**Project**

**Documentation**

**AND**

**Test Cases**

**For**

**Bressenhem’s Rasterization of Line**

**Using Java Script**

PART - 1

Documentation

1) - Introduction (3)

2) - Steps to run the Experiment (4)

3) - Code Description (5-6)

4) - Project Quiz (6-12)

**1: Introduction**

Rasterization of Line is implemented using the Bressenhem’s Line Drawing Algorithm using the D3 library for plotting the graph.

**Languages Used:**

HTML and CSS

JavaScript

**2: Steps to Run the Experiment**

1. You can proceed directly to the experiment using the default parameters of the size of frame buffer and the starting and ending coordinates, by clicking on **Run the Experiment**.
2. Fix the starting and ending coordinates. The default values are already filled in the input boxes. Once you have finalized the coordinates, you can click on **Run the Experiment** to begin with the experiment.  
   You can also change the values of the frame buffer but not width and height not more than 25 both.
3. Throughout the experiment, you can click on **Next** to go to the next step of the algorithm and **Previous** to go to the previous step.
4. Once the experiment ends, you can again experiment with a new line.

**3: Code Description**

Whole has been written in JavaScript using D3 Library.

Code contains **11** functions:

1. **Bresenham**

This function is main driving function of algorithm with the help of plotLineLow and plotLineHigh functions generates the set of coordinates of the line to be drawn.

1. **Datapicker**

This function picks up the data from the input box.

1. **previousStep**

This function takes the experiment one step back.

1. **nextStep**

This function takes the experiment one step forward.

1. **plotLineLow**

This is called by the Bresenham function to plot the line with negative slope.

1. **plotLineHigh**

This is called by the Bresenham function to plot the line with positive slope.

1. **generateXY**

This function generates the coordinate system.

1. **frameCreator**

This function fixes the frame size and calls the drawGraph function.

1. **drawGraph**

This function draws the Graph.

1. **drawLine**

This function draws the Line.

1. **Experiment**

This function is the wrapper function which generates the line coordinates and calls the drawGraph and drawLine

**4: Project Quiz**

Here the discussion is all about the dynamic quiz created independently created for MIPS Parser project.

It Contains following functions:-

- generate Question Container

- generate Result Container

- put containers

- put Result

- remove child

- generate Random Index

- get Content

- put Content

- check Answers

- submit Answers

- start Quiz

- submit Quiz

**4.1)-generate Question Container:-**

*Function Introduction:-*

This function is used to generate the container structure for each question and their options dynamically.

*Function Implementation:-*

Function takes the Question Id and Answer ID as parameter to assign to respective containers.

**4.2)-generate Result Container:-**

*Function Introduction:-*

This function is used to generate the container structure for Result for each question and their correct and user Answers dynamically.

*Function Implementation:-*

Function takes the Result ID as parameter to assign to respective containers.

**4.3) - put Containers:-**

*Function Introduction:-*

This function is used as wrapper over 4.1 and 4.2

*Function Implementation:-*

This function is randomly generates the number of container and then creates the number of containers using function 4.1 and 4.2 (discussed above).

**4.4) - put Result:-**

*Function Introduction:-*

This function is used to display the results for the quiz

*Function Implementation:-*

Function first creates variables used as ID for different sections. After that the User Answers (stored in global variable) is used to decide the text color and result status of particular question.

**4.5) - remove Child:-**

*Function Introduction:-*

This function is used to remove the child of Result and Quiz container to make a fresh start using the DOM property.

**4.6) - generate Random Index:-**

*Function Introduction:-*

This function is used to generate the Random number within a specified range using the JavaScript inbuilt function.

*Function Implementation:-*

This function uses the constant 11, which is the total number of question stored as JSON data to generate the Random Number. It then check, whether the number generated is generated earlier or not and stores and regenerate accordingly.

**4.7) - get Content:-**

*Function Introduction:-*

This function is used to place the options of a passed Question in random order.

**4.8) - put Questions:-**

*Function Introduction:-*

This function is used as wrapper over 4.7 and is used to put all the questions as the number of container.

**4.9) - check Answers:-**

*Function Introduction:-*

Functions used to validate and stores the score of User for their answers.

*Function Implementation:-*

This function is checking whether the user has skipped the question or not and if not skipped, it stores which option is selected by user and increase the score of user, if selected option is the right answer. It returns the above status to the caller function for storing purposes.

**4.10) - submit Answers:-**

*Function Introduction:-*

This function is used as wrapper over 4.9

*Function Implementation:-*

This function stores the status from function 4.9 and push it on global variable for further calculations.

**4.11) - start Quiz:-**

*Function Introduction:-*

This function is on-click listener of star button of quiz

*Function Implementation:-*

Functions alters the visibility of some containers and their content and places random number of question with randomize option. It also removes the previous questions or result (if exists).

**4.12) - submit Quiz:-**

*Function Introduction:-*

This function is on-click listener of submit button of quiz

*Function Implementation:-*

Functions alters the visibility of some containers and their content and places result with their result status. It also initializes all the global variables for the fresh start...

PART - 2

Test Cases

Input\_1 (Wrong range of Input)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Input Out of Bound | | | **Test Case ID** | Input-1 | | |  | |
| **Test Case Description** | | User provide wrong range of coordinates | | | **Test Priority** | Medium | | |  | |
| **Pre-Requisite** | | NA | | | **Post-Requisite** | NA | | |  | |
| Test Execution Steps: | | | | | | | | | | |
| **Sr. No** | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | | **Test Browser** | **Test Result** | |  |
| 1 | Launch application | | Open newindex.htm | Home page | Home page | | ff-66.0.4 | Pass | |  |
| 2 | Provide wrong range i.e. difference between the abscissa and ordinates. | | Fill input out of range i.e. greater the 25 | Alert with showing the warning | Alert with showing the warning | | ff-66.0.4 | Pass | |  |
|  |  | |  |  |  | |  |  | |  |

References:-

Software Testing Help Website

Source Code of Triple DES USING JS

Prepared By:-

Prince Pathak

]

Project Details:-

Project Domain – Computer Graphics

Issue Number – 256

Project Topic – Rasterization of Line

Submitted to:-

SRIP@IIITH