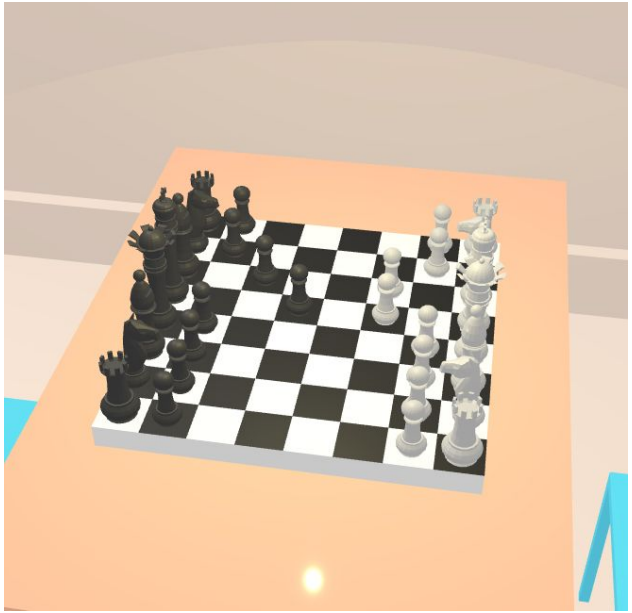
Chess Simulator

Authors: Daniel Cunningham, Zachary Rose, Zach Coomer, Colin Douglas

Our Project

- Create an interactable real-time 3D environment in WebGL which we decided to create a 3D chess environment

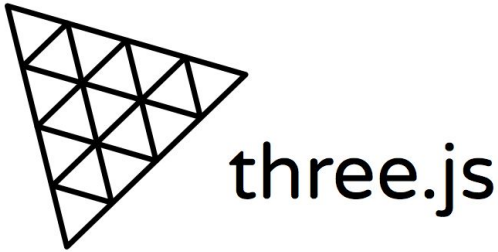


Original Goals

- Chessboard skeleton, where we create a static 3D chessboard and put pieces on the board
- Enhance the chessboard, add lighting and 3D model pieces
- Implement free camera controls, we also decided to allow for a user interface to allow for extra camera view options
- Implement chess rules, extra lighting, and AI

Libraries that we used

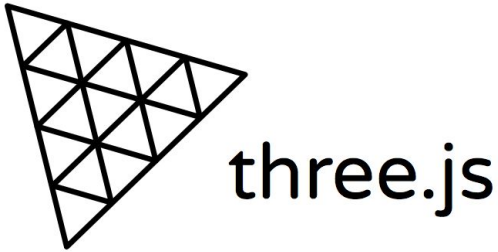
- WebGL
- Three.js
- The Stockfish API



WebGL

- Based On OpenGL
- Designed to leverage GPU of computer
- Capable of running in all commonly used modern browsers

Three JS

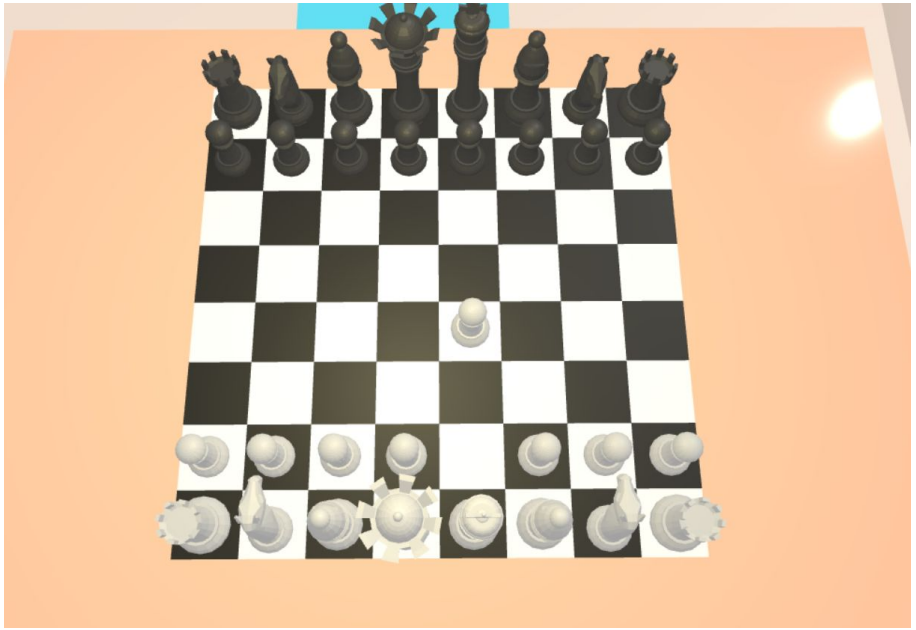


- “Higher Level” Cross Browser Library and API
- Allows for easier creation of 3D computer graphics in a browser environment while using WebGL

Stockfish Online Rest API

- Simple REST API that is accessed through an interface
- All that is needed is a FEN String, difficulty setting, and mode setting. There are three difficulties:
  - Easy - <1800
  - Medium - ≈1800
  - Hard - 2231
- Stockfish difficulties are taken from the research paper by Diogo R. Ferreira at web.ist.utl.pt/diogo.ferreira/papers/ferreira13impact.pdf

Example



FEN String

rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR b - - 1 1

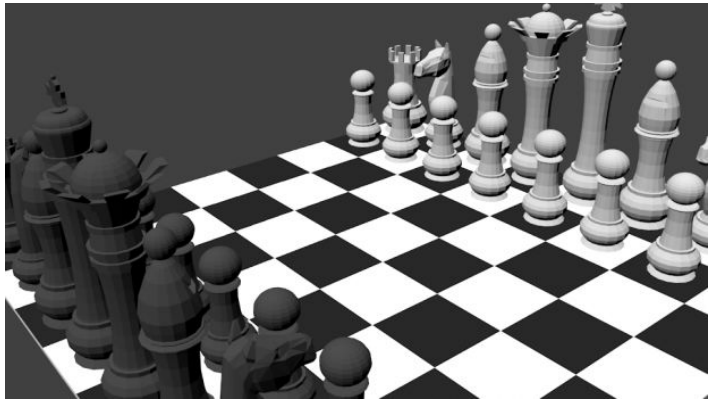
Request

https://stockfish.online/api/stockfish.php?fen=rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR%20b%20-%20-%201%201&depth=1&mode=bestmove

JSON Response

{"success":true, "data":"bestmove d7d5"}

Loading in Objects



- Firstly initializing backend features such as the array for board state
- Utilizing GLTF Loader from Three.js in order to load in objects and environment
- GLTF Loader allows for fast load times and easy transmission as it is optimized for asset delivery at runtime
- Adding Each Piece to the Backend board as we load it in and preparing it for use in the game
- Pieces obtained from: https://free3d.com/3d-model/low-poly-chess-black-white-48044.html

Piece Movement

- Local coordinates vs. world coordinates
- Raycasting
- Chessboard coordinate mapping
    - Prevent disarray on every axis
- Putting it all together

User Interface and Selection Avoidance

- Drop down menu that implemented cameras and game function tie ins
- Implementing logic that ties into the piece picking function to allow for only specific items to be selected and consequently interacted with

Backend Representation and Taking

- Each piece is a 3D Object mesh
  - Properties
- Some important arrays:
  - Pieces and Taken
- Updating the visible board

Now Onto the Demo!

Challenges

- Originally the Models were not appearing - We corrected lighting mistakes
- The Room and chessboard itself was originally selectable - In order to prevent this we instituted restrictions on what was selectable to just the pieces themselves
- Pieces originally were not moving correctly

Future Work

- Implement special chess moves such as: En Passant, Pawn Promotion, etc..
- Implement different types of board games
- Implement collisions and physics to allow pieces to be pushed and moved
- Implement animations
- Implement soundtrack