|  |  |
| --- | --- |
| 3D holographic Projector  Cornell Cup Entry | Team Name: PSU IEEE |

Project Abstract:

Visualizing 3D objects and scenes can be difficult, especially when trying to visualize 3D graphs in your Calculus 3 class. Unfortunately, really good 3D graphers don’t quite show off the 3 dimensionalities of the graph, as these graphs are mostly displayed on a 2-dimensional screen. There is a better intuition that comes from a 3-dimensional graph, and this is what our project is trying to achieve. Our plan is to use the raspberry pi, a small monitor, and an inverted square pyramid to display 3D graphs, and if time permits, a wide variety of 3D based applications. Based on a very simplified model found on YouTube, this device uses reflected light to give off the illusion of 3 dimensions, with the potential for a wide variety of applications, specifically in the math/science fields.

Figure The initial inspiration of the project

Challenge Definition:

The primary task to be accomplished is accurately graphing 3d functions of two variables, in a way that allows for viewing the graph as if it were a holographic projection. This serves two purposes. The more direct purpose is in the form of being able to visualize complex, two variable systems in a more tangible way, such that a person could gain an intuition of such systems, and be able to visualize them with ease. Another purpose, related to the first, delves deeper into ideas on how we learn. People learn through verbal or textual explanations primarily when studying in the STEM fields, but the biggest issue is trying to visualize certain concepts. Things in physics and math tend to be explained through equations, or very simplistic line drawings, because it is easier to convey these types of ideas on a chalkboard in that way. Visual learning is something that one loses very quickly in these fields, something that a holographic display can remedy. A good solution needs to, at the very least, be a proof of concept, showing off the viability of holograms as a useful learning tool for students, using relatively inexpensive, but easily scalable tools to suit whatever situation the visualizer is used in.

Proposed solution:

Our solution to all of this is to create a relatively cheap proof-of-concept 3D holographic visualizer, which allows students to look at graphs of 2 functions, and understand their behavior visually. This is a relatively simple way to demonstrate the capabilities of the device. Math is an ever expanding and ever abstract field, so the more that is accomplished with this device, the more that one can add on to it to expand its capabilities. When finished with the primary task, users should be able to input any two variable function and observe that function’s behavior from a given range of values. They should be able to zoom in and out of the graph, and the graph should adapt to that by generating points to fit with the scaling. It will likely not be able to work with graphs of 3+ variables (though that is a stretch goal). The demonstration will be of the above features, the main goal being to discuss the viability of the device as a learning.

Performance measures:

* Resolution of graphs (# of blocks it can generate to create the 3D effect)

Goal: 100 blocks

Ideal: 1000+ blocks

* Scaling speed (when zoomed in and out, time it takes to adapt to the scaling)

Goal: a few seconds

Ideal: Nearly instant

* Manipulation of the graph object
* Added features to assist in analysis of the graph

The last two do not have specific goals, the manipulation of the graph object is measured by whether it can be done or not, and the added features are limitless.

To check for the accuracy of the graphs and all other features, MATLAB or any other 3D graphing tool will be used.

Timeline and Milestones:

After every milestone, a meeting will be held to discuss how viable the idea still is, and where we can realistically expect the project to go. Also a good time to brainstorm other ideas.

Milestone 1: Development cycle (4 weeks)

This project will mainly use a game engine, so we need to develop an easy way to do most of the developing outside the raspberry pi, compile it for the raspberry pi, put it on the Pi, and run it successfully.

Figure Early stage testing

Checklist:

* Get the game engine on the team’s computers
* Compile a sample project to see its output
* Set up a repository on Github
* Load and compile the game engine on the raspberry pi
* Set up cross compiling tools to compile for ARM processors
* Compile a sample project on a normal computer and run it on the Pi

Tests:

* Compile all sample projects that come with the game engine on a separate device and run on the Pi
* Qualitatively examine performance (just to roughly gauge the limits of the Pi)

Deliverables: a raspberry pi capable of running programs with the intended game engine

Viability meeting: Discuss how viable the project is, any alternatives, readjusting milestones etc.

Milestone 2: Sample graphs (2 weeks)

To get more familiar with the game engine, and also check the 3d illusion capabilities of the device, we need to make a few sample programs that will take blocks and form them into a graph. These will be displayed on the raspberry pi. This will be a few sample hard coded graphs whose algorithms are taken from [here](https://catlikecoding.com/unity/tutorials/basics/building-a-graph/) (It is important to note that the game engine in use is not Unity).

Checklist:

* Create a program that displays a single cube in the correct 3 dimensional format
* Compile and run it on the Pi
* Use the linked tutorial to develop the needed algorithms for the grapher
* Add features to rotate view, zoom in, zoom out (no graph scaling)
* Compile and run the grapher on the Pi
* Adjust display parameters to fit the Pi monitor

Tests:

* Cube test (outlined above)
* Grapher test (outlined above)
* Stress test: test for # of cubes able to be generated while graph is static

Deliverables:

* Results of the stress test
* Proof of concept holographic visualizer

Viability meeting: Now that the idea is somewhat a reality, is this still viable? If so, what stretch goals should we look at? What embedded system optimizations should we anticipate using?

Milestone 3: Input, generating values for an eventual graph (6 weeks)

At this point we take a step back from the 3D effects, and begin to work on a back end to the final product.

Checklist:

* Set up the raspberry pi to run Octave
* Use GNU Octave to take functions of 2 variables from the user, a range of x values, a range of y values, and output the corresponding range of z values
* Create an algorithm that takes those initial values, and uses them as anchor points to figure out values for when zooming in and out (this is an attempt to avoid regenerating everything when zooming in and out)
* Create a feature to generate the equation of a line tangent to the graph at a point

Test:

* Compile for the raspberry Pi
* Runs successfully on the Pi

Deliverables:

* Raspberry Pi running Octave
* Accurate z outputs

Viability Meeting: Are there any features that we want to implement for easy analysis of the graph? Does the zoom in and zoom out feature work better than just regenerating a new set of outputs?

NOTE: This milestone (and the ones after) is where we reach VERY uncharted territory. None of these milestones are exactly something that we have ever done, but this has the greatest number of unknowns.

Milestone 4: The assembly of the entire application (3 weeks)

The idea here is to create a simple UI to accept the equation, link it to the previous milestone’s output, and see how the graphs are generated.

Checklist:

* Accept an input equation (assume for now the format is acceptable)
* Generate and deliver outputs to the main graphical program
* Create a surface of blocks, map each coordinate to a block
* Set the heights generated by the output
* Zoom in and Zoom out features

Test:

* Does effectively any graph work?
* Are there some graphs that don’t display as nicely?
* Does zooming in and out maintain the same quality of picture
* Do the rotation transformations maintain the integrity of the graph

Deliverables:

* A rough, but essentially complete version of the final product

Viability meeting: Look of the graph, any major bugs/things to improve. Reflect on the past milestone.

Milestone 5: Debugging (4 weeks)

This will be any debugging that needs to be done that does not impact the defined deliverables.

Feasibility & resources available:

This project is certainly feasible. Before attempts to make this project a reality were made, we worked on a similar, much more scaled down version of the project, that did display the 3D effect, but all the processing power was done through an external computer. The hologram was displayed through streaming video onto the Raspberry Pi, and then shown to project the picture onto the pyramid. The timeline is very much possible, with the only caveat being the various exams the team has throughout the school year, as well as the upkeep of the club that our team is affiliated with, IEEE Penn State chapter. Some resources needed would likely be books that we would use to optimize the item for embedded systems, which Penn State provides many of.

Potential Concerns:

The largest concern (other than time management between this and school) is compatibility checking between our computer and the raspberry pi for the various tools we use to make the device, the main one being the compatibility and feasibility of GNU Octave with the raspberry Pi, in terms of performance. Some alternatives would be to program everything ourselves (using Octave as a guide) or employing a different tool like MATLAB or Mathematica to do the calculations instead. The intent of the project is to keep things as open source as possible but if the use of proprietary work is needed, it will be used and credited accordingly. These alternatives will extend milestone 3 even longer, likely preventing us from even attempting stretch goals or more fun applications of a hologram. Most of these decisions will be made during milestone 3.

References:

“Documentation.” Urho3D, GitHub, urho3d.github.io/documentation/1.7/.

Eaton, John W. “GNU Octave.” Top (GNU Octave), GNU, 2018, octave.org/doc/v4.2.2/.

Flick, Jasper. “Building a Graph.” Catlike Coding, Catlike Coding, 13 Nov. 2017, catlikecoding.com/unity/tutorials/basics/building-a-graph/.

Mrwhosetheboss, director. Turn Your Smartphone into a 3D Hologram | 4K. YouTube, YouTube, 1 Aug. 2015, [www.youtube.com/watch?v=7YWTtCsvgvg&t=44s](http://www.youtube.com/watch?v=7YWTtCsvgvg&t=44s).