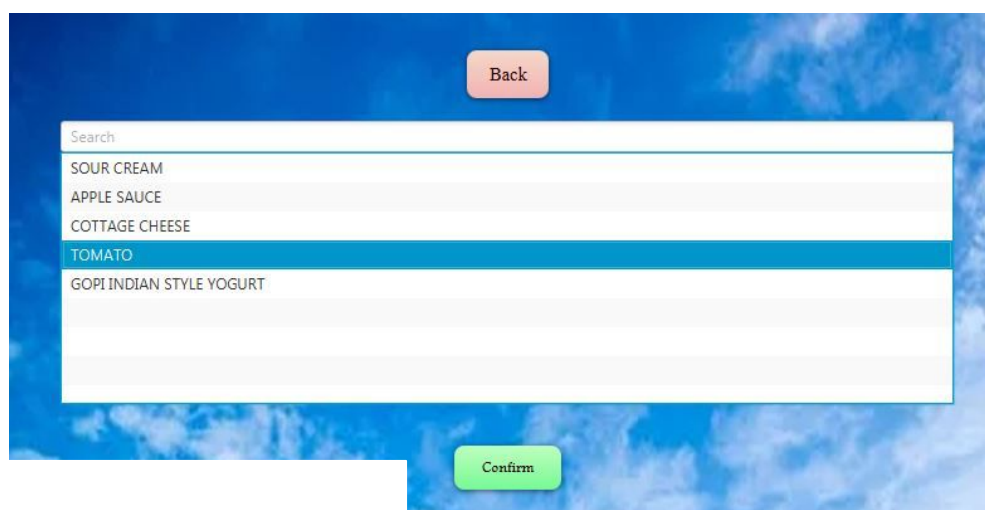
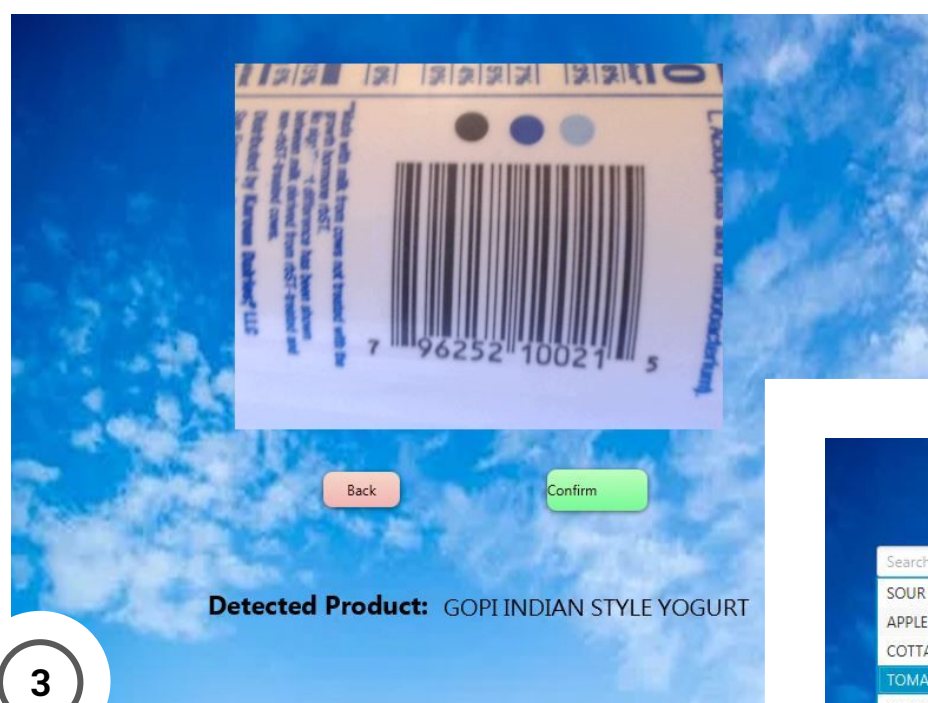
## Contribution

- Designed a recursive **pathfinding** algorithm to navigate between an arbitrary set of quadrilaterals
- Using **probability** and **calculus**, wrote core simulation logic to compute citizen infection using time integral approximations over distance
- Wrote logic to generate the city
- Wrote majority of the pygame UI for visualizing the simulation



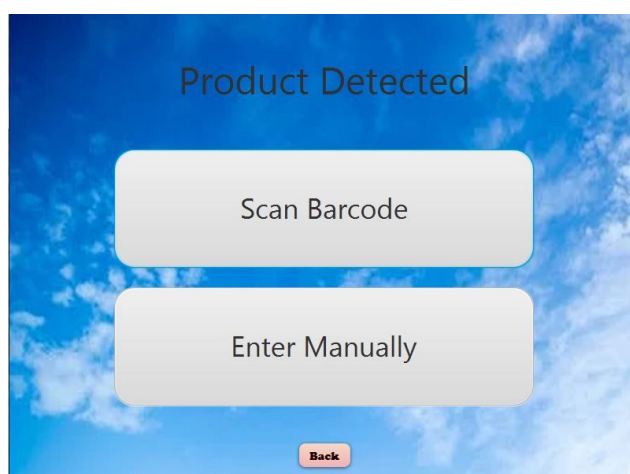


**1 Purpose**  
 A fridge management app developed for a hackathon project (UofT Hacks) with one partner. The app is a proof of concept for an embedded software that would run on a smart fridge allowing you to scan products as you put them in to track expiration dates and suggest recipes



## Captions

1. Main screen displaying products
2. Manual expiration entry if not deducible from vision code
3. Barcode product detection



## Technologies

- Java / JavaFX
- Google Zxing
- OpenCV



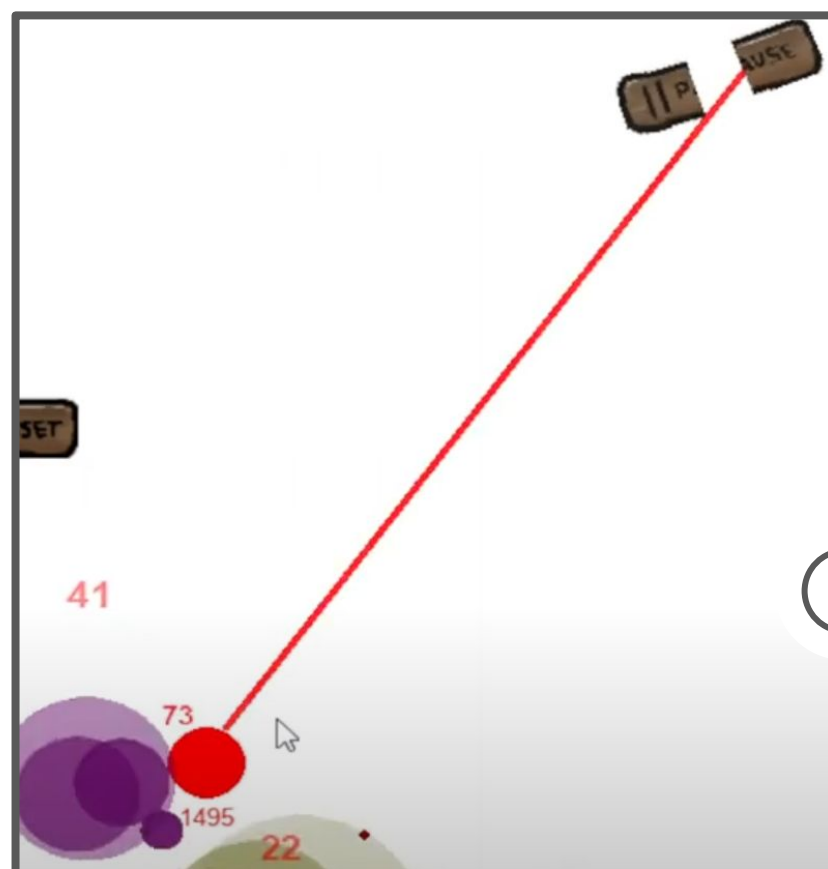
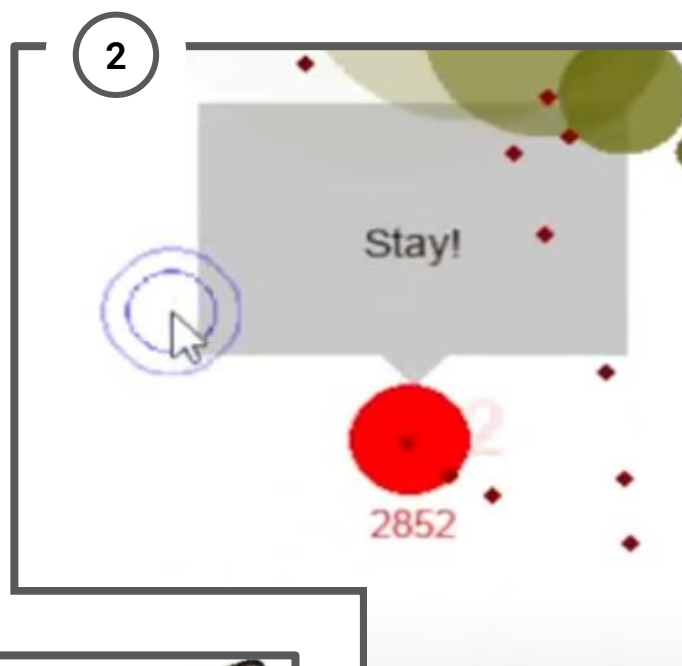
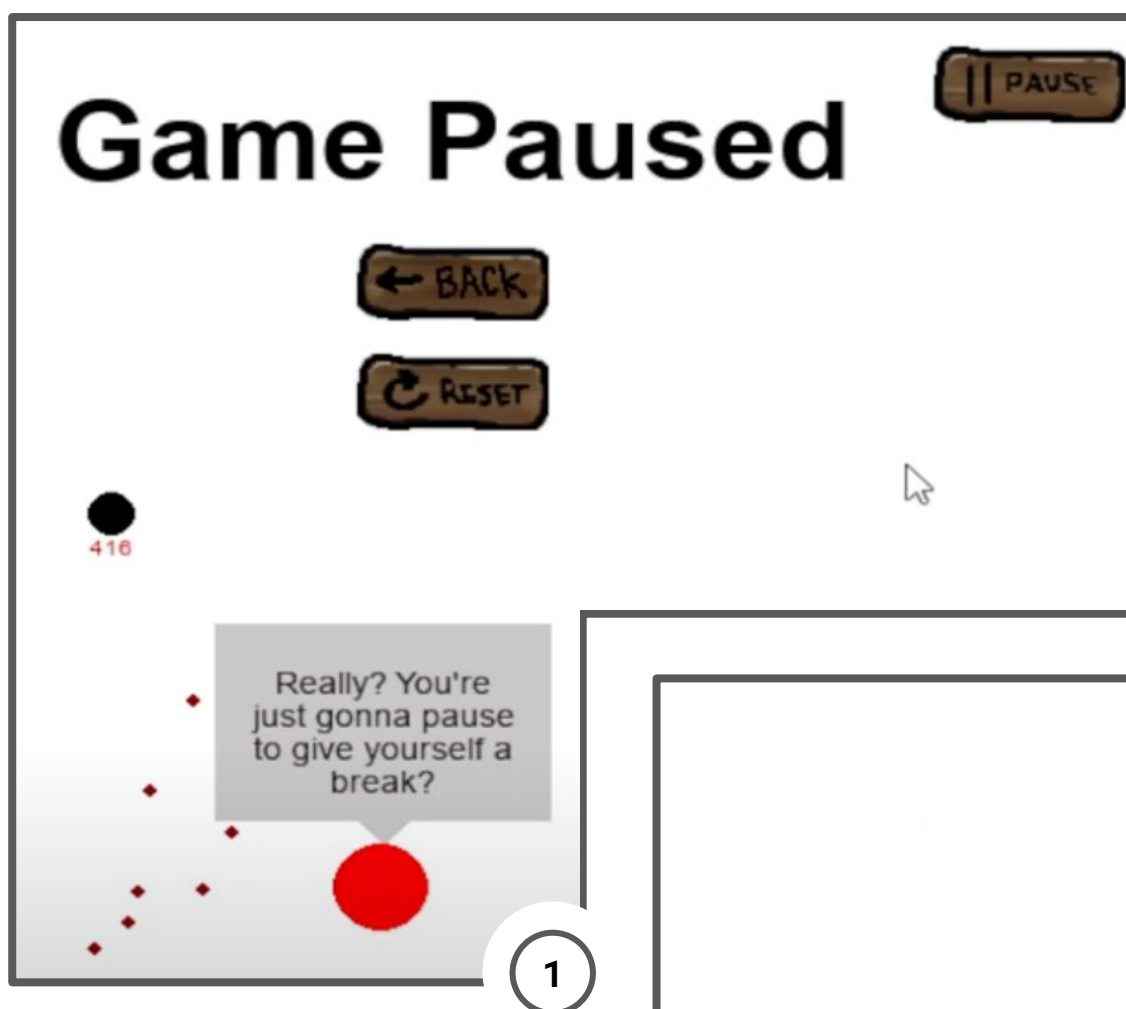
## Contribution

Wrote approximately 60% of the UI, and worked collectively with partner to solve the computer vision component of detecting barcodes and reading expiration dates



[AlekseyPanas/AdvancedFridge](https://github.com/AlekseyPanas/AdvancedFridge)





### Technologies

- Python
- Pygame
- Photoshop



### Purpose

A two person mini project created together with PyUnityd Co-president as a coding example and inspiration source for PyUnityd club members during a hackathon themed after Breaking the Fourth Wall

The game involves an arcade-style boss fight where the boss uses the pause and reset menu buttons to their advantage. The boss also occasionally traps your mouse pointer in a specific location

### Contribution

Wrote the entire game with the exception of the boss' shooting AI. Also designed and created button assets in Photoshop

### Challenges

Using Pygame meant that the speech boxes, pause menu, and other key features had to be implemented from scratch.

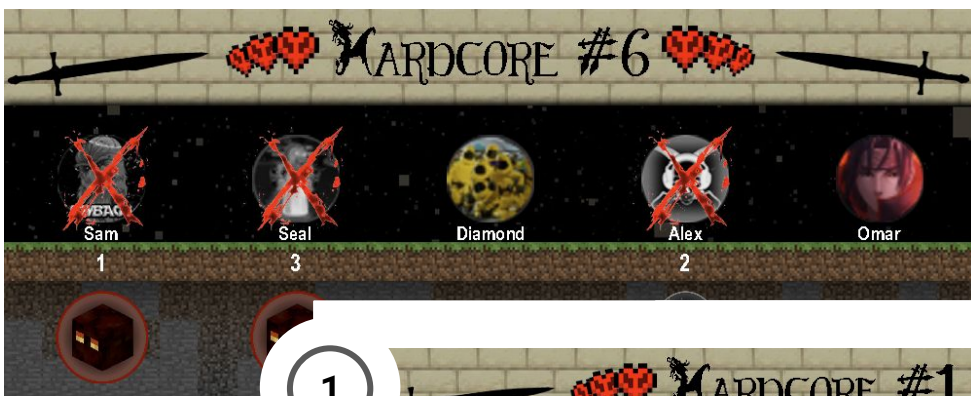
### Captions

1. Boss aware of game being paused
2. Boss traps mouse cursor
3. Boss destroys pause button
4. Boss knocks reset button out of pause menu so that the player has to avoid it



# Various Small Projects

2019 - 2022



```

{"hardcores": [
  {
    "players": [
      "Joshua",
      "Alex",
      "Sam",
      "Ben",
      "Kevin",
      "Emmit",
      "Mechanic"
    ],
    "deaths": [
      {"player": "Sam", "cause": ["creeper"]},
      {"player": "Ben", "cause": ["creeper"]},
      {"player": "Kevin", "cause": ["phantom", "fall"]},
      {"player": "Emmit", "cause": ["fall"]},
      {"player": "Alex", "cause": ["creeper"]},
      {"player": "Mechanic", "cause": ["tnt"]},
      {"player": "Joshua", "cause": ["tnt"]}
    ]
  }
],
"players": [
  "ItsYaBoiSchnozmo",
  "Kevin",
  "Sam",
  "Joshua",
  "Anson",
  "Alex"
],
"deaths": [
  {"player": "ItsYaBoiSchnozmo", "cause": ["creeper"]},
  {"player": "Kevin", "cause": ["lava"]},
  {"player": "Sam", "cause": ["creeper"]},
  {"player": "Joshua", "cause": ["lava"]},
  {"player": "Anson", "cause": ["lava"]}
]
    
```

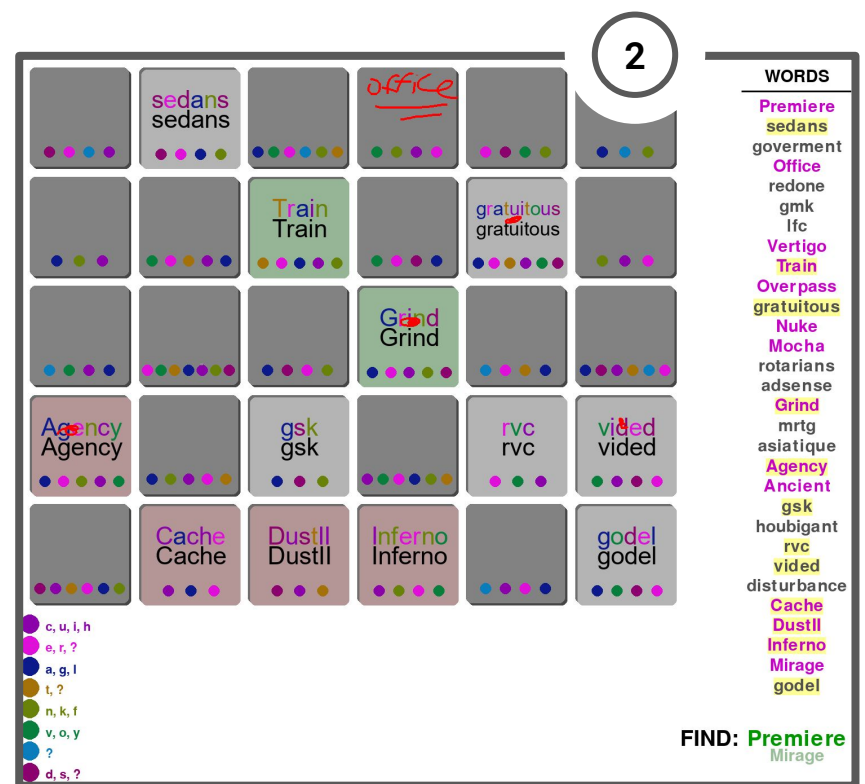


## Purpose





Various small solo Pygame projects completed over the years

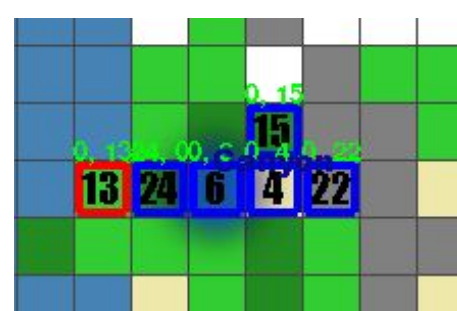
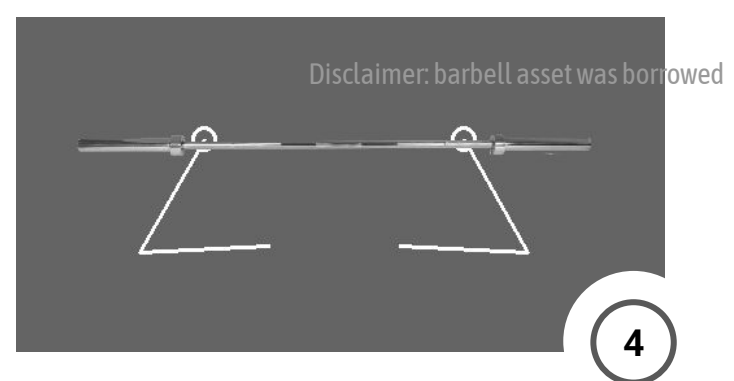
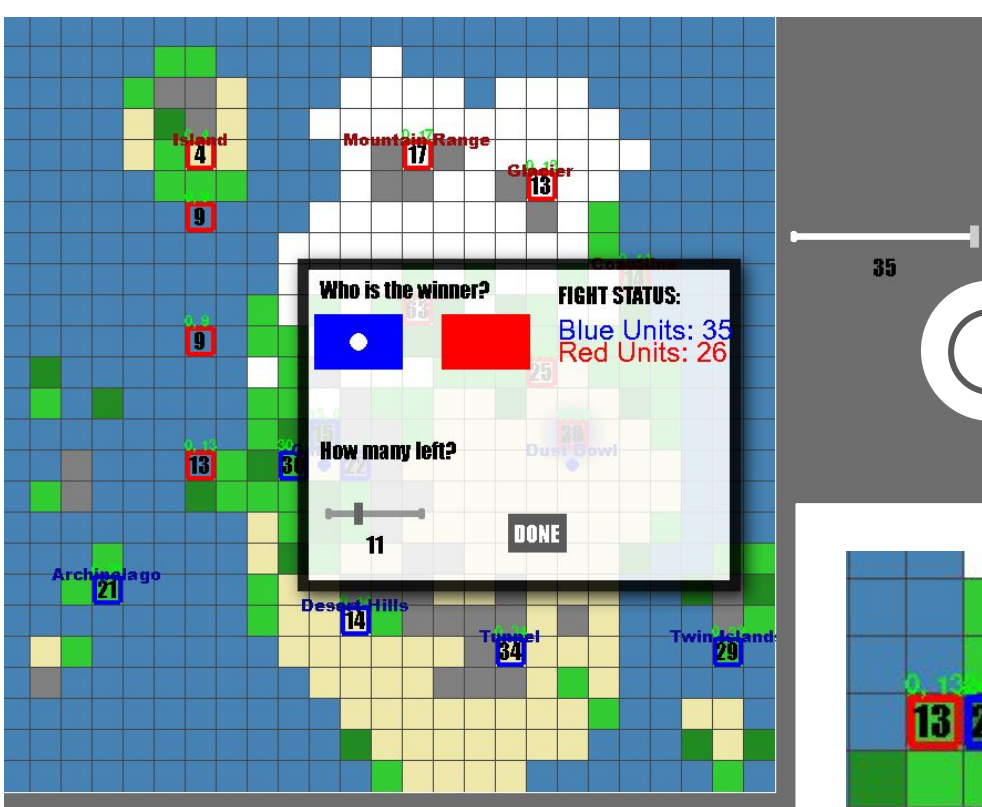
## Captions

- [2020] Converts JSON death logs from Minecraft Hardcore playthroughs hosted with friends into a visualization (using **OpenCV** for image filters and **Pygame** for drawing and exporting)
- [2022] Puzzle game initially created to make map choosing more fun when playing Counter Strike with friends. Purple words from the word list cycle at the bottom right with each revealed tile and will turn green if revealed while prompted. Colored dots help indicate letters and their order for hidden word tiles.
- [2019] A small top-view turn-based combat game where you control units and upgrade bases
- [2020] 2D inverse kinematics where the arms deform to reach specified locations.



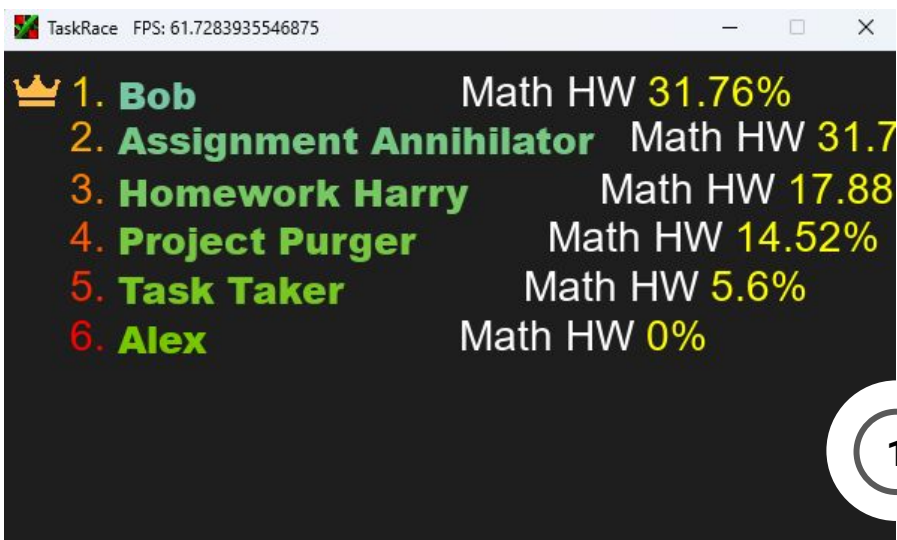
## Technologies

- Python / Pygame  
- JSON 
- OpenCV 



- [AleksyPanas/Hardcores](#)
- [AleksyPanas/RavenfieldSim](#)
- [AleksyPanas/LiftingSimulator](#)
- [AleksyPanas/CSMapChooser](#)

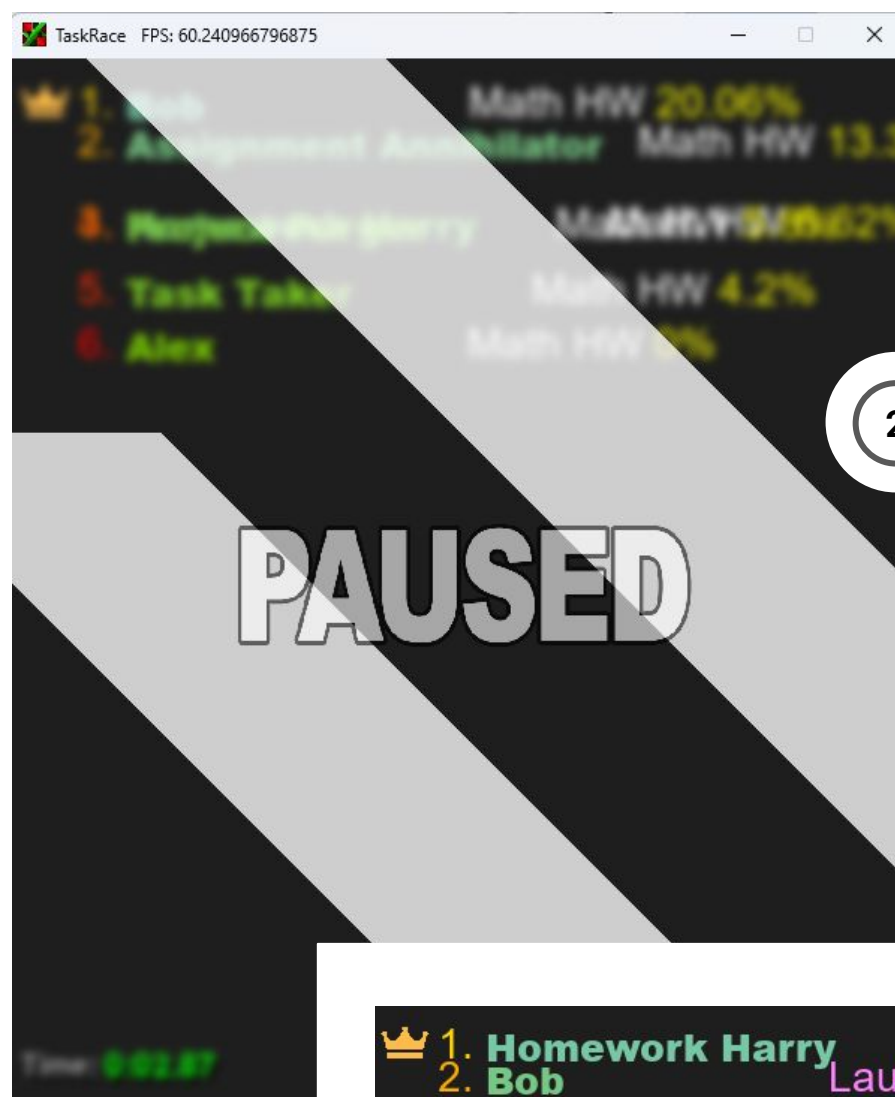




## Purpose

Implemented as a solo project based on an idea from childhood, TaskRace helps motivate doing mundane tasks by turning it into a race.

After inputting a set of tasks and their estimated time of completion, the app launches with a set of opponents who are supposedly doing the same tasks in some alternate universe, and you are competing against them. The premise is to get chores done faster (*works from experience*)



## Captions

1. Main app screen showing active percentages
2. Blur effect on pausing
3. Animated position switching



## Technologies

- Python
- Pygame
- PIL (Pillow)



## What's Ahead

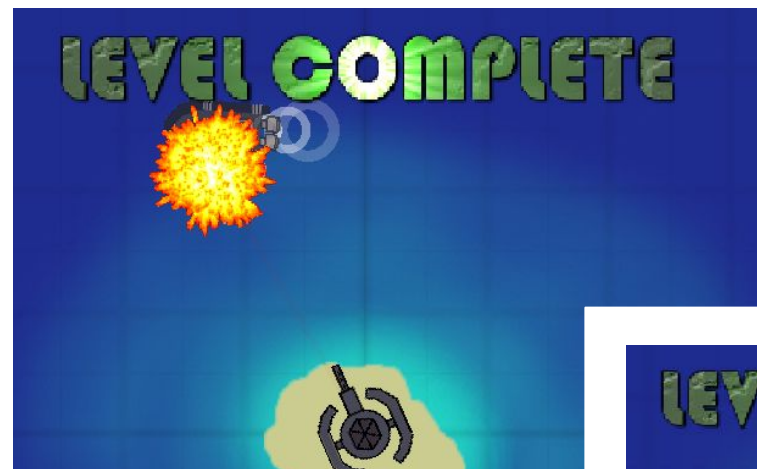
While I have already extensively used this app personally, there are plans to expand it to polish it as a finished product:

- Port to a web app
- Add a more sophisticated settings menu for configuring tasks and opponents
- Add matchmaking to go up against real opponents who have a task list of similar estimated length
- Add auto time estimation for common tasks

## Purpose

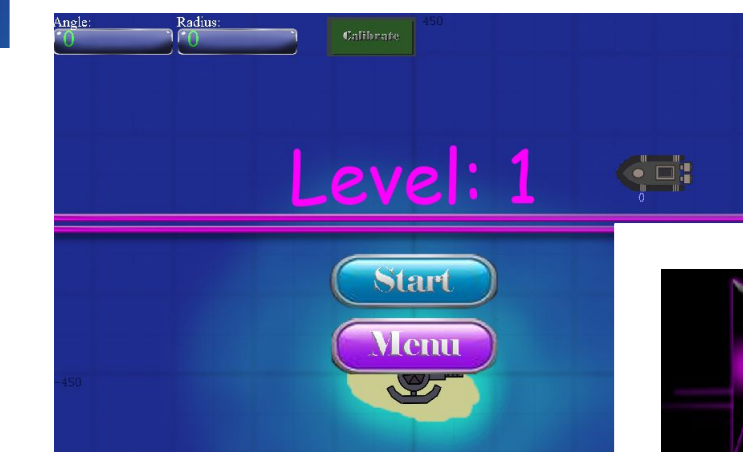
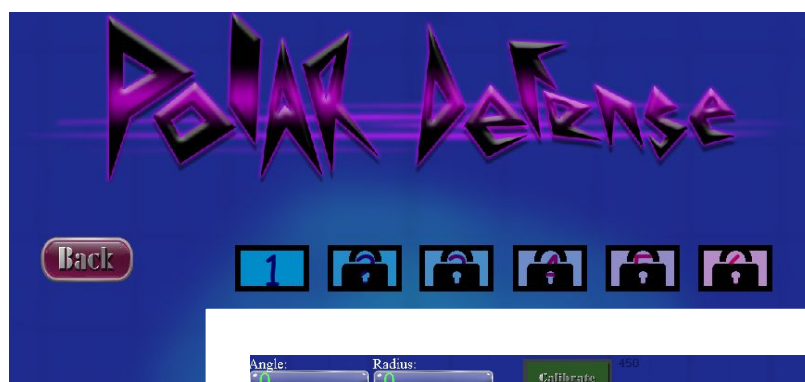
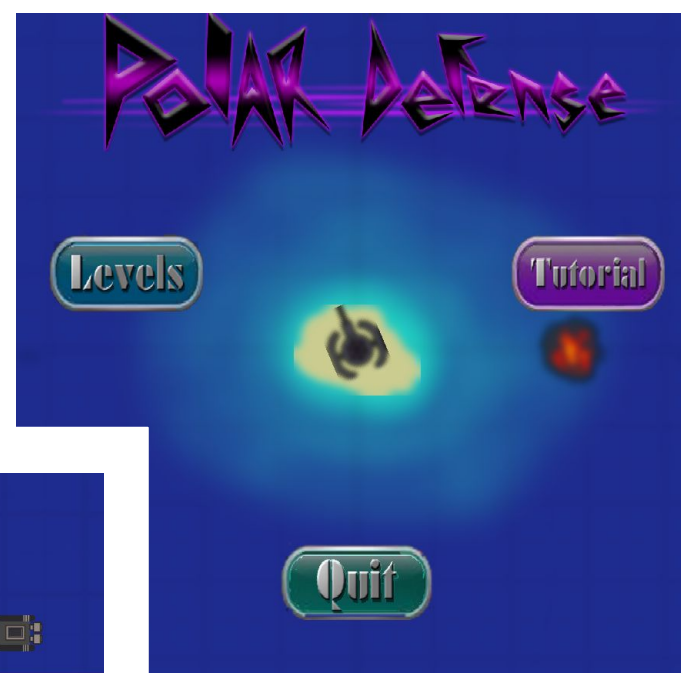
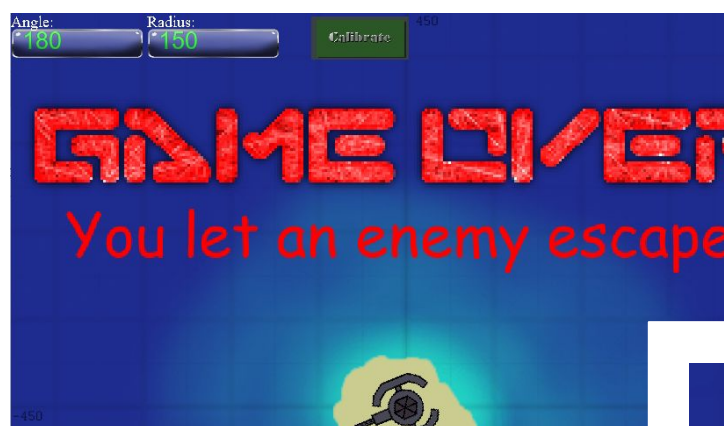
Submission for an open-ended extra credit assignment for a calculus class in high school with a task to develop an educational piece of media on a subject from the curriculum

Choosing polar coordinates as the topic, this small game was developed, requiring the player to convert cartesian coordinates to polar to fire the turret at various targets.



## Technologies

- Python
- Pygame
- Photoshop



## Contribution

Wrote all the game code while a partner created the boat, turret, and buoy assets.

The remaining assets, including the tutorial screen, were done collectively



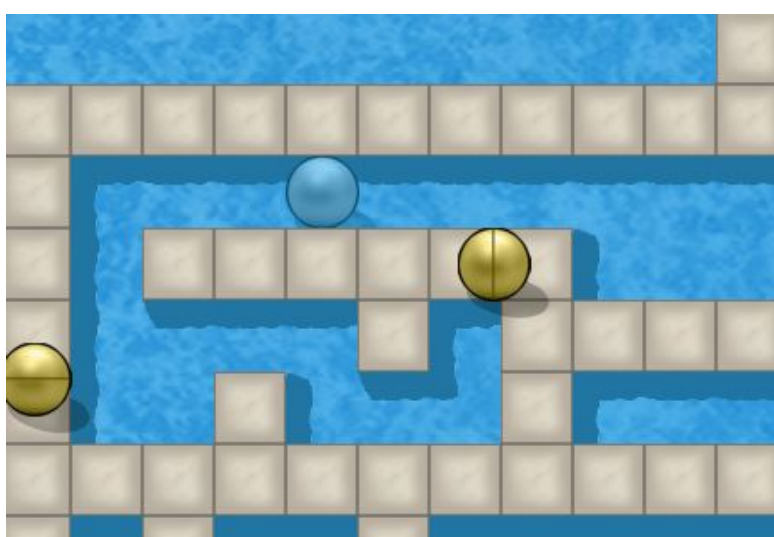




### Purpose

Clone of a Miniclip flash game called "SilverSphere" developed as a birthday present for my mom since she enjoyed that game and could no longer play it once flash got taken down.

This solo project features new levels and a simple level editor over the original game



Disclaimer: Majority of assets were borrowed from google images or the original SilverSphere Miniclip game

### Technologies

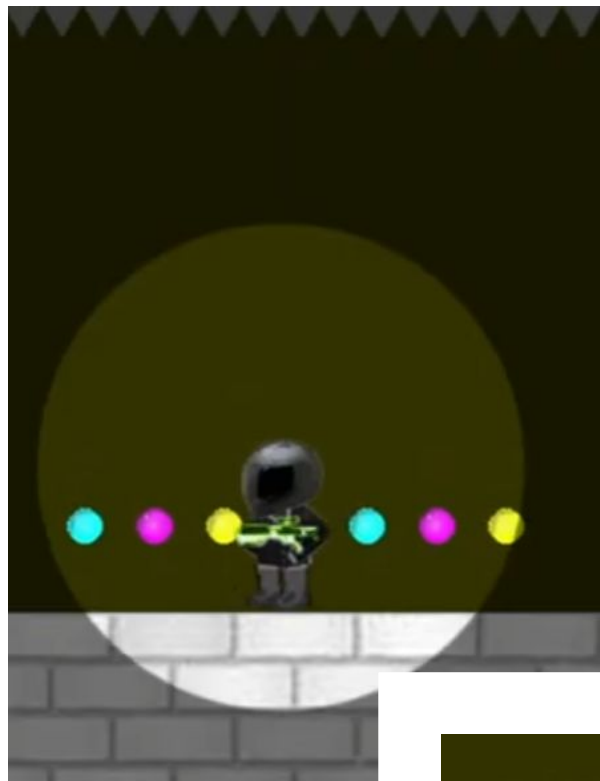
- Python
- Pygame
- Photoshop





### Purpose

Clone of a popular miniclip flash game called "Bubble Trouble" created as a solo personal project because I beat the original game and wished to make additional more challenging levels



### Technologies

- Python
- Pygame
- Photoshop

