Creating a simulator for the 2x2x4 Tower Cube

John-Michael Jeny Jeyaraj

December 2024

1 Introduction

The Rubik's Cube is the world-renowned 3D mechanical puzzle invented by Erno Rubik in 1974. This invention caused a massive impact on an international scale and catching the interest of many toy making companies. It was initially called 'Magic Cube' and used to teach about 3D geometry, before officially being patented in 1975. It is now considered the world's best selling toy [7], it won a German Game of the Year award [2] and many other awards in the UK, France and US.

It has entranced mathematicians and speedcubers alike, achieving the peak of its popularity in the 1980s, it still remains in the back of the mind of the general populace, even with the rise of mobile phones and tablets which made most toys completely obsolete. It was even touted as top 100 most influential invention.[9]

The complex and multifaceted nature of the Rubik's Cube and its use in innovating problem solving are the main motivator for this research. Many researchers have spent countless hours studying the secrets in this seemingly simple cube, but it's a gift that keeps on giving. Exploring the Rubik's Cube secrets comes with all the complicated bells and whistles of group theory. A challenge I am willing to take.

2 State of the art

2.1 Applications of the Mathematics

A well known article in the Rubik's cube research space is the 'Overview of Rubik's Cube and Reflections on Its Application in Mechanism' [11]. This article offers a surface level overview of the Rubik's Cube as more than just a puzzle toy; it has been adapted as a structural and functional model across various fields, including data encryption, robotics, and mechanical design. They have pin pointed multiple fields would be impacted significantly thanks to research on how to implement the maths behind the Rubik's cube characteristics. Evidently they only did surface level examples, however even if the application of the

group theory doesn't revolutionise the field, it would still be a new tool for scientists to use to deal with different situations. They give a good case study on why studying the complex maths involved with the Rubik's Cube could inspire scientists of different fields. The cube has very strict properties and algorithms which can studied to showcase the utility of group theory.

The Rubik's Cube's rotation patterns provide an analogy for complex encryption techniques, which could supplement already existing data encryption and security measures. By treating the cube's rotational steps as an encryption key, data can be scrambled and later decrypted by reversing these steps. The article mentions the use of chaotic maps and shifting algorithms to encrypt images, demonstrating the versatility of the Rubik's Cube model in cryptographic applications, where it acts as a non-linear, high-complexity system. This application leverages the randomness of rotations and permutations, making data more secure and challenging for unauthorized access. They didn't go into much detail if this method would be better than already implemented security measures, however this technique is definitely useful to give a different view of how to tackle cybersecurity problems for experts in the field.

There is also the emerging applications in Mechanical and Aerospace Design which can use the structure of the cube as a model for devices ranging from password locks to satellite systems, capitalising on its modular and configurable nature. In aerospace, it enables novel designs for deep-space exploration craft by leveraging the cube's adaptability and compartmentalization. The idea of using a "deformable magic-square-type" design for spacecraft suggests that the cube's structure may support versatile, compact, and organized space missions, where tasks can be distributed across modular units. Although the Rubik's cube will not be the catalyst for innovation in the space industry, it could serve as a supporting piece of evidence on the usefulness of modular space ships. Deep space exploration could be improved if you consider the utility of a flexible spacecraft that is able to adapt to more situations than our current rockets ensuring the safety of astronauts, which is key to the success in these space missions. Scientist could utilise the research on the movements of the Rubik's Cube to inspire new innovative technologies that depend on complex mathematical ideas from group theory.

The Rubik's Cube serves as a good enough object for testing robotic control systems, especially when coordinated with multiple sensors. It presents a multilayered problem that requires the robot to understand orientation, rotation, and color recognition—essential skills for more advanced tasks in automation and AI. This type of challenge can easily be expanded as required to test if the robot is able to handle complex tasks. The use of brain-computer interfaces (BCI) for solving the cube further highlights potential advancements in accessibility technologies, as it suggests new ways for people with physical limitations to interact with and control external devices. In the future this research would be of great help to people with reduced mobility, as having different ways of testing the robustness and problem solving capabilities of robots is of great importance.

We can use a Rubik's cube as a base level problem, and then expand and stretch the limits of the robot, by using different size and shape cubes. If the robot isn't able to keep up with a Rubik's cube, then it would demonstrate a lack of computing power and the ability of robotic manipulation of objects.

Research on the Rubik's Cube could lead to great innovations in Material Science and Analytical Devices. The cube inspires "equipment-free" methods for creating paper-based analytical devices, already showing a practical application in scientific fields that require low-cost, efficient diagnostic tools. This is an inventive way of taking inspiration from the cube's compartmentalized structure to fabricate simple, yet effective, devices for field diagnostics. This is by far the most useful application of cube, as tools for testing the robustness and analysing the effectiveness of new devices is paramount for innovation. Allowing producers to showcase the fidelity of their new creations and persuading investors with strong mathematical proofs to back up their models. Group theory's greatest asset is it's reliability, and it can be used for a wide variety of testing methods.

2.2 Rubik's Cube Simulators

There are many websites online offering a simulator for the basic NxNxN cubes [1], allowing users to easily manipulate the cube using their cursor. The complexity of their programs is the visual representation of the cube, often done with WebGL or javascript, which allows the user to control the camera and drag parts of the cube to make it rotate. However, it's glaringly obvious these websites avoid more complex cubes, such as the snake cube, mirror cube, or even the 2x2x4 tower cube. They are willing to do 2x2x3 and 2x2x5 and other uneven combinations [3] due to their simplicity. This is because the 2x2x4 configuration, and others like it, belongs to the shape-shifting cuboid family [4]. These puzzles are much more complex than a regular cube, as the puzzle can no longer neatly fit into a grid, more space is needed to allow the full range of motion and positions that is allowed by the tower cube.

The technical advancements made by the visual side of computing have been extraordinary in the past decade. 3D models can easily be represented by modern engines, which have exploded in complexity. An ability to represent a basic 3x3x3 structure with colors, isn't an impressive feat when there are games loading millions of polygons at a time. This isn't to downplay the work in to these online simulators. Even if there are 100s of these websites, it's definitely an interesting project to get your programming skills sharpened in front and back end development. The real challenge when building a interactive 3D model would be making a smooth camera system, rendering the cube correctly, and making high quality animations.

Most of these websites go a few step further than just being able to play with a Rubik's Cube. They may have a puzzle generator to allow intelligent players to train with different starting positions and may also have a leader board feature for competitive players, connecting data from players across the world. The

most interesting development done by online simulators is how ubiquitous the solver has become, because the algorithm for finding a solution for the Rubik's Cube isn't easy. Unfortunately I'm not able to see any of the code behind their programs or I would've checked extensively if their program follows God's Number or if they brute force the answer. They must use an algorithm as the solver can usually produce a solution within mere seconds.

2.3 Tower Cube Simulators or lack thereof

These simulators are few and far between for obvious reasons. Not only is the tower cube much less popular than the generic 3x3x3 Rubik's Cube, but the logic and the code for it is much more complicated. The biggest challenge is its shape-shifting nature, which requires much more difficult algorithms for rotations. Searching through many websites and many different videos to find what the current best simulator for a tower cube can do was a daunting task.

After extensive research online I settled on two interesting applications that go much deeper on the tower cube than a normal simulator: IsoCubeSim [6], which focused solely on the tower cube and, Permpuzzle [5], which had 100s of unique types of puzzles. Both were written in java for easier use of object orientated programming since the puzzle is a lot more complex. I had already imagined that a good simulator for the tower cube would be difficult to program, however I didn't expect how rare and obscures they would actually be. This shows there was a real need for a deeper look into not so popular tower cube, as it has lots of secrets of its own that a NxNxN cube couldn't replicate.

IsoCubeSim [6], is the first one I found. It simulates a tower cube, without any dynamic cameras, and probably uses a pattern focused program to do the rotation logic and then translates it into an image. A pattern focus would convey that program used the notion of stickers, which would be rotated by basic triangular rotations. Unfortunately this simulator has locked the shape-shifting capabilities of the tower cube, instead making it so that half turns of the left, front, right or back faces are completely impossible. This means that the face would make a 180 degrees turn, which has its own unique challenges. However, this program has avoided the difficulty of actually representing a tower cube, and instead opted for a half-baked version that is only able to simulate the tip of the tower cube iceberg. I don't mean to downplay the interesting logic they used for the rotations, as it was useful to see the 4 different moves already enabled by this improvement. Seeing the middle layers of the tower cube move separately from the edges was helpful in understanding how the stickers should rotate.

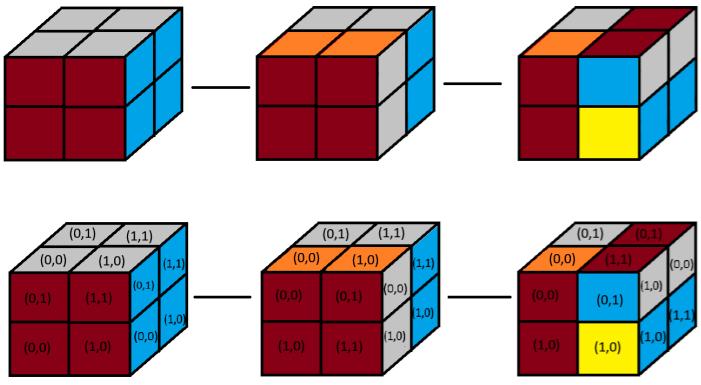
Permpuzzle [5] by Gelatinbrain is an application which has more breadth than depth. It has many unique versions of the Rubik's cube, such as the snake cube, and many different shape-shifting cubes. The simulator has a dynamic camera, smooth animations, and good rendering, making it on par with the best of the online simulators on the visual side. It even shows 2 angles of the

cube for a better observation of how the movements affect the unique puzzle you're looking at. It can take in a multitude of movements at once, and has a scramble feature that has a variable randomness and frequency, which could be used by researchers for finding interesting patterns. On the 3d model it also shows you what kind of movement would be done if you click in that position, this is especially useful because it has a lot of unorthodox puzzles. It has a huge library of puzzles, eclipsing any online simulator by an order of magnitude, each more bizarre than the last. The library is sorted by what type of puzzle it is, such as dodecahedron, snake cube, honey cube, etc... And even further sorted by what kind of rotations can be done on the puzzle, such as face turn, vertex turn and edge turn. Many of these puzzles move in unique ways, and it must've been difficult to program this, since so many of these puzzles couldn't even exist in real life. However, the 2x2x4 cube in its library isn't able to shape-shift. Why would this programmer avoid the real complexity of the 2x2x4 tower cube? I can't even begin to imagine how some of the puzzles in his library even begin to move, but the tower cube can be made in real life so it shouldn't be as difficult. As soon as I saw this, I understood this problem has a deeper issue than just being complex.

3 My algorithmic approach

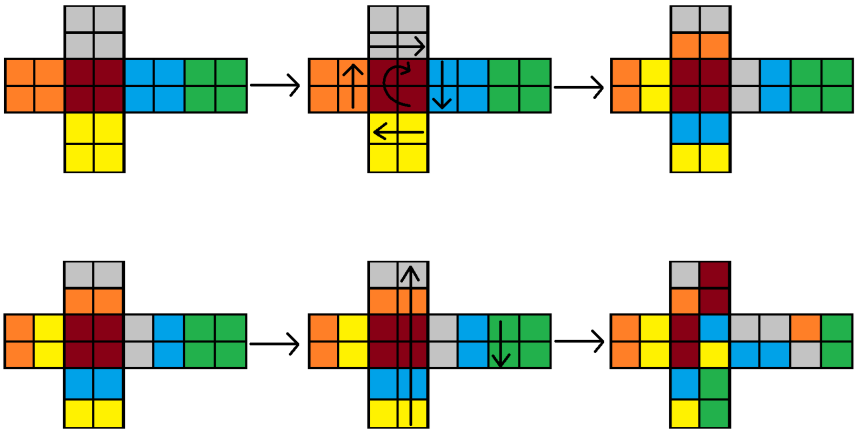
3.1 Rotation Logic

To begin with I needed to understand how to rotate stickers on a cube. To get used to it, I started with a 2x2x2 Cube. The rotations of the stickers were simple enough, but the logic of where it should be moved was much more difficult to figure out. There are 12 movements possible on the 2x2x2 cube: Up, Down, Left, Right, Front, Back; they should all have a similar logic, except back, which will have its own unique challenge.



In the above image, the movements done are F and then R.

After imagining how a 2x2x2 Cube would move in real life, I made simple drawings of the movements. I decided to label each sticker on the cube from (0,0) to (1,1) along with its colour, however the position is not to be printed out. The output would be a pattern using the colours of the rubik's cube, which can then be rotated using the rotation logic.

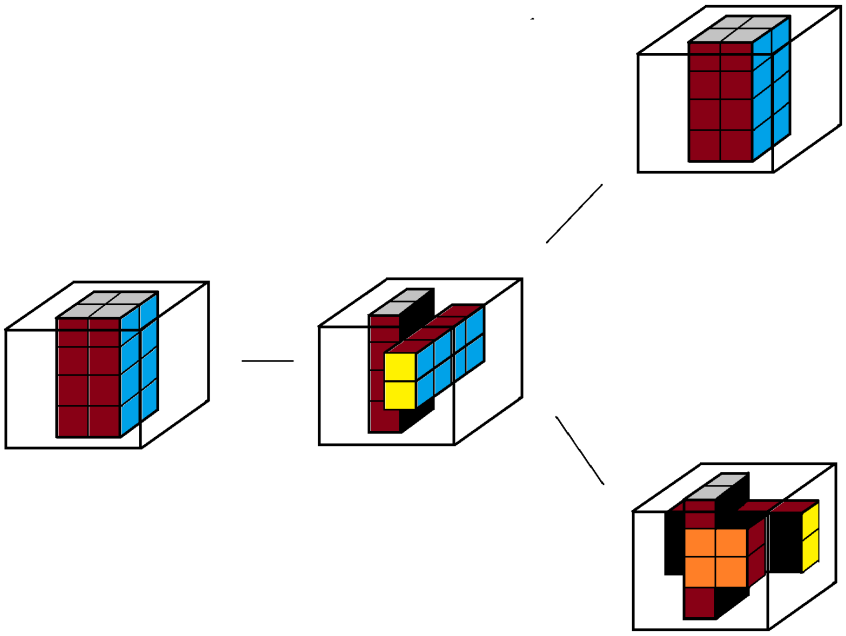


In the above image, the movements done are F and then R.

Now that I have settled on the output being a pattern, I can fully focus on the rotation logic. When rotating, the movement will either be clockwise or anticlockwise. The rotations will be done by triangle rotations, using a temp variable. This is quite simple, leaving the only challenge to be knowing exactly where the sticker should end up after the rotation. The biggest difficulty was the back side, as movements on it aren't obvious on a pattern. I manage to figure it out, by imagining the back part of the pattern to be flipped, when doing the logic.

3.2 Avoiding The Difficulties

The next step in figuring out the secret of the tower cube, was to make a version that avoids its shape-shifting quality. The shape-shifting quality of the cube is its biggest complexity. When the cuboid begins shape-shifting it can no longer be displayed as a simple pattern.



Any rotations on a side face as a magnitude larger number of possibilities than on a locked cube.

Each rotation on a side face, can come back to a normal and easy to program cuboid, or devolve into a wild shape that is difficult to display. This increased number of possibilities is the first difficulty in figuring out how each movement would affect the cuboid. The second being how to store the cuboid in a way that allows it to shape-shift, since not do we need to give it extra space for it to rotate into, but we also need have a structure that can also allow the stickers to move correctly. And the finally the third is how to output the tower cube, since the storing method for it will be much too complex for a pattern output.

Just like IsoCubeSim [6] and PermPuzzle [5], I can extend the 2x2x2 cube, adding 2 additional middle layers, and just focus on how this new cube would move, by locking the Left, Front, Right and Back movements to 180 degrees.

This had a strange result, because not only are we adding 4 new moves, that work differently from previously, we also have to change how the side faces would rotate. The new movements are simple enough, yet unique in their own way. They forego changing all the stickers on an edge, and instead focus on moving just 8 stickers without affecting the edge stickers. At first I wondered if I should add additional movements, such as the top and a middle layer at the same time, but I quickly realised that it was completely redundant. Forcing the user to

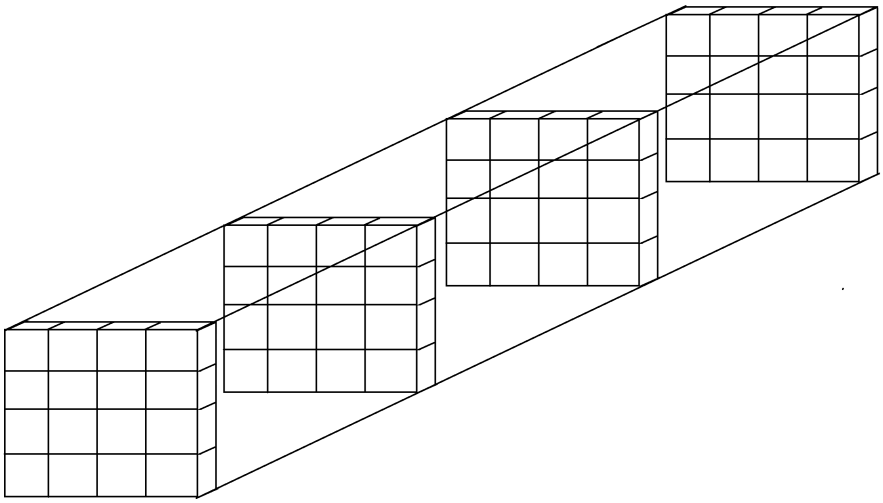
make one movement at a time, is the method done by every single simulator for a good reason: unnecessary complexity which will see little to no use.

So finally I made a 2x2x4 tower cube simulatorthat could match the state of the art for tower cube simulators, albeit not on the graphical side. However the technical side isn't lacking, as I have carefully fine tuned each movement to be not only precise but also optimised. The strange result from this test gave me a brand new idea on how to explore the problems of storing the tower cube.

3.3 Graph and Rotations

Indeed, thanks to making the locked version of a tower cube, the newest method I came up with for programming the tower cube was to store positions of little cubes in a graph that can be stuck together when displayed to mimic a real tower cube. I decided not to focus on the graphical side of the problem, as that can be done easily enough once the main functions are done, using python and a little more time.

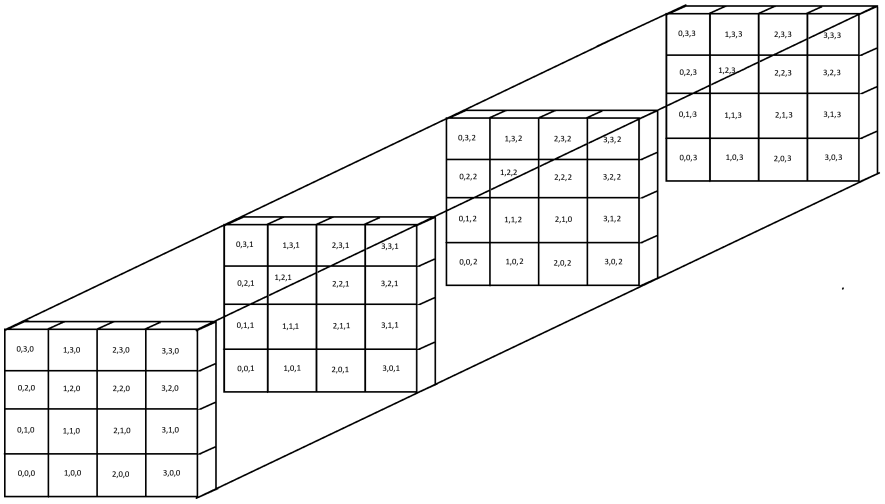
I started with a 4x4x4 grid to hold positions for objects.I focused on how the rotations would affect the cuboid and ensuring that I can keep track of the positions of the cubes effectively using object orientated programming. To begin I drew how I wanted to display the grid, deciding to separate the grid at points on the z axis, and marking exactly where each object would be.



Representation of a 4x4x4 grid in space separated at points on the z axis.

The next step was to give each object a position with an x,y and z coordinate. I decided to make a structure to hold an object, and will make 64 of these objects in for loops. In the structure I decided to hold the true value of the coordinate. I call it the true value, because that is the value that will rotate. The way I

will display the objects will not be using its true value, but instead using the normal value for the coordinates.

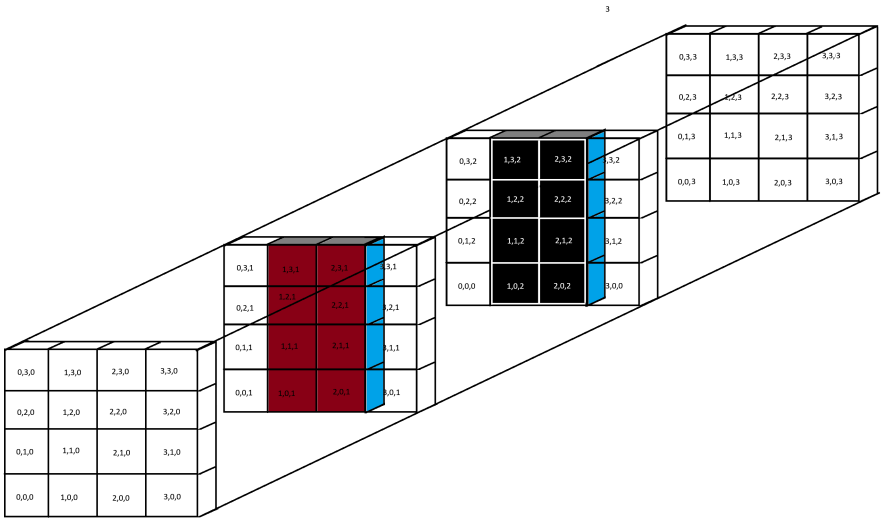


The 4x4x4 grid with each block having it's own coordinates.

Once I had the idea for holding the cube in a grid, the process of initialising a space for a tower cube was remarkably easy. And I immediately began figuring how to rotate this grid in a way that could be translated into movements for a tower cube. The key to the rotations was using the planes and axis of a graph. I could then move rotate the entire plane centered on the position (2,2) on that plane, whether that is (x,y) for z, (x,z) for y, or (y,z) for x, moving each cube individually using triangular rotation. Figuring out the logic for the rotation was exciting, as it got me closer and closer to finding the secret of the tower cube. The most important part here is making sure we always have 16 objects linked together to make up the tower cube.

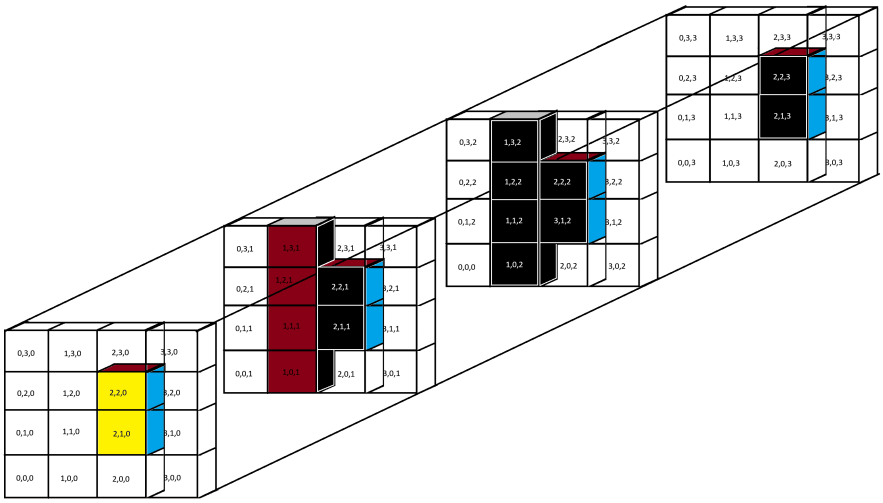
3.4 Stickers and Rotations

Once I finally finished rotating the objects in space, it was time to give each object a sticker. When looking at a real tower cube, there are 4 stickers for white and yellow, and 8 for the rest. However there's also another colour, which isn't exactly a sticker, black. This colour is extremely important to keep track off, as it shows that the cuboid is a singular object composed of other objects connected by black. Otherwise it would be like moving stickers in thin air.



Representing the grid where each object has colours on its faces.

When initialising the space, I made it so all objects had no colours. This was because, in the 4x4x4 grid, there is empty space to allow the tower cube to shape-shift as needed. I then initialise the tower cube, by adding colours to its faces depending on where the object starts. Remembering to colour black any faces that would point towards the center of the cube, therefore not having a sticker. This is where I realised a new problem, the movements of stickers aren't linked with the movements of objects when rotating the plane. The stickers follow the movement of the plane except for two at every time, for example the left and right stickers when rotating the x plane, the up and down stickers when rotating the y plane, and the front and back stickers when rotating the z plane.



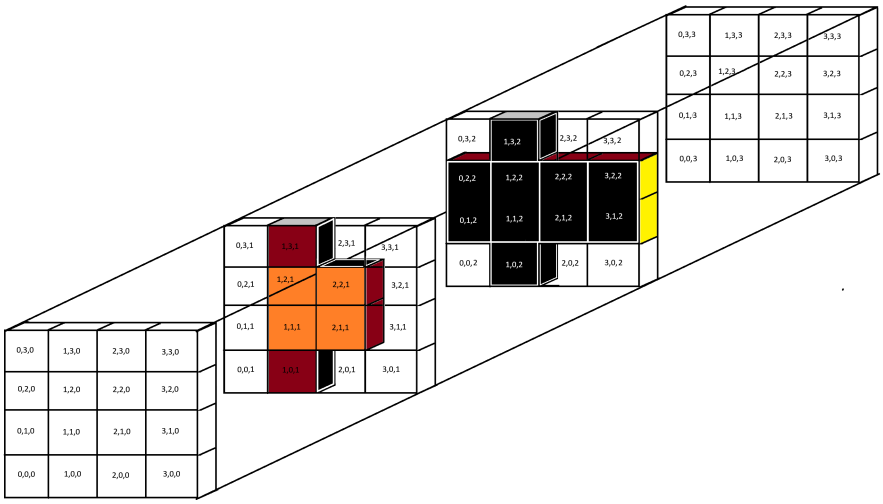
Rotating a plane also make the stickers rotate.

And with this I had completely grasped the shape-shifting quality of a tower cube. With extensive testing on the logic of the rotations, making sure to respect the space each object occupies in space, I have managed to match the complexity of a real tower cube. There's still one last problem to solve however, it's that there are certain positions aren't allowed to rotate, as the mechanism of a real tower cube doesn't allow that movement. This is no problem in a digital version, however the simulator must remain faithful.

3.5 Commutative Lock

This problem, which I have dubbed the commutative lock, stems from group theory. If the group isn't the correct size, then the number of permutations allowed on it is limited. This problem rears its head when the far edges of the tower cube aren't a 2x2, because the mechanism wouldn't be able to rotate until the black pieces are sticking together.

After thinking about the logic for how to handle this lock, I have realised that the only places this would apply for for the yellow and white edges. The logic still remains complex, however it brings down the number of cases by a large amount. The reason for this decision, is because if any of the side stickers made it to x,y or z = to 0 or 3, it would be absurd. So with an proof by contradiction we can easily prove that it can't be any colour except white or yellow.



In this position, yellow has 2 objects with black stickers on the right face between it and the center of the grid.

By definition if there is a red sticker on the face of an object there is at most one object with a black sticker on that same face between it and the center of the grid.

Then let's suppose red sticker was on the edge of a 4x4x4 piece, it would mean that there are 2 objects with a black sticker on that same face between it and the center of the grid. Which is a contradiction, and it's the same for Orange, Blue and Green.

4 My results

4.1 A Brand New Method

With this I have finally finished my 2x2x4 Tower Cube Simulator. This was an exciting journey; after extensive research online it seems that this has never been done before. An interesting observation was the problem became much easier once I had figured out a simpler way to represent the cube.

4.2 Rarity of Results

I believe the reason for the rarity of a tower cube simulators stems from two large issues. The unpopularity of the cube and the complexity of a shape-shifting cube.

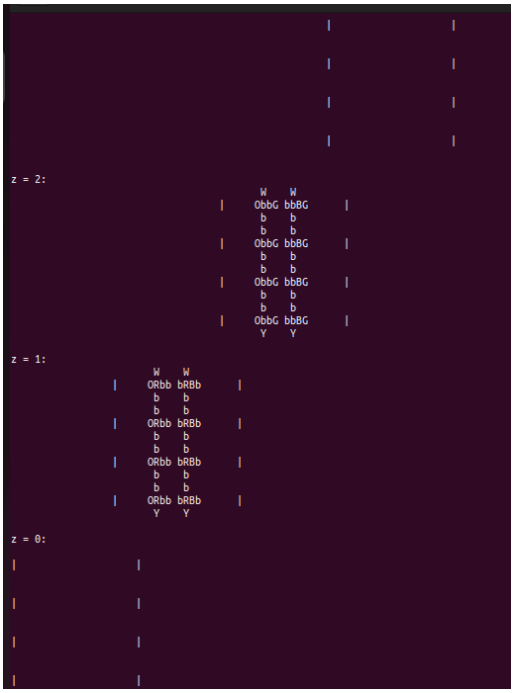
After looking around the PermPuzzle [5] application, I have come to realise the simplicity of the storing method for each puzzle, it uses matrices to store

all the data. This is the conventional way to think about a problem like the Rubik's Cube, however this method of storage is not good enough to handle the shape-shifting quality of a tower cube. The same can be said for IsoCubeSim [6], which is an over simplified version of a tower cube, not even allowing the user to see all sides of the cube at once. These programs have avoided the many problem of the tower cube, and in so fail to accurately depict the tower cube fully.

The Tower Cube is also one of the least discussed puzzles, not even mentioned on the Combination Puzzle [10] page on Wikipedia or even its own page. This unpopularity probably stems from the fact that it's not mass produced internationally, even the official Rubik's Cube[8] page, hides it in the beyond the cube section.

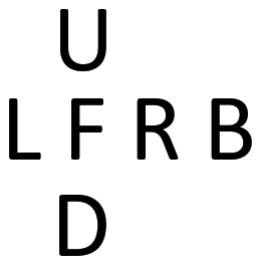
4.3 Usability of Results

When running my 2x2x4 Tower Cube Simulator, the user will be first met by a simple question, do they want to see the spatial movements of the grid, or how the patterns of each unique object moves on the grid. The spatial side forms the foundation of the entire program, so I have decided to allow a user to see the exact movements. I will focus on the Pattern Mode, for that is the best way of representing a tower cube using stickers and patterns.



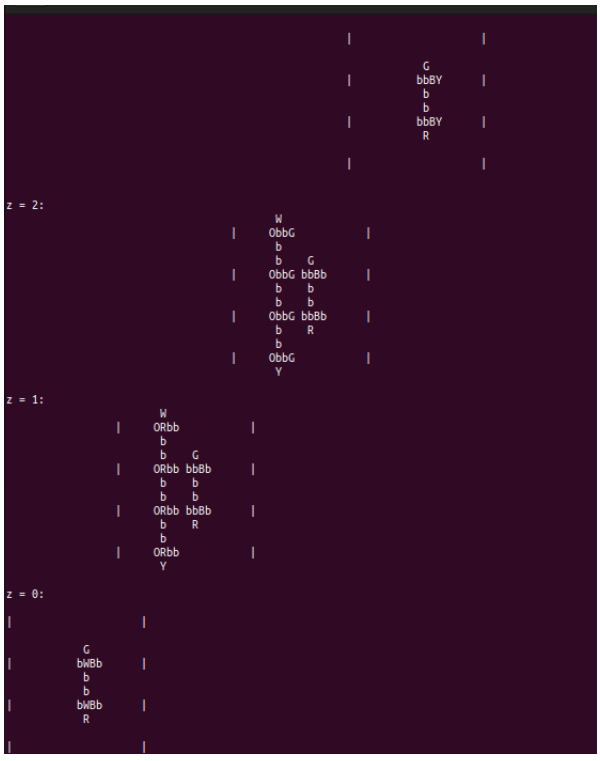
Representation of the patterns of each object on the 4x4x4 grid at the beginning.

This was the best way to represent the cube in C, this method is the most conducive to turning it into a graphical representation. Since each object can represented one at time using graphical tools. The patterns are represented as thus:

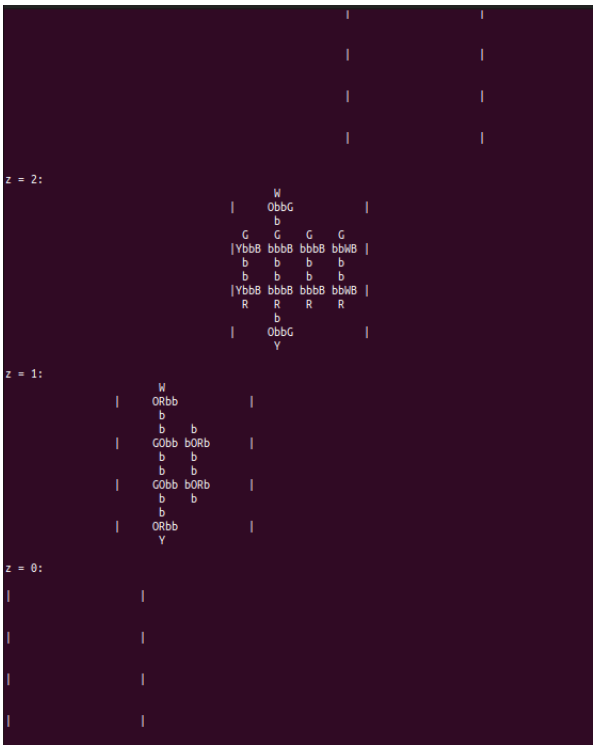


Using U for up, L for left, F for front, R for right, B for Back and D for down.

I believe the function of this program is obvious enough that most programmers could implement this a the rotation function for a graphical tower cube program. The moves are accurate and the pattern moves depending the on the object's true position. A showcase of movements, similar to the graph above:



Turning the base tower Cube using a R movement.



Turning the tower cube with M1 (upper middle) and also M2 (lower middle)

4.4 Possible Improvements and extensions

This program is fairly rudimentary, as most of the work was logical and theoretical. Here are couple of possible improvements for the future:

- A 3D representation of the cube, by integrating the pattern of each object
- Dynamic Camera movements, and smooth and fast animations
- A scrambler function that keeps the objects in the correct position
- A solver function that can solve the tower cube from any starting position even shape-shifted
- Finding out God's number for a tower cube
- A storage of the user's movements so that they can retrace their steps
- A way to see the cube for many different directions

5 Conclusion

The journey of developing a true simulator for the 2x2x4 Tower Cube has been enlightening. I was able to integrate algorithmic solutions and computational logic into my program to simulate the complex spatial mathematics hidden behind shape-shifting puzzles.

Unlike its counterparts, the Tower Cube's shape-shifting nature introduces a high degree of complexity, which is extremely difficult to achieve using a traditional matrix base approach. This is because a matrix cannot accommodate the irregularities of the cuboid and its dynamic constraints.

I am very proud of my simulator for the 2X2X4 Tower, which has gone above and beyond what any simulator that is currently easy to find on the internet has achieved. It's a solid foundation for many possible extension. I must also thank Cyril Banderier for giving me the idea of programming the 2x2x4 Tower Cube, he explained that it would provide a unique challenge. And it delivered, this mechanical puzzle has many intricacies and nuance that makes it a fulfilling computational challenge.

6 Journal de Recherche

6.1 20 questions

1. What is the maximum number of moves needed to solve a 3x3x3 Rubik's cube from any configuration? 2. What is 'God's Number' for the Rubik's cube and what does it represent? 3. What is the best way to represent a Rubik's Cube in a C program? 4. How many different configurations are there for a 3x3x3 cube? 5. If you swap two pieces of the cube, could it still be solvable? 6. If you change only one dimension of a Rubik's cube, how do the properties of the cube change? 7. For a tower cube 2x2x4, what is the maximum number of edges possible? 8. For a tower cube, do you need a completely different algorithm? 9. For a tower cube, what is the maximum number of moves to solve it starting from any configuration? 10. What is the relationship between the Rubik's Cube and Cayley graphs? Could it help visualise the cube's configuration in space? 11. Why is the Rubik's Cube so strongly related to group theory? 12. Is the Rubik's cube part of the abelian group? 13. How do the properties of an abelian group matter when solving a Rubik's cube? 14. What is the size of the group that represents a Rubik's Cube? 15. What sets of permutations can generate the group of the Rubik's Cube? 16. What is the smallest set of generators for the Rubik's Cube? (Are there multiple with the same size?) 17. Are there any interesting subgroups to the Rubik's Cube's group? 18. How do the groups for cubes of different dimensions compare to the group for the 3x3x3? 19. How can we use group theory to help compute an optimised algorithm to solve a given cube? 20. Can we use it to determine the solvability of this cube? 20. What are the best applications of the mathematics behind the Rubik's Cube

6.2 Frise Image

Frise Chronologique made using Smartdraw and images from the following: $https://old.maa.org/press/periodicals/convergence/mathematical-treasure-old-babylonian-area-calculationhttps://www.exploratorium.edu/brief-history-pi-phttps://www.shutterstock.com/image-vector/holy-bible-icon-flat-style-christianity-2392204939https://fr.wikipedia.org/wiki/Archim\%C3\%A8dehttps://medium.com/@gijovijayan_{2}2826/acharya-aryabhat-476-550-ce-was-an-indian-mathematician-who-calculated-pi-36f67446fe41https://chine.in/guide/liu-hui_{3}329.htmlhttps://chine.in/guide/chongzhi_{3}327.htmlhttps://oscarenfotos.com/2019/04/06/proporcion-aurea-y-fotografia/leonardo_{p}isano/https://fr.wikipedia.org/wiki/Nicolas_{d}e_{C}ueshttp://rudimatematici-lescienze.blogautore.espresso.repubblica.icompleanno-francois-e-adriaan/https://fr.wikipedia.org/wiki/Ludolph_{v}an_{C}eulenhttps://fr.wikipedia.org/wiki/William_{O}ughtredhttps://fr.wikipedia.org/wiki/John_{W}allishttps://fr.wikipedia.org/wiki/William_{J}ones_{\%}28linguiste\%29https://fr.wikipedia.org/wiki/Gottfried_{W}ilhelm//fr.wikipedia.org/wiki/Leonhard_{E}ulerhttps://www.popularmechanics.com/science/math/g26630324/wis-pi/$

6.3 CompteEstBon

- This program is an implementation of the game 'Le Compte Est Bon'.
- I stored the available starting numbers (1-10, 25,50,75,100) in an array, and I also stored the numbers from 101-999 for the target.
- I randomly pick 6 numbers form the starting numbers array and 1 from the target numbers array.
- Using a recursive function, I explore every single combination of the 6 numbers using the 4 basic operations (+,-,/,*), and store them.
- At each step I check if the new number is closer to the target number.
- At the end I output the closest number to the target number as asked by the instructions for the program.

References

[1] Calculators. 2x2xn cuboid. https://www.calculators.org/games/3d-rubiks-cube/. (accessed: 14.11.2024).

[2] P Rodney Carlisle. Encyclopedia of play in today's society. 2009.

[3] YO! development. Grubik's simulator. https://www.grubiks.com/. (accessed: 13.11.2024).

[4] Denes Ferenc. 2x2xn cuboid. https://ruwix.com/twisty-puzzles/2x2xn-cuboid-puzzles/. (accessed: 10.11.2024).

[5] gelatinbrain. Gelatinbrain puzzle simulator. https://github.com/Hypercubers/gelatinbrain/. (accessed: 25.11.2024).

[6] MICHAEL Z. R. GOTTLIEB. Isocubesim 4.1. https://mzrg.com/rubik/iso/. (accessed: 25.11.2024).

[7] T Jerome. Rubik's cube 25 years on: Crazy toys, crazy times. *https://www.independent.co.uk/news/science/rubiks-cube-25-years-on-crazy-toys-crazy-times-5334529.html*, 2007.

[8] Official rubik's cube page. https://www.rubiks.com/. (accessed: 04.12.2024).

[9] V D Stephen. Inventing the 20th century: 100 inventions that shaped the world. 2002.

[10] Combination puzzle. https://en.wikipedia.org/wiki/Combination_puzzle. (accessed: 16.11.2024).

[11] Da-Xing Zeng. Overview of rubik's cube and reflections on its application in mechanism. *https://cjme.springeropen.com/articles/10.1186/s10033-018-0269-7*, 2018.