Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Report on '**Lab Work 5'** [COMP 342]

Submitted by:

Chandan Kumar Mahato (31)

III-year, II semester

Submitted to:

Mr. Dhiraj Shrestha

Department of Computer Science and Engineering

Submission Date: May 11, 2023

Qlab5

1. ***Implement Cohen Sutherland Line Clipping algorithm***
2. ***Implement Sutherland Hodgeman Polygon Clipping algorithm***
3. ***Write a Program to Implement:***
    a. ***3D Translation***
    b. ***3D Rotation***
    c. ***3D Scaling***

Solution:

https://ckmwebgl.netlify.app

I have used JavaScript as the programming language and WebGL as the graphics library. **JavaScript** is a popular programming language that is used for web development and has the capability of creating interactive graphics and animations on the web. **WebGL** is a graphics library that enables high-performance 3D graphics library that enables high-performance 3D graphics rendering in web browser using JavaScript. These technologies provide a powerful platform for creating engaging and interactive graphics, which is essential for my lab work.

The code snippets for setting graphics environment in my chosen graphics library and programming language and display system resolution are as follow:

Source Code

https://github.com/ChandankMahato/Graphics_Lab_6th_Sem

Cohen Sutherland Line Clipping:

```
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);
      DrawObject(gl.LINES, 2, Green, vertexData, 0, vertexData.length);
      console.log(vertexData);
      vertexData = [];
      vertexData.push(...P1, ...P1_new, ...P2, ...P2_new);
      DrawObject(gl.LINES, 2, Red, vertexData, 0, vertexData.length);
      return;
    } else if ((regionCodeP1 & regionCodeP2) !== 0) {
      vertexData.push(...P1, ...P2);
      DrawObject(gl.LINES, 2, Red, vertexData, 0, vertexData.length);
      return null;
    } 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;
}
```

Sutherland Hodgemann:

```
function sutherLandHodgemann(Xw_min, Yw_min, Xw_max, Yw_max) {
  let vertexData = [];
  let P1 = [-0.6, -0.7, 0];
  let P2 = [0.9, 0.6, 0];
  let P3 = [0.4, 0.7, 0];
  let P4 = [-0.2, 0.6, 0];
  let P5 = [-0.4, 0.2, 0];
  vertexData.push(...P1,...P2,...P2,...P3,...P3,...P4,...P4,...P5,...P5);
  cohenSutherland(P1, P2, Xw_min, Yw_min, Xw_max, Yw_max);
  GLINIT();
  cohenSutherland(P2, P3, Xw_min, Yw_min, Xw_max, Yw_max);
  GLINIT();
  cohenSutherland(P3, P4, Xw_min, Yw_min, Xw_max, Yw_max);
  GLINIT();
  cohenSutherland(P4, P5, Xw_min, Yw_min, Xw_max, Yw_max);
  GLINIT();
  cohenSutherland(P5, P1, Xw_min, Yw_min, Xw_max, Yw_max);
  GLINIT();
}
```

Draw 3D Object:

```
let FrontFace, BackFace, LeftFace, RightFace, TopFace, BottomFace = [];

function DrawCube() {
  DrawObject(gl.TRIANGLES, 3, Yellow, BackFace, 0, BackFace.length);
  DrawObject(gl.TRIANGLES, 3, White, LeftFace, 0, BackFace.length);
  DrawObject(gl.TRIANGLES, 3, Gray, BottomFace, 0, BackFace.length);
  DrawObject(gl.TRIANGLES, 3, Red, FrontFace, 0, FrontFace.length);
  DrawObject(gl.TRIANGLES, 3, Green, RightFace, 0, RightFace.length);
  DrawObject(gl.TRIANGLES, 3, Blue, TopFace, 0, TopFace.length);
}
```

3D object Logic:

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

3D Translation:

```
function translate3DObject(Tx, Ty) {
  FrontFace = translateObject(FrontFace, Tx, Ty);
  BackFace = translateObject(BackFace, Tx, Ty);
  TopFace = translateObject(TopFace, Tx, Ty);
  BottomFace = translateObject(BottomFace, Tx, Ty);
  RightFace = translateObject(RightFace, Tx, Ty);
  LeftFace = translateObject(LeftFace, Tx, Ty);
  DrawCube();
}
```

3D Rotation:

```
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();
}
```

3D Scaling:

```
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();
}
```

Output and Screenshots:

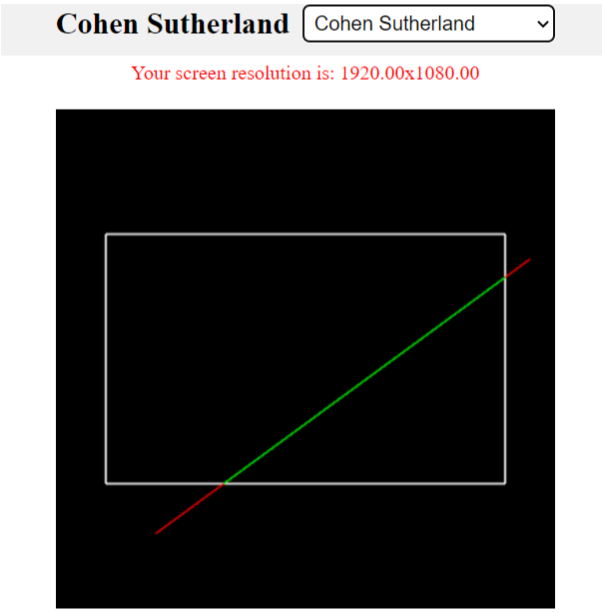

Fig 1: Cohen Sutherland

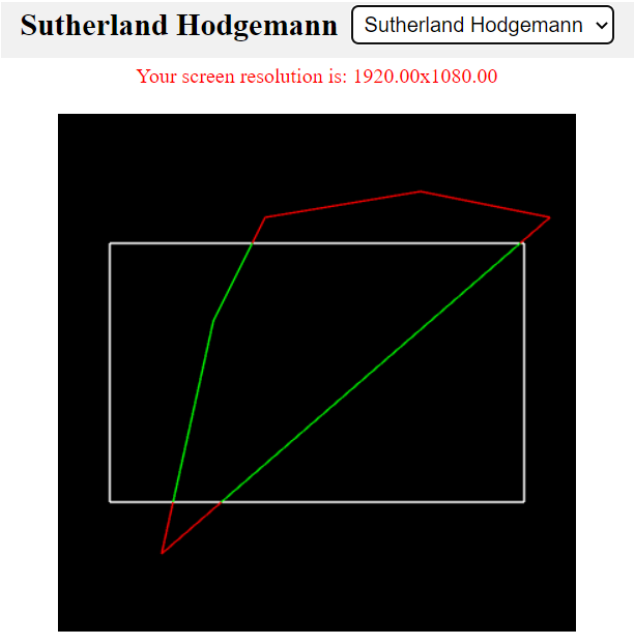

Fig 2: Sutherland Hodgemann

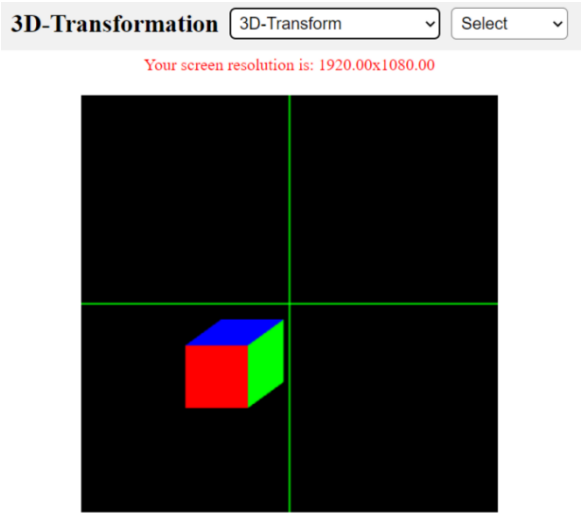

Fig 1: 3D Cube

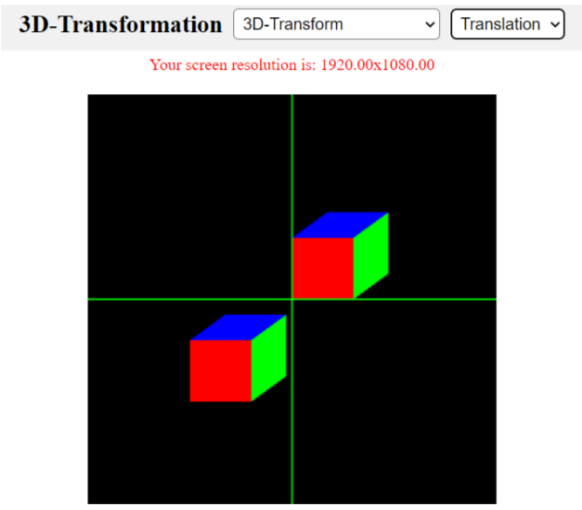

Fig 2: 3D Rotation

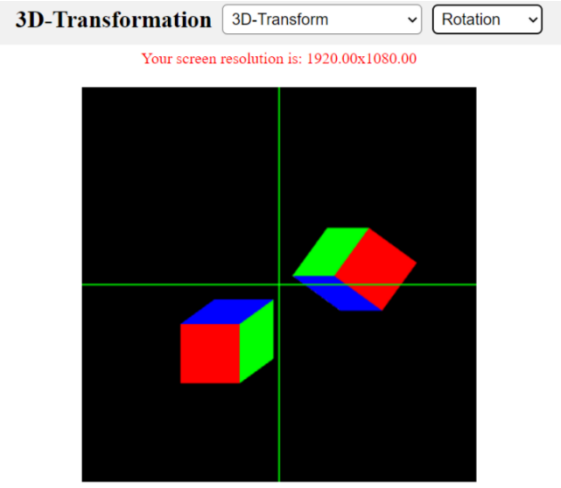

Fig 1: 3D Rotation

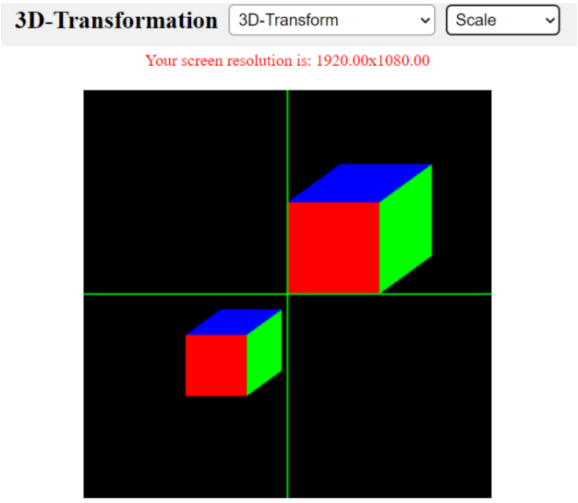

Fig 2: 3D Scaling

Conclusion

After implementing the Cohen Sutherland algorithm and Sutherland Hodgeman Algorithm and 3D Translation, Rotation, and Scaling programs, I have gained a deeper understanding of the fundamental concepts of computer graphics. By utilizing matrix transformations and basic mathematical operations, I was able to clip the line and polygon, also able to manipulate and transform 3D objects in various ways.

This exercise has helped me to develop a better understanding of the concepts involved line clipping and 3D graphics transformations and how to implement them using programming languages. Overall, this exercise has enhanced my skills in computer graphics.