Name: Mustafa Baran Ercan
ID: 2881055520 **Section**: CMPE_360_01

PROJECT 8 REPORT

• Finally, add some code that will make the finger and thumb move apart when 'f' is pressed and together when 'F' is pressed. The center of rotation should be at the wrist.

In order to complete this task, I added 4 more cubes, 1 for thumb and 3 for fingers. I positioned the thumb to the lower middle part of the wrist:

`mv = mult(mv, translate(1.0, -0.2, 0.0));`

and then added the rotation property so the center of rotation would be at the wrist:

```
mv = mult(mv, rotate(thumb,vec3(0, 0, 1)));
mv = mult(mv, translate(0.30, 0, 0.0)) //Translate so it moves the object but not rotat. cntr.
```

Then scaled the thumb so it would be wider:

`mv = mult(mv, scale(0.6, 0.1, 0.8));`

After that, I added the 3 fingers to the upper part of the end of the wrist and located them at the same distance from each other. Again, their rotation center was at the wrist. Then, I added the needed code for the thumb and finger so they would move apart or towards themselves when I pressed f/F.

```
//Claw Updates
if (shift && isPressed("F"))
{
   if (thumb > -8) thumb = (thumb - d);
   else  thumb = -8;
   if (finger < 8) finger = (finger + d);
   else finger = 8;
}
if (!shift && isPressed("F"))
{
   if (thumb < 30) thumb = (thumb + d);
   else thumb = 30;
   if (finger > -30) finger = (finger - d);
   else finger = -30;
}
```

One method is enough to move both the thumb and the fingers.
When f/F is pressed the thumb, and the fingers move to the opposite ways.

I also initialized them with these values so they would look more realistic at the beginning of the program.

`thumb = 15, finger = -15`

• x/X: rotate the arm on the X-axis so you can see it from different angles.

• y/Y: rotate the arm on the Y axis so you can see it from different angles.

To accomplish this, I added two more variables and two more rotation functions. I renamed the original shoulder as “shoulder_z” and added “shoulder_x = 0” and “shoulder_y = 0”.

```
mv = mult(mv,rotate(shoulder_x, vec3(1,0,0)));  //rotate around x axis
mv = mult(mv,rotate(shoulder_y, vec3(0,-1,0))); //rotate around y axis
mv = mult(mv,rotate(shoulder_z, vec3(0,0,1)));  //rotate around z axis
```

• a/A: to rotate the fingers on the x-axis in a positive direction.

• b/B: to rotate the fingers on the x-axis with a negative direction.

Those two are the opposite commands of each other. “A” rotates the fingers on the x-axis with a positive direction whereas “B” rotates the fingers on the x-axis with a negative direction. In order to rotate the fingers on the x-axis, I added the variable “finger_x = 0” and a function to rotate accordingly, here is the function:

```
//Finger Updates
if (!shift && isPressed("A"))
{
   if (finger_x < 30) finger_x = (finger_x + d);
   else  finger_x = 30;
}
if (!shift && isPressed("B"))
{
   if (finger_x > -30) finger_x = (finger_x - d);
   else finger_x = -30;
}
```

• m/M: to rotate the fingers on the y-axis with a positive direction.

• n/N: to rotate the fingers on the y-axis with a negative direction.

Those two are also opposite commands. Just like the previous, thus, I added one more variable "finger_y = 0" and a function to rotate the finger on the y-axis.

```
//Finger Y-Updates
if (!shift && isPressed("M"))
{
    if (finger_y < 30) finger_y = (finger_y + d);
    else  finger_y = 30;
}
if (!shift && isPressed("N"))
{
    if (finger_y > -30) finger_y = (finger_y - d);
    else finger_y = -30;
}
```

• t/T: toggle between solid and wire cubes.

First of all, I needed a Boolean variable to keep track of the state of the cubes. Therefore, I added var "isSolid = false;", which was initially set to false, so it means it is not solid, but a wire cube.

Then I added this to my render() function to set the variable accordingly:

```
var armShape = isSolid ? shapes.solidCube : shapes.wireCube;   // If isSolid is true, use solidCube, else use wireCube.
```

And to catch the event of pressing the key "t/T", I added this code to the handleKeyDown(event) function:

```
if (key == "T") {
    isSolid = !isSolid; // Toggle between wire and solid cube.
}
```

• p/P: toggle between perspective and ortho projections

First, I added a Boolean variable (“isPerspective”) to keep track of the projection type and set it to true initially.

Then, in order to toggle between perspective and orthographic projections, I changed the variable “p” into two states “p_p” and “p_o” meaning perspective and orthographic respectively.
After that, in the handleKeyDown(event) function, I added this if statement to catch and switch the Boolean variable.

```
if (key === 'P') {
    isPerspective = !isPerspective;
}
```

And finally, in the render function, I added this logic statement in order to choose between the two projections.

```
// Set the projection type. If isPerspective is true, use perspective, else use orthographic.
if (isPerspective) {
    gl.uniformMatrix4fv(projLoc, gl.FALSE, flatten(transpose(p_p)));
} else {
    gl.uniformMatrix4fv(projLoc, gl.FALSE, flatten(transpose(p_o)));
}
```

VIDEO

Drive link:

https://drive.google.com/file/d/1WGsygNpp2Z5b1oyYD8wCPFVonXx-8uw3/view?usp=sharing

Order:

I used the same order as in the pdf:

- f/F – Open/Close claw
- 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 a positive direction
- b/B: to rotate the fingers on the x-axis with a negative direction
- m/M: to rotate the fingers on the y-axis with a 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