Fall 2012

Three dimensional volumetric display

Scott Matthew Cook
James Madison University

Three Dimensional Volumetric Display

A Project Presented to

the Faculty of the Undergraduate

College of Integrated Science and Technology

James Madison University

in Partial Fulfillment of the Requirements

for the Degree of Bachelor of Science

by Scott Matthew Cook

Dec 2012

Accepted by the faculty of the Department of ISAT, James Madison University, in partial fulfillment of the requirements for the Degree of Bachelor of Science.

FACULTY COMMITTEE:

Project Advisor: Teate, Anthony Ph.D.
Professor, Integrated Science and Technology

Reader: Henriksen, Paul
Professor, Integrated Science and Technology

Reader: Radzwill, Nicole Ph.D.
Professor, Integrated Science and Technology

HONORS PROGRAM APPROVAL:

Barry Falk, Ph.D.,
Director, Honors Program

Contents

List of Figures

Figures

Images

Appendix

Acknowledgements

Special thanks needs to be given to the following people for helping with the idea, design, and build of this project. Without their help, the necessary performance and features of this project would not have been reached.

Dr. Anthony Teate – Advisor

Mark Starnes – Machine Shop

Morgan Hager – Machine Shop

Brad Fisher – Design / Build Support

Brian Walton – Mathematics Department

Abstract

Three dimensional (3D) displays are part of a growing market. These 3D displays are often seen in home entertainment and film industries. The most common three dimensional techniques require the use of glasses with special lenses to add the effect of "depth" in an animation. This project attempts to prove that a true three dimensional display can be made using low cost and accessible materials. The display is made from 625 Light Emitting Diodes (LEDs) on a Peggy 2LE LED matrix display. The Peggy 2LE board is spun around 15 Hz and held tight between two cylindrical bearings. Blender, an open source animation software, in tandem with Mathematica, a technical computing application, was used to convert three dimensional animation objects into a precise sequence of LED flashes. The sequence of abrupt LED flashes over the sweeping volume of the LED matrix created the appearance of a true three dimensional object. The three dimensional display created in this project had almost no loss of occlusion, enabling the display to be viewed with near perfect perspective at all angles without the need of special glasses. This report will include an introduction explaining how volumetric displays work as well as what other 3D technologies exist today. Then a generalized methodology explaining how the project was put together and finally the results of the project will be discussed.

Three Dimensional Volumetric Display

Introduction

There is a growing market for new applications of displays in media, entertainment, advertising, and marketing programs. A common and cheap method for displaying media content is through Light Emitting Diode (LED) arrays. These displays are relatively cheap and energy efficient. The main advantage to using LED displays is their massive scalability. For example, the Beijing 2008 Olympics used a 127m x 44m LED display for the opening ceremonies (Epistar Corp.). There are many applications for two-dimensional LED arrays. For example, LED displays are placed outside shopping centers to display advertisements for the companies within the center. Many LED displays are also used in sports arenas, downtown areas, and large conventions simply for entertainment and aesthetics for the public walking or driving through the venue. The newest development and research in multimedia displays is the addition of a third dimension.

The idea of creating a three-dimensional (3D) display is not new. However, many companies are pouring money into the development of better 3D technologies. One of the limitations to these technologies is that the user is required to wear special optic glasses to enable the illusion of adding depth to the video or image being displayed. Most 3D technologies today are transmitted onto a flat viewing plane. This only enables the “3D” image to be viewed from a limited number of angles. Movies, television shows, and even sporting events are now broadcasted in 3D (Matusik, Wujciech).

Companies and institutions such as SONY ®, RealFiction ®, and The University of Southern California (USC), Institute for Creative Technologies have explored the applications of 3D image processing. Their products have no loss of occlusion, enabling the display to be viewed with near-perfect perspective at all angles without the need of special glasses (Lawler, Richard). Each company uses a completely different approach for achieving true 3D visualizations. RealFiction has multiple commercial products that are sold at over $3,190 for base models. This project will combine the scalability, low-cost, and power efficiency of LED displays, with the growing technology of true three-dimensional displays to produce a unique 3D LED display.

Methods and Materials

This project made use of a handful of software applications as well as physical materials such as motors, aluminum framing, and bearings. The three main pieces of software included Blender, an open source animation software suite; Mathematica, a technical computation application, and Arduino, "an open-source electronic prototyping platform" (arduino.org). The methodology for this project will be broken up into two segments: Software and Hardware. The software section will explain how the Blender, Mathematica, and Arduino software was used. The hardware section will explain the components of the Peggy 2LE and the aluminum framing built to hold the project together.

Software

This section will walk through how a three dimensional cube was able to be translated from animation software to a virtual representation of that figure on the physical 3D volumetric display.

Blender

Blender is a completely open source animation software suite that allows users to create very complex animations. Blender was used in this project to design and animate geometric figures, which were later turned into the visualizations shown on the spinning LEDs.

Blender has a very steep learning curve. However, it was capable for producing all of the geometric figures needed in this project for representation on the physical project. Blender uses three dimensional objects which contain information about the vertices, faces, and textures of that object. These objects are referred to as mesh objects and are fully editable. For example, a mesh object can be scaled, rotated, skewed, extruded, animated, and so on. However, the most important data needed for the LED display was the vertices' locations. Wavefront OBJ files are a very friendly format that breaks a three dimensional objects down into its basic structures such as lines, vertices, faces, and so on. A three dimensional object, such as a cube, is inserted into a project, and the Wavefront file is exported as shown in Figure 1.

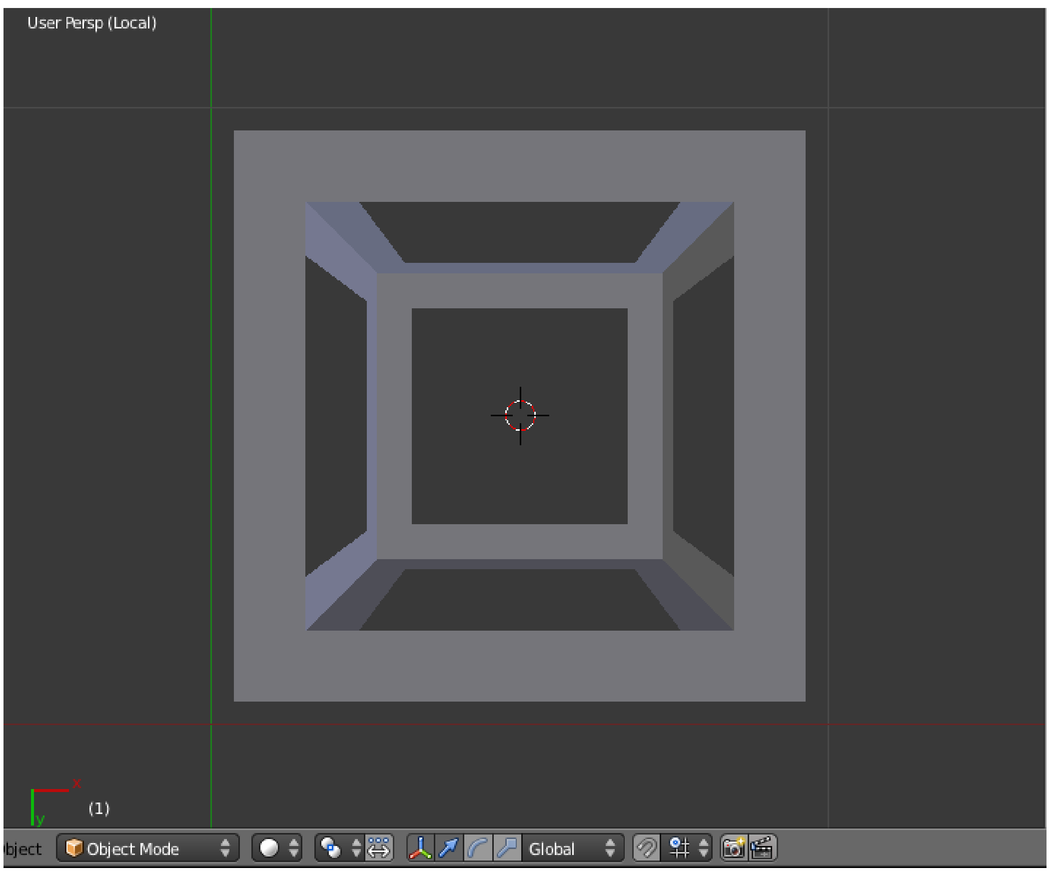


FIGURE 1 SHOWS A CUBE USED IN BLENDER WHICH WAS LATER EXPORTED AS A WAVEFRONT FILE

Mathematica

The next step was to export the vertex data from Blender, and import that data into Mathematica. A typical mesh object can contain around 500 vertices. The vertex data, once imported into Mathematica, looks as follows:

$$\begin{pmatrix} 0.140946 \\ -0.292049 \\ 0.207951 \end{pmatrix} \begin{pmatrix} 0.140946 \\ -0.343413 \\ 0.270539 \end{pmatrix} \begin{pmatrix} 0.140945 \\ -0.381581 \\ 0.341944 \end{pmatrix} \begin{pmatrix} 0.140945 \\ -0.405084 \\ 0.419424 \end{pmatrix} \begin{pmatrix} 0.140945 \\ -0.41302 \\ 0.5 \end{pmatrix}$$

The figure above shows five sets of data coordinates. Each matrix contains an X, Z, and Y component respectively. Notice that all of these values are between the values of {0,1} for X and Y; and between {0,-1} for Z. An animation space and reference was chosen in Blender that represented the volumetric space for the physical volume that the Peggy 2LE LED board swept as it spun. There needed to be a conversion from Blender units to the {0,25} Peggy 2LE board scale. This was done by taking the absolute value of each number in the matrix, and multiplying it by 25 as shown below. Then the "ceiling" was taken for each number which rounds a real number into its upper limit as an integer.

$$\begin{pmatrix} 4 & 0 & 3 \\ 4 & -2 & 3 \\ 4 & -3 & 3 \\ 4 & -5 & 4 \\ 4 & -7 & 6 \\ 4 & -8 & 7 \\ 4 & -9 & 9 \\ 4 & -10 & 11 \\ 4 & -10 & 13 \\ 4 & -10 & 15 \\ 4 & -9 & 17 \\ 4 & -8 & 19 \\ 4 & -7 & 20 \\ 4 & -5 & 22 \\ 4 & -3 & 23 \\ 4 & -2 & 23 \\ 4 & 0 & 23 \\ 4 & 3 & 23 \\ 4 & 4 & 23 \\ 4 & 6 & 22 \end{pmatrix} \ldots$$

After the conversion into the Peggy scale of 0 to 25, some "massaging" of the data had to be done to get it into the correct format for processing. As seen above, the new list of vertex points was in one single list, undifferentiated. This list could have errors and they had to be eliminated. A few things were done to this list. Any rows that had values greater than 25 or less than 0 were deleted. This was because the Peggy 2LE board only has 25 LEDs, and it has to stay within that range. Thus, if the object in Blender goes outside the animation space, this code will correct and delete those values.

Next, the data points had to be categorized into "slices". As the Peggy board spins, the LEDs will light up roughly 30 times per rotation. These are considered slices. The slices must be calculated by taking the animation file and grouping the 3D object's vertices into their respective slice ranges/angles. For example, if there were 8 slices per rotation, the slices would be broken into $\pi/8$ bins.

$$0 <= \text{slice position} <= \frac{\pi}{8}$$

$$\frac{\pi}{8} <= \text{slice position} <= \frac{\pi}{4}$$

$$\frac{\pi}{4} <= \text{slice position} <= \frac{3\pi}{8}$$

$$\frac{3\pi}{8} <= \text{slice position} <= \frac{\pi}{2}$$

$$\frac{\pi}{2} <= \text{slice position} <= \frac{5\pi}{8}$$

$$\frac{5\pi}{8} <= \text{slice position} <= \frac{3\pi}{4}$$

$$\frac{3\pi}{4} <= \text{slice position} <= \frac{7\pi}{8}$$

$$\frac{7\pi}{8} <= \text{slice position} <= \pi$$

The Peggy community is fairly large and some very nice packages have been written and included in the Arduino library for Peggy users. The Arduino library is a codebase that contains useful packages to extend the functionality of an Arduino device such as the Peggy 2LE. One of these packages enables the user to light up the entire 25 x 25 grid with only 25 numbers. Each number represents a row on the Peggy board. It might seem counterintuitive to think that one number could represent the state of 25 LEDs (in a row), but Windell Oskay created a method for accomplishing that goal.

Take for example a row where the first, fifth, and twenty-fifth lights need to be on. The zero-based indexes to represent these positions are {0, 4, 24}. To calculate the number to represent position 0, 4, and 24 the following function was used.

$$2^{n_0} + 2^{n_1} + \cdots + 2^{n_k} \qquad (1)$$

where n are the positions of the LEDs to be turned on, and n_k is the last LED position to be turned on. An example is shown below for the positions 0, 4, 24 in a single row using Equation 1.

$$2^0 + 2^4 + 2^{24} = \mathbf{16777233}$$

The function used to calculate the LED positions from a single integer (such as the one above) simply checked for the first largest factor of 2^n that was less than the number for the row. For the number above, $2^{24} = 162777216$ is the largest. Take the difference and what's left is 17. Repeat the process: $2^4 = 16$ is the next biggest that is less than 17. Finally, what is left over is 1, and that is simply $2^0 = 1$. From this, one can pull the LED positions from the exponential powers... 0, 4, and 24.

This had to be implemented into the Mathematica code. A list of integers defining which LED positions were to be enabled was created. This was done by the following code for 50 slices. The snippet of code below shows the function, *g[x_]* , which makes a new list called slicelist, that stores the radial position of each vertex that lies between the individual slice range.

```
g[x_] := (
slicelist = {};
temp = {};
Do[
    If[x ≤ newlist[[j, 4]] ≤ x + (Pi / 50) , temp = Append[temp, newlist[[j, All]]]];
    If[temp ≠ {}, slicelist = Append[slicelist, temp]];
    temp = {},
    {j, Length[newlist]}
];
slicelist
)
```

As seen above, the slices include a range of positions between $\pm\frac{\pi}{50}$ per slice. This enabled the Mathematica code to grab vertices that weren't exactly aligned with the virtual rotation, and group them into their respective $\pm\pi/50$ bins. The following figure below shows a "top-view" representation of the individual bins inscribing a cube. Notice that the bins on the outer-most side of the figure are much larger than the bins closer to the center of the bin. As the bins increase in size from the center of the figure to the outside of the figure, the resolution is also decreasing.

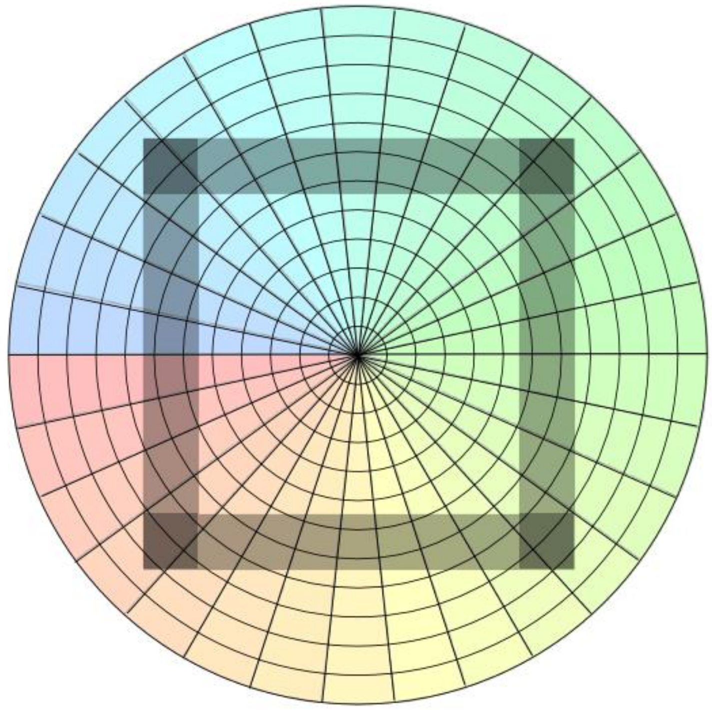


FIGURE 2 SHOWS A VIRTUAL REPRESENTATION OF THE AREAS THAT THE 30 SLICES SPAN

From the Blender mesh object, came the vertices; Mathematica then categorized the vertices into defined bins / slices. Equation 1 was also implemented in Mathematica to produce a list (array) of numbers that represent the LED positions to be turned on for an animation. Figure 3 on the following page shows how three dimensional images are rendered in Blender. Then how the wire frame and vertex information is viewed in Blender. It also shows how the vertices look when they are stripped as simple point data in Mathematica.

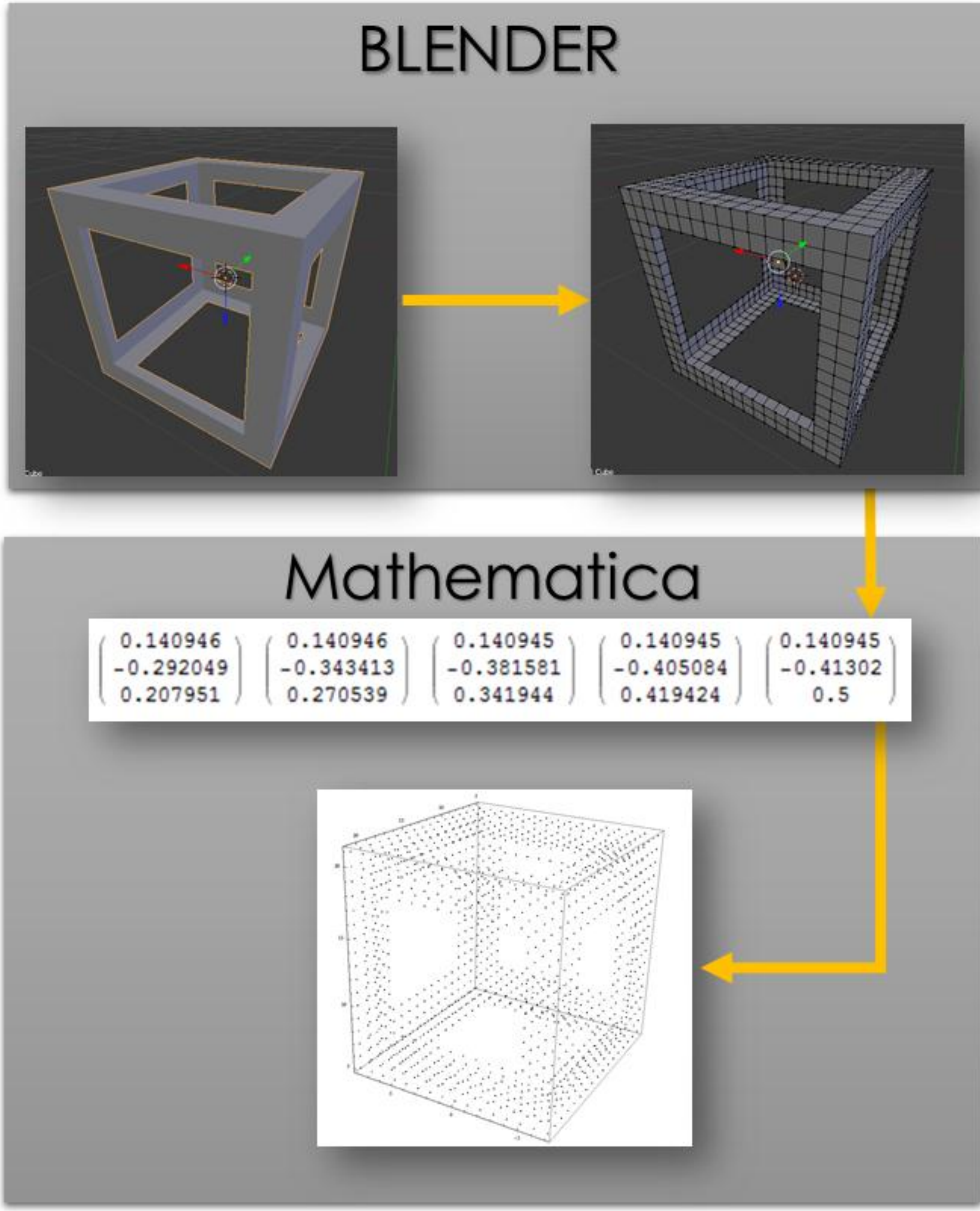


Figure 3 Process flow of three dimensional image data from Blender to Mathematica. Top left shows a solid three dimensional mesh object in Blender. Top left shows the same object as a wire frame which includes the vertex and line data. The data is output from Blender and input into Mathematica as a matrix of vertices. The vertices are then conditioned and transformed into the "Peggy" coordinate system. The image on the bottom shows the individual vertex / point data for the vertices pulled from the Blender file after conditioning.

Arduino

Arduino software was used to communicate with the Peggy 2LE board. As mentioned earlier, the Arduino and Peggy 2LE community are rather large and many software programs and packages have been created to help developers and inventors use the Peggy board in creative ways.

It is important to understand how Equation 1 was used in Mathematica to produce the “Arduino-ready” code. A single number can be used for each row on the Peggy to light up any combination of LEDs. For example, using Equation 1 produced a single value that lit up a certain combination of LEDs for a single row on the Peggy board. Because there are 25 rows, there need to be 25 numbers per slice. As mentioned earlier, there are 30 slices per revolution which would equate to a list of 750 numbers per revolution.

A script created by Windell Oskay, the creator of the Peggy 2LE, was used to process the numbers Mathematica outputs (that referenced which LEDs to light up per row). The refresh rate time between slices can be set in the Arduino code. The Arduino code and interface allow the user to directly compile the code to the Peggy 2LE board through a USB to TTL cable. Once the row data is exported out of Mathematica, it is inserted into the Arduino code. The Arduino code is written so to cycle through the row data and animate the Peggy 2LE. When the Arduino code is ready for uploading, the code is sent to the Peggy 2LE and compiled.

Hardware

Peggy 2LE

The main component of this project was the two dimensional display that was used to create the volumetric display. Resolution, frame rate, viewing angle, size, and intensity were all key aspects of the 2D array that affected the performance of and quality of the final product. Many options for the 2D array were considered including Liquid Crystal Displays (LCD) and Organic Light Emitting Diodes (OLEDs) (Geffroy, Bernard). However, these options were disregarded due to their low refresh rates. Refresh rate is defined as the speed at which the display can show individual frames. Projectors are often used to create volumetric displays, but the projectors must have an extremely high (up to 5 kHz) refresh rate to project the sliced images onto a rotating screen (Favalora, Greg). The solution to this problem was to use cheap, efficient, and highly controllable LED arrays. Light Emitting Diodes are advantageous because they refresh (turn on/off) as quickly as the current passes through them.

The Peggy 2LE is versatile and contains a lot of documentation about how to operate and even hack it. The unfortunate part was that it is requires a lot of knowledge about microprocessing, electrical engineering, and programming in C. A lot of time was spent learning the basics of Arduino and how it interacted with the Peggy 2LE. The figure below shows the schematic for the Peggy 2LE board without any of the components.

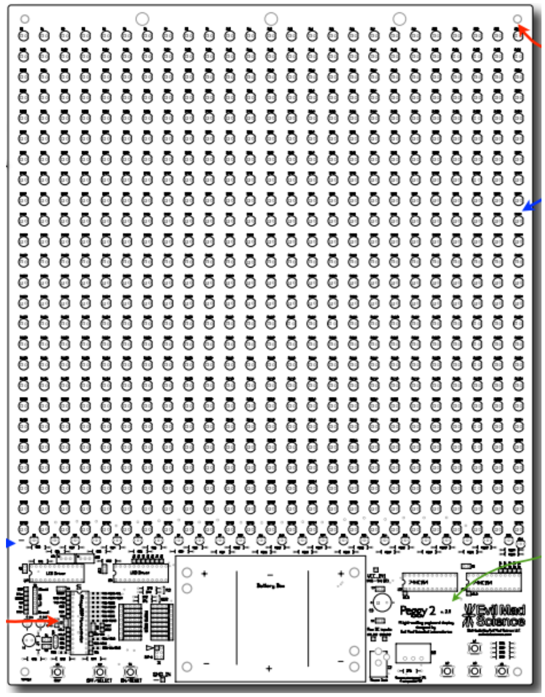
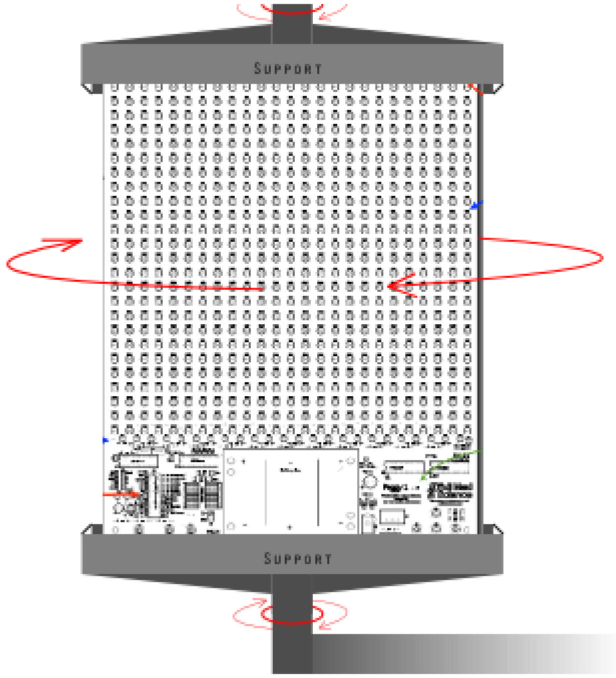


FIGURE 4 THE PEGGY 2LE 2.3 MANUAL SCHEMATIC

The Peggy 2LE is capable of refreshing around 2.5Hz. If the Peggy is spun around the axis at roughly 15Hz with 30 slices, the refresh rate is around 3Hz which the Peggy 2LE can handle. There are faster ways to refresh the board but more microprocessing knowledge is needed to accomplish it. There are 25 LED columns, which means that the Peggy 2LE will rotate around the 13th column. The odd number of columns makes it very easy for symmetrical designs. The LED brightness is also adjustable. However, the faster the refresh rate, the dimmer the LEDS are going to show. The image below shows how the Peggy 2LE rotates around the axis.

FIGURE 5 VOLUMETRIC DISPLAY ROTATIONAL SCHEMATIC / DESIGN



To explain how the display shows geometric figures, a simple example of a single point will be explained. When the LED array is spinning at a given frequency, a single LED on the exterior can light up and turn off every time the array spins 360º. This enables a single point to be visible only from the 'front' of the display. The 'back' of the display would show nothing. To accommodate for this, a second LED from the opposite side of the array will light up when it passes the same 360º mark. Think of the LED array as two displays, halved through the middle. If the array is spun clockwise, the right side of the array will light a single point every 360º, and the left side will come by 180º later and light up the same point. Thus, it will be visible from either side of the volumetric display.

The Peggy 2LE came with an instruction manual on how to assemble the board. It took close to 14 hours of soldering time. The picture below shows the completed Peggy 2LE board showing a two dimensional image of a spinning cube (Windell Oskay).

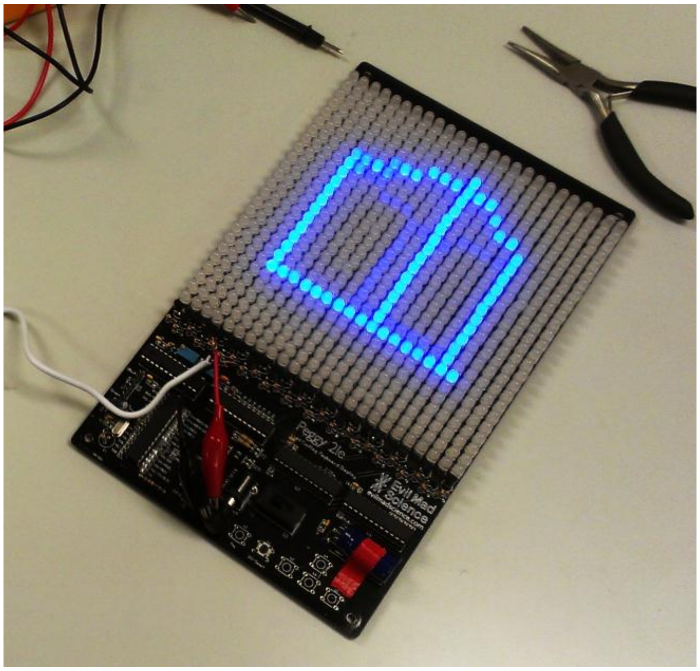


FIGURE 6 ASSEMBLED PEGGY 2LE

Hardware and Machining

The design and build of the framing and enclosure was especially challenging for this project. Many drawings were created and tossed out for improved designs. Parts were ordered and later discarded because they did not perform as expected. The re-design process ended up being relatively costly, which lead to decisions being made based off of cost rather than performance. However, the build was completed and is structurally sound and cost effective.

The first thing designed and built was the frame for the Peggy 2LE. This frame was a relatively simple design, much like a picture frame for a photo. The Peggy 2LE slipped right into the half inch bar stock frame, and was fastened tightly. Two ¼" rods were press fit into the top and bottoms of the frame. The alignment was an issue early on in the build, but was re-machined. There is a small empty space under the Peggy 2LE which was designed to hold the external battery pack needed to power the board as it was spinning

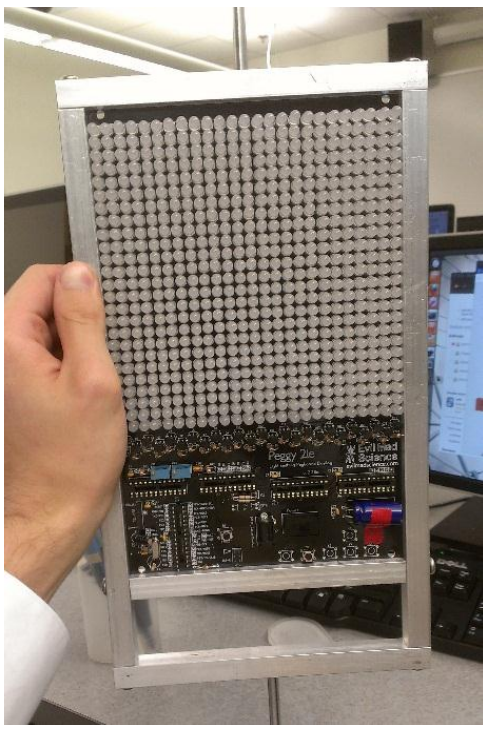


FIGURE 7 PEGGY 2LE FRAME. MADE FROM 0.5" ALUMINUM BAR STOCK. QUARTER INCH SHAFTS WERE PRESS FIT INTO THE TOP AND BOTTOM. EMPTY SPACE BELOW PEGGY WAS FOR MOUNTING AN EXTERNAL BATTERY PACK TO POWER THE PEGGY WHEN IT WAS ROTATING.

The major components of the project's physical features are the aluminum framing structure, bearings, and motor. The aluminum framing structure will be discussed first.

T-Slotted Aluminum Framing System

A few designs using solid 2x4" aluminum bar stock were drawn up early in the project design process. The solid aluminum would have been very strong, relatively cheap, and rigid. However, the machinists that were collaborating on this project did not have the time or resources to machine the aluminum bar stock to the specifications and design that was provided.

An alternative solution was proposed using a T-slotted aluminum framing system from McMaster-Carr as show in in Figure 8. The framing system is extremely robust, customizable, and inexpensive. All of the design features and requirements needed for the project were able to be accomplished using the t-slotted system.

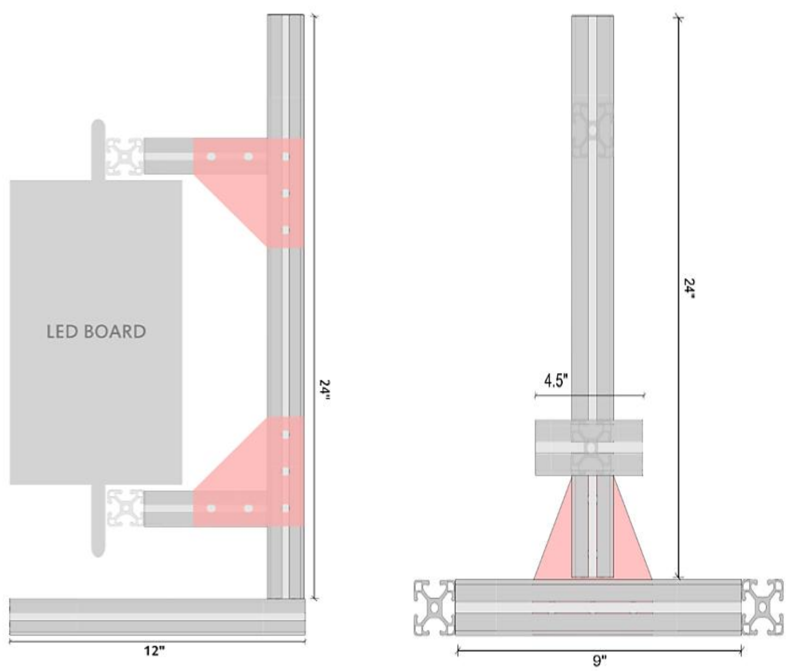


FIGURE 8 ABOVE LEFT SHOWS THE PROFILE VIEW FOR THE T-SLOTTED ALUMINUM FRAMING STRUCTURE. ABOVE RIGHT SHOWS THE FRONT OF THE STRUCTURE. THE LIGHTER COLORED PIECES ARE FASTENERS AND CONNECTORS.

The parts were ordered for the design in Figure 5. Assembly was simple and the rigidity of the structure was strong. Next, the Peggy Board had to be secured to the T-slotted system. The board also had to spin very fast, so the connections and rotational speeds had to be maintained at constant rates with minimal vibrations.

Bearings

It is worth noting the selection of the bearings for this project because of its critical nature to how the entire framing system reacts to high rotational speeds and vibrations. The first design for the bearing system involved two small, high speed, ball bearings. These bearings were rigid and had to be press-fit into a fixture to hold them tight. The original design did just that; it was designed in a way that the bearings would be press fit very tightly into a cavity in aluminum blocks, and the rods from the Peggy Frame would slip into the bearings. This design, however, had many flaws. There is an entire field of study dedicated towards aligning, setting, and placing bearings; the original design did not account for standard practices for bearing alignment (Maslen, Eric). If the original design was used, vibration could have been a serious issue. Also, the press fit rods in the Peggy frame would have start to come loose if they were not aligned exactly. The solution to this problem was using self-aligning pillow block bearings.

Self-aligning pillow block bearings are normal bearings that are “pre-mounted” in a block so that they are easily fastened/bolted to a device. The pillow blocks purchased for this project had a ±5º tolerance that allowed the bearings to be ever so slightly, misaligned. This solution was needed for the project because it was cost- and time-effective. The pillow blocks also mounted directly onto the T-slotted framing structure and enabled stability and high-speed rotational velocities for the Peggy frame. The figure below shows the specification diagram for the pillow block high speed cylindrical bearings.

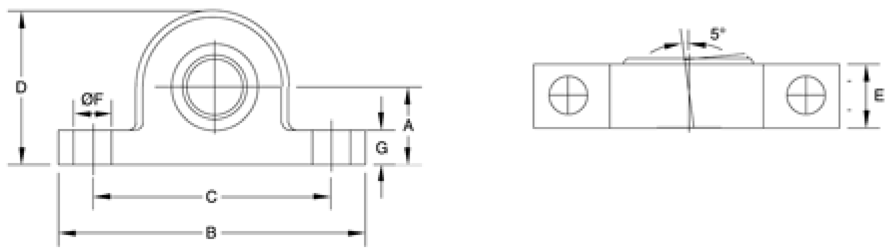


FIGURE 9 SPECIFICATION DIAGRAM FOR THE SPYRAFLO PILLOW BLOCK CYLINDRICAL BEARING. TOP SHOWS MOUNTING ON T-SLOTTED SYSTEM (SPYRAFLO IMAGE).

Motor

The motor chosen to drive this project was a high torque 48mm stepper motor from Phidgets©. The motor provided the necessary speeds and power needed to rotate the Peggy frame around 15Hz. The stepper motor was chosen over a DC motor and Servo motor because of its continuous and steady rate of rotation as well as its high torque.

The stepper motor connected, via USB, to the "1063 – PhidgetStepper Bipolar 1-Motor". The motor controller had a user-friendly interface with open-source code to edit. The source code was downloaded and edited in Visual Studio 2010. Parameters such as acceleration, speed, and step counts were changed in the source code for more suitable control of the stepper motor.

Carefully thought out designs went into connecting the 5mm shaft from the stepper motor to the ¼” shaft of the Peggy Frame. However, there were absolutely no gears, pulleys, or vertical female/female pieces that converted imperial units to metric units. For this reason, a custom piece was made to overlay the 5mm shaft and increase its diameter to ¼”. This method most likely would have worked, but a simpler, less elegant solution was implemented using duct tape. A comprehensive list of build materials can be seen in Appendix 1.

Results

The results of the hardware will be discussed, followed by the presentation, fluidity, and perception of the 3D image translation from the software to the volumetric display.

All of the LED hardware and machined framing worked properly. The original designs led to solid builds. A majority of the time in this project went to researching for the appropriate materials and components for this project. The Peggy frame is able to spin around 14Hz with minimal vibrations. The acceleration of the stepper motor was set to 4000 1/16th steps per sec^2. The stepper controller's software has a built in function for acceleration. The stepper motor also decelerates as it reaches the end of its target position, allowing the Peggy display to gently come to a rest.

When rotating at 14Hz the entire structure was stable with minimal vibrations. Counter weights were used to balance the Peggy and the frame. After the counter weights were installed, the structure was even more stable approaching its target rotational velocity of 15Hz. The stepper motor can produce speeds slightly over 15Hz, but requires a higher current limit for the stepper motor to operate at those speeds.

The number of slices per revolution are limited to the refresh rate of the Peggy 2LE (Oskay, Windell). If the Peggy 2LE is spun at 14Hz and has 30 slices, the Peggy has to refresh at, $\frac{1}{13Hz} \div 30 \cong 2.6\,ms$. The Peggy 2LE is capable of refreshing this quickly in between slices, but cannot refresh much quicker without more knowledge of microprocessors and the Arduino library code. Because of these limitations, all the animations were created for 30 slices at roughly 13 rps. At these rates, a three dimensional object is easily visible. The following image shows a three dimensional cube being displayed. It is difficult to capture the true depth of the visualization in a two-dimensional image because of the timing difference between the refresh rate of the LEDs and the shutter rate of the camera.

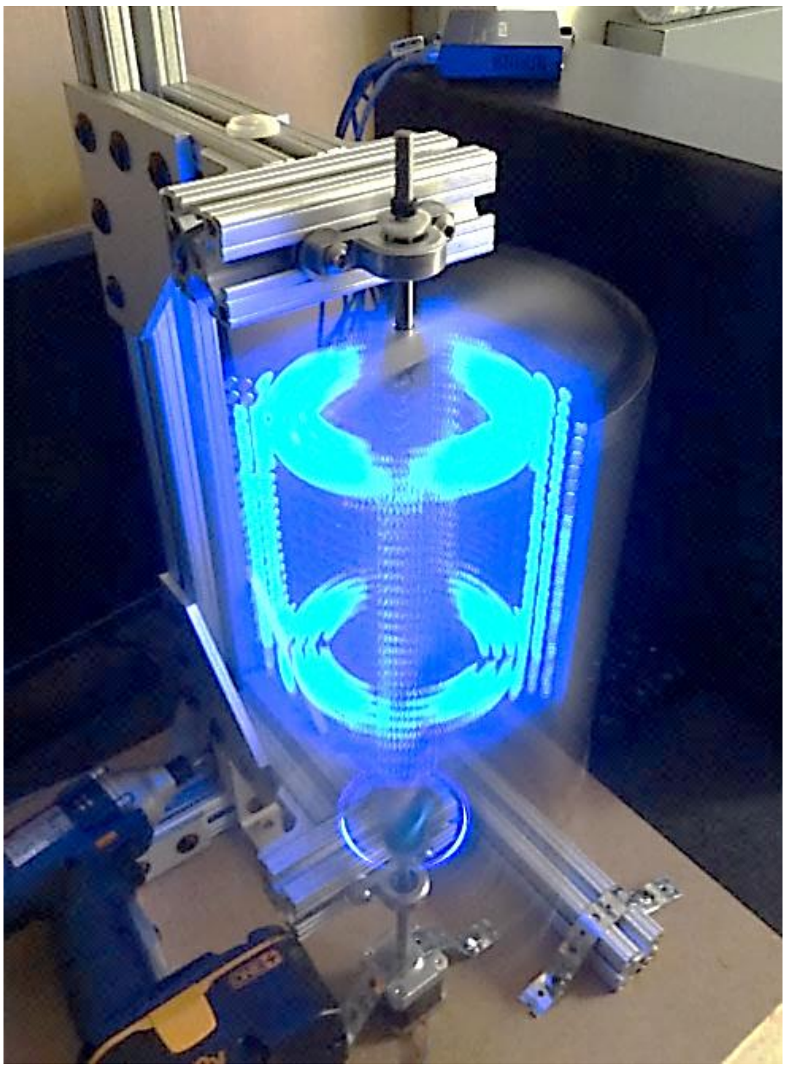


FIGURE 10 VOLUMETRIC DISPLAY SHOWING A THREE DIMENSIONAL REPRESENTATION OF A CUBE IMPORTED FROM BLENDER ANIMATION SOFTWARE

Full animations of geometric objects, scrolling text, and other animations were clearly visible from almost all viewing positions around the display. The hardware and framing for the display was structurally sound, minimizing vibrations and feedback.

Discussion

The idea of a volumetric display is not new, but this project uses an original technique to accomplish a three dimensional effect. The theory behind volumetric displays led to the conception of the method used to create this project. The most significant portion of this project was the methodology because of its originality and reproducibility.

The three main pieces of software used were Blender, Mathematica, and Arduino. With these three pieces of software, all of the calculations, modeling, animating, and processing were done. Blender, as stated earlier, has a very steep learning curve and is probably not the best tool for users that are not experienced with animation software. However, it is extremely robust and contains every feature one would need to make and export three dimensional animations. Thus, for the scope of this project, Blender worked well.

Mathematica has very good documentation but is not open source, and is relatively expensive. Mathematica handled the large data sets of vertices well, and performed fast and accurate calculations. Python was considered at the beginning of the project. Python was considered at the beginning stages of this project because it is open source and has a very active user base. This would mean that any future groups with experience using Python would be able to work on the project without the necessity of purchasing software. A large portion of time spent on this project was converting the Wavefront animation file data into the Peggy 2LE code through Mathematica.

The Arduino user community is a very exciting, growing, and active community with thousands of new and interesting projects coming out yearly. Arduino has great documentation, and the Peggy 2LE makes use of its great interface and library. The Arduino-ready Peggy 2LE was great for hacking and will serve well as a learning tool or for future extensions of this project.

The completed project can be deemed successful. Animations created in Blender are easily viewable on the volumetric display. Similar displays can cost upwards of $3,000 (Dreamoc Holographic Display Solutions, Joel Solloway). Many of the more expensive displays have better performance, full color, smoother images, etc. However, where this project stands out is with its unique design at a relatively low cost. Many volumetric displays use mirrors and projectors to cast out a 3D image. This project uses a “D.I.Y.” LED matrix that is very customizable and programmable. With good planning and reference materials, this project could be created for under $350. The performance, cost, and simple repeatable design of this project is what makes it unique.

Works Cited

Arduino. Computer software. *Arduino*. Vers. 1.0.2. Hosted by Google Code, n.d. Web. <http://arduino.cc/en/Main/Software>.

Blender. Computer software. Vers. 2.64a. N.p., n.d. Web. <Blender.org>.

"EPISTAR Corporation." *Epistar- Solutions for LED Lighting, LED Applications, Co-activation Service*. N.p., n.d. Web. 04 Nov. 2012. <http://www.epistar.com.tw/_english/02_case/03_detail.php?SID=14>.

Favalora, Greg, Joshua Napoli, and Deirdre Hall. "100-million-voxel Volumetric Display." *SPEI Digital Library* 4712 (2002): n. pag. Print.

Fry, Ben, and Casey Reas. *Processing*. Computer software. Vers. 2.0 Beta 6. Hosted by Media Temple, n.d. Web. <http://processing.org/download/>.

Geffroy, Bernard, Philippe Le Roy, and Christophe Prat. "Organic Light-emitting Diode (OLED) Technology: Materials, Devices and Display Technologies." *Polymer International* 55.6 (2006): 572-82. Print.

Lawler, Richard. "Sony's 360-degree RayModeler 3D Display Brings Its Glasses-free Act to LA, Plays Breakout (video)." *Engadget*. N.p., 28 July 2012. Web. 15 Sept. 2012. <http://www.engadget.com/2010/07/28/sonys-360-degree-raymodeler-3d-display-brings-its-glasses-free/>.

Matusik, Wujciech, and Hanspeter Pfister. "3D TV: A Scalable System for Real-time Acquisition, Transmission, and Autostereoscopic Display of Dynamic Scenes." *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH*

2004 23.3 (2004): 814-24. *ACM Digital Library*. Web. 2 July 2012. <http://dl.acm.org/citation.cfm?id=1015805>.

Oskay, Windell. "Evil Mad Scientist Laboratories." *Evil Mad Scientist Laboratories*. N.p., n.d. Web. 12 Sept. 2012. <http://www.evilmadscientist.com/2008/programming-peggy-2-0>.

Oskay, Windell. *Peggy2LE Arduino Library*. Computer software. *Evil Mad Scientist*. Vers. 0.30b. Hosted by Google Code, n.d. Web. <https://code.google.com/p/peggy/>.

Specification Diagram for Spyraflow Pillow Block Cylindrical Bearing. Digital image. *Spyraflo Precision Bearings*. N.p., 4 Nov. 2009. Web. 8 Oct. 2012. <www.spyraflo.com>.

Wofram Mathematica. Program documentation. *Wolfram Mathematica*. Vers. 9. N.p., n.d. Web. <www.wolfram.com/mathematica>.

Appendix

Appendix 1: Table of Materials

Item	**Description**
Rosin Solder	Lead-free
Soldering Iron	High Quality, <8V to ensure board is not burned
Spur Gears	Translate power from motor to board
T-Slotted Aluminum framing w/ connectors	About 8ft … customizable
Stepper Motor	High Torque (4.8 kg * cm) Bipolar Stepper
Stepper Motor Controller	Comes with computer interface for easy programming
Self-Aligning Pillow Block Cylindrical Bearings	High speed, lubricated
2200mAh External Battery Pack	Powers the Peggy 2LE as its spinning. Takes >750mA @ 5V
Peggy 2LE	The LED matrix 625 LEDs