CS 174A Final Project – Hypercubes

Team AR
Ryan Adisasmito-Smith, 505164357, drraumirr@gmail.com, GH: Ssirrikh
Venkata Sai Sriram Sonti, 904599241, sriramsonti1997@gmail.com, GH: Sriram-Sonti
Ayush Patel, 204822908, ayushp@g.ucla.edu, GH: ayush-patel

Description:

Our idea was to create a space where users could interact with 4-dimensional objects (specifically, hypercubes) to help them visualize and build an intuition for how 4-dimensional objects work. Users would see an ordinary 3-dimensional cube and a hypercube next to each other and would be able to control the rotations of the cubes, the lighting, and the placement of the camera. We intend for our webapp to be used as a learning tool.

How to use:

Upon opening the web app, the user sees 3-d wireframes of an ordinary 3-d cube as well as a 4-dimensional hypercube rotating.

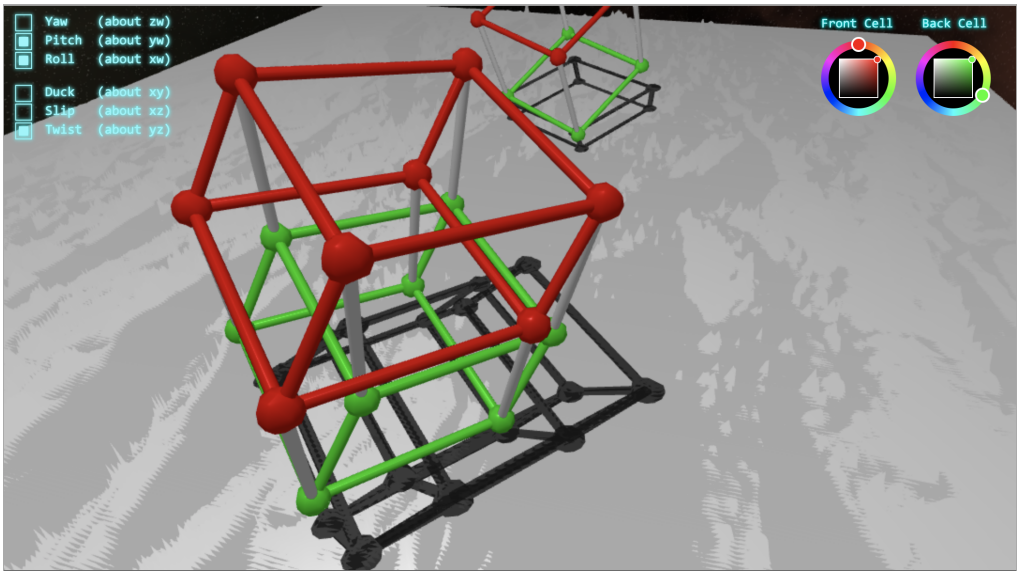

Fig 1. 4-dimensional hypercube rotating about yw, xw, and yz directions.

With the controls on the top left, the user can control the rotation of the two cubes about the mentioned axes. The rotations about x, y, z, and combinations thereof apply to both cubes but rotations involving the fourth axis, w, apply only to the hypercube.

As can be seen from the image, the 3-d cube has two colored faces – green and red by default. The hypercube similarly has two cube-faces. The colors of these faces can be controlled from the top right. The user can also press n to switch the colors to white. Pressing n again switches the colors back to what they were before.

Pressing m stops the rotation in place. Pressing n again re-starts the rotation.

The user can press b to turn the solid wireframes to ordinary wireframes, as seen below:

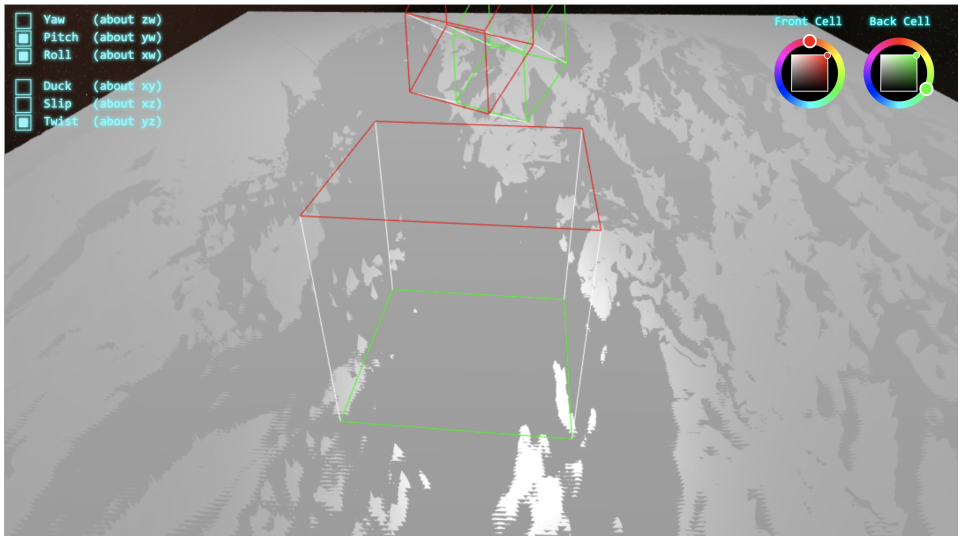

Fig 2. 3-dimensional wireframe cube

At the center, the user can see a spherical light source. It's movement can be controlled with the following keys: y to move it up, h to move it downwards, g to move it left, and j to move it to the right by 1 unit. Further, by pressing q the user can change the color of the light to a randomly generated color. Pressing q will change the color back to white. Pressing it again will generate a new random color for the light. By pressing l, the user can induce a bloom effect in the light source.

Beyond these, we have the standard movement controls used for this course. Spacebar to move the camera up, z to move it down, w for forward movement, s to move it back, and s and d for, respectively, left and rightward camera movement. The user can also hit , and . to roll the camera left or right, respectively.

Implementation of hypercube and its rotation

The math behind the hypercubes involves quaternions, which consist of a real part, w, and imaginary parts x, y, and z. For the quaternion math, we relied on a library found online, which came with functions for performing operations like addition, scaling, dot product, invert, and rotate on quaternions. In hypercube.js, we first create a cube and hypercube – they are just expressed as a data structure consisting of 8 and 16 vertices, and 12 and 32 edges respectively. We also keep track of which vertices and edges are to be considered front vertices/edges and which are back vertices/edges. This is to help us color code them later on. Then, based on which axes of rotation the user has selected, we rotate the cube and hypercube. Our functions for rotation of hypercubes (specifically, for calculating coordinates of vertices after rotation) were adapted from the aforementioned quaternion library.

Advanced features – Light source, skybox, and shadow mapping

For our advanced feature we wanted to show the effect had by light on the cube/hypercube and what shadows each shape casts on its surroundings. We chose this to give the users a more immersive look into how the shapes behave while rotating.

The spherical ball of light consists of a subdivision sphere with a source of light inside it. The sphere and the source of light start off at the origin to begin with, and when the user hits any of the y, g, h, or j keys, they move together because their position is defined by the same variables (which start off with value 0). Similarly, their color is defined by the same variable (starts off white and then changes to a random value by using Math.random()%255 when the user presses q. For the bloom effect, we defined a new class called BloomEffect in hypercube-dependencies.js which implements, in the glsl_fragment() function, a Gaussian blur applied over the light source. However, we were not able to get the implementation to work in time. We tried alternate approaches, including by rendering just the light in a second scene, blurring that second canvas, and then drawing it on top of the main scene. Unfortunately, that approach did not work either.

We used a skybox to give a visually continuous surface around and top of the hypercubes. This adds in to the futuristic vibe that we were aiming for. This is done by having 5 thin cubes around the hypercubes (top, right, left, back, and front).

For the shadow map, we worked off the mapper used by TA Jonathan Mitchell in his project. The shadows fall on the displacement map at the bottom.

Graduation feature – rocket flying with banner (work in progress)

For the extra feature for graduating seniors, we will implement a rocket ship in the background of our scene that will fly past with a banner that has a congratulatory message.

Github Code Repository

Our code can be found at https://github.com/ayush-patel/CS174A.

Team member contributions

Ryan Adisasmito-Smith – implementation of the cubes and their rotation including the quaternion library, as well as scene set-up
Commits:

- Initial code including quaternion math and scene set-up: https://github.com/ayush-patel/CS174A/commit/d451dcfffa611a994bb46ec3e0f2b8f778d3122c (Note: the code in this commit was written entirely by Ryan Adisasmito-Smith, even though Ayush Patel committed it to GitHub)
- Added dynamic recoloring and displacement map: https://github.com/ayush-patel/CS174A/commit/761f96787fb331791c162fa9e9ecb1360c531a5a
- Simplifications to the quaternion math: https://github.com/ayush-patel/CS174A/commit/265e09fbb789910dbe8d2e908d5da4834c554c45
- Replaced the default camera set-up with orbit controls that allow us to view from different angles: https://github.com/ayush-patel/CS174A/commit/1e7fa892b8a841e3654b9d1a86a91dc013ce2db5

Venkata Sai Sriram Sonti – implementation of the movable color-changing light source, and blur effect (implementation did not work), readme, and report

- Added a stationary light source: https://github.com/ayush-patel/CS174A/commit/76dbcb426fe1724c7fafb62204dafc53a4adc903
- Made light source moveable: https://github.com/ayush-patel/CS174A/commit/82866b5d2743ec3f01cc1d2b3bc75ab0eb028503
- Code for Gaussian blur applied onto the light source (incomplete implementation): https://github.com/ayush-patel/CS174A/commit/857771da0859937814810c4152f15e7607202c0e,

https://github.com/ayush-patel/CS174A/commit/99b56a8f3e0c6c276f672c7db579766943b5c313

- Made light capable of changing to random color upon pressing ‘q’: https://github.com/ayush-patel/CS174A/commit/9eb0ea352cc9495ed86006bd6a80bdba107358f8
- Added readme (also wrote the project report): https://github.com/ayush-patel/CS174A/commit/822dfbf996c894b098e9be34fb45c3361ec470b4

Ayush Patel – implementation of shadow mapping, alignment, add skybox surfaces around the cubes, mid-point presentation

- Added surfaces (thin cubes acting as planes) around the hypercubes upon which the shadows could fall: https://github.com/ayush-patel/CS174A/commit/6e8706e089336fd43e389c059f65b8895cc3139d
- Added switches to allow light to move in z-direction as well: https://github.com/ayush-patel/CS174A/commit/c71e5ec43bdb87f8a2b8e9afea17eccad04ebdcf
- Integrated background boxes with displacement map: https://github.com/ayush-patel/CS174A/commit/4ac1d926349c0f6c861174673e0c8f612e0e568e
- Integrated shadow mapper into our scene: https://github.com/ayush-patel/CS174A/commit/e3a28dabf53a1f007f8bb3f617cc70fc87fd13ee, https://github.com/ayush-patel/CS174A/commit/ed1907e13b080e45a78cddf7695c974f5ce5a71a, and https://github.com/ayush-patel/CS174A/commit/f72e0fdfd0661814e422fbcea7e5ee564336d842
- Background images (galaxy sky box) integrated and aligned. Camera perspective aligned. https://github.com/ayush-patel/CS174A/commit/4a7ea1a00f593be63af62fd33fbc156b028e375a