Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Computer Graphics(COMP 342) – Lab 5

Submitted To:

Dhiraj Shrestha Sir

Department of Computer Science and Engineering

Submitted By:

Mission Shrestha (54)

CE-2019 Year/Sem: 3rd Year/2nd Sem

Submission Date: 26th May, 2023

Implementation of Cohen Sutherland Line Clipping algorithm & Sutherland Hodgemann polygon clipping algorithm

1. CohenSutherland

Source Code:

```
function cohenSutherland(P1, P2, Xw_min, Yw_min, Xw_max, Yw_max) {
  let x0 = P1[0];
  let y0 = P1[1];
  let x1 = P2[0];
  let y1 = P2[1];
  let vertexData = [];
  let P1_new = [...P1];
  let P2_new = [...P2];
  let m = (y1 - y0) / (x1 - x0);

  let regionCodeP1 = computeRegionCode(x0, y0, Xw_min, Yw_min, Xw_max, Yw_max);
  let regionCodeP2 = computeRegionCode(x1, y1, Xw_min, Yw_min, Xw_max, Yw_max);
  while (true) {
    if ((regionCodeP1 | regionCodeP2) === 0) {
      vertexData.push(...P1_new, ...P2_new);
      draw(gl, vertexData, "line");
      console.log(vertexData);
      vertexData = [];
      vertexData.push(...P1, ...P1_new, ...P2, ...P2_new);
      draw(
        gl,
        vertexData,
        "line",
        `void main(){ gl_FragColor = vec4(1, 0, 0, 1);}`
      );
      break;
    } else if ((regionCodeP1 & regionCodeP2) !== 0) {
      vertexData.push(...P1, ...P2);
      draw(
        gl,
        vertexData,
        "line",
        `void main(){ gl_FragColor = vec4(1, 0, 0, 1);}`
      );
```

```
      break;
    } else {
      let x, y;
      let regionCode = regionCodeP1 !== 0 ? regionCodeP1 : regionCodeP2;
      if ((regionCode & 1) !== 0) {
        x = Xw_min;
        y = y1 + m * (x - x1);
      } else if ((regionCode & 2) !== 0) {
        x = Xw_max;
        y = y1 + m * (x - x1);
      } else if ((regionCode & 4) !== 0) {
        y = Yw_min;
        x = x1 + (y - y1) / m;
      } else if ((regionCode & 8) !== 0) {
        y = Yw_max;
        x = x1 + (y - y1) / m;
      }

      if (regionCode === regionCodeP1) {
        regionCodeP1 = computeRegionCode(x, y, Xw_min, Yw_min, Xw_max, Yw_max);
        P1_new = [];
        P1_new = [x, y, 0];
      } else {
        regionCodeP2 = computeRegionCode(x, y, Xw_min, Yw_min, Xw_max, Yw_max);
        P2_new = [];
        P2_new = [x, y, 0];
      }
    }
  }
}

function computeRegionCode(x, y, Xw_min, Yw_min, Xw_max, Yw_max) {
  let code = 0;
  if (x < Xw_min) {
    code |= 1;
  } else if (x > Xw_max) {
    code |= 2;
  }
  if (y < Yw_min) {
    code |= 4;
  } else if (y > Yw_max) {
    code |= 8;
  }
  return code;
}
```

2) Sutherland Hogdemann

```
function sutherLandHodgemann(P1, P2, P3, P4, P5, Xw_min, Yw_min, Xw_max, Yw_max) {
  let vertexData = [];
  vertexData.push(
    ...P1,
    ...P2,
    ...P2,
    ...P3,
    ...P3,
    ...P4,
    ...P4,
    ...P5,
    ...P5
  );
  cohenSutherland(P1, P2, Xw_min, Yw_min, Xw_max, Yw_max);
  cohenSutherland(P2, P3, Xw_min, Yw_min, Xw_max, Yw_max);
  cohenSutherland(P3, P4, Xw_min, Yw_min, Xw_max, Yw_max);
  cohenSutherland(P4, P5, Xw_min, Yw_min, Xw_max, Yw_max);
  cohenSutherland(P5, P1, Xw_min, Yw_min, Xw_max, Yw_max);
}
```

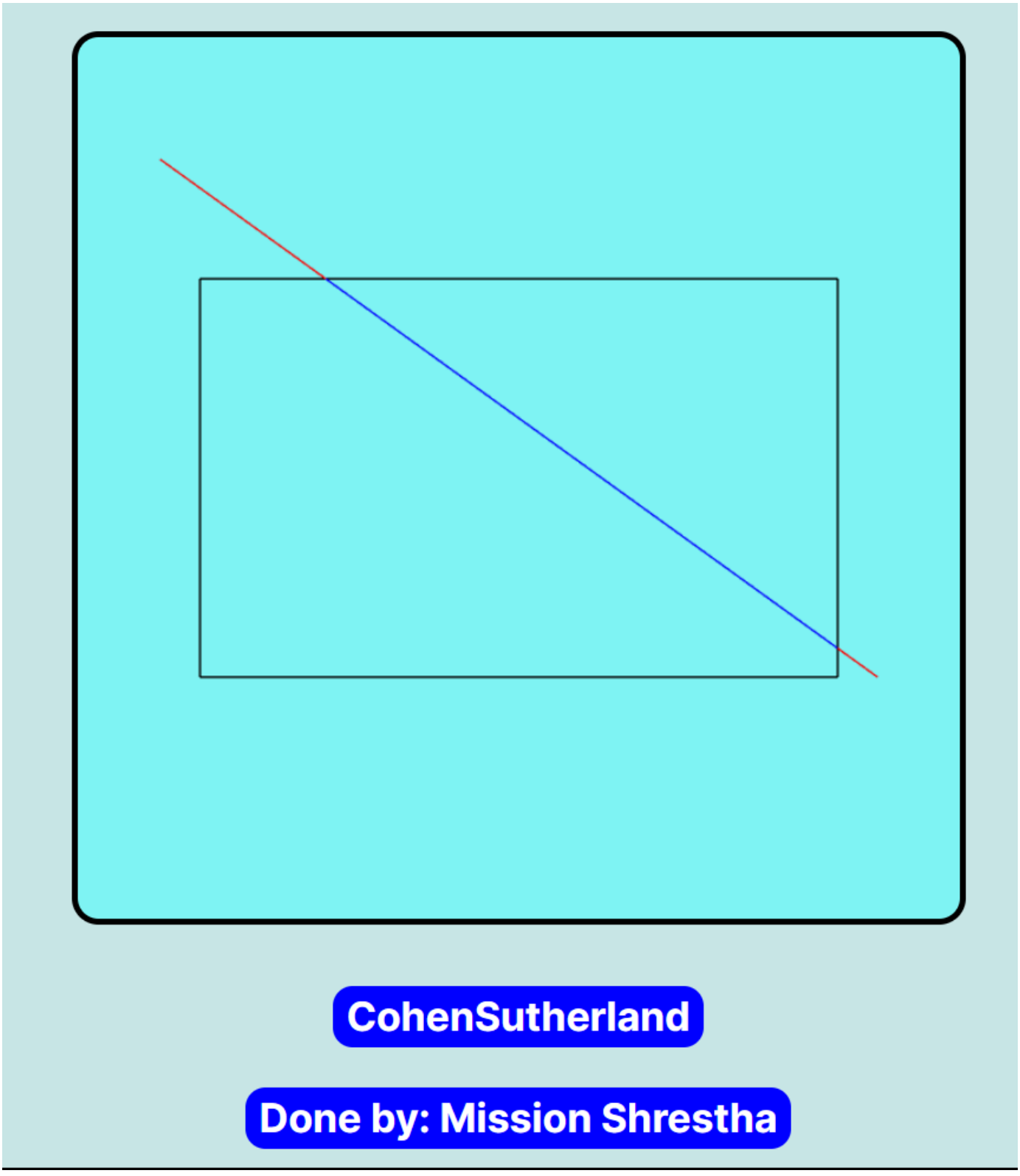
View Port:

```
function viewPortVertex(Xw_min, Yw_min, Xw_max, Yw_max) {

  const vertexData = [];
  let bottomLeft = [Xw_min, Yw_min, 0];
  let topLeft = [Xw_min, Yw_max, 0];
  let topRight = [Xw_max, Yw_max, 0];
  let bottomRight = [Xw_max, Yw_min, 0];
  vertexData.push(
    ...bottomLeft,
    ...topLeft,
    ...topLeft,
    ...topRight,
    ...topRight,
    ...bottomRight,
    ...bottomRight,
    ...bottomLeft
  );
```

```
  return vertexData;
}
```

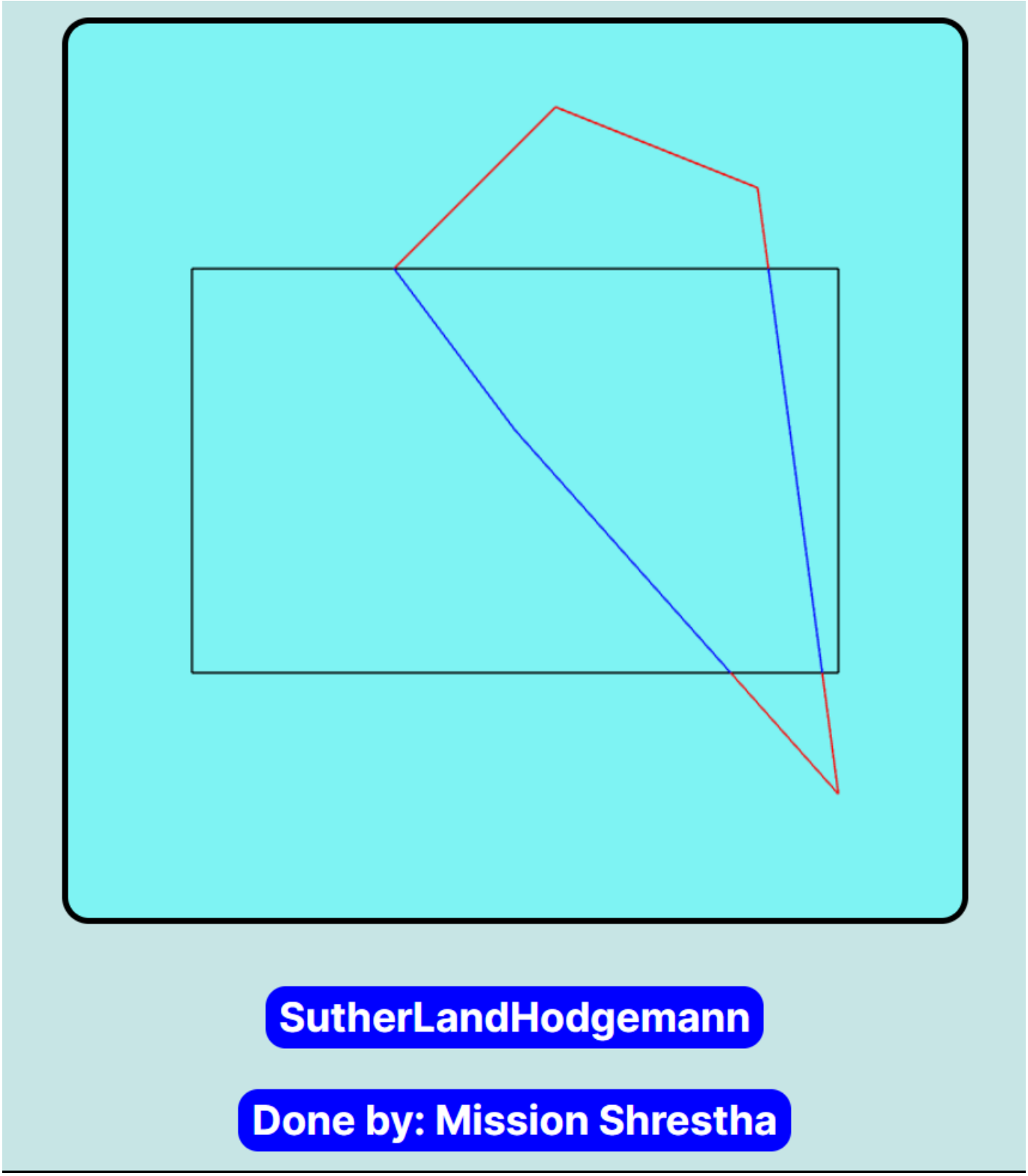
Output:

CohenSutherland:



a.

SutherLandHodgeman:



b.

Program Implementation of :
3D Translation
3D Rotation
3D Scaling

Source Code:

Draw 3d:

```
let frontFace,
  backFace,
  leftFace,
  rightFace,
  topFace,
  bottomFace = [];

function DrawCube() {
  draw(backFace, 'triangle',`void main(){gl_FragColor = vec4(0, 0, 0,
1);}`);
  draw(leftFace, 'triangle',`void main(){gl_FragColor = vec4(1, 1, 1,
1);}`);
  draw(bottomFace, 'triangle',`void main(){gl_FragColor = vec4(1, 0, 1,
1);}`);
  draw(frontFace, 'triangle',`void main(){gl_FragColor = vec4(1, 0, 0,
1);}`);
  draw(rightFace, 'triangle',`void main(){gl_FragColor = vec4(0, 1, 0,
1);}`);
  draw(topFace, 'triangle', `void main(){gl_FragColor = vec4(0, 0, 1,
1);}`);
}

function draw3DObject(O, H, W, L) {
  // P2 P4
  // P1 P3
  let [x, y] = [O[0], O[1]];
  let P1 = [x, y, 1];
```

c.

```
  let P2 = [x, y + H, 1];
  let P3 = [x + L, y, 1];
  let P4 = [x + L, y + H, 1];
  let P5 = createVertex(P3, W / 2, W / 2);
  let P6 = createVertex(P4, W / 2, W / 2);
  let P7 = createVertex(P2, W / 2, W / 2);
  frontFace = [...P1, ...P2, ...P3, ...P2, ...P3, ...P4];
  backFace = translateObject(frontFace, W / 1.75, W / 2.4);
  rightFace = [...P3, ...P4, ...P5, ...P4, ...P5, ...P6];
  leftFace = translateObject(rightFace, -L, 0);
  topFace = [...P2, ...P4, ...P7, ...P4, ...P6, ...P7];
  bottomFace = translateObject(topFace, 0, -H);
  DrawCube();
}

function createVertex(A, Tx, Ty) {
  let vertexData = [
    ...translateObject(
      rotateObject(
        -Math.PI / 20,
        translateObject(translateObject(A, Tx, Ty), -A[0], -A[1])
      ),
      A[0],
      A[1]
    ),
  ];
  return vertexData;
}
```

Transformation:

```
function translate3DObject(Tx, Ty) {

  frontFace = translateObject(frontFace, Tx, Ty);
  backFace = translateObject(backFace, Tx, Ty);
  topFace = translateObject(topFace, Tx, Ty);
  bottomFace = translateObject(bottomFace, Tx, Ty);
```

d.

```
  rightFace = translateObject(rightFace, Tx, Ty);
  leftFace = translateObject(leftFace, Tx, Ty);
  DrawCube();
}

function rotate3DObject(angle) {
  frontFace = rotateObject(angle, frontFace);
  backFace = rotateObject(angle, backFace);
  topFace = rotateObject(angle, topFace);
  bottomFace = rotateObject(angle, bottomFace);
  rightFace = rotateObject(angle, rightFace);
  leftFace = rotateObject(angle, leftFace);
  DrawCube();
}

function scale3DObject(Sx, Sy) {
  frontFace = scaleObject(frontFace, Sx, Sy);
  backFace = scaleObject(backFace, Sx, Sy);
  topFace = scaleObject(topFace, Sx, Sy);
  bottomFace = scaleObject(bottomFace, Sx, Sy);
  rightFace = scaleObject(rightFace, Sx, Sy);
  leftFace = scaleObject(leftFace, Sx, Sy);
  DrawCube();
}
```

Tranform Object:

```
function translateObject(objectData, Tx, Ty) {

  let vertexData = [];
  let translationMatrix = [...[1, 0, Tx], ...[0, 1, Ty], ...[0, 0, 1]];
  vertexData.push(...matrixMultiplication(translationMatrix, objectData,
3));
  return vertexData;
}

function rotateObject(angle, objectData) {
```

e.

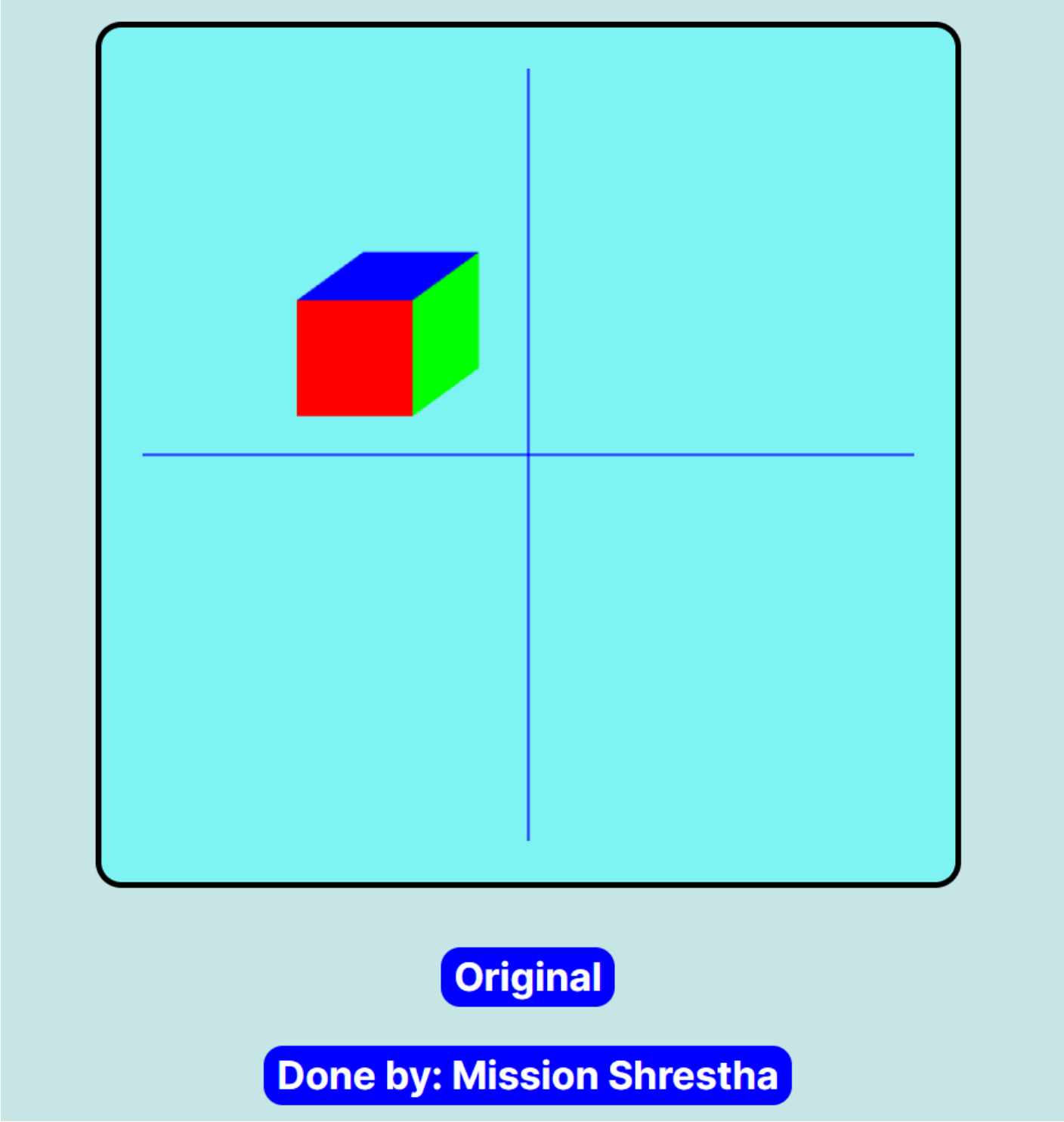
```
  let vertexData = [];
  let cos = Math.cos(angle);
  let sin = Math.sin(angle);
  let rotationMatrix = [...[cos, -sin, 0], ...[sin, cos, 0], ...[0, 0,
1]];
  vertexData.push(...matrixMultiplication(rotationMatrix, objectData, 3));
  return vertexData;
}

function scaleObject(objectData, Sx, Sy) {
  let vertexData = [];
  let scalingMatrix = [...[Sx, 0, 0], ...[0, Sy, 0], ...[0, 0, 1]];
  vertexData.push(...matrixMultiplication(scalingMatrix, objectData, 3));
  return vertexData;
}

function matrixMultiplication(Transformer, coordinates, numElements) {
  let result = [];
  for (let i = 0; i < coordinates.length; i += 3) {
    for (let j = 0; j < 3; j++) {
      let sum = 0;
      for (let k = 0; k < numElements; k++) {
        sum += Transformer[j * 3 + k] * coordinates[i + k];
      }
      result.push(sum);
    }
  }
  return result;
}
```

f.

Output:

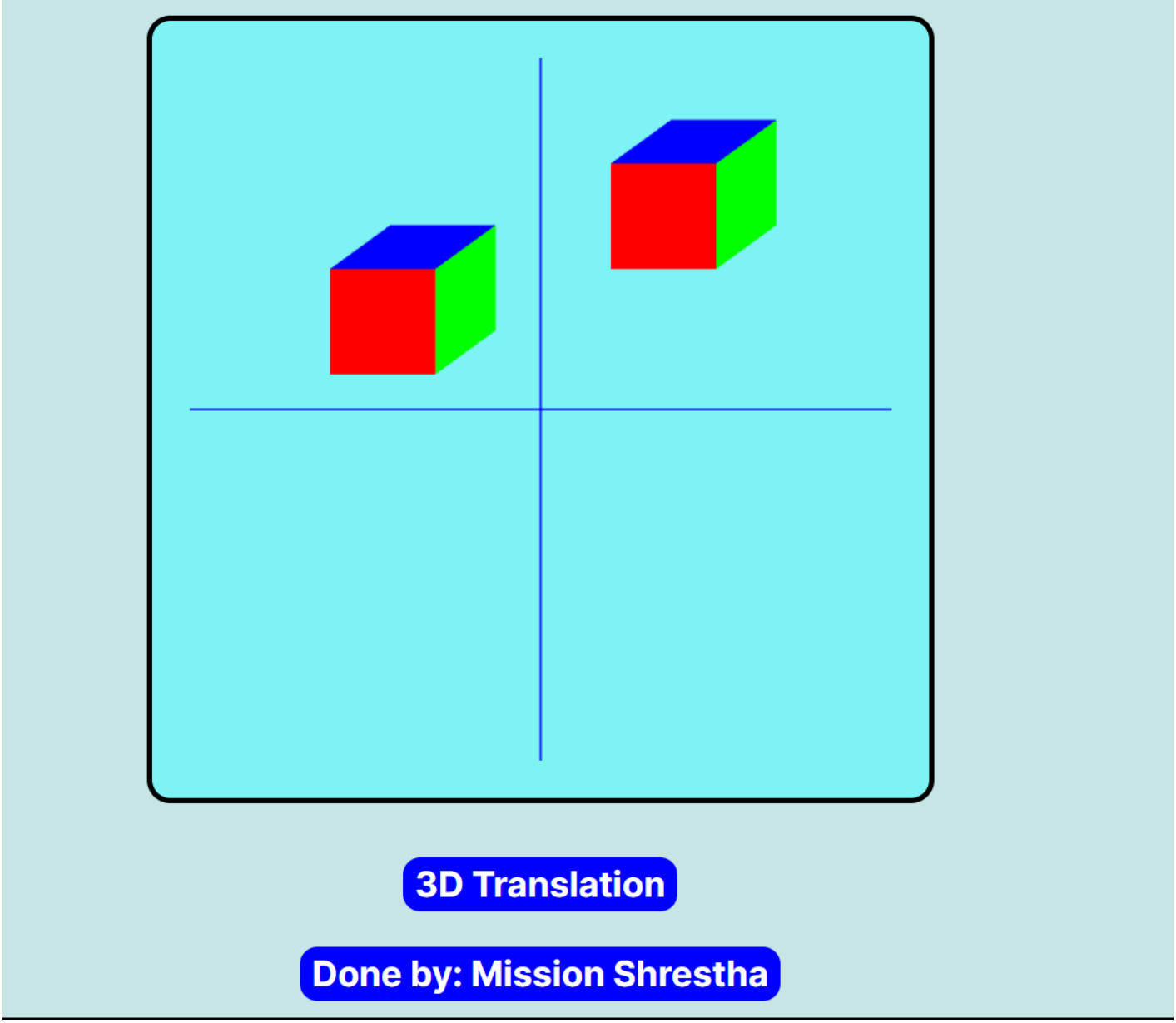
g.

Original:



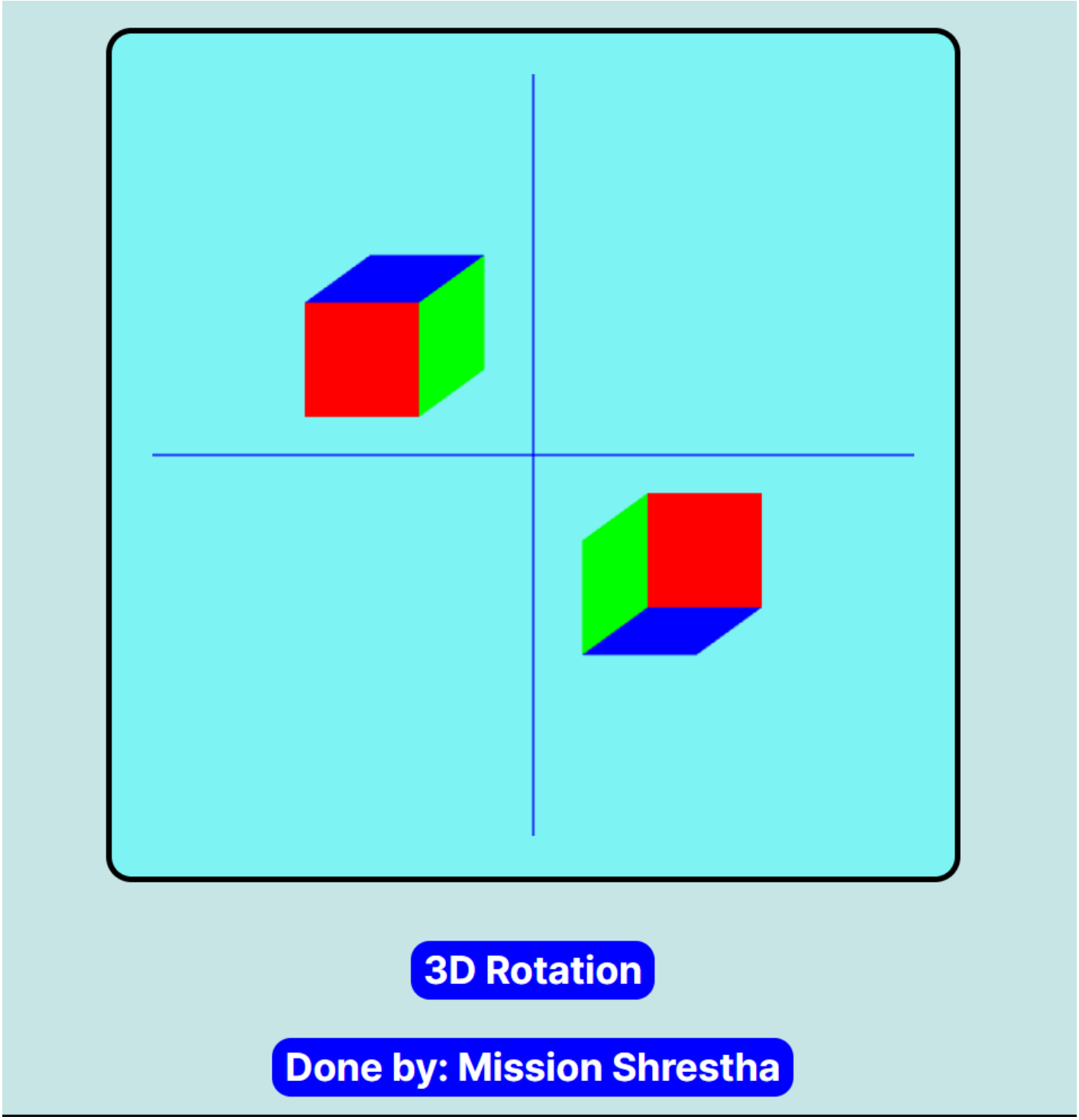
h.

3D Translation:



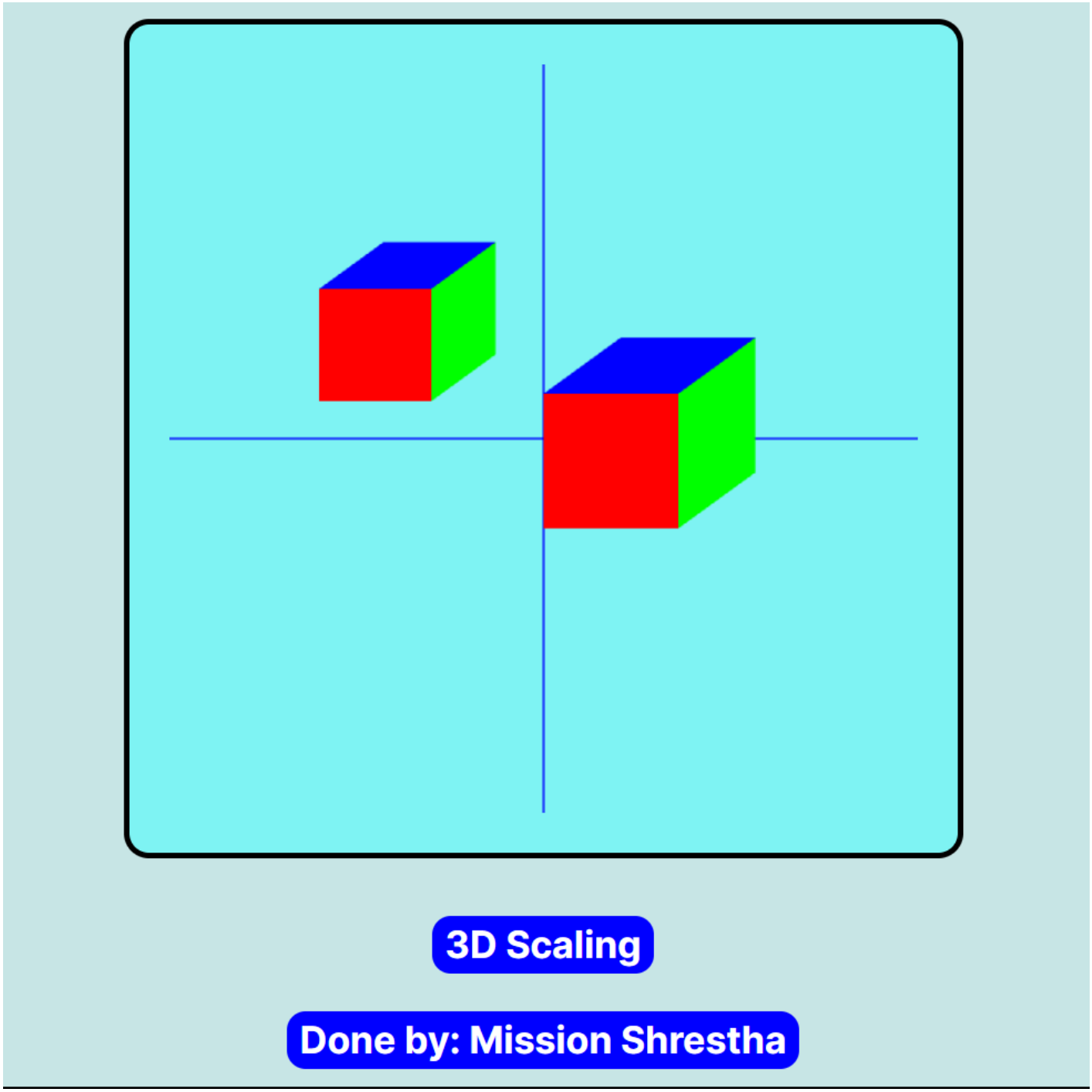
i.

3D Rotation



j.

3D Scaling:



k.

Conclusion:

In conclusion, LAB 5 involved the implementation of two popular clipping algorithms namely Cohen Sutherland Line Clipping algorithm and Sutherland Hodgemann polygon clipping algorithm. In addition, a program was developed to perform 3D translation, rotation, and scaling on any three-dimensional shapes.