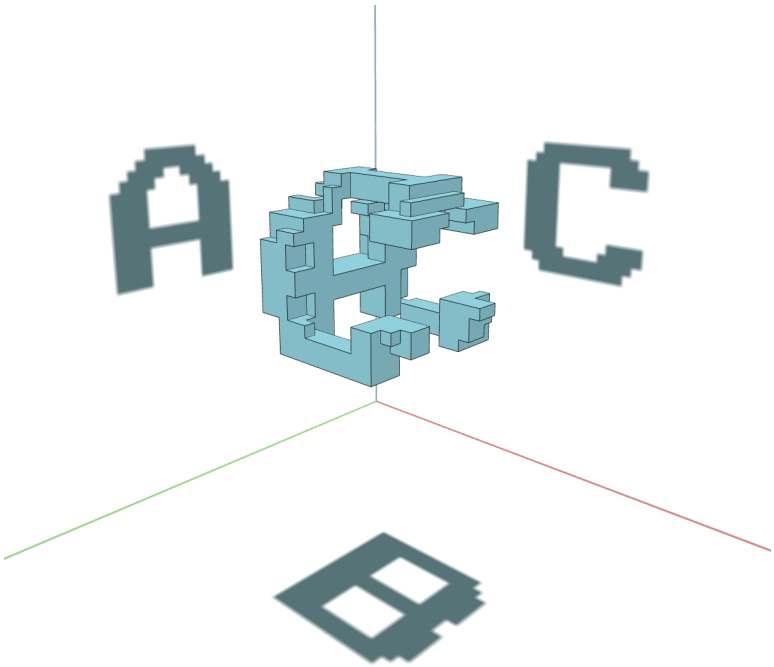
Designing and Analysing Triplets

S.H.J. Duijn

April 16, 2024



Abstract

This research delves into the creation and analysis of triplets—a unique form of 3D objects capable of casting three distinct shadows when illuminated from different angles. Drawing inspiration from mathematical and artistic concepts, this study investigates methods for generating triplets from user-defined 2D input shapes and evaluates the impact of input shape characteristics on shadow accuracy. Through the development of a specialized application for triplet design and experimentation, significant insights are gained. Results indicate a correlation between input shape thickness and triplet error scores, with higher fill percentages leading to improved accuracy. Notably, techniques for enhancing the visual appeal by selectively removing cubes show promising results. Through an exploration of the creation and manipulation of triplets, this study provides valuable insights into their unique properties and the potential for further modification to enhance their visual appeal and add additional intriguing characteristics.

Utrecht University

Contents

1 Introduction

Since the rise of computers and video games, three-dimensional art has become increasingly popular. Take, for example, the rotating donut (or torus) in the terminal, created by Andy Sloane in 2006 [29], using ASCII characters, or the intricate 3D fractals generated through ray marching [11]. Shadows, too, can be harnessed as an art form, conveying the intended meaning or emotion of an object solely through projected shadows. Triplets combine these concepts to create intriguing shapes that possess interesting properties.

Triplets are 3D objects capable of casting three, possibly different, shadows when illuminated from different angles. For instance, light from above can cast a shadow on the ground below the object, while light from the side may produce a different shadow. Similarly, light directed towards the front of the object could result in a third shadow. What makes triplets interesting is their ability to project three, potentially distinct, 2D shapes from a single 3D object.

This research investigates the process of creating a 3D triplet shape from three 2D input shapes defined by the user as silhouettes. The objective is to generate a triplet capable of casting three shadows that exactly match the input shapes. The resulting object could also be rotated to sequentially reveal all three shapes when looking at it directly.

However, it may not always be feasible to create a shape capable of casting all three predefined silhouettes. Depending on the input shapes, it might be impossible to produce an object which shadows precisely match the input silhouettes. One aspect of this research aims to explore the influence of input size and thickness on the resulting triplet shape and shadow accuracy.

Another area of investigation is the exploration of methods to enhance the visual appeal of the resulting triplet shape. This could involve selectively removing cubes while preserving the same shadows.

To facilitate these explorations, an application has been developed for creating and visually inspecting triplet shapes. Please refer to Appendix B for details on this application. Additionally, a font has been developed for use as input shapes in the experiments.

This document will begin by exploring relevant concepts such as the history of mathematics and art, font design, and the utilization of shadows in art. Subsequently, the method employed for triplet creation and evaluation will be introduced. Following this, experiments will be conducted to assess the impact of varying input shape thickness and to explore methods of enhancing triplet shape appeal by cube removal. Finally, conclusions will be drawn, and potential topics for future research will be discussed.

2 Related Work

2.1 The Combination of Math, Computer Science and Art

Creating visually appealing shapes and figures by combining mathematics, computer science, and artistic expressions has a rich history. This section will briefly introduce some of these shapes and figures.

A classic example where simple mathematical definitions create beautiful figures is fractals. Fractals are "exactly the same at every scale or nearly the same at different scales" as defined by Benoit B. Mandelbrot. A well known fractal and often used for creative and aesthetic purposes, is the Mandelbrot set, named after Benoit B. Mandelbrot. In 1980, Mandelbrot started to explore how this fractal can be calculated with the help of a computer [17, 18]. However, the fractal has been explored before in 1905 by Pierre Fatou, a mathematician who specialised in recursive equations. A recursive equation can be seen as a series of calculations where the next calculation is based on the previous one, with a predefined starting value.

A Mandelbrot set is a collection of complex numbers that remain bounded when recursively applying the quadratic map $Z_{n+1} = Z_n^2 + c$ to them, starting with $Z_0 = 0$. A complex number c can be written as $c = x + yi$, where x and y are real numbers that can be plotted as coordinates on a 2D plane. Points (x, y) that belong to the Mandelbrot set are typically represented as black dots on this 2D plane. Points that will diverge to infinity, and thus do not belong in the Mandelbrot set, can be colored white. Or, to get a more interesting image, can be colored based on the number of iterations before it diverges to infinity, see Figure 1a. In practice, a maximum number of iterations needs to defined. When this is

reached and Z_{n+1} remains within bounds, we can say that the point belongs to the Mandelbrot set. To check whether complex number Z_{n+1} remains within bounds, the absolute value of this number can be used. If the absolute value is less than or equal to 2 it is bounded. Additionally, a three-dimensional extension of the Mandelbrot set, known as the Mandel*bulb*, also exists. This intriguing shape, when neatly colored, can resemble fantastical landscapes or planets [33].

Another well known fractal is the Sierpiński triangle, named after the Polish mathematician Wacław Sierpiński [25]. Sierpiński described the fractal in 1915 but similar patterns already appeared in the 13th century as a common motif in inlay stonework [30]. The Sierpiński triangle consists of an equilateral triangle which is recursively subdivided into smaller triangles. A simple method to create a Sierpiński triangle is to start with an equilateral triangle and divide it into four smaller congruent equilateral triangles, remove the middle one and repeat the process for the other three. Keep repeating to infinity. An interesting property of the Sierpiński triangle is that the area of the shape is zero when the recursion reaches infinity. There are other methods to create the Sierpiński triangle, like shrinking and duplication, Pascal's triangle, towers of Hanoi and using a fractal tree configuration [27, 3].

Fractal trees are another aesthetically pleasing shape created using recursion. It starts with a root branch which splits into two or more branches at certain angles to the root branch. Recursively repeat this branching process for the new branches. Depending on the angles used for branching, tree-like structures can arise. Interestingly, it is possible to create a Sierpiński triangle by using three branches with a 120° degree angle between them. Fractal trees are a very simple concept that can create beautiful and intricate figures. Three-dimensional fractal trees can resemble a real tree quite well with the right configurations [21].

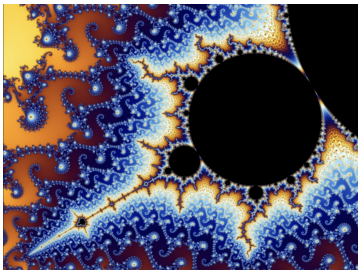
A more recent technique called ray marching makes it possible to render three-dimensional fractals with low computational cost [28, 11]. Ray marching is a technique similar to ray tracing. For each pixel on the camera plane, a ray is cast until it hits an object in the scene. With ray marching, the ray is advanced in steps (marching), until it "hits" or is close enough to an object. The amount the ray is advanced can be just a small delta value. But a smarter way to do this is to make use of a signed distance function (SDF). This SDF gives the closest point to the surface of an object in the scene. Now, calculate the SDF at each ray marching step, and safely move this number along the ray without hitting any objects. If the SDF becomes very small, it can be assumed that the object is hit. When there are a lot of objects in the scene this becomes expensive to calculate. But, even with only one object in the scene, it is possible to make it look like there are an infinite amount of them. This is done by modifying the SDF, for example by taking the modulo of x and y of our point before calculating the SDF. By modifying the SDF, it is possible to create the Sierpiński triangle and Mandelbox, a fractal with a square box shape found in 2010 by Thomas Lowe [14]. See Figure 1b for an artistic example of this.

The golden ratio, also known as the golden number or divine proportion, is an aesthetically pleasing ratio between two numbers that manifests in nature and has a rich historical presence. The ratio equals $\frac{1+\sqrt{5}}{2} = 1.618...$ and is usually written as the Greek letter phi: ϕ. It has a close connection to the Fibonacci sequence, a series of numbers where the next number is the sum of the previous two: $\{0, 1, 1, 2, 3, 5, 8, 13, 21, ...\}$. The ratio between each number and the previous gradually equals ϕ, the golden ratio.

The first known mention of it, in 300 BC, was by the ancient Greek mathematician Euclid [13]. Much later, in 1509, a book was published by an Italian mathematician Luca Pacolio called *De Divina Proportione*. With illustrations by Leonardo da Vinci, the ratio was praised to represent divine simplicity and orderliness [13]. The golden ratio can be found back in architecture designs from ancient civilizations like the Greeks and Egyptians, for example in the Parthenon in Athens and the Great Pyramid of Giza. People argue that it can also be seen in the composition of paintings from artists like Michelangelo and (unsurprising) Leonardo da Vinci. *The Last Supper* and *The Vitruvian Man* are often used as examples of this proportion [7].

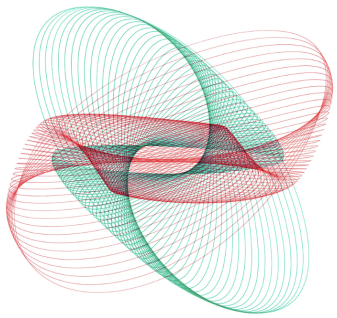
The golden ratio can be observed in natural phenomena, from the arrangement of leaves on a stem or the spiral pattern of a seashell and pine cone and sunflower seeds. Golden ratio enthusiasts argue that this is the reason the ratio looks pleasing to the eye when used in graphical design or practical objects. It is important to note that while the golden ratio has been widely celebrated, its application is sometimes subjective and can be a matter of interpretation [24].

Another concrete and tangible example of math combined with art are harmonographs or "pendulum drawings". These are two-dimensional figures consisting of many curves that form interesting



(a) Mandelbrot set.

(b) Ray marching fractal art by Pete Linforth.



(c) Harmonograph.

Figure 1: Three examples of art that uses math.

shapes, some of which even give the feeling of depth. They can be created using mechanical constructions, the simplest of which is a pendulum. This is a weighted pendant on a cord that can draw in sand or on a piece of paper. If the pendant is swung, it can produce beautiful shapes.

Other mechanical devices are harmonographs and spirographs. A harmonograph is a device that appeared in the mid-19th century and consists of a plane with a piece of paper suspended in the air by the four corners of the plane [32]. A pen rests lightly on the paper to be able to draw. This is a so called *lateral* harmonograph. It basically consists of two pendulums, where one moves along one axis and the other moves along a perpendicular axis. Different patterns are created by varying the frequency and phase of the pendulums relative to one another, see Figure 1c. A spirograph is a geometric drawing toy that consists of gears with holes for a pencil. When rotated inside other wheel-like gears, the pencil follows a path that creates interesting figures, so called hypotrochoids and epitrochoids.

These harmonographs resemble the recently developed Spiroplots [8]. Instead of using mechanical devices, spiroplots are simulated on the computer. They are defined by a number of points connected by edges. Each edge will then consecutively rotate a around its center by a predetermined number of degrees. After each edge rotation, connected points will have moved and a dot is drawn for the moved points. After all edges have rotated, repeat. Do this many times and captivating figures can arise. This is another example of figures created with simple mathematical definitions.

2.2 Shadows in Art

A shadow is fundamentally the result of the absence of light. There needs to be a light source to create a shadow, where the shadow is defined by some object that is obstructing the path of light. There always needs to be two objects and a light source to create a shadow. One object, often referred to as the occluder, obstructs the light source, while another object, such as a wall or surface, becomes the canvas upon which the shadow is cast.

The usage of shadows in art, philosophy and entertainment goes a long way back. In art, shadows were used to add depth and dimension to compositions or even to convey certain emotions. The use of shadows not only allowed artists to add depth but also to create atmospheric tension and evoke emotions. For example chiaroscuro, a technique mastered by artists like Caravaggio during the Baroque period, introduced a heightened sense of drama through the heavy contrast between light and shadow. More recently, Andy Warhol, with his work *66 Shadows*, and Joseph Beuys have explored the idea of the shadow as the revelation of utter human emptiness as the powerful doppelgänger of our "self". Philosophers have discussed shadows as a metaphorical concept. Plato's Allegory of the Cave, from his work "The Republic", delves into the symbolic nature of shadows. In this allegory, prisoners confined in a cave perceive only the shadows cast by objects behind them. The shadows represent a distorted reality, an illusion that the prisoners mistake for the true forms. Plato uses this to present the philosophical concept of the path from ignorance to enlightenment [31].

In modern and contemporary art, shadows continue to serve as a way to convey emotion. The use of shadows in photography and film further expand the narrative possibilities, allowing for nuanced storytelling through the manipulation of light and darkness.

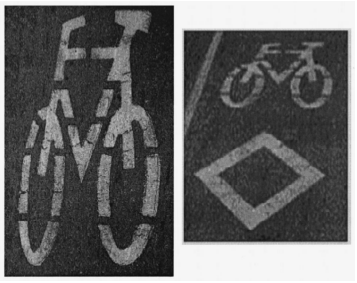
Another form of entertainment that makes use of shadows is shadow play. Shadow play probably

originates from China and Indonesian islands of Java and Bali and is still a living folk tradition in China, India, Iran and Nepal [22, 5]. It is a way of storytelling where the audience looks at a translucent screen which has silhouettes projected on it from behind, often combined with a musical composition. These silhouettes are cast by intricate puppets which are controlled by the puppeteers or can be cast by the bodies of the actors themselves.

Shadows can leave a lot to one's imagination, which can be a positive point because the viewer can depict the play in their mind as very realistic. The silhouettes can give the illusion that things behind the screen are happening for real. Although shadow play has its limitations, it is possible to do things which are not possible with normal actors. A shadow could hide behind another and suddenly appear out of nowhere. It is also possible to use colored translucent film with the puppets to add colors to the projected silhouettes. Using your hands to create silhouettes that resemble various animals or figures is a widely known usage of shadow play. To this day, shadow play is still liked in various cultures by both children and adults.

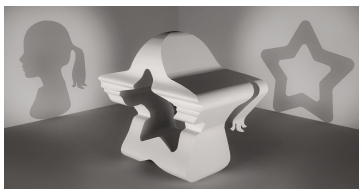
Shadows are also used by artists where the actual shadow is the main art piece instead of the object it was projected by. When looking at the object itself, it might not be immediately clear what known shape the object represents or the object might appear distorted. But when looking at the shadow, cast via a light at a certain angle, it becomes clear. This is known as anamorphic shadow art. Anamorphic art appears in 16th century when artists, mathematicians, and philosophers start using perspective to blur the boundary between illusion and reality [10].

Anamorphosis is a type of distorted representation of an image or object that only appears as intended when viewed from a specific angle or through the use of special devices, like a curved mirror surface. Bernard Pras is an artist who uses anamorphosis, he makes installations from all kinds of objects, from toothbrushes to pianos. Theses installations only reveal the intended essence of the art piece when viewed or photographed at a specific angle, see Figure 2b for an example. Some anamorphic art pieces use shadows to reveal the intended shape. Take for example the Star Wars shadow art from Red Hong Yi, a Malaysian artist and architect. This Star Wars shadow art is made from several everyday objects on long pins, which reveal figures of Star Wars characters when light is projected from the right angle. Without shadows, it is hard to see what the collection of objects should represent and looks like randomly scattered objects on pins. A 3D model made by Ivana Bajšanski, shows two different silhouettes when light is projected from two different angles see Figure 2c. This is closely related to triplets, which show three, possibly different, silhouettes of letters.



(a) Bicycle lane marking. Appears normal only when viewed at the right angle (about 20°degrees). Source: Anamorphic images. [10].

(b) Art installation that reveals a face when viewed at a specific angle. Made by Bernard Pras.



(c) Object which produces two different shadows when light is cast at the right sides. By Ivana Bajšanski.

Figure 2: Examples of anamorphic art.

2.3 Font Design and History

To delve into the history of font development we need to go back to a time when books and manuscripts were only written by hand. The style of writing, or letterform, was calligraphy-like, with long and elegant characters. Books were a luxury and generally reserved for the elite. But, when people in middle-class began learning to read, the demand increased. In the mid-15th century the goldsmith Johannes Gutenberg developed a printing press which used metal letter blocks, which made it durable

and reusable [16]. Now, books could be produced in a relatively fast manner. Gutenberg took inspiration from the presses used in East Asia, where wooden blocks were used. No one knows when the first press was invented, but the oldest known printed text originates in China during the first millenium A.D [34]. Gutenberg's press used the so called *Blackletter* calligraphy letterform, a wide and thick calligraphy-like letterform which was based on the hand written manuscripts during Gutenberg's time [9, 19]. A downside of this was that the amount of text on a page was limited due to the wide letterform. In 1470, Nicolas Jenson solved this issue by creating the fist Roman typeface. With this simpler letterform, more text could fit on a page and thus increasing printing speed. In 1501, the first italic typeface was created to save even more space. This received quite some critique due to its poor readability, but italics are still used to emphasize text.

In the 18th century the focus shifted to improve readability. Fonts were created by William Caslon and John Baskerville that have more contrast in thickness between horizontal and vertical strokes. These typefaces made letters more distinguishable from one another, improving readability. Improvements were also made to ink and printing presses which made the resulting prints clearer. As another attempt to improve readability, serifs were introduced by two type designers, Firmin Didot in France and Giambattista Bodoni in Italy. Another concept, referred to as slab serifs, was introduced in 1815 by Vincent Figgins. Compared to normal serifs, slab serifs are mostly horizontal lines at the end strokes. Around the same time a font without serifs was developed which gained popularity in the use of advertisement headers. Around 100 years later another sans serif typeface was introduced by Edward Jonson and is still in use today by the London underground signs. In the 20th century some well known fonts were developed like *Copperplate Gothic* and *Helvetica*. As well as minimalistic typefaces like *Futura* and *Optima* [4].

When digital screens were introduced, the first digital font was developed in 1968 by Rudolf Hell called *Digi Grotesk*. The fonts developed around that time were stored in bitmaps, which were hard to read at small sizes. These were also popular in video games and nowadays referred to as pixel fonts. They were mostly monospaced and constructed of cells on a grid. A few years later however, the first vector fonts were developed, these encode the outline structure in a mathematical fashion which improved readability and reduced file size. *TrueType* fonts were created in 1980, which allowed computers and output devices, like printers, to use the same file. In 1997, this same idea was applied to Mac and PC, with one font format *OpenType*, to support both devices. This *OpenType* format was later updated to variable fonts for the web, which are able to change size and weight depending on the design they are used in [35].

We have seen that a font describes not only the style but also the size and weight of a character, and may include serifs; small features at the end of strokes. In the present there are five basic classifications of font types: script, display, monospaced, serif and sans serif. Script style resembles handwriting and can range from formal to very casual and are suitable for screens. Display typefaces are very wide in appearance and are mostly used for headlines and titles. Monospaced typefaces are spaced equally, meaning that every character takes up the same amount of space. They are used for displaying code and headlines, and suitable for typewriters.

In 2005, a study concludes that fonts with small serifs (5% of the character size) are slightly more legible that sans serif fonts (those without serifs) [2]. They do note that this is mostly because serifs lead to greater letter spacing. However, 6 years later, a study from 2011 shows a slight, yet noteworthy, readability advantage in sans serif fonts compared to serif fonts [20]. This knowledge is already applied in practical contexts, such as traffic signs, which use a sans serif font.

A font can come in various weights, ranging from light and medium to bold. Some fonts even offer extra-light or extra-bold options. A study from 1940 found no difference in reading speed between normal and bold weight [23]. However, 70% of the readers expressed a preference for the normal-weighted font. Another study from 1942 similarly indicates that the optimal reading speed falls somewhere between normal and bold weights [15]. With an extra-bold weight, reading speed tends to decrease.

Letters can also be used as entertainment or creative outlet. This is shown by Erik Demaine, he combines letters with paradigms from math, puzzles, or nature. Erik Demaine has some exquisite examples of this, such as creating letters using a square piece of paper folded n number of times and cut only once, creating characters by solving Sudoku puzzles, or using Voronoi diagrams, among others [6].

2.4 Triplets

The combination of art, computer science and mathematics to create beautiful figures, shadows to illustrate certain shapes which may not be apparent at a first glance, and typography that is designed to be clear and consistent, bring us to triplets.

In 1998, at the *ACM Symposium on Computational Geometry*, O'Rourke presented the question what the conditions on three input shapes need to be in order to create a valid triplet [1]. The definition of a valid triplet is that the three shadows cast by the triplet should exactly resemble the three predefined desired shapes. For example, consider the letters `X`, `X` and `O`, for these letters it is probably not possible to create a valid triplet. Due to the two `X`'s, there needs to be material in the middle of the shape to create the cross-section of the `X`, but for the `O`, the middle of the triplet needs to empty. A combination that should be possible is `F`, `J` and `O`, for example.

In 2009, Walderveen et al. tries to answer the question by O'Rourke, exploring the validity and connectivity for triplets with rectilinear polygons (polygons whose edges are parallel to one of the primary axes) [12]. The validity can be computed with a sweep-plane algorithm with a time-complexity of $O(n^2 \log n)$ where n represents the total complexity of the three rectilinear shapes. The connectivity can be checked by employing the sweep-plane algorithm again. This can be done in $O((n^2 + k) \log n)$ time, where n is the total number of vertices in the rectilinear shapes and k represents the complexity of the triplet.

More recently, Roggen has created a tool to calculate triplets on the computer and he provides proofs for the connectivity of the final object and its combinatorial complexity [26]. He also showed that 70% of all three-letter combinations in the alphabet can create valid volume-connected triplets. The 26 letters of the alphabet were defined on a 5x5 grid were it is possible to only enable a triangular part of a cell. A tool was developed in Unity to create triplets where the users can define the three shape planes themselves. The tool has support for exporting the triplet to a 3D model format and also allows triangular half-cells to be used. Some limitations of the 5×5 grid are that the predefined letters are not very pleasant to read. For example, some letters are the same when rotated, like `I` and `H`, as well as `N` and `Z`, and `P` and `Q`.

This research aims to construct a better-looking font with a larger grid size, 14×14, and explore methods to improve the percentage of possible letter combinations. Techniques for creating more interesting triplet shapes will also be explored.

3 Requirements and Methods to Build a Triplet

A triplet is a three-dimensional shape that is able to cast three, possibly different, shadows. The input for a triplet consists of these three desired shadow shapes. They will be called shape planes further on. The shadow that is cast along an axis of a triplet will be called a shadow plane. For a correct triplet the shape planes and corresponding shadow planes should hold the same three shapes.

3.1 Shape Planes

In theory, a shape plane could be any two-dimensional shape. But, a shape plane that is defined by two or more separate components will naturally give a triplet that also has two or more separate components. This is undesirable due to the difficulty to produce this triplet in real life, for example with 3D-printing or woodworking. The two components should somehow be connected. A solution could be to use a thin transparent piece of plastic to connect them. But, because this gets complicated quickly and is outside the scope of this research, only triplets that consist of one component will be considered. And thus, shape planes should also consist of one component, or in other words, be connected.

For the purpose of this research, a shape plane is defined by a 2D grid of square cells. A cell in this grid can either have an enabled or disabled state. When a cell is enabled, the corresponding triplet's shadow plane should have a shadow at this cell's location. To satisfy the connectivity constraint, a valid shape plane should be edge-connected. This means that it should be possible to reach each enabled cell from any enabled cell by traversing through top, bottom, left or right neighboring enabled cells.

3.2 From Shape Planes to Triplet

When three valid input shape planes are given, a triplet can be created. The dimension of the triplet will depend on the dimensions of the input shape planes. When all three shape planes are square and of the same dimension, $n \times n$, the resulting triplet will be built from a 3D grid of $n \times n \times n$ cubes. It is also possible to build a triplet that is not cubical with $n \times m \times p$ cubes. The input shape planes then need to have dimensions of $n \times m$, $n \times p$ and $m \times p$. For this research, only square input shapes with the same dimensions are considered, meaning that the resulting triplets will be cubical.

To create a triplet from the input shape planes, a simple and straightforward approach can be used by starting with a 3D grid of cubes and removing cubes which obstruct the light were they should not according to the shape planes. That is, each shape plane is assigned to an axis of the triplet. The cubes along such axis are removed that match the location of disabled cells in the corresponding shape plane. For example, the shape plane assigned to the x-axis will cast a shadow on the yz plane, so all the cubes along the x-axis with yz coordinates that match coordinates in the shape plane of disabled cells, are removed. An example of the steps taken to create a triplet can be viewed in Figure 3.

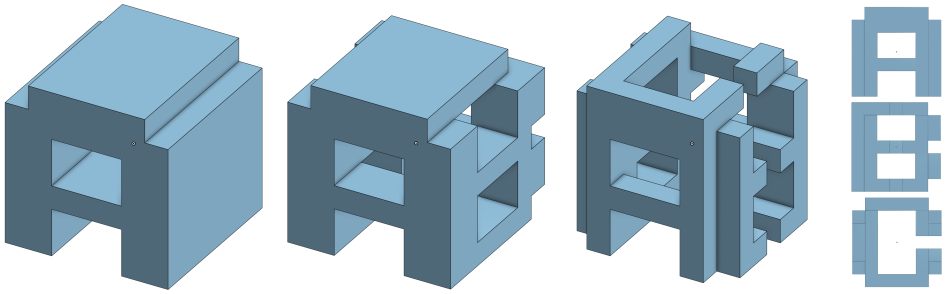


Figure 3: An example of how a triplet can be generated with letters `A`, `B` and `C` as the shape planes. First, the letter `A` is made visible along one axis by removing material from the starting cube. This process is repeated for the letters `B` and `C` along different axis which results in the correct shadow planes.

Using this approach, a triplet where the shadow planes match the input shape planes, will always have the maximum number of cubes. It might be possible to remove cubes while still keeping the shadow planes correct, but adding cubes will always result in shadow planes with erroneous cells. However, it is not always possible to create a triplet where all the shadow planes are as desired and correspond to the input shape planes.

Even if all three input shape planes are edge-connected it is still possible to get a triplet that consists of multiple components, depending on the connectivity property. The connectivity can be vertex, edge or volume-connected. If a triplet is vertex-connected then all cubes in the 3D grid can be reached from any other cube by traversing through neighboring cubes where a vertex coincides with at least one of the eight vertices of the current cube. A triplet is edge-connected if the edge of a neighboring cube coincides with an edge of the current cube, and volume-connected if the faces coincide. Thus, if a triplet is volume-connected then it is also vertex and edge-connected. Likewise, if a triplet is edge-connected, it is also vertex-connected. See Figure 4 for an example.

If a triplet is not connected according to the specified connectivity constraint (vertex, edge or volume), it will have multiple components. To deal with this problem, only the component with the most cubes will be kept and smaller components will be removed from the triplet. This will cause the shadow planes to deviate from the input shape planes but it ensures the preservation of the connectivity constraint.

3.3 Triplet Evaluation

To assess the accuracy of a triplet, an error score will be calculated. Accuracy is determined by how well the desired shape planes align with the resulting shadow planes. The score is obtained by counting the number of grid cells in the shape plane that match those in the corresponding shadow plane. This

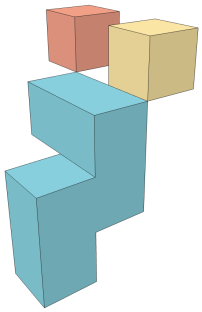


Figure 4: An example of the connectivity of a 3D shape. The blue shape is volume-connected. The shape consisting of blue and yellow cubes is not volume-connected but is edge-connected. The shape consisting of all cubes is only vertex-connected due to the orange cube.

score is then converted to a ratio of the total number of cells in the grid.

$$Score = \frac{Wrong\ cells\ in\ shadow\ plane}{Grid\ width \cdot Grid\ height}$$

With three shadows, three scores are generated, and their sum yields the final error score. In the case of a perfect triplet, this error score equals zero. In theory the maximum error score could be 3, if all cells in all three shape planes are enabled but the triplet is empty, for example.

3.4 Improving Triplet Error Score

To increase the likelihood of obtaining a triplet with low or perfect error score, several strategies can be employed, one of which is rotating the input shape planes. The rotation of a shape plane can yield a different triplet with more accurate shadows, which results in a lower error score, see Figure 5. The resulting shadow will also be rotated, but this is no problem because the user can rotate the triplet itself to the correct orientation. Each shape plane can be rotated by 0°, 90°, 180° and 270° degrees. Considering these possibilities for each of the three shape planes, a total of $4^3 = 64$ potential input combinations are possible to experiment with in search of a perfect triplet. Note that rotating a shape plane might not give a different triplet when the shape exhibits rotational symmetry.

Figure 5: An example where rotating a shape plane results in more accurate shadows. When the `M` is rotated 90° degrees clockwise, the shadow of the `B` becomes whole, unlike when the `M` is not rotated.

Another strategy is to mirror an input shape if this results in a different shape plane grid. This can also give a different triplet and thus increasing the chance of obtaining a perfect triplet. An interesting property of a triplet is that changing the input order of shape planes will be the same as mirroring a shape plane. This is due to the triplet being defined in a cube. Changing the shape plane input order will be used here instead of mirroring.

With three input shape planes, a triplet can be constructed in $4^3 \cdot 2 = 128$ configurations by rotating shape planes and altering their order. Each configuration will be tested, and the triplet with the lowest error score among them will be selected as the final output.

4 Varying Input Shape Thickness

The error score in a triplet is influenced by the characteristics of the input shapes, which can vary in size, ranging from large to small, and exhibit diverse morphologies, such as being rounded and bulbous or lanky and branching. This section aims to investigate the impact of input shape thickness on the error score of triplets.

First the 26 letters in the Latin alphabet will be considered as input shapes, where thickness can be seen as the **boldness** of a font. Then, thickness will be defined as the percentage of cells used in an input shape which will be generated randomly.

4.1 Input Shape Generation

To test whether shape thickness has an influence on the triplet error score, it is desirable to have a diverse set of thick and thin shapes. Ideally, the shapes should be close to figures that people will use to create a triplet. For example, silhouettes of letters, animals or common symbols like a heart, arrow or cross. It is desirable to have multiple shapes with varying thickness for the same figure.

4.1.1 Font Design

The examination of thickness will initially involve the 26 letters of the alphabet, as they inherently possess variations in thickness through bold and regular versions. Letters can be creatively used to form interesting triplets, such as incorporating initials or company name abbreviations. This makes it compelling to explore what the optimal thickness for letters will be to achieve triplets with either perfect or minimal error scores.

Three versions of each letter will be created, each varying in thickness: thin, medium and thick. The letters will be defined on a 14×14 grid. This dimension was selected because it is possible to define all three thickness levels with this size, and it gives enough room to create visually pleasing and distinct letters.

Inspiration for creating this font with three types of thickness was taken from pixel fonts, as the font will be defined on a grid of square cells. The cells should be edge-connected, and the font should be monospaced. Monospaced here means that each letter should extend to all edges of the 14×14 grid. This will help achieve triplets with a low error score. When a letter does not extend to all edges, such as a narrow `I`, and is used in combination with letters that do extend to all edges, such as `M` and `W`, there will always be an error. This is because it is not possible to cast a shadow to the sides of the narrow `I`, as required by the `M` or `W`. The font will only contain uppercase characters, as they are easier to define on a grid and more distinct.

Because the goal is to create distinct letters, it was chosen to make vertical lines thicker than horizontal lines. This makes sure that for example the `I` and `H`, will not have the exact same shape when one of them is rotated 90° degrees.

To facilitate the creation of the pixel grid font, a dedicated application has been developed for the easy comparison of letters and thickness variations. It also offers the convenience of importing and exporting fonts, simplifying adjustments to letter designs and their use within the main application. This application helps to ensure that the letters are distinct and easily legible. An example of the font created can be seen in Figure 6. The full font and the three thickness versions can be seen in Appendix A.

4.1.2 Random Generation

To obtain more insightful results, it was decided to also define a set of input shapes by using randomly generated shape planes. Another perspective on the *thickness* of an input shape plane, is the number of enabled cells in the shape grid, referred to as the fill percentage. Using random shape planes makes it possible to have shape planes with varying fill percentages. This approach is also expected to yield more comprehensive results as it theoretically covers all possible input shapes when using a sufficient number of random samples.

The average fill percentages for the thin, medium, and thick font are 31%, 57%, and 74%, respectively. Exploring additional fill percentages provides a more insightful understanding of the influence of fill percentage. Observations indicate that randomly generating shapes with a fill percentage lower

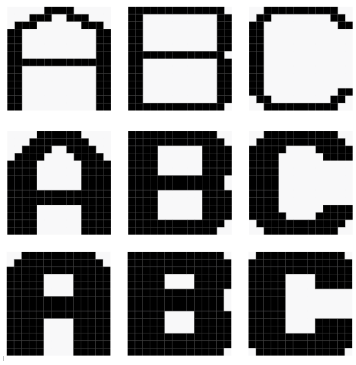


Figure 6: The letters `A`, `B` and `C` of the created font, with varying thickness from thin to thick.

than 30% is not practical because with 30%, the shapes become small and thin which makes it almost impossible to create a triplet that can cast all three shadows accurately. Thus, the fill percentage will start at 30%. It will be incremented in intervals of 5% and halted when either no change in error score is observed or when only perfect triplets are consistently produced.

To generate a random shape plane, it must adhere to the constraint of edge-connectedness and ensure that at least one cell is enabled at each edge of the grid. To guarantee the latter, initially, one cell at each edge will be randomly enabled. An example of such an initial shape plane can be seen in Figure 7. It is important to note that when a cell is selected in the corner, a cell for an adjacent edge might also be chosen in that same corner. This scenario allows for the possibility that only two cells are enabled, both in opposite corners, while still satisfying the constraint that at least one cell is enabled at each edge.

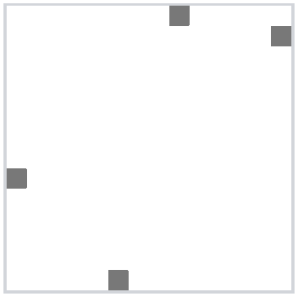
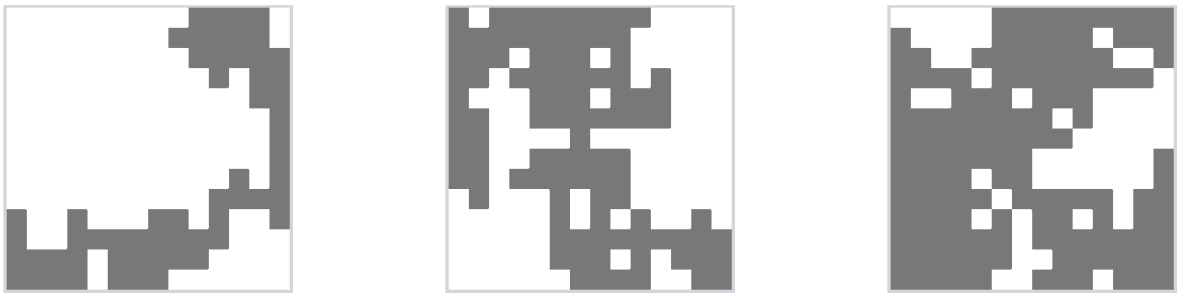


Figure 7: An example of the initial state of a random shape plane before randomly enabling neighboring cells.

The subsequent step involves randomly selecting a disabled neighboring cell of an enabled cell and enabling it. This process is repeated until the desired fill percentage is achieved. One drawback of this approach is that it does not guarantee edge-connectedness. It is possible that the edge cells will not be connected. To address this issue, an unconnected shape plane is discarded, and the process is repeated until a shape plane which is edge-connected is obtained. An advantage of this approach is that the shapes encompass a wide variety of potential inputs for a triplet. In theory, they could represent every possible edge-connected shape. A few example shapes with varying fill percentages can be observed in Figure 8.

Another approach is to initially connect the edge cells to ensure edge-connectedness. This can be achieved by randomly selecting one of the four edge cells and traversing into the shape plane in a straight line until it aligns with a cell to the left or right of the line, at which point the two cells are connected with a straight line. Continue traversing until another edge cell is aligned. Connect this other edge cell in the same manner. This process may leave one cell disconnected. This cell will be connected to the nearest enabled cell. This manner of connecting these edge cells was used because it is simple to implement and keeps the number of cells needed to connect all edge cells low. After connecting the edges, continue adding neighboring cells randomly until the desired fill percentage is reached.



30% 50% 70%

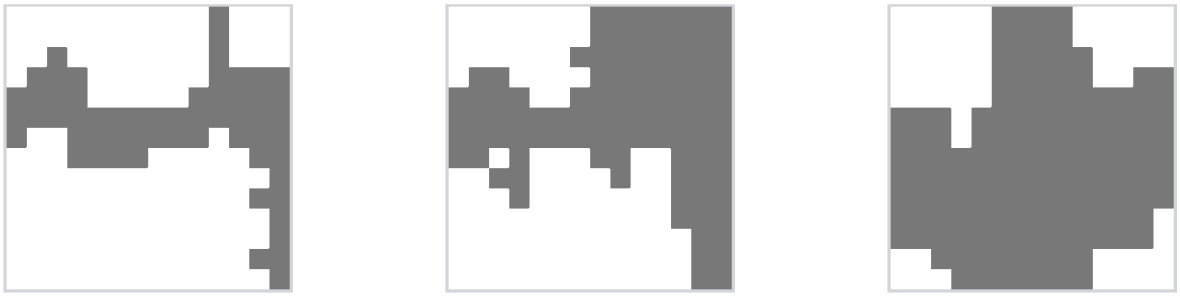
Figure 8: Examples of random generated shape planes with varying fill percentages on a 14×14 grid.

The aim is to create shapes that are close to shapes that would be used in the real world to create triplets, for example letters, playing card suits, arrows, emoji, and so on. But, upon observing the randomly generated shapes, it became evident that they can exhibit erratic patterns and often contain small holes. To address this, the decision was made to assign weights to all possible neighbors that could be enabled. These weights are used as a probability for randomly selecting a neighbor to enable. They are proportional to the number of enabled neighbors:

$$Weight_{ij} = \frac{Number\ of\ enabled\ neighbors}{4}$$

This can be seen as: *"The more neighbors a disabled cell has, the more likely it is to be chosen to be enabled"*, aiming to create more rounded and less capricious shapes. After looking at the resulting shapes using these weights, it was decided to strengthen the weight by raising it to the power of two.

Some examples of these neighbour weighted shape planes with initially connected edge cells can be observed in Figure 9. A tool has been developed to inspect the randomly generated shape planes with varying fill percentages and grid sizes.



30% 50% 70%

Figure 9: Examples of random neighbor weighted generated shape planes with varying fill percentages where the initial state contains connected edge cells on a 14×14 grid.

4.2 Results

For the font input, a triplet was built for each unique three-letter combination of the 26 letters, where repetition of the same letter in a combination is allowed. This results in a total of 3276 unique combinations. This process was carried out for each font thickness and connectivity type. The average error score and the percentage of perfect triplets over the 3276 combinations was recorded. The results can be observed in Table 1.

Looking at this table it is clear that a bold or thick font performs better than a thin one. The thick font has a significantly smaller error score than the thin font. The percentage of perfect (error score of zero) triplets for the thick font is 98.69%, which means that almost all three-letter combinations result in a perfect triplet. Compare this to the thin font, with only 16.27% of perfect triplets, and it is evident that a bolder font outperforms a thin font.

	Volume-connected		Edge-connected		Vertex-connected	
	Error score	% Perfect	Error score	% Perfect	Error score	% Perfect
Thin font	0.1629	16.27%	0.1629	16.27%	0.1558	16.30%
Medium font	0.0545	40.29%	0.0545	40.29%	0.0521	40.38%
Thick font	0.0003	98.69%	0.0003	98.69%	0.0003	98.69%

Table 1: The average error score and the average percentage of perfect triplets is shown for triplets built from the 3276 unique combinations of the 26 letters of the alphabet. The results are presented for each combination of connectivity type and font thickness.

For the randomly generated shape planes with varying fill percentage as input, the same pattern can be observed. Triplets were built using randomly generated shape planes, starting with a fill percentage of 30% and ending at 70%, with intervals of 5%. Multiple grid sizes were used to explore whether this has any impact on the error score. Grid sizes range from 14×14 to 30×30 with increments of 4. The connectivity constraint for the triplets is volume-connected. For each combination of fill percentage and grid size, 4000 triplets were built with three randomly generated shape planes. The results for the average error score can be seen in Figure 10, and the average percentage of perfect triplets in Figure 11.

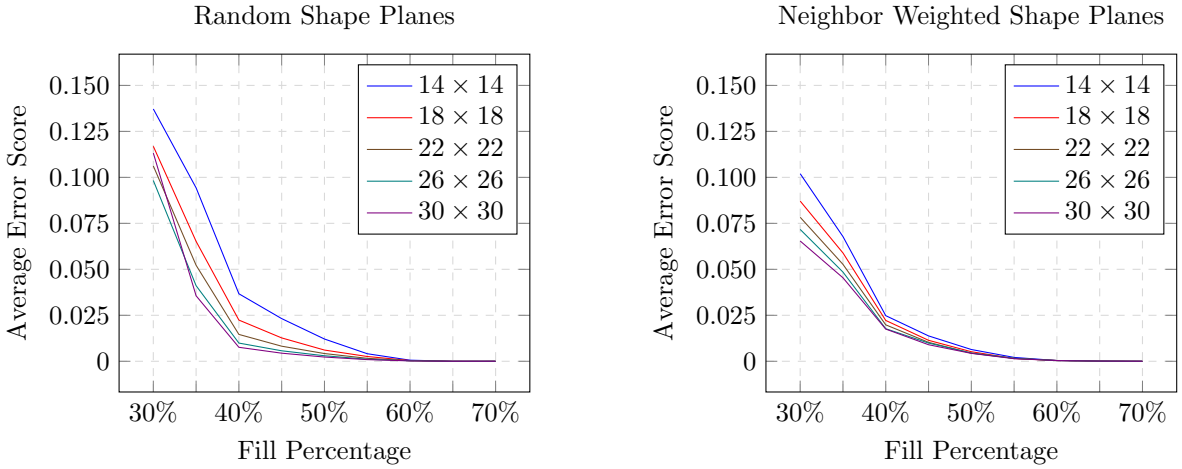


Figure 10: The average error score is plotted against the fill percentage of the input shape planes. Multiple grid sizes are utilized, ranging from 14×14 to 30×30.

These plots reveal that a higher fill percentage leads to triplets with a lower error score. For random shape planes, the initial error score at a fill percentage of 30% is slightly higher than that of the neighbor-weighted shape planes. In both cases, the error score rapidly decreases until a fill percentage of around 40% is used. For shape planes with a fill percentage of 60% or higher, the average error score reaches zero.

The plots depicting the average percentage of perfect triplets exhibit the same trend, showing a correlation between the increase in fill percentage and the percentage of perfect triplets. Around a fill percentage of 70% or higher, all triplets achieve an error score equal to zero.

Concerning grid size, it can be observed that there is minimal variation in the error score as the grid size increases. Both random and neighbor-weighted shape planes exhibit lower error scores with larger grid sizes compared to smaller ones. However, when examining the average percentage of perfect triplets, smaller grid sizes appear to be slightly more effective for random shape planes.

The decrease in error scores with higher fill percentages and larger grid sizes can be verified by calculating the error score using the fill percentage as a probability. The probability for an empty cell in a shape plane is denoted by f, representing the fill percentage as a probability (e.g., 0.3 or 0.5). An error occurs for this cell if it is impossible to cast a shadow on that location, determined by the other two shape planes. They must have disabled cells in the corresponding row or column of the filled cell.

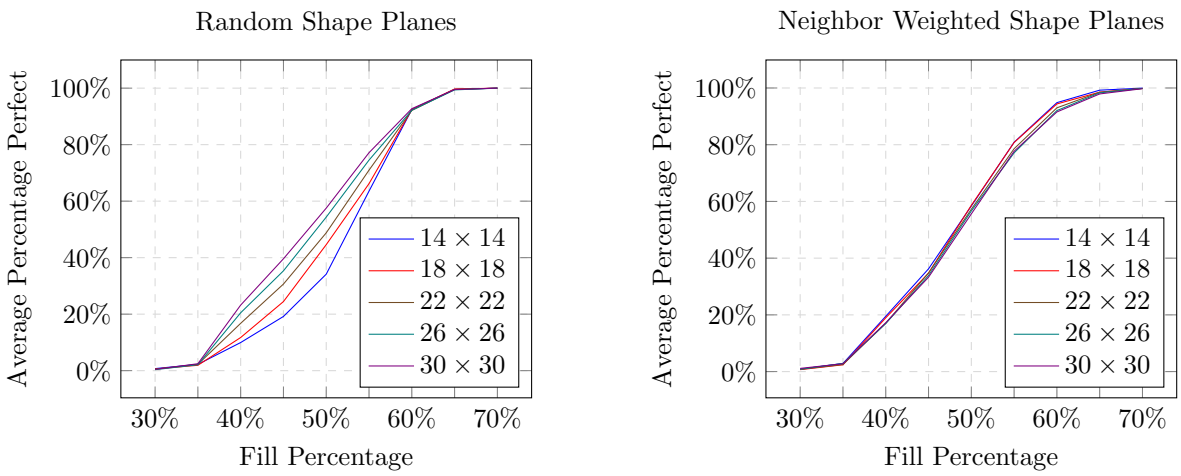


Figure 11: The average percentage of perfect triplets is plotted against the fill percentage of the input shape planes. Multiple grid sizes are utilized, ranging from 14×14 to 30×30.

In other words, the entire beam must be empty for the cell to result in an error. The probability for one cube in that beam to be empty is $1 - f^2$, as f^2 represents the probability of the cube being filled. The probability for the entire beam to be empty is $(1 - f^2)^g$, where g represents the grid size. The average error score can be approximated as $f \cdot (1 - f^2)^g \cdot 3$. Figure 12 illustrates this function with the same fill percentages and grid sizes as before.

This figure reveals a consistent trend of decreasing error scores with higher fill percentages, and larger grid sizes corresponding to lower error scores. However, the error scores for low fill percentages and small grids are higher than those for randomly generated shape planes. Large grid sizes show lower error scores at small fill percentages. That the error scores do not align exactly with those of generated shape planes can be attributed to the reality that the triplet shadow planes are not independent of each other, and to the fact that the calculated error score applies to all shape planes.

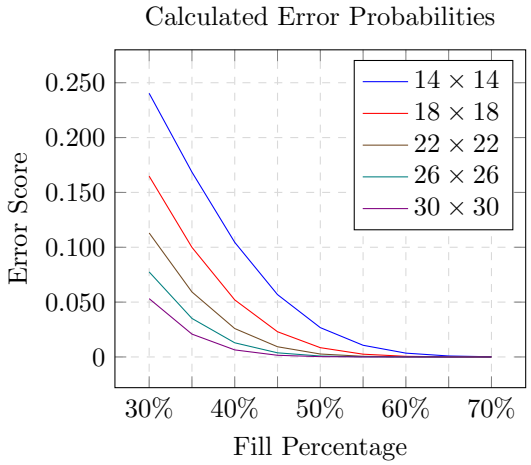


Figure 12: The calculated error score is plotted against the fill percentage of the input shape planes for grid sizes ranging from 14×14 to 30×30.

5 Removing More Cubes

The construction process of triplets produces an output with the maximum number of cubes while ensuring accurate shadow planes. This process involves cutting away, from each direction, an extrusion of the inverted desired shape plane. Specifically, an extrusion of the disabled cells in the shape plane.

In other words, an extrusion of the *inverted* shape plane is 'cut away' in each direction, allowing the enabled cells in the shape plane to remain as cubes in the triplet and cast shadows. Because of this the resulting triplet volume will always contain the maximum number of cubes. However, it could be possible to reduce the number of cubes while maintaining correct shadow planes. This section seeks to determine the minimum number of cubes required for the creation of perfect triplets using the 14×14 grid sized letters.

To answer this, cubes will be removed from perfect triplets created from the medium-boldness letter combinations, of which the font can be seen in Figure 20. There are 1344 perfect triplets for this font. Here, 'perfect' signifies triplets with an error score of zero. Since these triplets' shadow planes are error-free, the logical approach is to only remove cubes that preserve the shadow planes when removed. This ensures that the triplet stays error-free.

The process of selecting a cube for removal from the triplet grid volume involves a random choice from all possible removable cubes. As previously mentioned, a cube can only be removed if doing so maintains the same shadow planes, ensuring the triplet stays error-free. Additionally, cubes situated on the interior should be retained to avoid creating a hollow shape. It does not make sense to remove cubes on the inside since this would not be visible to the observer. It is also undesirable for 3D printing because it makes the shape harder to print and the mesh will inherently be hollow when printed depending on the infill structure generated by the 3D printing software. While hollowing might make sense in some contexts, like reducing the weight or adjusting the center of gravity, it is not applicable here. Therefore, only cubes with a face adjacent to an empty cube, or 'air', can be selected for removal.

Another aspect to keep in mind is that removing a cube might introduce edge-contacts. Edge-contacts appear when an edge shares more than two faces, see Figure 13. These edge-contacts make the model not be a manifold anymore.

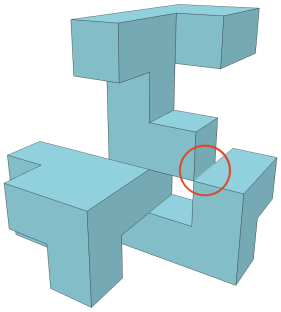


Figure 13: An example of an edge-contact in a triplet. The highlighted edge is shared by four faces.

A manifold 3D model is characterized by well-defined geometry at every point on its surface, presenting a smooth, continuous structure without self-intersections or irregularities. Slicing software, responsible for generating walls, infill, and support structures, and converting models into 3D printing instructions, prefers manifold models due to their ease of interpretation and processing. This preference ensures a more accurate and reliable printing process. Conversely, non-manifold models may show issues like self-intersections, holes, or irregularities. Such challenges make it difficult, or even impossible, for slicing software to process the model into 3D printing instructions. For this reason, cubes that cause edge-contacts when removed, are not permitted to be removed.

Using these constraints for cubes that can be removed, a set can be computed containing all possible cubes for removal. From this set, a cube is randomly selected and removed from the triplet volume. Then, the set will be recalculated and another cube will be removed. This continues until the set is empty, meaning that there are no cubes that are on the outside and can be removed without altering the shadow planes or introducing edge-contacts.

Utilizing random selection, the number of cubes within a triplet after consecutive removals until further removal is impossible, may not reach the absolute minimum. Instead, it represents a local minimum. To approach closer to the absolute minimum, the minimum value is used from ten runs for each medium-boldness letter combination.

Upon removing cubes from all letter combinations, the results for ten letter combinations are summarized in Table 2. The average minimum number of cubes across all triplets is 333, with an

average maximum number of cubes removed being 390. See Figure 14 for a visualisation of three example triplets before and after removing cubes.

Letters	Initial	Removed	Minimum
AGR	775	437	338
BFG	662	339	323
BLW	677	342	335
CRW	870	508	362
DNU	990	582	408
.	.	.	.
.	.	.	.
.	.	.	.
FLM	616	303	313
GNN	991	564	427
HIP	542	257	285
HLN	680	334	346
MPS	713	398	315
Average	**723**	**390**	**333**

Table 2: The initial and (best local) minimum number of cubes for ten triplets from perfect medium-boldness letter combinations. The best local maximum number of removed cubes is shown and the average over all 1344 perfect medium-boldness letter combinations.

Upon reviewing these results, it becomes evident that the minimum number of cubes required to maintain perfect shadow planes is relatively low. Considering the total number of cubes that could potentially be used, calculated as $14 \cdot 14 \cdot 14 = 2744$, the average minimum number of cubes needed, 333, accounts for only 12.1% of this total. However, it is not practical to use all cubes, as doing so would result in a square cube with all shadow planes being a square as well. But, these results suggests that, on average, more than half of the initial number of cubes can be removed.

This indicates that a triplet constructed using the method outlined here—taking intersections of the three input shape planes—retains a significant number of cubes that could still be removed while maintaining correct shadow planes. This leaves room to further modify a triplet, for example, enhancing its complexity and visual appeal.

6 Reducing Recognisability of Triplets

In the previous section, it became clear that, on average, more than half of the initial cubes could be removed for perfect medium-boldness letter combinations. However, upon examining the three example triplets in Figure 14, it could be said that attempting to remove the maximum number of cubes, results in thin, erratic, and unappealing triplets. Consequently, these triplets are also difficult to 3D print and are fragile due to thin structures. This section aims to make triplets more interesting by selectively removing only certain cubes.

What makes a triplet interesting? Perhaps its size, the shapes of its projected shadows, or keeping the center of gravity in the middle. Arguably the most intriguing aspect of a triplet is its ability to cast three different shadows, which may not be immediately apparent when examining the triplet object itself. The observer must observe the triplet in the correct orientations to view all three shapes. However, sometimes it is easy to recognize all shapes at a glance in a triplet. For instance, in the triplets shown in the left column of Figure 14, it is straightforward to identify the letter `P` in the `MPS` triplet as well as the letters `B` and `F` in the `BFG` triplet, and `R` in the `WRC` triplet.

6.1 Reducing Number of Cubes in a Slice

This motivates the objective of making triplets less recognizable by reducing the number of cubes in a slice. A slice, in this context, refers to all the cubes within a plane of the triplet volume while keeping

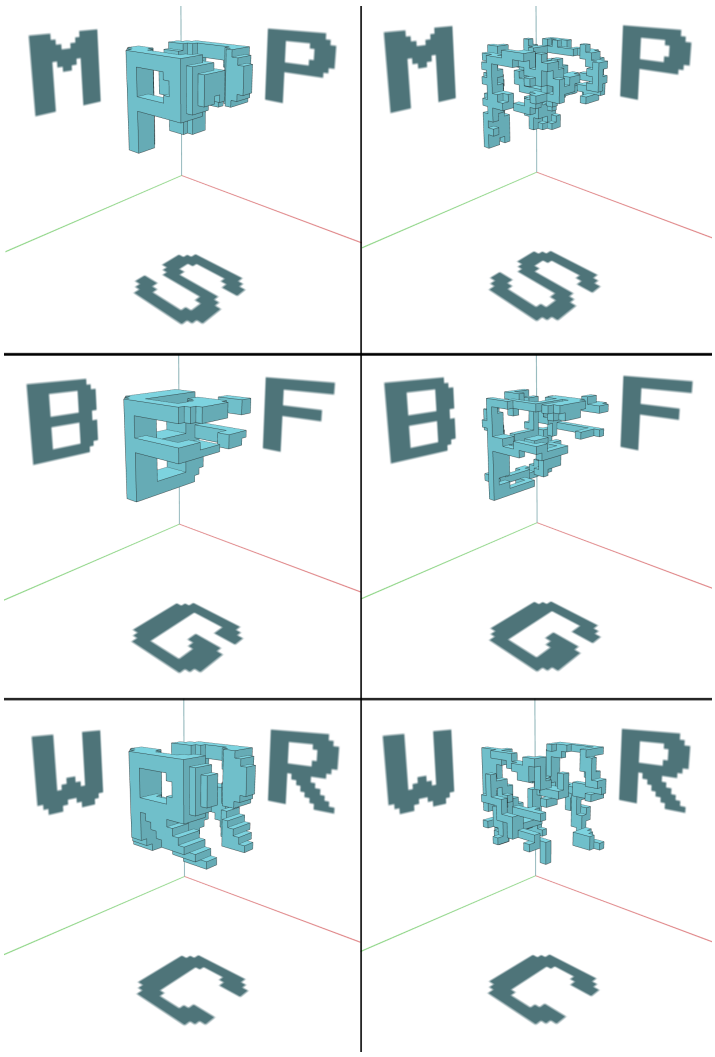


Figure 14: Examples of three triplets from the letters `MPS`, `BFG` and `WRC`, before (left column) and after (right column) removing the (local) maximum number of cubes.

the other axis constant. For example, a slice could encompass all the cubes in the XY plane where z equals 3. If most of the cubes of a shape plane are contained within one slice, an observer only needs to examine this plane to determine which shape is being displayed. Even from an angle, it can be easy to recognize. By reducing the number of cubes in a slice, the shape will be formed by structures in the triplet which are further away in depth. As a result, the observer must hold the triplet in precisely the right orientation for the structures to align perfectly, revealing the intended shape.

To achieve this objective, cubes for removal will not be selected at random but instead will use a weight to determine the probability of being selected. Each cube that is allowed to be removed will be assigned a weight based on the number of cubes in that slice of the triplet. More precisely, the weight will be calculated as the ratio between the number of cubes in the slice and the number of cells in the corresponding shadow plane. When the number of cubes in the slice equals the number of cells in the shadow plane, the slice will exactly resemble the shadow plane, and the weight will be 1, indicating that removing a cube in that slice is preferable. Calculating the ratio of the number of cubes in a slice to the number of cells in the corresponding shadow plane ensures equal weighing regardless of the shape's fill percentage, whether high or low. For instance, the letter `I` with a low fill percentage and the letter `M` with a high fill percentage will be weighted equally. Since there are three planes, three weights will be determined, and their average will yield a final weight ranging from 0 to 1. The full equation will be:

$$SliceWeightCube_{ijk} = \left(\frac{\sum_{x=0}^{g}\sum_{y=0}^{g} cube_{xyk}}{cellsShadowPlane_{XY}} + \frac{\sum_{x=0}^{g}\sum_{z=0}^{g} cube_{xjz}}{cellsShadowPlane_{XZ}} + \frac{\sum_{y=0}^{g}\sum_{z=0}^{g} cube_{iyz}}{cellsShadowPlane_{YZ}}\right) \cdot \frac{1}{3}$$

Here, $\sum_{x=0}^{g}\sum_{y=0}^{g} cube_{xyk}$ can be seen as the sum of cubes in the XY plane slice with depth k and grid size g. $cellsShadowPlane_{XY}$ represents the number of enabled cells in the XY shadow plane. Note that this weight depends on other cubes in the triplet volume. This means that when a cube is removed, the weights for other cubes that depend on the removed cube need to be recalculated.

6.2 Avoiding Removing Cubes that Introduce Edges

Another improvement that could be made while removing cubes, is to keep the number of edges to a minimum. Since having a lot of edges can make the triplet look erratic and messy, it is preferable to keep flat surfaces intact. For example, when there is a flat surface section on the outside of the triplet, removing a cube from this area might be deemed unsightly and irregular. It would introduce a hole. This removal would increase the number of edges surrounding the affected cube from 0 to 12, as depicted in Figure 15. Removing a cube which keeps the number of edges of a triplet minimal is preferable.

Note that here, each edge has a unit length, so the triplet is seen as the union of unit cubes. For example, a rectangular parallelepiped with a width, height and length of $1 \times 1 \times 3$, would have $4 + 4 + 3 \cdot 4 = 20$ edges.

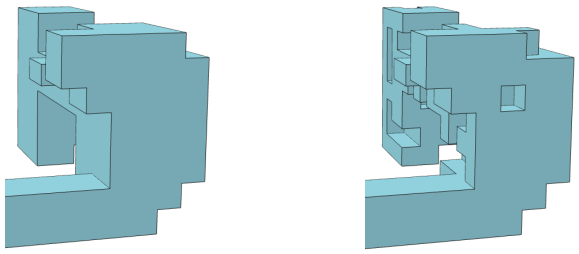


Figure 15: An example where a cube is removed from a flat surface.

The weight to keep the number of edges minimal is calculated by determining the difference in edges when a cube is removed. For example, the optimal choice would be to remove a cube situated on an otherwise flat surface. Removing such a cube would decrease the number of edges by 12. Conversely, removing a cube from a flat surface as depicted in Figure 15, would add 12 edges. Therefore, the difference in edges when removing a cube ranges from -12 to $+12$.

To ensure a value between 0 and 1, the weight for each cube is determined as follows:

$$EdgeWeightCube_{ijk} = \frac{12 - \Delta Edges}{24}$$

where $\Delta Edges$ stands for the difference in number of edges of a triplet after removing $cube_{ijk}$.

Note that removing a cube from a flat surface will result in a $\Delta Edges$ value of 12, indicating that the weight of such a cube is 0. Therefore, any cube that would increase the number of edges by 12 will not be selected for removal. This precaution is intentional, as it is undesirable to remove a cube from a flat surface.

6.3 Combining Weights

Now, there are two weights for each cube, a slice fill ratio weight and a number of edges weight. These need to be combined to get one final weight and use that as a probability for selecting a cube for removal. The most straightforward approach is to just take the average of the two. But, when looking at the resulting triplet shape, it was found that the difference to a triplet shape which used random selection cube removal, was minimal, see Figure 16a and 16b. The slice fill ratio is visibly reduced but it still exhibited cubes removed from flat surfaces and looked erratic and scraggly.

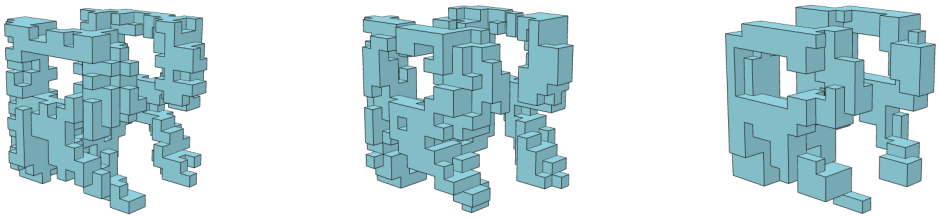
The two weights can also be combined using a ratio. For example, 20% can be taken from the slice fill ratio weight and 80% from the edge weight. The weights can also be scaled, allowing for the strengthening of the weights, making it even less likely for a cube to be chosen for removal if it introduces edges or resides in a slice with few cubes.

Using this, an equation can be constructed as follows:

$$FinalWeight = (r \cdot EdgeWeightCube + (1 - r) \cdot SliceWeightCube)^p \tag{1}$$

Where r expresses the ratio between $EdgeWeightCube$ and $SliceWeightCube$ when combined. p determines how strong the weights are, meaning initially small weights are even small, and large weights are even larger.

To determine the best values for r and p, exploration of various combinations was undertaken in the developed application to find the optimal balance between reducing the number of cubes in a slice and keeping the number of edges to a minimum. Optimal would be to create a triplet of which the shapes are not easy to recognise when looking at the triplet shape, but the triplet also does not look erratic due to a lot of edges. While fine-tuning these numbers, care was taken to ensure that the triplet did not contain excessively fragmented structures and that it remained visually pleasing. After experimentation, $r = 0.6$ and $p = 9$, resulted in visually interesting triplets that were not erratic, see Figure 16c.



(a) (b) (c)

Figure 16: Examples of triplets where 250 cubes are removed with different weight calculations for selecting cubes for removal. (a) Random selection. (b) Weighted with $SliceWeightCube$ and $EdgeWeightCube$ combined using the average of the two. (c) Weighted using Equation 1 with $r = 0.6$ and $p = 9$.

6.4 Weighted Removing Results

To assess the difference between removing cubes using weights and randomly removing cubes, an experiment was conducted. This involved trying to remove 250 cubes from all perfect triplets generated from medium-boldness letter combinations. It was chosen to remove 250 cubes because this number is less than the average minimum required cubes (see Table 2), but still high enough to affect the triplet shape sufficiently. Note that there might be triplets for which the maximum number of cubes to remove is less than 250. In these cases the maximum number was taken. The average number of removed cubes over all letter combinations is 244 for random removal and 247 for weighted removal, which is close to the desired 250.

The number of edges and the maximum slice fill ratio of all slices were recorded before and after removing the cubes. These numbers were averaged over ten runs for each three-letter combination. See Table 3 for the results from ten randomly selected letter combinations, along with the average over all 1344 perfect medium-boldness letter combinations.

Upon examining this table, it becomes evident that for both random and weighted removal, the slice ratio decreases, and the number of edges increases after removing 250 cubes. The increase in number of edges makes sense because the initial triplet exhibits a substantial number of flat surfaces, see Figure 17a. The difference between random removal and weighted removal is minimal concerning the maximum slice fill ratio. The slice fill ratio for weighted removal is slightly lower, indicating that the weight did indeed have an effect. The marginal difference could stem from the fact that there are fewer cubes that can be removed without compromising the shadow planes around a slice fill ratio of

	Before removing		Random removed 250		Weighted removed 250	
	Slice ratio	**Edges**	**Slice ratio**	**Edges**	**Slice ratio**	**Edges**
AGR	0.91	952	0.64	1758	0.61	966
BFG	1.00	580	0.72	1412	0.62	742
BLW	1.00	476	0.74	1438	0.63	705
CRW	1.00	1028	0.79	1895	0.73	1006
DNU	1.00	772	0.85	1926	0.83	954
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
FLM	1.00	372	0.74	1266	0.66	685
GNN	1.00	1016	0.81	2072	0.79	1107
HIP	1.00	464	0.69	1200	0.60	725
HLN	1.00	408	0.74	1394	0.66	676
MPS	0.94	902	0.67	1585	0.58	937
Average	**0.99**	**705**	**0.73**	**1545**	**0.66**	**852**

Table 3: The maximum ratio of number of cubes in a slice to the number of cells in the corresponding shadow plane and the number of edges, averaged over ten runs for each medium-boldness three-letter combination. Values are shown from before removing cubes and after removing 250 cubes randomly and using weights with Equation 1 with $r = 0.6$ and $p = 9$. The average is shown from all 1344 perfect letter combinations.

0.7. Alternatively, it could be that randomly removing cubes is already effective in reducing the slice fill ratio.

However, upon evaluating the number of edges, weighted removal results in significantly fewer edges than random removal. The average number of edges for weighted removal is slightly higher than before removal, and even lower for some letter combinations like CRW. Although it is slightly higher, the slice fill ratio is reduced substantially. This demonstrates that the weight had a pronounced effect on the triplet shape.

When inspecting various example triplet shapes, as depicted in Figure 17, it becomes evident that utilizing weighted removal yields different triplet shapes than random removal. The triplets generated through random removal could be characterized as a disorganized assemblage of cubes, whereas those formed via weighted removal exhibit smoother surfaces, are tidier and more structured.

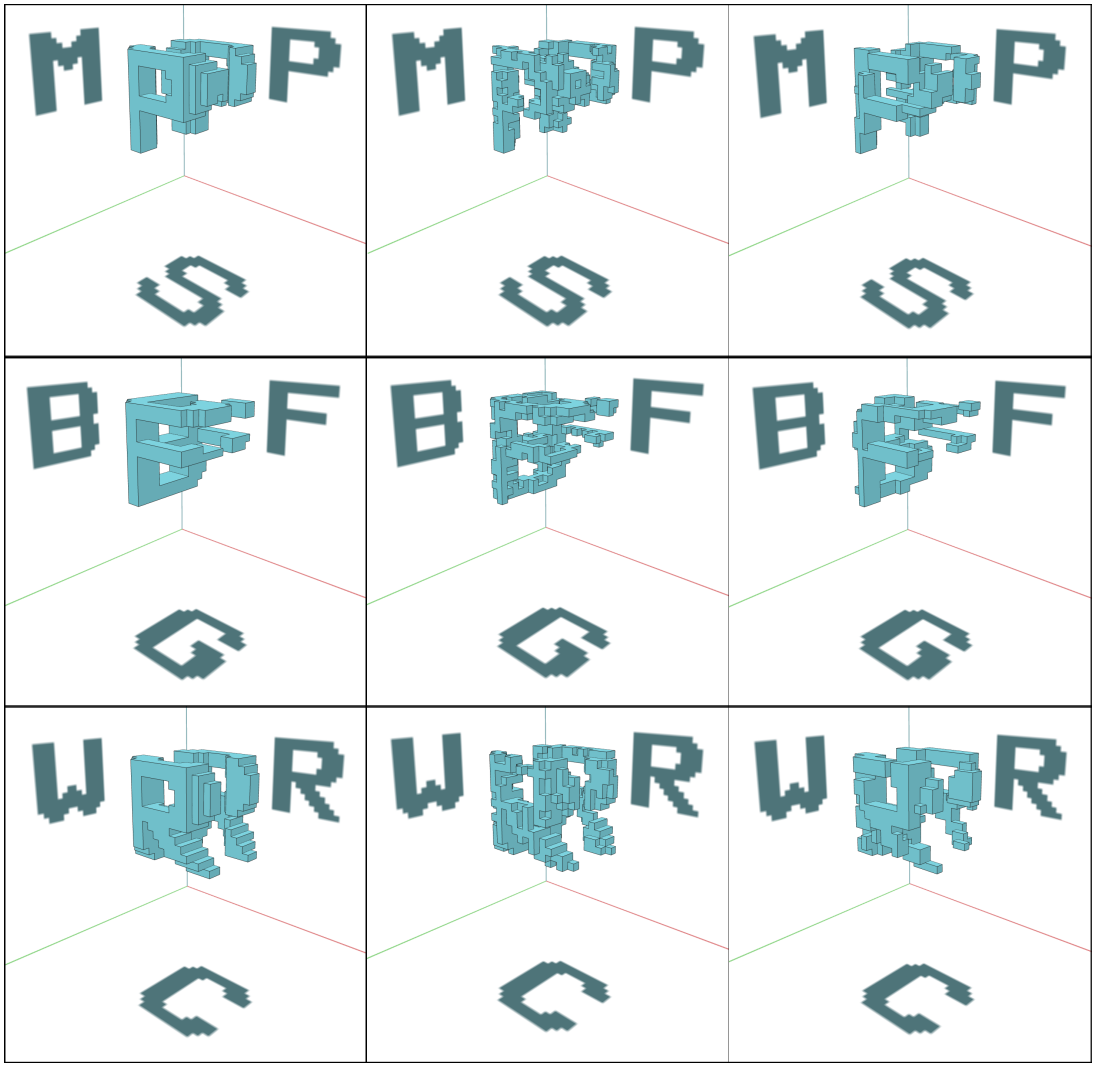
7 Conclusions

This research examined the intersection of mathematics with art and font design, presented a method to construct and assess triplets, investigated the impact of input shape fill percentage on triplet error scores, and explored a technique to enhance the visual appeal of triplet shapes. Additionally, an application was developed to play and experiment with triplets.

Initially, an overview of the history of mathematics, perspective, and computer science in art and font design was provided. This overview highlighted the increasing popularity of mathematical and graphical computer science concepts, encompassing topics such as the golden ratio, the Mandelbrot set, fractals, and harmonographs. Additionally, the significance of shadows in art projects that involve illusions was discussed, along with the evolution of font design from traditional calligraphy to modern fonts tailored for screen usage. These three concepts resulted in the introduction of the triplet—a 3D shape projecting three shadows.

To explore the influence of input shape thickness on triplet error scores, a monospaced pixel font with three distinct boldness styles was developed. Subsequently, a technique for randomly generating semi-realistic input shapes was developed and utilized.

A correlation between the 'thickness' of input shapes and triplet error scores was found. Specifically,



(a) (b) (c)

Figure 17: Examples of triplets where 250 cubes were removed. Column (a) shows the triplets without removal, column (b) after random removal and column (c) after weighted removal using Equation 1 with $r = 0.6$ and $p = 9$.

higher fill percentages of input shapes resulted in lower error scores, with fill percentages of 70% or higher yielding perfect triplets with zero errors. Surprisingly, larger grids for input shapes led to marginally lower error scores, contrary to initial expectations. This might be against intuition but can be explained by noting that the larger the grid, the lower the influence of a single erroneous cell on the total error score. These findings were further validated through calculations of expected error scores based on probabilities for various fill percentages.

The technique used to construct triplets gives a shape that uses the maximum number of cubes for correct shadows. Remarkably, for perfect triplets generated from the medium boldness font, it was possible to remove half of the initially required cubes while retaining correct shadows. This creates room for experimentation with triplet shapes.

So, efforts were made to enhance the visual appeal of triplets by adjusting the ratio of cubes in a slice to the number of cells in the corresponding shadow. This involved using weights that consider both the maximum slice fill ratio and the number of edges of a triplet. By removing 250 cubes from medium boldness letter triplets using these weights, it was possible to create triplets characterized as more enigmatic, yet still structured, compared to randomly removing the same number of cubes.

7.1 Future Research

Although this research has explored intriguing aspects of triplets, there are opportunities for further investigation to broaden the understanding of triplet creation.

Presently, letters and randomly generated shapes serve as input. While these encompass a diverse range of shapes, gaining insights from commonly used input shapes, such as symbols, primitives, and other signs converted into a 2D cell grid, could provide additional value. While the initial aim was to generate shapes randomly to cover all potential forms, using large grid sizes led to the emergence of recurring patterns. It may be beneficial to employ a set of primitive shapes and randomly selecting a few for placement at various rotations and locations on the grid. This approach could offer a broader variety and more realistic shapes for input.

Limitations arise from the use of a grid for both input and output shapes. For instance, smooth curves cannot be expressed in a triplet, and fine details are constrained by the input grid size. Although increasing the input grid size could give higher resolution, it may significantly impact performance. Alternatively, allowing triangles within input shape cells, composed of three of the four corners of a cell, could facilitate diagonal structures and slightly rounded shapes. See Figure 18 and this demo application for an example.

Another limitation is the lack of exploration into combining different input shape dimensions to create non-cubical triplets. For example, combining input shapes with dimensions $n \times m$, $n \times p$, and $m \times p$ could yield a valid triplet with dimensions $n \times m \times p$. This would allow the usage of a font where letters have different widths.

To further reduce the error score and accommodate various input shape dimension combinations, incorporating thin cells between each cell of the input shapes could be beneficial. This would enable thin structures to display parts of one shadow without interfering with the other two, see Figure 18 for an example.

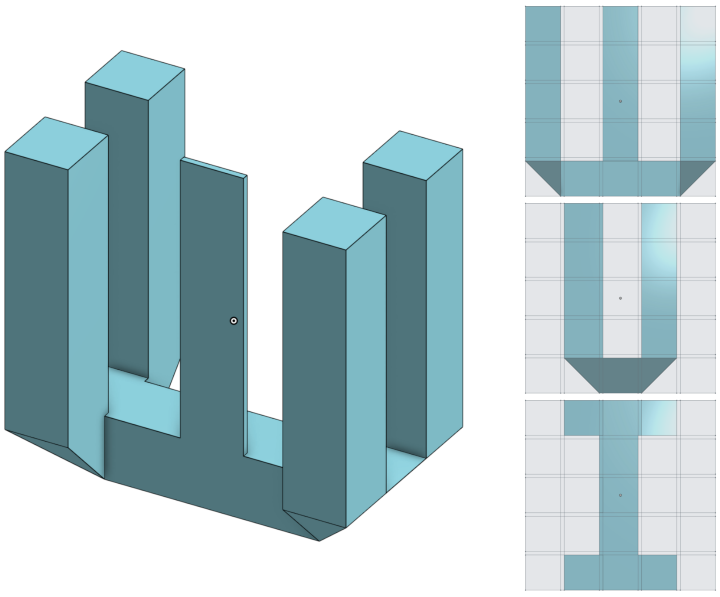


Figure 18: An example were thin intermediary cells are utilized to make the letter combination `WUI` possible, where the `U` and `I` are narrower than the `W`. Half cells are also used to accommodate the `U` and `W` shape.

Exploring alternative methods for cube removal is another potential research topic. Removing predefined structures at a time, rather than individual cubes, could improve the overall visual appeal of triplets by eliminating protruding cubes. Additionally, one could consider the physical properties of triplet shapes, such as the center of gravity. By strategically removing cubes to maintain the center of gravity at the center, triplets could be suspended on a wire and rotated to display all three shadows sequentially using a single light source. Furthermore, optimizing triplets for 3D printing by minimizing the need for support structures presents another direction for future research.

References

[1] Pankaj K. Agarwal and Joseph O'Rourke. "Computational Geometry Column 34". In: *International Journal of Computational Geometry and Applications* 08 (1998), pp. 637–642.

[2] Aries Arditi and Jianna Cho. "Serifs and font legibility". In: *Vision research* 45.23 (2005), pp. 2926–2933.

[3] Michael Barnsley, John E Hutchinson, and Örjan Stenflo. "V-variable Fractals and Superfractals". In: *arXiv preprint math/0312314* (2003).

[4] Roy R Behrens. *Fonts and Logos: Font Analysis, Logotype Design, Typography, Type Comparison, and History*. 2001.

[5] Fan Pen Chen. "Shadow theaters of the world". In: *Asian Folklore Studies* (2003), pp. 25–64.

[6] Erik D Demaine and Martin L Demaine. "Fun with fonts: Algorithmic typography". In: *Theoretical Computer Science* 586 (2015), pp. 111–119.

[7] Keith Devlin. "The myth that will not go away". In: *Mathematical Association of America* (2007).

[8] Casper van Dommelen, Marc van Kreveld, and Jérôme Urhausen. "Spiroplots: a New Discrete-time Dynamical System to Generate Curve Patterns". In: *Bridges 2020 Conference Proceedings*. Tessellations Publishing. 2020, pp. 353–360.

[9] Stephan Füssel. "Gutenberg and today's media change". In: *Publishing research quarterly* 16.4 (2001), pp. 3–10.

[10] James L Hunt, BG Nickel, and Christian Gigault. "Anamorphic images". In: *American Journal of Physics* 68.3 (2000), pp. 232–237.

[11] Prutsdom Jiarathanakul. *Ray marching distance fields in real-time on webgl*. Tech. rep. Citeseer, 2012.

[12] JJA Keiren, Freek van Walderveen, and Alexander Wolff. "Constructability of trip-lets". In: *Abstracts 25th European Workshop on Computational Geometry (EuroCG'09, Brussels, Belgium, March 16-18, 2009)*. 2009, pp. 251–254.

[13] Mario Livio. *The Golden Ratio: The story of phi, the world's most astonishing number*. Crown, 2008.

[14] Thomas Lowe. *What is a Mandelbox*. 2010. URL: https://sites.google.com/site/mandelbox/what-is-a-mandelbox.

[15] Matthew Luckiesh and Frank Kendall Moss. *Reading as a Visual Task*. D. Van Nostrand Company, Incorporated, 1942.

[16] John Man. *The Gutenberg Revolution*. Random House, 2010.

[17] Benoit B Mandelbrot. *The Fractal Geometry of Nature*. Vol. 1. WH freeman New York, 1982.

[18] Benoit B Mandelbrot, Carl JG Evertsz, and Martin C Gutzwiller. *Fractals and Chaos: the Mandelbrot set and beyond*. Vol. 3. Springer, 2004.

[19] Marshall McLuhan et al. *The Gutenberg Galaxy: The making of typographic man*. University of Toronto Press, 2011.

[20] Carmen Moret-Tatay and Manuel Perea. "Do serifs provide an advantage in the recognition of written words?" In: *Journal of Cognitive Psychology* 23.5 (2011), pp. 619–624.

[21] WI Newman, DL Turcotte, and AM Gabrielov. "Fractal trees with side branching". In: *Fractals* 5.04 (1997), pp. 603–614.

[22] Inge C Orr. "Puppet theatre in Asia". In: *Asian Folklore Studies* (1974), pp. 69–84.

[23] Donald Gildersleeve Paterson and Miles Albert Tinker. *How to make type readable*. 1940.

[24] James Pommersheim. *Number Theory: A Lively Introduction with Proofs, Applications, and Stories*. John Wiley & Sons, 2010.

[25] Clifford A Reiter. *Sierpinski Fractals and GCDs*. Elsevier, 1998, pp. 169–175.

[26] Tim Roggen. *Trip-lets: Constructability of trip-lets in theory and in practice*. Dept. of Information and Computing Science, Utrecht University, 2023.

[27] Dan Romik. "Shortest paths in the Tower of Hanoi graph and finite automata". In: *SIAM Journal on Discrete Mathematics* 20.3 (2006), pp. 610–622.

[28] Vinícius da Silva and Tiago Novello. "Real-time rendering of complex fractals". In: *Ray Tracing Gems II: Next Generation Real-Time Rendering with DXR, Vulkan, and OptiX* (2021), pp. 529–544.

[29] Andy Sloane. *Have a donut.* 2006. URL: https://www.a1k0n.net/2006/09/15/obfuscated-c-donut.html.

[30] Ian Stewart, WS Anglin, and Kim Williams. "The mathematical tourist". In: *The Mathematical Intelligencer* 19 (1997), pp. 39–49.

[31] Victor I Stoichita. *Short History of the Shadow.* Reaktion Books, 1997.

[32] Robert J Whitaker. "Harmonographs. ii. circular design". In: *American Journal of Physics* 69.2 (2001), pp. 174–183.

[33] Daniel White. "Further exploration of the 3D mandelbulb". In: *The Unravelling of the Real 3D Mandelbrot Fractal* (2009). URL: https://www.skytopia.com/project/fractal/2mandelbulb.html.

[34] Endymion Porter Wilkinson. *Chinese History: a manual.* Vol. 52. Harvard University, Asia Center, 2000.

[35] Tom Wright. "History and technology of computer fonts". In: *IEEE Annals of the History of Computing* 20.2 (1998), pp. 30–34.

A Font

Figure 19: Thin version of the font.

Figure 20: Medium version of the font.

Figure 21: Thick version of the font.

B Triplet Designer Application

An application has been developed to create and visually inspect triplets. The goal was to create an application that is user friendly and is easy to understand. It can be viewed at www.sivanduijn.com/triplet-designer.

It contains functionality to draw your own input shape planes on a user specified grid size. It is also possible to create triplets from letters using the font described in Appendix A with three boldness levels. The user can also randomly generate shape planes. Here, the neighbor weighted random generation is used. Cube removal is supported where the weighted removal is used. The created triplet shape can be exported to a .stl file.

Heavy calculations, like triplet creation and cube removal, are done in web assembly which is compiled from Rust code.