CS 606: Computer Graphics Assignment 1 Report

Chinthan Chandra
IMT2020109
IIIT Bangalore
chinthan.chandra@iiitb.ac.in

I. Introduction

The aim of the assignment is to provide an introduction to the graphics pipeline and 2D interactive graphics rendering along with translation, rotation, and scaling in 2D. The assignment revolves around making a Pacman game similar to the one in Google Doodle, but without the ghost movements. The game is on x-y plane and various 2D objects are rendered based on the specifications provided in the assignment document. A few of the geometric shapes rendered are circles(Pacman), squares(grids), and triangles(ghosts).

II. Libraries

The assignment is made using JavaScript and WebGL. I have used the node package `live-server` to run the folder with code as a server to build dynamically.

III. Code Design and Explanation

We were given a boilerplate code to make the use of the graphics pipeline easier. I have built on this code along with making some modifications to it.

A. Modifications

1) Vertex Shader: The vertex shader program given to us took coordinates to be in clip space. Working with clip space involves values between -1.0 to 1.0. Since we have a grid, etc. working with models in pixel space seemed to be a better option for various reasons. The modification made to the code is multiplying the `aPosition` with the transformation matrix and giving point size to the pipeline instead of being common for all. After the multiplication, the coordinates are converted to clip space by dividing the multiplied coordinates with the resolution of the screen, which gives a number between 0 and 1. I then multiply it by 2 and subtract by 1 to make it between -1.0 to 1.0.
Reasons to use pixel space:

- The clip space coordinates range between -1.0 to 1.0, whereas the pixel space coordinates have a longer range depending on the canvas size.
- Rendering of objects on a rectangle canvas like circle makes it be elongated along the bigger axis of the canvas. So a circle looks like an ellipse instead. This happens as both axes are mapped from -1.0 to 1.0, but this does not mean they have the same value. But pixel space does not have this issue.
- The problem of the limited precision of floating point numbers in JavaScript.

2) Scene: I have not used the scene class to add primitives and render them.

B. Additions to boilerplate

I have made an additional primitive, square to make the creation of grid cells easier.
For each object in the game, I have created a model. The models I have created are:

1 Pacman
2 Grid - consists of the center points of each grid cell.
3 GridColored- consists of the maze layout.
4 Ghosts- the ghosts in that maze layout.
5 PowerPellets- points of size 13.
6 Pellets- the grid points which the Pacman has accessed.

All these models make up a Maze object. Each of these objects has a render function that renders them if they do not directly store the vertex positions. Each Maze object takes in parameters such as the width, height, and layout of the maze, Pacman's center, and radius, grid cell size, and ghost positions as grid coordinates. The maze object stores all the possible orientations in a `mazeArray`. This is used to check moves of the Pacman are possible or not. Maze also has a `currentMaze` variable which updates as and when the maze is rotated.

1) Interactions: The objects which have interactions are Pacman and Maze. Each of them has a process event to handle keyboard events. For the modifying mode, mouse events are handled by Pacman object's `modifyPacman` function. It checks if the move is accessible or not by checking in the maze depending on the grid. If it is not accessible, the latest coordinates are used to place the Pacman.

2) Transformation: Every object has a transform instance where all the transformations take place. The GridColored and Ghosts are made up of a list of simpler objects or primitives and the transform instance is found in these primitives rather than the object itself. GridColored consists of a list of squares and Ghost consists of a list of Ghost which inherits a triangle. Transform consists of a set of functions that apply the required transformation based on the interactions. Global rotation i.e the rotation of the maze is handled by the functions `globalRotation` and `globalTranslation`. The Pacman's movements in the grid are handled by `translateToOriginScaleRotate` and `translation`. All these functions update the `modelTransformMatrix` every time.

- **Rotation of Maze:** I rotate all the objects about the center of the maze using `globalRotation`. Since the maze can be a rectangle, rotation may put it out of the canvas. To fix I have used `globalTranslation` to correct this. For the Pacman, I have also used translation correction as the Pacman's original center is rotated but the rest of the translations also need to be reflected after rotation to preserve location. For the ghosts to preserve orientation, I translate to its center rotate it by the angle the maze is rotated and translate it back by calling `translateRotate` function.
- **Rotation/translation of Pacman:** For every translation, I update the coordinates of the Pacman by checking with maze orientation at that time. For each interaction of the function `translateToOriginScaleRotate` is called to orient the Pacman appropriately and `translation` is called to move it. Every translation adds a pellet using the Pacman's present coordinates to indicate the grid has been accessed. For scaling part, whenever the Pacman is on a power pellet position, the `scale` variable of Pacman's transform is updated to $[1.5, 1.5, 1.5]$ which gets the Pacman scaled accordingly in `translateToOriginScaleRotate` function.

This sums up the majority of the code I have written. The orientation of the ghosts is preserved as instructed. The orientation of the Pacman is not, but it can be done too just by translating the Pacman to the center and rotating it by the angle depending on the direction. My implementation just renders the Pacman in the original angle it was in the first frame.
When the maze configuration is changed, only the Pacman is reset to its initial position when that maze configuration is accessed again. The maze stays in the same rotated position if it was rotated. The green pellets which indicate that the grid cell was accessed are not reset. These can be done too, by just clearing the pellets array in the Pellets object, and to reset the maze's orientation, we just set the angle to 0 and set the `modelTransformMatrix` to identity in all objects.

IV. Assumptions

The following assumptions are taken:

- The ghosts color changes to its original color once the pacman moves from the power pellet grid cell.
- The maze configurations reset means the pacman goes back to its starting position and the maze retains its orientation.
- The pacman's positions changed due to mouse events in modifying mode by picking it and placing it in a different grid position does mean that the grid cell was accessed.
- If the pacman is in a power pellet grid cell, any modification to pacman's position by picking is not allowed.
- If the maze is in modify mode, switching of configurations is not allowed.

V. Implementation

I have used 3 maze configurations of size 30×25, 25×30, and 30×30. The grid cells are of 40 pixels each in the first two and 30 pixels in the last. So the dimensions for each maze are, $1200, 1000$, $1000, 1200$, and $900, 900$. The ghost and power pellet positions are hard coded as grid coordinates. The pictures for the same are shown below in Figure 1,2,and 3.

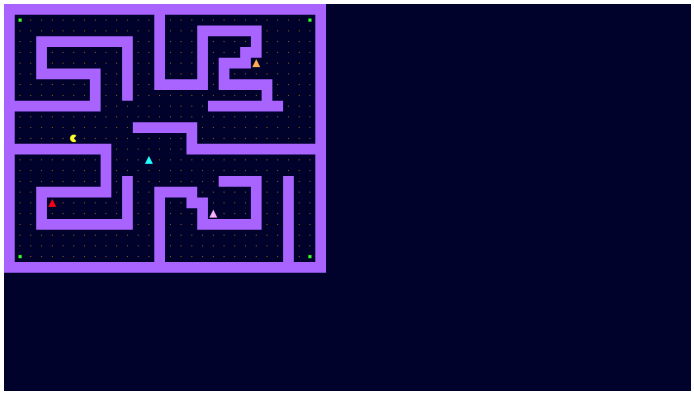

Fig. 1: First Maze Configuration with 1200x1000 dimension

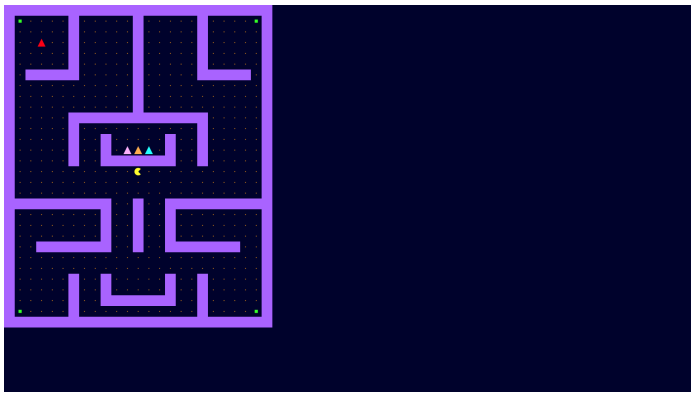

Fig. 2: Second Maze Configuration with 1000x1200 dimension

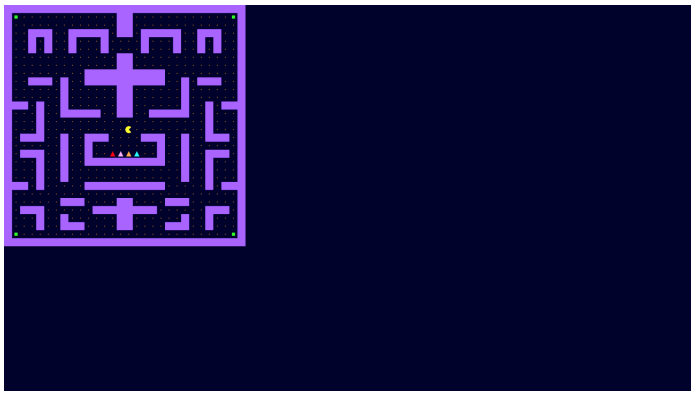

Fig. 3: Third Maze Configuration with 900x900 dimension

Green pellets are added as pacman moves to show that the grid is accessed, Figure 4 shows the same.

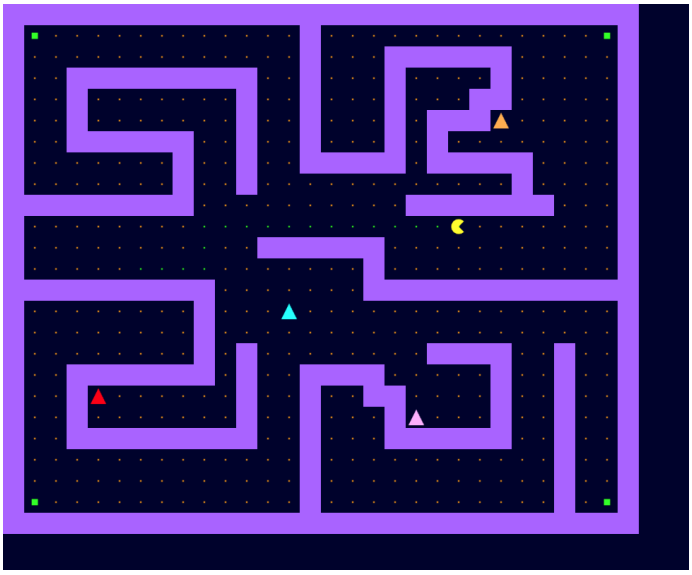

Fig. 4: Green Pellets

The monsters turn blue when Pacman eats the power pellet and pacman grows to 1.5x its size, shown in figure 5.

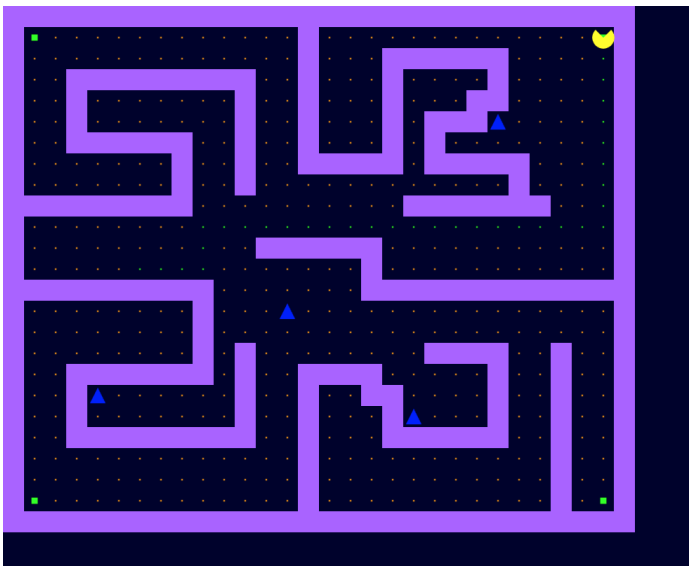

Fig. 5: Monsters Turn Blue

They return to their original color once the pacman moves out of the grid cell, shown in figure 6.

The maze rotates, ghosts retain their orientation and location, the pacman retains its location, shown in figure 7.

When the maze is changed and after few changes I come back to the same maze, it is reset i.e the pacman is made to move to its initial position but the maze preserves orientation, shown in figure 8.

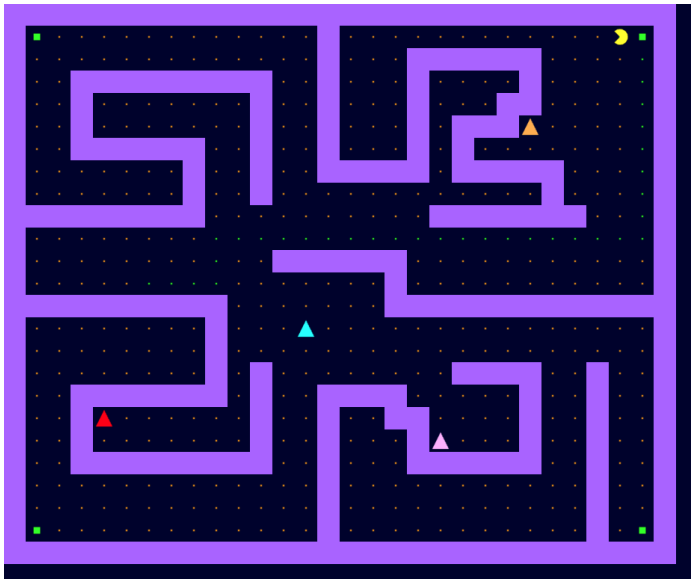

Fig. 6: Monsters change to their original color. Pacman goes its original size.

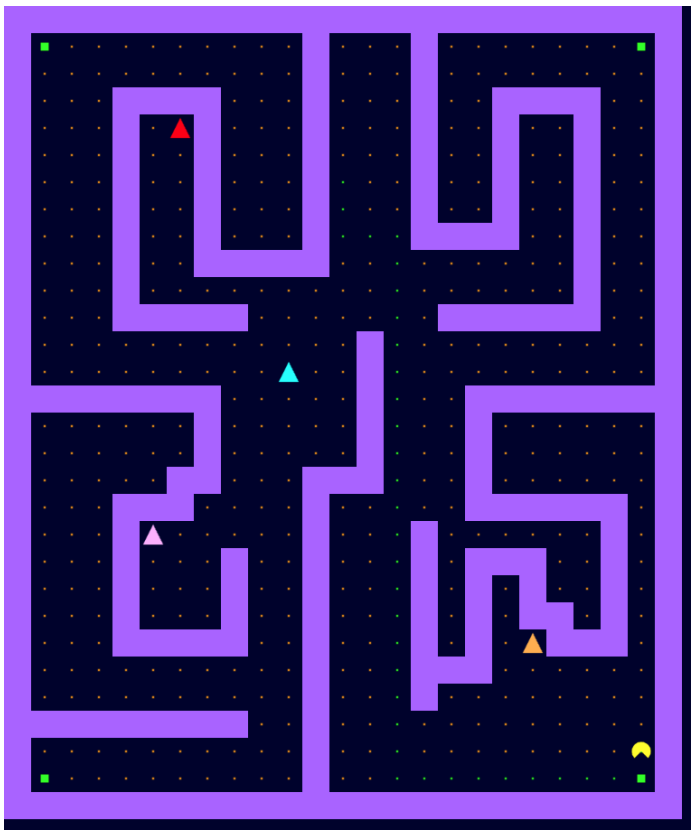

Fig. 7: Rotated maze.

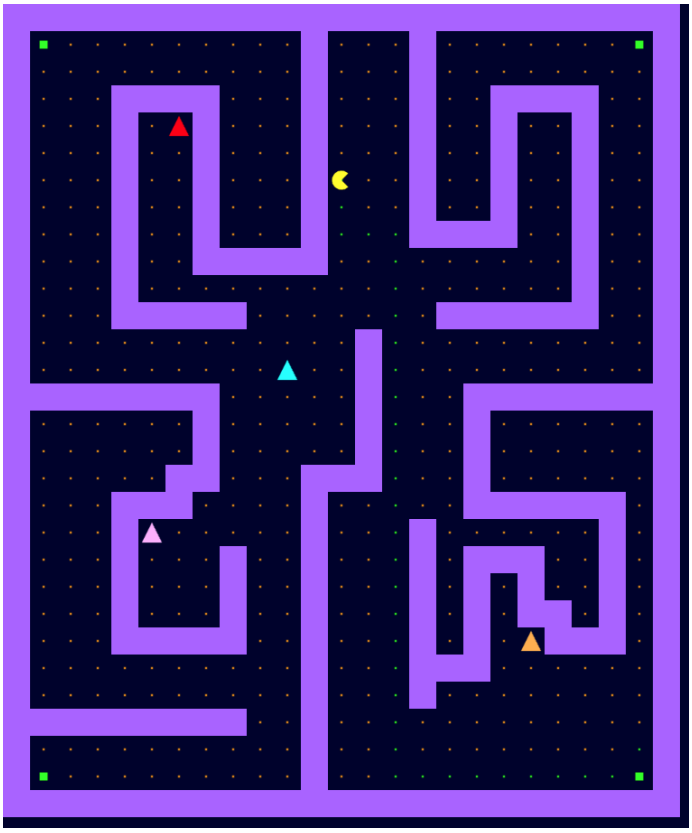

Fig. 8: Reset maze.

VI. Interactions

The following are the set of interactions that are implemented:

1. Arrow keys for the movement of Pacman.
2. '(' and ')' keys for rotating the Pacman by 45°.
3. '[' and ']' keys for rotating the maze by 90°.
4. 'c' or 'C' key to cycle between different maze configurations.
5. 'm' switches the Maze to modify mode.
6. Mouse tracking is done in modify mode to place Pacman in a grid cell.

VII. Questions

1. **What did you do to ensure that the objects (Pac-Man, ghosts) do not change orientation or location when the maze rotates?**
   Answer. To retain the orientation of any object when the maze rotates, the transformation of translating the origin to the center of the object and rotating it and translating back needs to be done. This ensures the original orientation is preserved.
2. **What did you do differently when rotating the maze vs Pac-Man?**
   Answer. Rotating the Maze involves rotation of all objects. This is easy as I just need to move all the objects around the center of maze and translate them by some distance to ensure that the maze is completely still in canvas. Whereas while rotating the Pacman I had to translate the origin to the center of the Pacman rotate it accordingly, move the origin back and then apply the translations as required.
3. **If you were asked to scale the maze, what would be the steps you would implement?**
   Answer. Scaling the maze, this depends on how it is done i.e. whether I am scaling the maze about every grid cell point or just scaling the whole maze to make it look bigger like a zoom option. The latter would be just applying a scaling matrix transformation on all objects. If I were to scale the maze by taking every grid cell center this would make the maze look rather stuffed than scaled. The objects would look different depending on the scale along each axis. They may be elongated along one axis or deformed if the scale values differ by a lot.

VIII. Conclusion

The assignment gave an idea of how the grpahics pipeline works and how graphics rendering is done. How transformations have to applied in series using matrices and also how to do interactive graphics rendering in 2D space. This report covers all the particulars of the code, assumptions and implementation, however for a more clearer explanation the demo video can be used.