[LINK TO GITHUB REPOSITORY](https://github.com/Eliaznizzle/4DChess)

^ PROJECT FILES ARE HERE ^

Plan

* Develop a set of rules for Four-Dimensional Chess
* Develop software that allows players to play Four-Dimensional Chess against each other over an internet connection
* Study how adding dimensions to a game affects the process of programming and playing it

08/01 2020

Promotion and castling have yet to be implemented, but the game is otherwise perfectly playable.

The UI can use a lot of work. I’m currently learning to use adobe illustrator to create some basic UI elements. I hope to illustrate my own chess pieces by the deadline but I might have to end up using someone else’s. Remember to give credit, me!

Netcode is implemented. There are currently no failsafes if someone disconnects during a game, but play between two peers works flawlessly. For debugging it is however currently disabled in the current build, and launching it will not show the networking interface but rather just immediately launch a version of the game that lets the local player take infinite turns. To reenable networking functionality, set singlePlayerTest to false in the Manager component of the “Manager” game object.

09/01 2020

Looking at creating some sound effects.

Looking at <http://labbed.net/software/labchirp/>

Turns out that’s not something I want to do at all

I’m going to use this <https://freesound.org/people/mh2o/sounds/351518/>

It is in the public domain

Messing with postprocessing

Adding a vignette effect does a lot

Might use bloom and chromatic aberration to create some neat transitions

[https://www.freevector.com/chess-icons-20806#](https://www.freevector.com/chess-icons-20806) is free to use if I give credit in the form of a link to the main site

11/01 2020

The graphics from freevector.com ended up being problematic due to their non-square aspect ratio. The current plan is to modify some of the graphics from <https://commons.wikimedia.org/wiki/Category:SVG_chess_pieces>

Pre-logs, I implemented a funky effect. When a piece is captured, it flies off to the side using Unity’s physics engine. When it hits the edge of the screen I want it to bounce and stay within the visible game’s borders. The presence of a camera is not very important since it never moves, but it is used to dynamically change the view depending on the user’s monitor aspect ratio. I am currently writing a script that controls four colliders to automatically move to the edges of the camera.

13/01 2020

The sprites from Wikimedia commons have been modified and implemented. I didn’t like the look of the black pieces so I duplicated the white ones and inverted them.