Jason Tieu
304047667
CS174A
Dis 1A

Assignment 1 Part 1

1.

A: $p = 2i_A + (-2)j_A$

B: $p = 3i_B + 0.5j_B$

C: $p = -4i_C + 3j_C$

2.

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

3.

```
modelMatrix.setAsIdentity();
modelMatrix *= Translate(1,1,1);
modelMatrix *= Scale(1,1,2);
```

4.

$[0.5\ 2.5\ 2]^T$

5.

A: $\begin{matrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$ B: $\begin{matrix} 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 3 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$ C: $\begin{matrix} 0 & 0 & 1 & 2 \\ 0 & 0.5 & 0 & 3.5 \\ -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{matrix}$ D: $\begin{matrix} 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 3 \\ -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$

6.

For arbitrary tilted line y = mx + b

$$M = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & b \\ 0 & 0 & 1 \end{matrix} * \begin{matrix} \cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{matrix} * \begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix} * \begin{matrix} \cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{matrix} * \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{matrix}$$

M = Translate(0,b,0) * RotateX(θ) * Scale(1,-1,1) * RotateX(-θ) * Translate(0,-b,0)

For arbitrary tilted line y = x - 1 and θ = 45

In OpenGL Shader code:

```
modelMatrix.setAsIdentity();
modelMatrix *= Translate(0,b,0);
modelMatrix *= RotateX(θ);
modelMatrix *= Scale(1,-1,1);
modelMatrix *= RotateX(-θ);
modelMatrix *= Translate(0,-b,0);
```

For arbitrary tilted line y = x - 1 and θ = 45

$$M = \begin{matrix}1&0&0\\0&1&-1\\0&0&1\end{matrix} * \begin{matrix}\cos 45&-sin45&0\\\sin 45&cos45&0\\0&0&1\end{matrix} * \begin{matrix}1&0&0\\0&-1&0\\0&0&1\end{matrix} * \begin{matrix}\cos 45&sin45&0\\-sin45&cos45&0\\0&0&1\end{matrix} * \begin{matrix}1&0&0\\0&1&1\\0&0&1\end{matrix}$$

$$M = \begin{matrix}0&1&1\\1&0&-1\\0&0&1\end{matrix}$$

For arbitrary tilted line y = x - 1

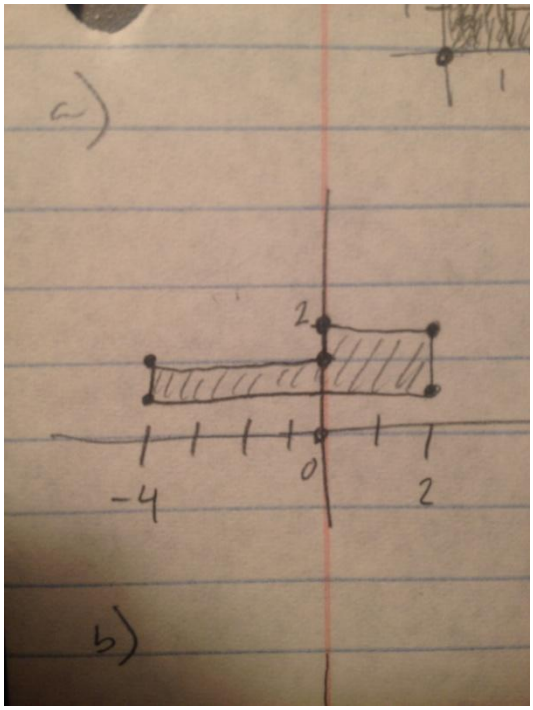
In OpenGL Shader code:

```
modelMatrix.setAsIdentity();
modelMatrix *= Translate(0,-1,0);
modelMatrix *= RotateX(45);
modelMatrix *= Scale(1,-1,1);
modelMatrix *= RotateX(−45);
modelMatrix *= Translate(0,1,0);
```

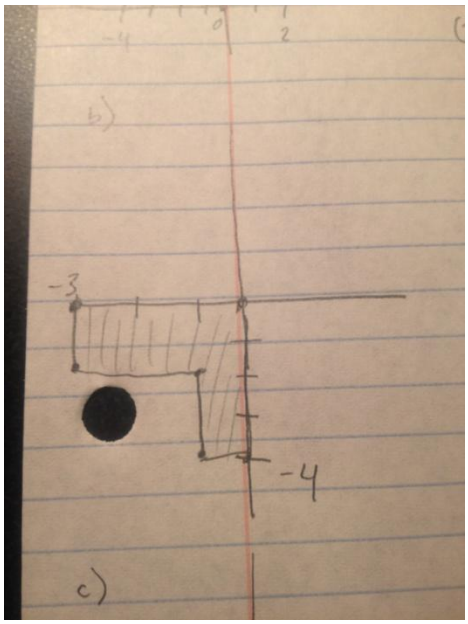
7.

a)L' = ABCL

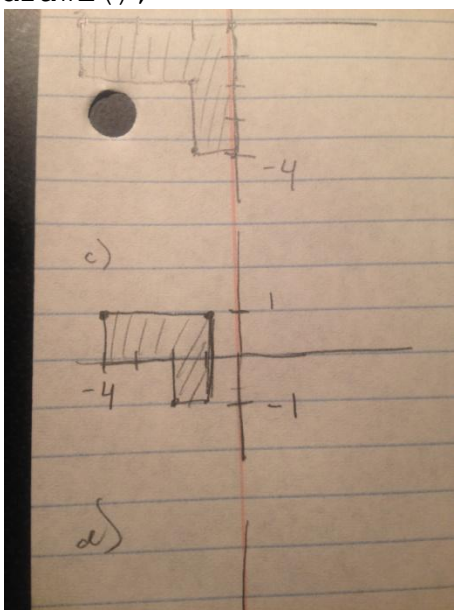
```
modelMatrix *= Scale(2,1,1);
modelMatrix *= Translate(1,1,0);
modelMatrix *= RotateZ(90);
drawL();
```




```
b) L' = CADL
modelMatrix = matrixStack.pop();
matrixStack.push(modelMatrix); //saves for part c
modelMatrix *= RotateZ(90);
modelMatrix *= Scale(2,1,1);
modelMatrix *= Scale(-1,1,1);
drawL();
```




```
c) L' = CBDL
modelMatrix = matrixStack.pop();
matrixStack.push(modelMatrix); //saves for part d
modelMatrix *= RotateZ(90);
modelMatrix *= Translate(1,1,0);
modelMatrix *= Scale(-1,1,1);
drawL();
```




```
d) L' = DCCADL
modelMatrix = matrixStack.pop();
modelMatrix *= Scale(-1,1,1);
modelMatrix *= RotateZ(90);
modelMatrix *= RotateZ(90);
```

```
modelMatrix *= Scale(2,1,1);
modelMatrix *= Scale(-1,1,1);
drawL();
```