CG ASSIGNMENT 1

- Prakhar Rastogi

IMPLEMENTATION:

Grid:

- I created 3 different grid/maps manually by constructing 2D arrays of 0s and 1s. 0s represents the parts of the map that you can't travel to i.e the walls, while the 1s represent the parts which you can traverse.
- The 3 grids can be switched on pressing the key 'c' using the function change_grid()
- I drew the grid according to the following steps:
    - For every cell in the grid matrix, I drew a square of some particular box size.
    - The cells that have value 0, drew a blue square. These represent the walls.
    - The cells that have value 1, drew black square. These represent the paths that can be traversed.

Pacman:

- The pacman was simply created in a similar fashion as we create a circle, just, we changed the angle range between which the circle was defined to (145,405).
- Movement of pacman:
    - The pacman can move in four directions: up, down, left and right.
    - Movement is handled by the move_it() function. It's argument is just the direction in which we want to move.
    - According to the direction passed, we shift the pacman by one unit by applying translateTransform/translateRotatateTransform to translate/rotate it to the next calculated point.
    - Before the translation/rotation takes place, we have a function is_wall() that tells us whether the next calculated point is a wall or not. According to this, we either move the pacman or let it stay at the same position.
- Rotation of pacman:
    - We have handled rotation in two ways:
        - If let's say the pacman is currently moving right, then we press up. So first the pacman will rotate about it's axis to face up and then translate to the next calculated point. This is done using the translateRotateTransform function in transform.js.
        - In order to rotate a pacman at it's own position, I made a function rotate_it() that rotates the pacman by first translating it to its origin position, rotating it, then moving it back to the original position. This too is implemented using the translateRotateTransform function.

- Teleportation of pacman:
    - This is implemented in the tp_maar() function. Using the click coordinates, we calculate the closest grid point, and then check if there is a wall at that place in the map or not using the is_wall() function. If there is no wall, we simply translate the pacman to the new point. If there is , then that move is invalidated.

Pellets:

- Similar to how I created the map grid, I created a 3 different pellets_grid matrix corresponding to each of the 3 maps, whose all entries are zero, except the locations where the power pellet is supposed to be. There it is 2.
- The 3 pellets_grid can be switched on pressing the key ‘c’ using the function change_pellet_grid()
- Creation of pellets:
    - The pellets are created using the initiate_pellets() function. We create the pellets by traversing the pellets_grid matrix.
    - We also have a global pellets array which hold all the pellets.
    - If the cell has value = 0, w create a white pellet and add it to the pellets matrix.
    - If the cell has value = 2, we create a larger yellow pellet i.e the power pellet and add it to the pellets matrix.
- Changing color of pellets:
    - This is done using the change_pellet_color() function.
    - We call this function everytime the pacman moves in the move_it() fucn.
    - We first check if the traversed pellet was visited or not.
    - This check is done using the pellets_grid matrix. In this, 0 represents a unvisited pellet while 1 represents a visited one.
    - If the traversed location had a unvisited pellet, we delete that pellet, insert a new yellow pellet at that same location, add it to the pellets matrix and mark that location as visited i.e, pellet_grid[location] = 1.

Ghosts:

- The ghosts are constructed using a new ghost class which is basically just a simple triangle.
- Color change of ghost:
    - The color change is implemented using the toggle_ghosts function. It takes into input Boolean values true and false. If it’s true, a normal ghost is drawn, if its false, a ghost with changed color is drawn.
    - Every time a pacman moves we first check if the traversed pellet is a power pellet or not. If it is, we pass the value false in the the toggle_ghost() else we pass the value true in toggle_ghosts().

Controls and their implementation:

- Movement is controlled using the arrow keys, we simply call the move_it() function with the parameter according to the key pressed.
- Changing of Maps: The grid is changed using the 'c' key. We switch the grid and the pellets grid, remove all primitives from the scene and then add them again.
- Rotation of pacman only: The pacman can be rotated by 90 degrees in either directions using the '(' and ')' keys. I simply call the rotate_it() function to implement this.
- Teleportation of pacman: On pressing the key 'm', the whole game goes into modify mode i.e we set the mouse_mode=true , where if we click on any point in the map, we check if the mouse_mode =true, ie we are in a modify mode, then we call the tp_maar function and translate the pacman to its new points.
- Rotation of whole thing:
    - This is done using the '{' and '}' keys.
    - We keep track of the current orientation of the grid, on pressing the button, we change the orientation to the next one.
    - Then we uses the scene.primitives to get all the elements in the game and then rotate+translate according the orientation they were previously in. This is done using the rotateTranslate function. It first brings to origin, rotates, then translates to new point.
    - If the pacman was at location (i,j), we shift it to another (j,i).
    - Finally, the grid, pellets_grid, pellets matrices are also transformed using the rotate_all_left() and rotate_all_right() functions so that the game can work properly.

To make sure that the ghosts and pacman do not change orientations/locations, I did as follows. For the ghosts, it was quite simple, the rotating all of the scene.primitives using forAll function worked well in this case. For the pacman, I kept track of its location, orientation then according to the rotation type, I removed the previous pacman, and added a new one at the corresponding new location and with same orientation.

For pacman, to rotate it , simply used the rotateTranslateTransform function. The only thing different is that there is no translation just rotation about its axis.

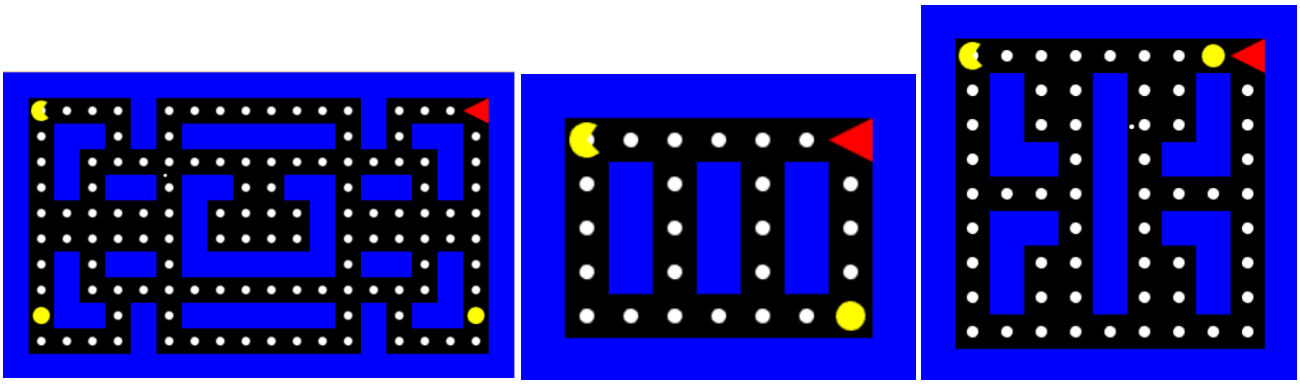
For the maze, to rotate all of it, I used another function rotateTranslate. It translated all of the primitives to new points as well as rotates it. In order to rotate the maze, if I had simply done rotation of all primitives, nothing would have changed, each of the squares, pellets, pacman, ghosts would have rotated at their own places. This wouldn't give us the required rotation of maze, thus needed translation too. Did it using the following code:

```
if(grid_orient == 1)
{
    scene.primitives.forEach(function (primitive) {
        primitive.transform.rotateTranslate(grid_angle,box_size*grid.length,0)
    })
}
else if(grid_orient ==2)
{
    scene.primitives.forEach(function (primitive){
        primitive.transform.rotateTranslate(grid_angle,box_size*grid[0].length,box_size*grid.length)
    })
}
else if(grid_orient ==3)
{
    scene.primitives.forEach(function (primitive) {
        primitive.transform.rotateTranslate(grid_angle,0,box_size*grid[0].length)
    })
}
else{
    scene.primitives.forEach(function (primitive) {
        primitive.transform.rotateTranslate(grid_angle,0,0)
    })
}
```

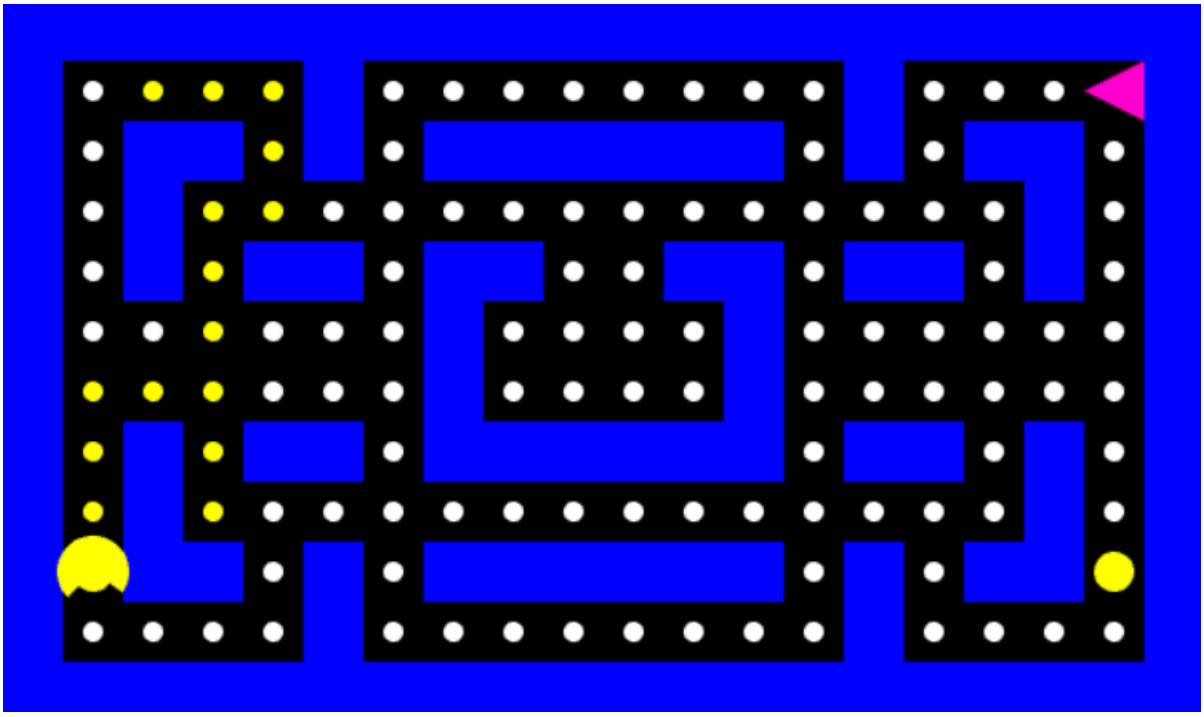
For scaling the maze, itself, I would do what I had done for the rotation of maze. Just the thing that would change is that for all primitives I would use the function translateRotateScaleTransform() to simply scale all of them and display. (translateRotateScaleTransform() was the function which we used to scale the pacman when it encountered a ghost)

Working:

Shown below are the 3 map orientation:



This is what happens when the pacman encounters a power pellet: (scaled to 1.5 its size and color of ghost also changed)



Rotation of map: