Cmpe 360: Computer Graphics
Project 8 Report

Neslihan Pelin Metin
71047171244

Project 8

Introduction

In this Project we were given a robotic arm and we had to give this arm 3 fingers and one thumb. In addition, we were asked to implement following keyboard controls:

- x/X: to rotate the arm on the X axis so you can see it from different angles
- y/Y: to rotate the arm on the Y axis so you can see it from different angles
- a/A: to rotate the fingers on the x-axis with positive direction
- b/B: to rotate the fingers on the x-axis with negative direction
- m/M: to rotate the fingers on the y-axis with positive direction
- n/N: to rotate the fingers on the y-axis with negative direction
- t/T: toggle between solid and wire cubes
- p/P: toggle between perspective and ortho projections
- f: finger and thumb move apart
- F: finger and thumb move closer

Fingers & Thumb

In render function, to add a finger I start by initializing the finger view transform(mvf). This step is essential, especially for the finger rotation. I then saved the value of mvf into stack to use it for the other fingers. After that, I defined the position of the finger's joint, it's position and dimension. In a similar way, I defined finger's upper part. Finally, I added other fingers and thumb by changing the necessary positions.

```
if(isEqual){
   mvf = mv;
}
else{
   mvf = mult(mvf, translate(0.0, 0.0, 0.0));
}
//finger1
matStack.push(mvf);

   //Position lowerFinger Joint1
   mvf = mult(mvf, translate(1.0, 0.2, 0.0));
   //Elbow Joint
   mvf = mult(mvf, rotate(lowerFinger ,vec3(0,0,1)));
   //Position lowerfinger Cube
   mvf = mult(mvf, translate(0.3, 0, 0.4));
   //Scale and Draw Forearm
   matStack.push(mvf);
      //boyut
      mvf = mult(mvf, scale(0.6, 0.3, 0.3));
      gl.uniformMatrix4fv(mvLoc, gl.FALSE, flatten(transpose(mvf)));
      gl.drawArrays(armShape.type, armShape.start, armShape.size);
   //Undo Scale
   mvf= matStack.pop();
```

```
        //Position upperFinger Joint1
        mvf = mult(mvf, translate(0.3, 0, 0.0));
        //Elbow Joint
        mvf = mult(mvf, rotate(upperFinger ,vec3(0,0,-1)));
        //Position Forearm Cube
        mvf = mult(mvf, translate(0.3, 0, 0));
        //Scale and Draw Forearm
        matStack.push(mvf);
           //boyut
           mvf = mult(mvf, scale(0.6, 0.3, 0.3));
           gl.uniformMatrix4fv(mvLoc, gl.FALSE, flatten(transpose(mvf)));
           gl.drawArrays(armShape.type, armShape.start, armShape.size);
        //Undo Scale
        mvf = matStack.pop();

     mvf = matStack.pop();
```

Controls

1. x/X

When “x” is pressed, we are rotating from the arm on the positive x-axis by multiplying the view transform(mv) with the rotation matrix.

```
   if (isPressed("X")) {
      // Rotate the arm around the X-axis
      isEqual = true;
      mv = mult(mv, rotate(d, vec3(1, 0, 0)));
   }
```

2. y/Y

When “y” is pressed, we are rotating from the arm on the positive y-axis by multiplying the view transform(mv) with the rotation matrix.

```
if (isPressed("Y")) {
      // Rotate the arm around the Y-axis
      isEqual = true;
      mv = mult(mv, rotate(d, vec3(0, 1, 0)));
   }
```

3. a/A

When “a” is pressed, we are rotating from the wrist on the positive x-axis by multiplying the finger view transform(mvf) with the rotation matrix.

```
  if (isPressed("A")) {
      // Rotate the fingers on the x-axis positive direction
      isEqual = false;
      mvf = mult(mvf, rotate(d, vec3(1, 0, 0)));
   }
```

4. b/B

When “b” is pressed, we are rotating from the wrist on the negative x-axis by multiplying the finger view transform(mvf) with the rotation matrix.

```
if (isPressed("B")) {
      // Rotate the fingers on the x-axis negative direction
      // Adjust the angle increment as needed
      isEqual = false;
      mvf = mult(mvf, rotate(d, vec3(-1, 0, 0)));
   }
```

5. m/M

When “m” is pressed, we are rotating from the wrist on the positive y-axis by multiplying the finger view transform(mvf) with the rotation matrix.

```
   if (isPressed("M")) {
      // Rotate the fingers on the y-axis positive direction
      isEqual = false;
      mvf = mult(mvf, rotate(d2, vec3(0, 1, 0)));
   }
```

6. n/N

When “n” is pressed, we are rotating from the wrist on the negative y-axis by multiplying the finger view transform(mvf) with the rotation matrix.

```
   if (isPressed("N")) {
      // Rotate the fingers on the y-axis negative direction
      isEqual = false;
      mvf = mult(mvf, rotate(d2, vec3(0, -1, 0)));
   }
```

7. t/T

To be able to toggle between wire and solid cubes, I made the armShape variable accessible by all the functions, and then when “t” is pressed it checks which shape is armShape and changes its shape accordingly.

```
   if (isPressed("T")) {
      // Toggle between solid and wire cube types
      if (armShape == shapes.wireCube) {
          armShape = shapes.solidCube;
      } else if (armShape == shapes.solidCube){
         armShape = shapes.wireCube;
      }
   }
```

Neslihan Pelin Metin
71047171244

8. p/P

To be able to toggle between perspective and ortogonal projection, I have created a Boolean variable named isPerspectiveProjection. Next, when "p" is pressed it checks which projection should the program use and uses it accordingly.

```
   if (isPerspectiveProjection) {
      gl.uniformMatrix4fv(projLoc, gl.FALSE, flatten(transpose(p_perspective)));
   } else {
      gl.uniformMatrix4fv(projLoc, gl.FALSE, flatten(transpose(p_orthogonal)));
   }
```

```
   if (isPressed("P")) {
      // Toggle between perspective and ortho projections
      isPerspectiveProjection = !isPerspectiveProjection;
  }
```

9. f

By changing the coordinates of the lower and upper parts of the fingers, I successfully implemented creating an animation effect as if the hand is opening.

```
if (!shift && isPressed("F")) //f opening
   {
      //finger
      if (lowerFinger > -90){
         lowerFinger = (lowerFinger - d);
         upperFinger = (upperFinger - d2);
      }
      else lowerFinger = -90;
      //thumb
      if (lowerThumb > -90){
         lowerThumb = (lowerThumb - d);
         upperThumb = (upperThumb - d2);
      }
      else lowerThumb = -90;
   }
```

10. F

By changing the coordinates of the lower and upper parts of the fingers, I successfully implemented creating an animation effect as if the hand is closing.

```
if (shift && isPressed("F")) //F closing
   {
      //finger
      if (lowerFinger < 90){
         lowerFinger = (lowerFinger + d);
         upperFinger = (upperFinger + d2);
      }
      else  lowerFinger = 90;
      //thumb

      if (lowerThumb < 90){
         lowerThumb = (lowerThumb + d);
         upperThumb = (upperThumb + d2);
      }
      else  lowerThumb = 90;
   }
```

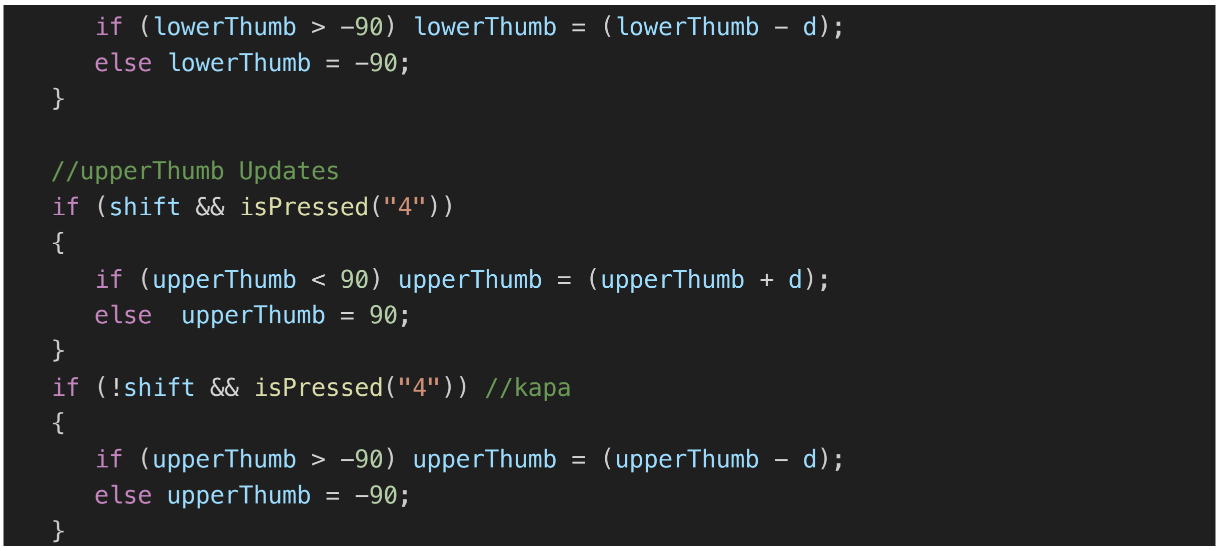
11. Extras

I also implemented extra functionality to clearly understand the opening and closing animation of the hand. When we press 1 and 3, we are allowed to control the lower parts of the fingers and thumb. If they are pressed without the shift, then fingers go up while thumb goes down. If they are pressed with the shift, then fingers go down while thumb goes up. In a similar way, 2 and 4 allows us to control the upper parts of the fingers. If they are pressed without the shift, then fingers go down while thumb goes up. If they are pressed with the shift, then fingers go up while thumb goes down.

```
   //lowerFinger Updates
   if (shift && isPressed("1")) //kapa
   {
      if (lowerFinger < 90) lowerFinger = (lowerFinger + d);
      else  lowerFinger = 90;
   }
   if (!shift && isPressed("1")) //aç
   {
      if (lowerFinger > -90) lowerFinger = (lowerFinger - d);
      else lowerFinger = -90;
   }

   //upperFinger Updates
   if (shift && isPressed("2")) //aç
   {
      if (upperFinger < 90) upperFinger = (upperFinger + d);
      else  upperFinger = 90;
   }
   if (!shift && isPressed("2")) //kapa
   {
      if (upperFinger > -90) upperFinger = (upperFinger - d);
      else upperFinger = -90;
   }

   //lowerThumb Updates
   if (shift && isPressed("3"))
   {
      if (lowerThumb < 90) lowerThumb = (lowerThumb + d);
      else  lowerThumb = 90;
   }
   if (!shift && isPressed("3")) //aç
   {
```



```
      if (lowerThumb > -90) lowerThumb = (lowerThumb - d);
      else lowerThumb = -90;
   }

   //upperThumb Updates
   if (shift && isPressed("4"))
   {
      if (upperThumb < 90) upperThumb = (upperThumb + d);
      else  upperThumb = 90;
   }
   if (!shift && isPressed("4")) //kapa
   {
      if (upperThumb > -90) upperThumb = (upperThumb - d);
      else upperThumb = -90;
   }
```

Drive Link

You can use the following link to see my result:

https://drive.google.com/file/d/1dIECzr6YwdX-c-1_jRfa3zWw5OY1V-Bf/view?usp=drivesdk

Order: x, y, a, b, m, n, t, p, p, f, F, 1, !, 2, ‘, 3, ^, 4, +