Project 3D Graphics Processing Engine: Diffuse Reflection, Shadows, and Decoration

Prasaanth Radhakrishnan
Computer Engineering Department
San José State University (SJSU)
San José, CA, USA
Email: prasaanth.radhakrishnan@sjsu.edu

***Abstract*—In the Advanced Computer Design course, I completed multiple projects with respect to graphics processing. These ranged from interfacing GPIO pins with LEDs and switches to doing diffused reflection on an LCD display. Some of these tasks are making a screensaver with rotating squares and a forest. Another task was to draw a 3D coordinate system along with a cube and the top half of a sphere. The final project required us to integrate all of these features along with diffused reflection which was accomplished and will be shown in this report.**

I. Introduction

This project required the use of the LPC1769 microcontroller. The LPC1769 is a Cortex-M3 microcontroller with 512kB of Flash memory and up to 70 General Purpose Registers. This report will briefly introduce all the projects done in this course and a detailed analysis of the Final project, which is on the 3D Graphics Processing Engine.
The projects done in this course are as follows:

- Generating screen saver of rotating squares
- Generating forest screen saver
- Generating 3d coordinate system
- Generating Cube
- Generating Sphere
- Generating Diffuse reflection on top of the cube
- Generating ray equation and shadow for the cube
- Generating a tree on the frontal surface of the cube using a Linear Algorithm
- Generating S alphabet on the top half of the sphere

The difficulties in completing these projects concerned the mathematical computations required to generate each pattern or object. It also needed the proper interfacing with the hardware, which will be listed in the upcoming sections.

II. Methodology

This section will talk about the technical objectives and challenges in completing the project which would give a detailed enumeration of the goals of the project. The next part would be about the problem formulation and design which is the description of the design used to solve the problem statement.

A. Objectives and Technical Challenges

1) Screen savers of Rotating squares and Forest: For the rotating squares problem, we had to assume lambda as 0.8 which is a reduction factor to reduce each side of a square to 80 percent and rotate it in an anticlockwise direction up to 10 levels to form a pattern. Then continue making these patterns until the user stops it manually to simulate a screen saver on the LED screen.

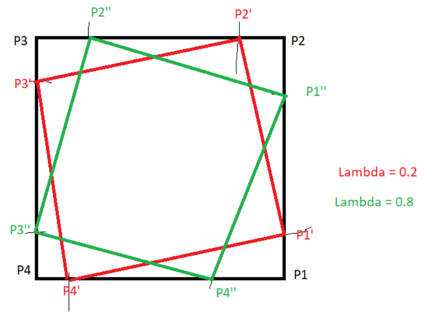


Fig. 1. Sample rotating square with a size reduction(lambda) of 0.8 or 0.2

For generating the forest pattern, a similar pattern had to be followed where each line was reduced to 80 percent and rotated in both clockwise and anti-clockwise directions up to 8 levels to generate a tree-like design. Then the design had to be continued until the user gave input hence ending the screensaver. This was done using a recursion method to generate the trees. The new lines would be generated with the following C code:

2) Generating 3D coordinate System, Cube and Sphere: This required the generation of a 3D coordinate system on a 2D LCD display, which required transformation from a 3D to a 2D view. In addition, it required assuming a camera location which is the eye location and can be adjusted based on the picture we would like to see.

Then, a cube had to be generated with its center at the origin, which is (xw, yw,zw) = (0,0,0). The formula for generating one point of the cube is where the world-to-viewer function would convert a 3D coordinate to 2D to be viewable on the LED screen, and its result or return value is assigned to the x and y variables.

A 3D sphere had to also be generated on top of the cube as seen in figure 3. This sphere needed to have 8 to 10 levels

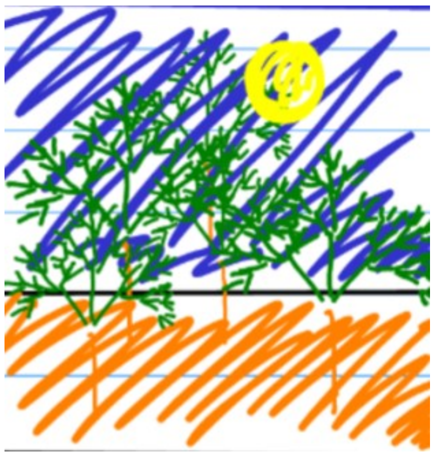


Fig. 2. Sample forest requirement

with decreasing radiuses and all points interconnected to each other.

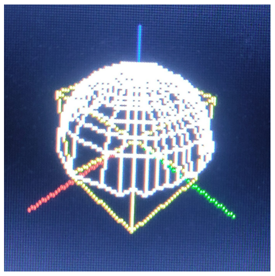


Fig. 3. Generating a 3D coordinate system with a Sphere and a Cube on top of that

3) Diffuse Reflection, Shadows, and Decoration: This project was to complete a 3D shading model and diffuse reflection computation on the LPC1769 microprocessor platform for the purpose of designing 3D Graphics Processing Engine.

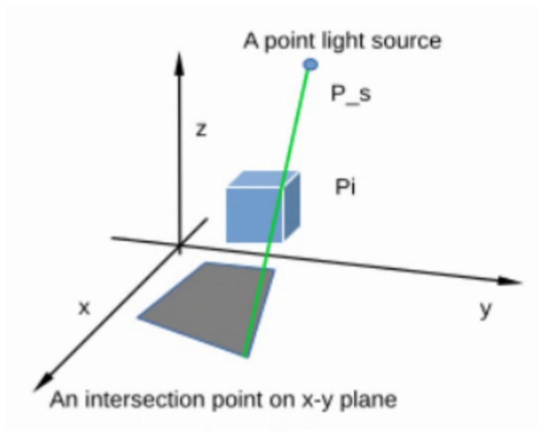


Fig. 4. Generating diffuse reflection on a cube

Then we had to compute diffuse reflection on the top surface of the cube, using one primitive color, I used red for the cube and dark blue to generate the shadow. Considering a light source, ray equations had to be computed to the cube from the light source to generate a shadow on the xw-yw plane and I used gold lines to highlight these lines. Using a scaling equation, I could scale up the reflection factor on the color to show diffused reflection. The results will be shown in the next section.

B. Problem Formulation and Design

To replicate all these problems, MCUXpresso had to be set up, and the C files had to be edited in its workspace. This included changing the pins of the GPIO to the connections made on the hardware. It also required the addition of functions to perform each of the operations. I made use of switch functions to be able to select whatever process needed to be performed easily.

The LCD display used by me is the ST7735R display. It has 10 pins and the basic block diagram is shown below.

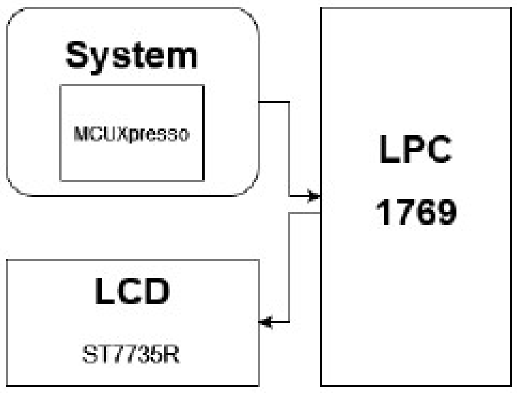


Fig. 5. Basic block diagram of LPC1769, LCD display and the MCUXpresso running on the system

This also increased easy code readability, and updates were easy to make. A sample block diagram and pin connections were made so the connections could be made quickly and the pins could be configured appropriately on MCUXpresso. Detailed block diagrams and connections will be shown in the next section.

III. Implementation

This section is organized into two main sections, which are the hardware design and the software design. The hardware design has images of the prototype system along with its block diagrams and the software will explain the flow charts, algorithms, and Pseudo code for all the projects.

A. Hardware Design

The LPC1769 is a microcontroller that has over 70 GPIO pins and the list of its pins are as given in the diagram below in Figure 6.

The ST7735R LCD display is a TFT display with 128x160 color pixels. The model I used has 10 pins and the TFT display looks as shown in Figure 7.

The Pin connection block diagram between the LPC1769 and the LCD display is as shown in Figure 8.

The pin connections between the LPC1769 and the ST7735R TFT display are as shown in Table I.

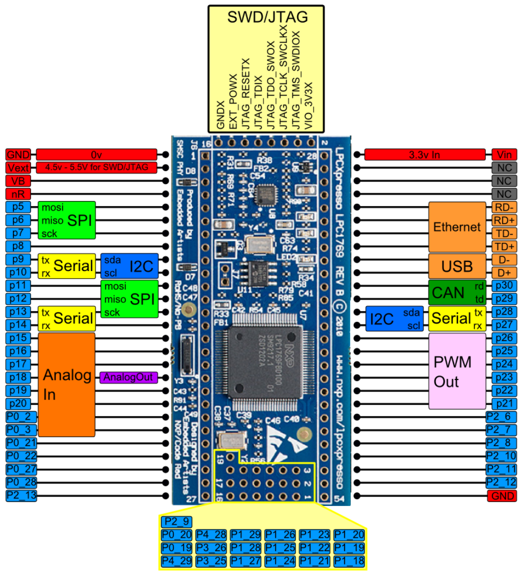


Fig. 6. Pin Diagram of LPC1769 microcontroller

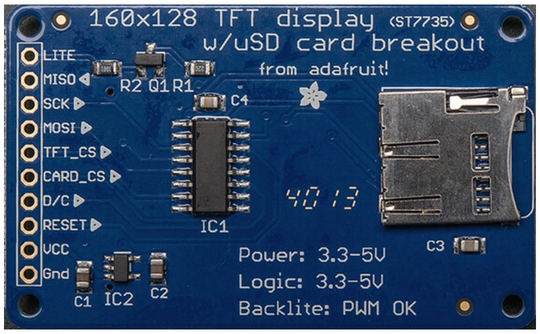


Fig. 7. Image of ST7735R 10 pin TFT display

B. Software Design

This section would explain each of the projects with a bit of pseudo-code to show how the implementations work.

1) Screen savers of Rotating squares and Forest: For the Rotating squares project, the lambda value was set at 0.8 to reduce the square by 20 percent in each iteration. The C code

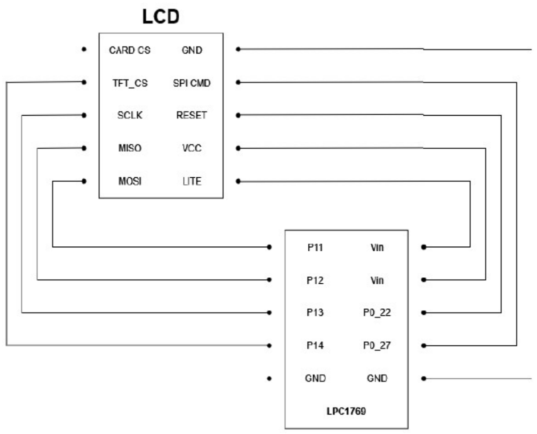


Fig. 8. Pin connection as a Block diagram between LPC1769 and ST7735

LCD ST7735R	LPC1769
LITE	Vin
MISO	P12
SCK	P13
MOSI	P11
TFT-CS	P14
CARD-CS	OPEN
D/C	P0.27
RESET	P0.22
VCC	Vin
GND	GND

TABLE I
PIN CONNECTIONS BETWEEN LCD ST7735R AND LPC1769.

for generating the new x and y points from the existing line whose endpoints are (x1,y1) and (x2,y2) is:

xn1 = x1 + lambda * (x2 - x1)
yn1 = y1 + lambda * (y2 - y1)

The result is captured on my LCD display for a single square with 10 iterations on the inner squares as as shown in figure 9.

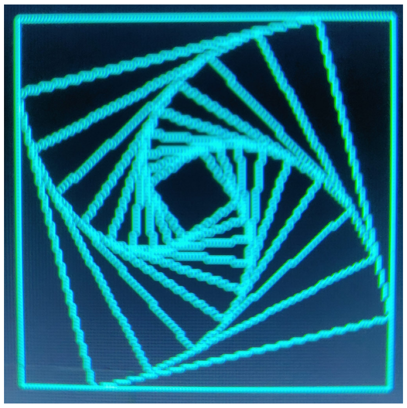


Fig. 9. Rotating square screen capture with a size reduction(lambda) of 0.8 or 0.2

For computing the points of the tree, the basic methodology was to first transform the line that is by reducing the link by 20 percent or 80 percent of its original length. The rotation had to be done in both directions that are clockwise and anti-clockwise directions. Consider T as the transitional matrix and R as the rotation matrix. The final result for the translational and rotational shift of the tree can be given by equation 1.

$$\mathbf{T}^1 = \mathbf{T}^{-1} * \mathbf{R} * \mathbf{T} \tag{1}$$

Where T1 is the new point. Rotating it in both clockwise and anti-clockwise directions would require both sin and cos rotations and the equation to calculate these rotations with reduction can be done as shown in the C code lines below. The function uses recursion to reduce its length and angle so it can keep track of its previous branch node.

x1 = x + treeLen * cos(angle * PI / 180)
y1 = y + treeLen * sin(angle * PI / 180)

The result of these calculations and display on the LCD screen with 25 trees and lambda of 0.8 is as shown below in figure 10.

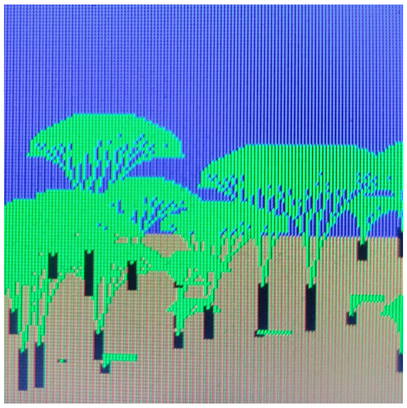


Fig. 10. Forest screen saver as captured on LCD

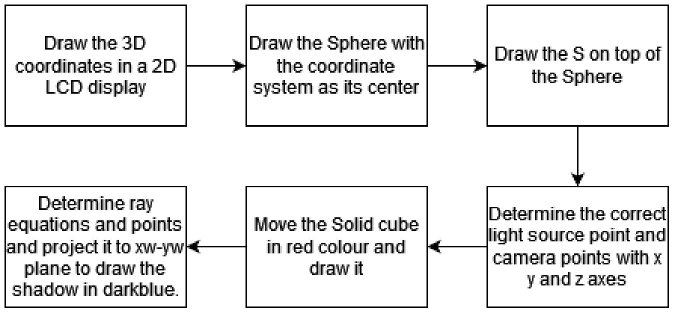
2) Generating 3D coordinate System, Cube and Sphere: This section deals with developing the 3D coordinate system on a 2D LCD screen. It first involves deciding the four points for the coordinate system. In the virtual world consider the points (0,0,0), (180,0,0), (0,180,0), and (0,0,180) as the four points to which the coordinate points are connected. According to figure 3, we can see the coordinates in Red for the X-axis, Green for the Y-axis, and Blue for the Z-axis. These points should be converted to the viewer world and then it can be viewed on the LCD. Its C code for drawing the lines is given as follows:

```
drawLine(x[0],y[0],x[1],y[1],RED)
drawLine(x[0],y[0],x[2],y[2],GREEN)
drawLine(x[0],y[0],x[3],y[3],BLUE)
```

For generating the cube, the three starting points along with the size of the square are given as arguments to the function, and the corresponding points are calculated with respect to all three axes. Which gives a total of 8 points and interconnecting them would give the shape of the cube.

The top half of a sphere was then generated on top of the cube. I created a sphere with R = 100. It had eight levels of cross-section contours. The most significant contour C 1, is a circle on the x w-y w plane, then a set of parallel smaller contours with the exact location of their center but with Z w distance of 10 from one to the other. The maximum radius of this 3D Sphere is 100 and was consecutively reduced by 10 and interconnected to each other. Its final result is as shown in Figure 3.

3) Diffuse Reflection, Shadows, and Decoration: This project was the final project to be completed. It was an integration of all the projects done up until now along with some more additions. The algorithm was as follows. The additions were to generate a solid cube with diffused reflection, consider a light source and calculate the shadow of the cube on the Xw-Yw axis. This should be generated away from the coordinate axis because on the coordinate axis, a top half of a sphere must be generated along with the letter S printed on top of it. The frontal face of the Cube also had to have a tree displayed on it. The flowchart of events are as shown in figure 11.


Draw the 3D coordinates in a 2D LCD display
Draw the Sphere with the coordinate system as its center
Draw the S on top of the Sphere
Determine the correct light source point and camera points with x y and z axes
Move the Solid cube in red colour and draw it
Determine ray equations and points and project it to xw-yw plane to draw the shadow in darkblue.
Fig. 11. Flow Chart of the Diffuse Reflection, Shadows, and Decoration Project

It also required calculation of the intensity of the reflectivity and the brightness of the cube, the C code for that can be given as follows:

```
double rcos = sqrt(pow((zPs - zPi), 2))
return (reflectivity[i] * 255 * (rcos/r)) / pow(r, 2)
```

After using many functions such as calculating intensity, DDA algorithm for drawing lines, diffused reflection, shadow calculation, fill triangles, rotating the tree, and drawing the S alphabet on top of the sphere. The final output on the LCD display looks like this after the completion of the project. It is shown in figure 12. It shows the solid cube with primitive red colour, the shadow in darkblue, the sphere and coordinate system with the letter S, the tree on the frontal face of the cube and the light source with the ray points to the shadow.

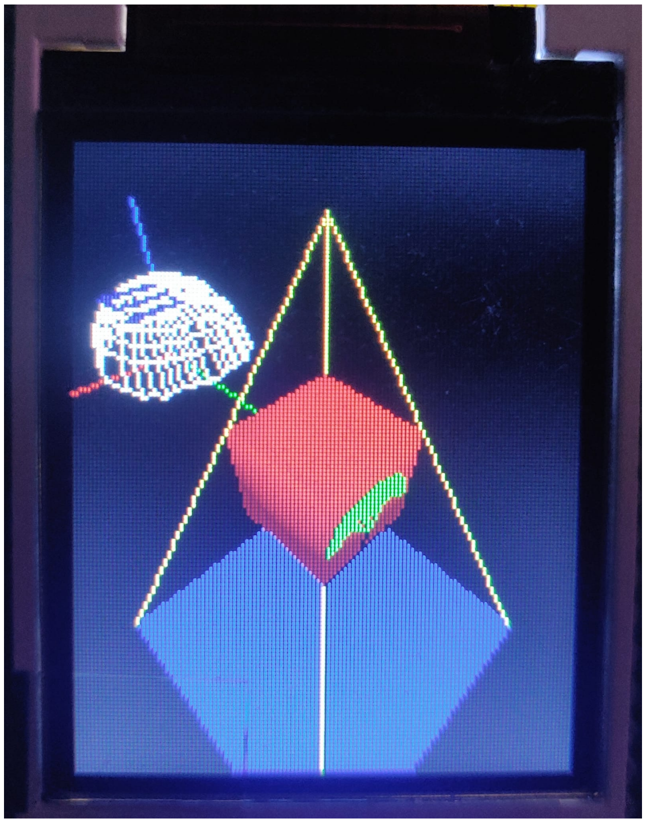


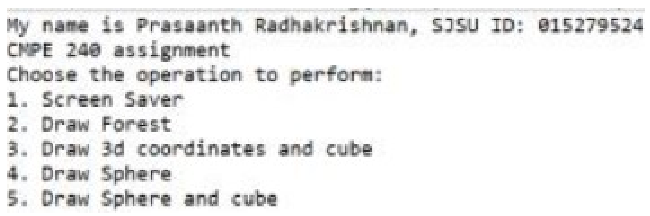
Fig. 12. Final output of Diffuse Reflection, Shadows, and Decoration Project

IV. Testing and Verification

The testing and setup consist of a lot of steps. They can be listed as follows:

- Downloading the MCUXpresso IDE from the NXP website.
- Creation of a suitable workspace.
- Import the project from the GitHub repository named 3D graphics processing, link provided in the Appendix section.
- Import the necessary libraries, under debug options set the necessary fields for debugging in C/C++.
- Connect the probe from a laptop USB port to the port on the LPC1769 till its detected.
- Flash the code onto the Micro-controller and run the code.

I made use of switch statements to easily choose what operation to perform and display on the screen of the LCD. A screen capture of the MCUXpresso window with the same has been attached below in the figure 13.



```
My name is Prasaanth Radhakrishnan, SJSU ID: 015279524
CMPE 240 assignment
Choose the operation to perform:
1. Screen Saver
2. Draw Forest
3. Draw 3d coordinates and cube
4. Draw Sphere
5. Draw Sphere and cube
```

Fig. 13. Screen capture of the output terminal in MCUXpresso

While all these connections have been connected, the block diagram would look as shown in figure 5. The image of it along with the prototype system and the laptop with MCUXpresso open and the output of the final output being shown is attached in figure 14.

V. Conclusion

In brief, I would like to conclude by saying that all these projects gave me a lot of experience to learn about Graphics Processing. I learned about generating screensavers on an LCD display and the formulae required to generate them. Converting mathematics to code to execute in C language proved to be very beneficial for me.

The second project with the 3D coordinates, sphere, and cube gave me ample information about 3D to 2D coordinate conversion and how to generate them using basic mathematics. Being able to place them one on top of the other and generating them all at the same time with clarity and calculating the math behind them proved to be a tough task for me.

The final project with Diffused reflection, Shadows and Decoration was the toughest of all the tasks and took the most effort to complete. Calculating the mathematics and the equations for each of the items to be completed gave me satisfaction when all of them worked in sync. I learned about how diffused reflection and shadows work and in detail about 3D graphics processing and the amount of work that goes into rendering even the most basic of shapes like cubes or trees.

Further improvements are not needed in terms of the projects but more optimization in terms of storage and I/O improvements are something that needs to be looked into in the future. Other data structures apart from Arrays can be used to limit space usage and make the images render even faster. Better logic could also be written with fewer lines and can result in lesser execution times.

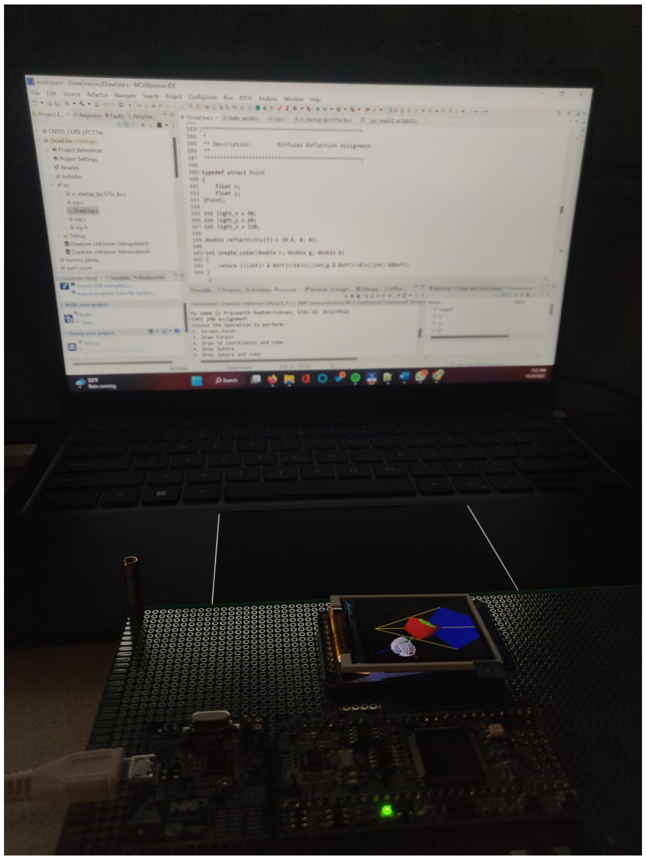


Fig. 14. Laptop with MCUXpresso and connected to LPC1769 and LCD with probe

Acknowledgment

I sincerely thank my Professor, Harry Li for all his guidance and support in completing these projects. I also thank the Teaching Assistant Bruce Jiang for his continued support and for helping me with my doubts. I would also like to take all my classmates for being very supportive and proactive in completing the project and assisting me.

References

[1] H. Li, "Author Guidelines for CMPE 146/242 Project Report", *Lecture Notes of CMPE 146/242*, Computer Engineering Department, College of Engineering, San Jose State University, March 6, 2006, pp. 1.

[2] H. Li, "Steps to import, build, debug and flash on LPC1769 platform", *Lecture Notes of CMPE 240*, Computer Engineering Department, College of Engineering, San Jose State University, May 7, 2019, pp. 1.

Appendix

Source Code: https://github.com/PrasaanthRadhakrishnan/3D-Graphics-Processing