Experiment Code Documentation

Introduction

This document captures the details of the experiment and how it has been implemented in Javascript

PROJECT

Domain Name: Computer Science
Lab Name: Computer Graphics
Experiment Name: Clipping Line

Inspired By: Cohen Sutherland Algorithm

Code Explanation

File Name :Cohen-Sutherland.js
This file has the details behind an iterative implementation of the algorithm mentioned above. Functions have been explained below

1. Function init() :

1.Initializes Global variables and store user input data.
2.Clears the canvas.

2. Function begin() :

1. Called when the start button is pressed.
2. Calls init() and initializes the variables.
3. Stores coordinates of the line and pushes them to a stack.
4. Calls rep() , which fills the canvas with coordinates and begins the experiment

3. Function next_iter():

1.Generates out codes from 0000 to 1001 corresponding to the 9 regions the line is present in.
2. Checks with each side of the bounding rectangle, whether a part of the line intersects that side by calling clip().

4.Function prev_iteration ():

1. Restores the line which(if) was clipped in the previous iteration, which allows the user to understand the functioning of the algorithm.
2. Complements the switch case present in clip()
3. Prints the coordinates of the line if accepted or rejects the line in the remaining case.

5.Function clip():

1. Checks on every iteration, whether the line has the be clipped and new coordinates of intersection must be updated or not.
2. The iteration starts from the left side of the bounding(clipping) rectangle.

6. Function set_outcode() and decbin()

Generate the out codes for each end point of the line and returns the value to decide which part of the line has to be clipped.

The latter returns a binary string of length 4 , for the corresponding out code generated.

8. Function : side_left etc.

These functions check with each side, whether the line has the be clipped or not and push the new intersections in remaining stack.

9. Function find intersections() and check():

Finds the intersections based on the out codes generated ,the orientation of the line (slope) and returns the intersections list.

10. Function drawSqaure(), drawGrid(),drawLine():

1. drawSqaure() draws the bounding rectangle.
2. drawGrid() divides the canvas into 9 regions , as needed in the **Cohen Sutherland algorithm**.
3. drawLine() draws the line to be clipped .

11. Function rep():

1. draws the clipped line on after each iteration has completed and coordinates have been updated. Colors the accepted line in crimson red if accepted.
2. Fills the canvas with the current coordinates of the line being clipped and the clipping box(bounding rectangle).

12. jQuery numLimit:

1. Automatically prevents the user to enter a value greater than a permissible limit as per the canvas.

13. Function main():

1. This function uses WebGL to render the canvas and resize the canvas and the rectangles and display coordinates present in it to make the webpage responsive and hence contribute towards a better UI.

14. Function resize():

1. Resizes the canvas to fit in any webpage of dimension n*m , which prevents the need to scroll every time.

Idea behind the algorithm:

The **Cohen–Sutherland algorithm** is a computer-graphics algorithm for line clipping. It uses a Divide and Conquer strategy. The algorithm divides a two-dimensional space into 9 regions and then efficiently determines the lines and portions of lines that are visible in the central region of interest.(in our case , the bounding rectangle)

The algorithm includes, excludes or partially includes the line based on whether:

- Both endpoints are in the viewport region (bitwise OR of endpoints = 00) trivial accept .
- Both endpoints share at least one non-visible region, which implies that the line does not cross the visible region. (bitwise AND of endpoints ≠ 0): trivial reject.

- Both endpoints are in different regions: in case of this nontrivial situation the algorithm finds one of the two points that is outside the viewport region (there will be at least one point outside).
- The intersection of the outpoint and extended viewport border is then calculated (i.e. with the parametric equation for the line), and this new point replaces the outpoint. The algorithm repeats until a trivial accept or reject occurs.

An overview of how out codes help in clipping the line

For any endpoint (x , y) of a line, the code can be determined that identifies which region the endpoint lies. The code's bits are set according to the following conditions:

- First bit set **1** : Point lies to **left** of window $\mathbf{x} < \mathbf{x_{min}}$
- Second bit set **1** : Point lies to **right** of window $\mathbf{x} > \mathbf{x_{max}}$
- Third bit set **1** : Point lies below(**bottom**) window $\mathbf{y} < \mathbf{y_{min}}$
- fourth bit set **1** : Point lies above(**top**) window $\mathbf{y} > \mathbf{y_{max}}$

File Name : support2.css
File Description : This File Contains the designing and formatting as seen in the html page.