

# **A Study on the Comparison between HTML5 and OpenGL in Rendering Fractal Tree**

by

**Mehbuba Zerine Khan**

A project submitted in partial fulfilment of the requirements for the degree of  
Masters of Science in Computer Science & Engineering



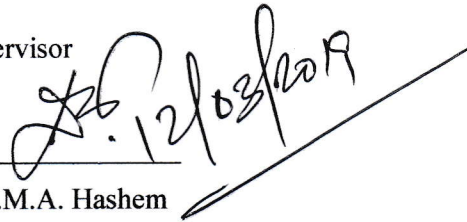
Department of Computer Science and Engineering  
Khulna University of Engineering & technology  
Khulna 9203, Bangladesh

**March 2019**

## Declaration

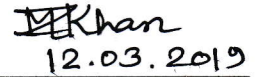
This is to certify that the project work entitled "A Study on the Comparison between HTML5 and OpenGL in Rendering Fractal Tree" has been carried out by Mehbuba Zerine Khan in the Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh. The above project work or any part of this work has not been submitted anywhere for the award of any degree or diploma.

Supervisor



Prof. Dr. M.M.A. Hashem

Candidate



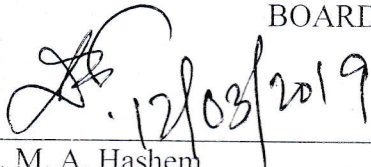
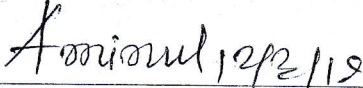
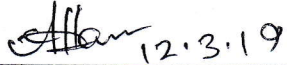

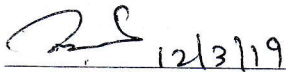
Mehbuba Zerine Khan

Roll: 1407502

## Approval

This is to certify that the project work submitted by Mehbuba Zerine Khan entitled "A Study on the Comparison between HTML5 and OpenGL in Rendering Fractal" <sup>Tree</sup> has been approved by the board of examiners for the partial fulfilment of the requirements for the degree of Masters of Science in Computer Science & Engineering in the Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh in March 2019.

### BOARD OF EXAMINERS

1.  12/03/2019  
 \_\_\_\_\_  
 Dr. M. M. A. Hashem  
 Professor  
 Department of Computer Science and Engineering  
 Khulna University of Engineering & Technology, Khulna-9203  
 Supervisor  
 (Chairman)
2.  12/02/19  
 \_\_\_\_\_  
 Dr. Md. Aminul Haque Akhand  
 Head & Professor  
 Department of Computer Science and Engineering  
 Khulna University of Engineering & Technology, Khulna-9203  
 Member
3.  12.3.19  
 \_\_\_\_\_  
 Dr. K. M. Azharul Hasan  
 Professor  
 Department of Computer Science and Engineering  
 Khulna University of Engineering & Technology, Khulna-9203  
 Member
4.  12/03/19  
 \_\_\_\_\_  
 Dr. Pintu Chandra Shill  
 Professor  
 Department of Computer Science and Engineering  
 Khulna University of Engineering & Technology, Khulna-9203  
 Member
5.  12/3/19  
 \_\_\_\_\_  
 Dr. Kamrul Hasan Talukder  
 Professor  
 Computer Science and Engineering Discipline  
 Khulna University, Khulna  
 Member  
 (External)

## **Abstract**

This is the era of web applications as for each desktop application, there is a corresponding web application being developed. Each web application consists of interactive graphical user interface and some of them requires in-browser rendering. In this period, it's high time to study the abilities of modern-day web applications on handling graphical operations. As both HTML5 and OpenGL are strong tools for graphical operation and both depicts rendering capabilities on different platforms, in this project, they have been compared thoroughly. To measure the effectiveness and compare the results from HTML5 and OpenGL, Fractals are considered to be drawn on web platform and on desktop graphical program. In this project, A simple HTML5 web page is implemented along with a C++ based command line program is also implemented to render fractal trees. HTML5 and OpenGL both performed significantly well in case of rendering fractal trees where HTML5 fell a little bit short in case of rendering time in case of large number of iterations. As the number of iterations increased rapidly, the rendering time required by the HTML5 increased but it performed on par with OpenGL in rendering quality.

## **Acknowledgements**

First of all, I would like to express my grateful thanks to the almighty to complete my task. I would like to express my deepest sense of gratitude and sincere thanks to my respected supervisor Dr. M.M.A. Hashem, Professor, Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh for providing me the opportunity of working under his kind supervision. For his proper guidance, co-operation, invaluable suggestions and constant encouragement throughout this research work it is easy to find the right way to fulfil the desire research. I will remember his inspiring guidance and cordial behaviour forever in my future life.

I should take this opportunity to express my sincere thanks to all the teachers and staffs of this department for their valuable advice and moral support in my research work. I wish to convey my hearty thanks to all my friends and class fellows for helping me according to their ability.

I wish to thank my parents and husband for their great understanding and support.

## Contents

	Declaration	i
	Approval	ii
	Abstract	iii
	Acknowledgements	iv
	List of Figures	vi
<b>CHAPTER I</b>	Introduction	1
	1.1 Background	1
	1.2 Motivation	1
	1.3 Problem Statement	2
	1.4 Specific Objective	3
	1.5 Methodology	3
	1.6 Organization of the Project	5
<b>CHAPTER II</b>	Literature Review	6
	2.1 HTML5	6
	2.1.1 Canvas	6
	2.2 OpenGL	8
<b>CHAPTER III</b>	System Architecture	9
	3.1 System Model	9
	3.2 Fractal Tree	12
<b>CHAPTER IV</b>	Implementation	12
	4.1 Preparing HTML File	12
	4.2 Drawing Fractal in HTML5	12
	4.3 Drawing Fractal in OpenGL	14
<b>CHAPTER V</b>	Experimental Analysis	15
	5.1 Experimental Setup	15
	5.2 Experimental Results	15
<b>CHAPTER VI</b>	6.1 Conclusions	18
	6.2 Future Work	18
	References	19
	Appendix-A	22

## LIST OF FIGURES

<b>Figure No</b>	<b>Description</b>	<b>Page</b>
1.1	Fractal drawing of a tree leaf.	2
1.2	Proposed system for comparison of performance between HTML5 and OpenGL.	5
3.1	Structure of the system.	9
3.2	SVG image before zoom	10
3.3	SVG image after zooming	11
3.4	Fractal Drawing Techniques	12
4.1	Flowchart of Fractal Tree drawing	13
4.2	Fractal Tree drawn by HTML5	14
4.2	Fractal Tree drawn by OpenGL	16
5.1	Rendering Time on multiple web browsers	17
5.2	Comparison of execution time on different web-browsers	18
5.3	Comparison of HTML5 and OpenGL on rendering time with respect to number of iterations in fractal trees	19

*Dedicated to my son, parents, husband and*

*Honourable supervisor sir. . .*



## **CHAPTER I**

### **Introduction**

#### **1.1 Background**

With the advancement of internet and technologies, web applications are rapidly becoming more complex and sophisticated. Once, desktop computers were considered as the sole platform for the development and deployment of applications. Now, people are more intrigued about using web applications instead of desktop applications.

Web applications now provide various intricate features to the users which were previously only available to desktop applications. Rendering of 3D graphics is one of the complex and intensive work for any platform. Web applications these days have earned the ability to render these intensive 3D graphics on the go. As the number of devices with graphics processing units (GPUs) are increasing in number, the web developers can harness the power of these GPUs more accurately than ever to produce wonderful graphical environments for general users.

#### **1.2 Motivation**

The World Wide Web Consortium (W3C) introduced the world a new era of web through HTML5 in 2014 [1, 2]. The earlier versions of HTML could only perform simple tasks such as creating static pages and manipulating static data, lacking the ability to handle dynamic data. Moreover, the web interface were not rich and interactive enough in earlier versions of HTML. To handle these shortcomings, W3C released HTML5. It provides a lot of advanced features to the general users as well as to web developers such as local file handling, image operations over pixels and support for complex 2D & 3D graphics. The features of HTML5 are supported in most of the popular browsers such as Chrome, Safari, and Mozilla Firefox etc.

On the other hand, OpenGL is the most popular development environment for producing eye popping 2D and 3D graphics applications in current industry. In 1992, it was first introduced to the developers. Since then, it helped the graphics developers to achieve astonishing results in rendering complex graphical structures and produced numerous number of applications both in desktop platform and application programming interfaces (API) [3]. OpenGL provides the necessary graphics processing functions such as

rendering, texture mapping and other visualization tools that helps the speedy development of graphical applications. Developers can easily harness the raw power of GPUs through these various functions of OpenGL in most of the popular platforms such as desktop and workstations environment.

### 1.3 Problem Statement

A fractal is an endless pattern. Fractals are vastly mind-boggling design patterns that are self-comparable crosswise over various scales. They are made by rehashing a straightforward procedure again and again in a progressive feedback circuit. Driven by recursion, fractals are pictures of great dynamic. Geometrically, they exist in our known dimensions. Fractal designs are to a great degree, well-known, since nature is brimming with fractals. For example: trees, streams, coastlines, mountains, mists, seashells, sea tempests, and so forth. Fig. 1.1 shows the fractal drawing of a tree leaf [4]. Theoretical fractals, for example, the Mandelbrot Set – can be produced by a PC figuring a straightforward condition again and again. Both HTML5 and OpenGL have the capabilities to render fractals in modern day platforms.

HTML5 is being considered in case of browser-based rendering with respect to other rendering platforms such as CSS3, WebGL etc. However, OpenGL is used in rendering in most of the desktop environments and its rather useful for heavy graphics-based operations. For simple object rendering, browser rendering can be a new area of

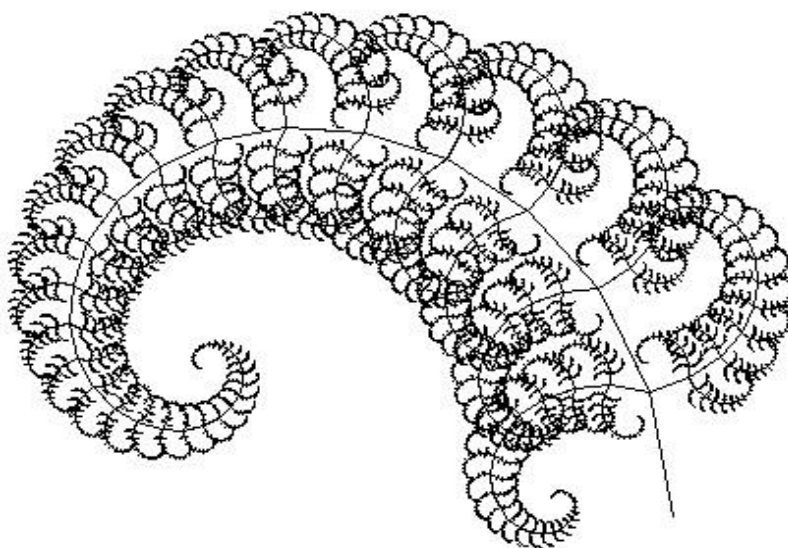


Fig 1.1: Fractal drawing of a tree leaf [4].

study that requires light weight computations from the user which can be performed by a CPU rather than having a GPU to handle these operations in case of using OpenGL. Developing an application as a web application can allow it to run on wider array of platforms than what would have been possible if it had been written as a normal desktop application.

#### **1.4 Specific Objective**

The key objective of this project is to study the comparison of HTML5 and OpenGL in case of simple object rendering. To reach the goal the study is carried out with the following specific objectives:

- Investigate HTML5 for better understanding of its features.
- Investigate OpenGL for its underlying structure and features.
- Investigate the effect of HTML5 on rendering simple objects (i.e fractal tree) in web browsers.
- Investigate the effect of OpenGL on rendering simple objects (i.e fractal tree) in desktop environment.
- Performance comparison between the rendering ability of HTML5 & OpenGL on simple fractal tree.

#### **1.5 Methodology**

As the web evolves, so do its underlying technologies. Browsers continuously gain new features while deprecating old ones, fix bugs, and implement newer versions of technologies such as HTML, JavaScript, HTTP and CSS. HTML5 is not just the new version of the web's markup language, but a collection of new features for a more modern web. These new features enable users to interact with web sites in different ways, while helping developers have an easier time delivering content to users. HTML5 includes new CSS features that allow developers to style a web page differently, new JavaScript features that bring new capabilities to a web page, and new markup that allows for a semantic web. New semantic elements give developers a way to better structure web pages through HTML tags that better describe the content. It has been possible to render images in a web page since the beginning on HTML. This is done through the use of the <img> tag, but this is solely for rendering an image and is not a new feature. We are instead concerned with more elaborate graphic technologies.

To properly draw or render an item in a browser, HTML5 uses ‘Canvas’, a raster-based technology that renders pixels to a screen. <canvas> element is used to draw graphics, on the fly, via JavaScript. The <canvas> element is only a container for graphics. Thus, JavaScript will be used to actually draw the graphics. An HTML file is maintained which contains three sections such as HTML, CSS, JavaScript sections respectively. The HTML section is used to declare the necessary tags in order to create a web page with a canvas container. The CSS section is used to properly decorate the environment on the web page. JavaScript sections contains the necessary function and variables to draw the simple objects (i.e fractal tree) in the selected canvas area of the web page.

Similarly, a C++ based OpenGL program is written in CodeBlocks IDE in Microsoft windows environment to draw the similar simple objects (i.e fractal tree) drawn in web browser. The proposed system is depicted in fig. 1.2.

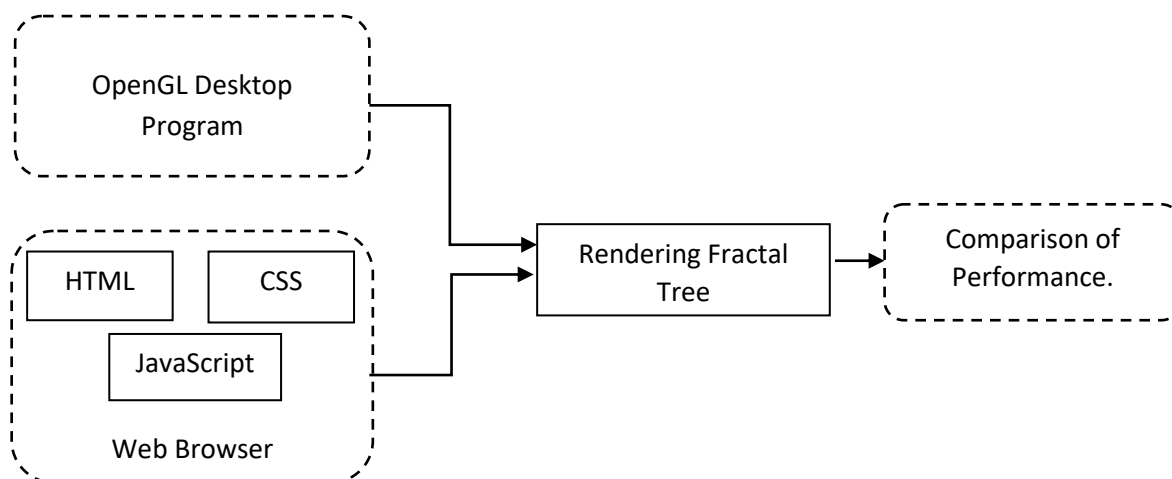


Fig 1.2: Proposed system for comparison of performance between HTML5 and OpenGL.

## 1.6 Organization of the Project

The rest of this project is organized in five chapters, which are as follows:

Chapter 2 Provides literature review related to graphical rendering technologies such as HTML5 and OpenGL and a brief overview on fractal tree.

Chapter 3 Specifies problem formulation that are essential to implement the project.

Chapter 4 Introduces the implementation of the project. Contains all codes and functions related to HTML5 and OpenGL.

Chapter 5 Contains experimental studies details, result analysis, discussion for

future work.

Chapter 6 Conclude the project work and recommendation some future work.

## CHAPTER II

### Literature Review

#### Introduction

Web is the fastest growing resource that is rapidly and constantly used across almost every platform. To this day, some research work has been performed on the evaluation of 3D graphics technologies in the field of web applications [5]. Among them, various technologies and techniques has been measured for effectiveness in case of rendering graphical objects and their shortcomings has been studied thoroughly.

#### 2.1 HTML5

As HTML5 is the newer version of HTML, it helps us creating interactive and rich webpages. HTML has grown drastically from simply emphasizing on production of audio, video and animations to providing offline functionality, local storage and geo location on any client-side database.

The development of HTML5 gives rise to a wide variety of multimedia applications. Without any help of proprietary techniques from the browser it supports animations and can play audio and video. For web developer and web designer the new features provided by the HTML5 would add up new values. HTML5 provides cross platform, which is designed to display webpages on Smart TV, Tablet, PC, Smartphone etc. So many websites as well as browser designers are adopting HTML5 elements. The main temptation for the web developers and browsers is that someone can create rich web pages, web-based applications and enhanced forms without mastering or licensing multiple proprietary techniques.

##### 2.1.1 Canvas

One of the most interesting new elements in HTML5, `<canvas>` provides an area of the screen which can be drawn upon programmatically. It enjoys widespread support, being available in the most recent versions of Chrome, Firefox, Internet Explorer, Opera, and Safari as well as Mobile Safari and Android Browser. The basic approach to drawing on a canvas is simple: acquire a graphics context, and use the context's API to affect your changes. In the current HTML5 specification, the only defined context is "2d". The 2d context provides basic drawing primitives such as `fillRect`, `lineTo`, and `arc`, as well as more

complicated features such as Bezier curves, color gradients, and copying in an existing image.

To combine video and animations on webpages HTML5 uses <canvas> element for drawing graphics using java script. To present 2D/3D graphics script is used and graphics is contained in the canvas. To make graphics heavy pages render fast, various types of methods for boxes, texts, drawing paths, images are being used.

The role of Graphics Processing Units (GPUs) in mobile devices has been getting wide attention, as they provide huge acceleration benefits for compute-intensive apps that require fast rendering or 3D gaming. GPUs utilize low power and are far more efficient than CPUs for performing the complex mathematical calculations needed for graphics rendering (fewer compute cycles).

Web Graphics Library (WebGL), based on HTML5 and OpenGL ES 2.0 standards, is a JavaScript API for rendering rich and interactive 3D graphics without installing any additional software [6,7]. It is designed to efficiently utilize the GPU to perform the intensive rendering computations that enables the creation of real-time interactive data visualizations. With the availability of GPUs on almost every modern mobile devices, combined with the emergence of industry standards such as WebGL, it has opened up enormous opportunities in developing visually appealing and rich experience apps, and are expected to play a significant role in mobile computing.

WebGL is also used in some geographical applications where it is being used as a tool for visualization [8]. As WebGL is a variation of OpenGL and shares some of its capabilities on the web platform, it also shares some features with HTML5. Both HTML5 and WebGL have been considered to be used in medical image rendering [9].

In this project, HTML5 is being considered in case of browser-based rendering with respect to other rendering platforms such as CSS3, WebGL etc. [10]. However, OpenGL is used in rendering in most of the desktop environments and its rather useful for heavy graphics-based operations. For simple object rendering, browser rendering can be a new area of study that requires light weight computations from the user which can be performed by a CPU rather than having a GPU to handle these operations in case of using OpenGL. Developers are now preferring web version of applications to develop which

gives them an opportunity to reach a large number of users rather than developing desktop-based applications [11].

## **2.2 OpenGL**

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications [12]. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

HTML5 and OpenGL, both possess rendering capabilities of objects and they produce marvellous graphical contents. Still, HTML5 works better in web architecture, whereas OpenGL is developed only considering desktop and workstations, and it utilizes the Graphical processing resources of those platforms completely.



## CHAPTER III

### System Architecture

#### Introduction

In this chapter, the architecture of the graphic system is described as well as the features of HTML5 and OpenGL will be discussed.

#### 3.1 System Model

As the web evolves, so do its underlying technologies. Browsers continuously gain new features while deprecating old ones, fix bugs, and implement newer versions of technologies such as HTML, JavaScript, HTTP and CSS. HTML5 is not just the new version of the web's markup language, but a collection of new features for a more modern web. These new features enable users to interact with web sites in different ways, while helping developers have an easier time delivering content to users. HTML5 includes new CSS features that allow developers to style a web page differently, new JavaScript features that bring new capabilities to a web page, and new markup that allows for a semantic web. New semantic elements give developers a way to better structure web pages through HTML tags that better describe the content. It has been possible to render images in a web page since the beginning on HTML. This is done through the use of the <img> tag, but this is solely for rendering an image and is not a new feature. We are instead concerned with more elaborate graphic technologies. We want to render fractal trees with the help of HTML5 and also OpenGL to compare the outcome on the basis of graphics rendering capabilities of these platforms.

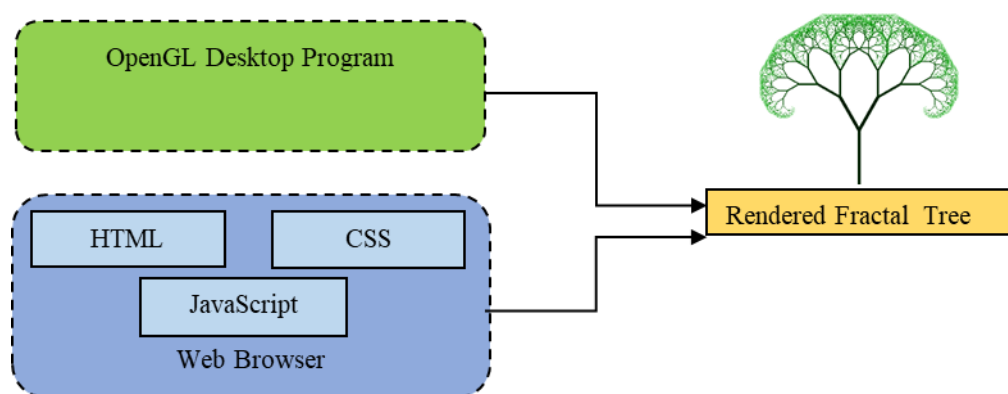


Fig 3.1: Structure of the system.

In Fig. 3.1, the structure of the system is shown. The experiment is divided into two sections, developing an OpenGL desktop program and creating a HTML file with CSS and JavaScript in order to draw fractal trees. The resultant fractal trees will be compared between the competing platforms. There are certain features which the web technologies and desktop environment that are used in our experiment holds. HTML5 provides advanced set of features for its users such as unique ways to render graphics to produce interactive and animated graphical elements in web browsers. HTML5 achieves this through CSS3, Canvas [13], Scalar Vector Graphics (SVG) [14,15], and WebGL.

To properly show videos and animations on web browsers [13], HTML5 utilizes <canvas> element. It draws different sort of things in a webpage by the help of java script. The 2D/3D elements in the webpage is contained within <canvas>. CSS3 provides rich outlook to the webpage by designing menus, buttons and other basic elements. For faster loading speed of intense graphical elements in a web page, various forms of methods are used.

One of them is SVG. The prominent feature of SVG is that the quality of the images remains unchanged even after compression or enlargement of it. As can be seen in Fig. 3.2, the SVG image of a butterfly remains unchanged even after continuous zooming. The zoomed image, circled in red, in Fig. 3.3 shows that SVG remain undistorted and unchanged after manipulation of its size.



Fig 3.2: SVG image before zoom.



Fig 3.3: SVG image after zooming.

On the other hand, WebGL is based on OpenGL ES which lacks many of the features that regular OpenGL has [15]. There are a number of other features that OpenGL has that WebGL does not have such as 3D textures, vertex array objects [16].

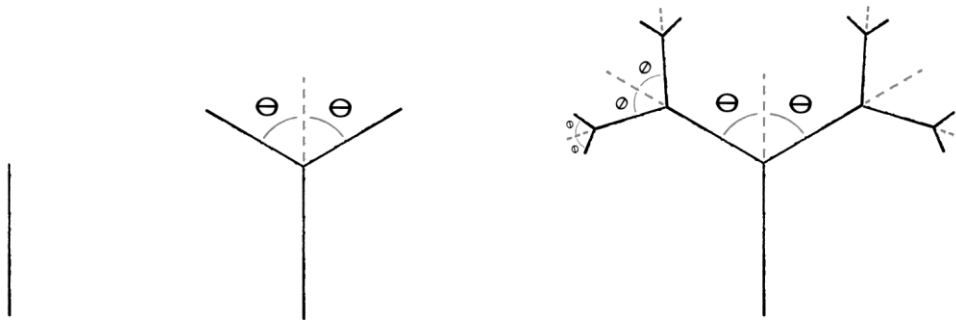
OpenGL, another way to say "Open Graphics Library," is an application programming interface (API) intended for rendering 2D and 3D designs [17]. It gives a typical arrangement of directions that can be utilized to oversee designs in various applications and on different stages. OpenGL code is generally cross platform, giving the graphics developers the flexibility to develop and publish features rich graphical content simultaneously in Windows, Mac or cell phones. Current GPUs are specifically optimized for OpenGL to perform better in intense graphics operation. Features of OpenGL includes illustration of polygons, colouring different shapes, texture mapping and transformation of polygons and controlling the movement of them.

This is an important area of research since web browsers are evolving rapidly, as are graphics in general. It is also an important area of study because many applications that were previously standalone applications that ran natively on desktops and laptops are moving into the web. For example, Microsoft's popular Office suite is now available completely inside a browser with no installation necessary. Adobe has also done the same with their Creative Cloud software as an alternative to installing photoshop on one's computer that renders SVG.

### 3.2 Fractal Tree

A *fractal tree* is defined recursively by symmetric binary branching. The trunk of length 1 splits into two branches of length  $r$ , each making an angle  $q$  with the direction of the trunk. Both of these branches divide into two branches of length  $r^2$ , each making an angle  $q$  with the direction of its parent branch. Continuing in this way for infinitely many branching, the tree is the set of branches, together with their limit points, called *branch tips* [18]. The steps [19] to draw a simple fractal tree are given below:

1. Draw a line.
2. At the end of the line, (a) rotate to the left and draw a shorter line and (b) rotate to the right and draw a shorter line.
3. Repeat step 2 for the new lines, again and again and again.



In Fig. 3.4, each branch is determined by a string of symbols  $L$  and  $R$  specifying the choice of direction taken along the tree to reach the branch. A branch determined by a string of  $n$  symbols has length  $r^n$ ; a branch tip is determined by an infinite string of symbols. Most of the analysis in results from converting eventually periodic symbol strings of branch tips into geometric series for the  $x$  and  $y$  coordinates of the branch tips, and making appropriate interpretations.

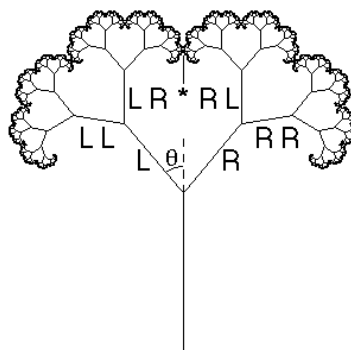


Fig 3.4: Fractal Drawing Techniques.

## CHAPTER IV

### Implementation

#### Introduction

In this chapter, the implementation of the graphic system architecture is discussed and the rendering factors are also being addressed thoroughly.

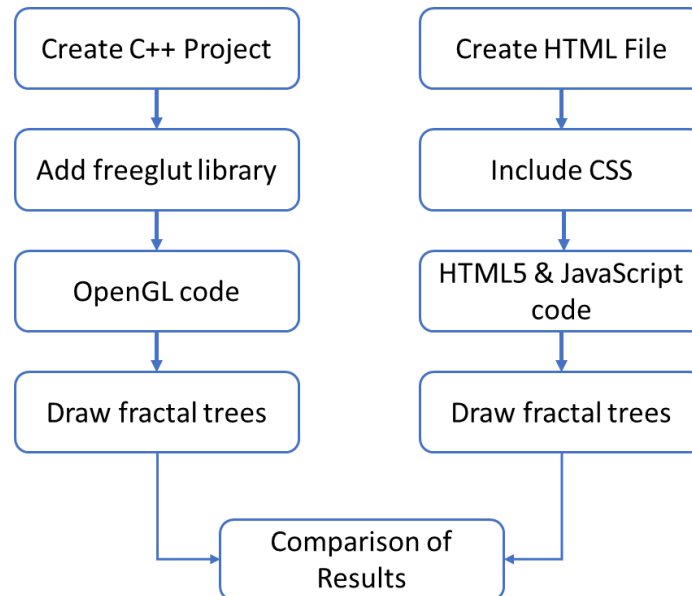


Fig 4.1: Flowchart of Fractal Tree drawing.

To prepare the system, we need a HTML file for implementing HTML5 along with a C++ coded program for OpenGL program implementation which can be seen in Fig. 4.1. The details of the functions involved in rendering, are discussed in their corresponding subsections.

#### 4.1 Preparing HTML File

To properly draw or render an item in a browser, HTML5 uses ‘Canvas’, a raster-based technology that renders pixels to a screen. <canvas> element is used to draw graphics, on the fly, via JavaScript. The <canvas> element is only a container for graphics. Thus, JavaScript will be used to actually draw the graphics. An HTML file will contain three sections such as

- HTML
- CSS

- JavaScript

The HTML section is used to declare the necessary tags in order to create a web page with a canvas container. The CSS section will be used to properly decorate the environment on the web page. JavaScript sections will contain the necessary function and variables to draw the simple objects (i.e. fractal tree) in the selected canvas area of the web page.

## 4.2 Drawing Fractal in HTML5

To draw the fractal in HTML5, canvas element is first called by

```
<canvas id="canvas"></canvas>
```

We set the margin and padding to zero as we don't want to obstruct the view of the user while rendering fractal trees. Canvas variables and elements are at first initialized in the `<script>` element.

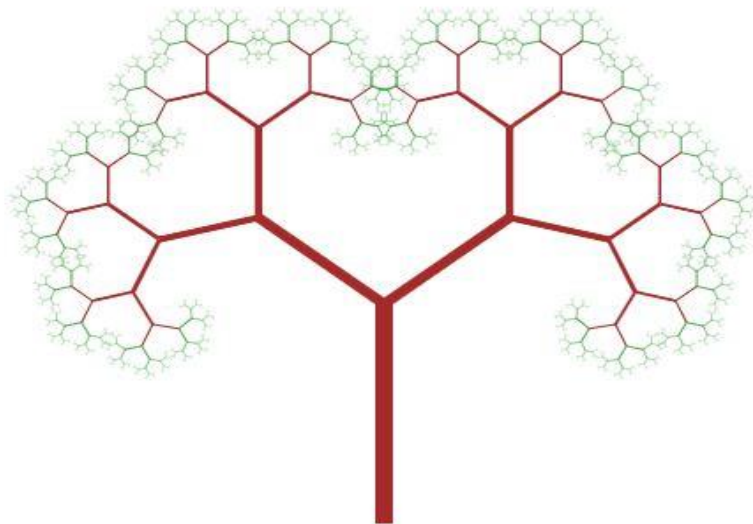


Fig 4.2: Fractal Tree drawn by HTML5.

Three user functions are created namely `init()`, `branches()` and `get_endpoint()`. These functions are declared and defined in the JavaScript block of the HTML file. In `init()` function, the basic parameters are initialized such as length of the trunk, divergence angle, reduction rate etc. Also, the beginning co-ordinate of the fractal drawing is initialized in this function. In `branches()` function, the drawing co-ordinate of the branches of the fractal tree is calculated. This function is called recursively to continuously draw shapes in the canvas. The `get_endpoint()` function continuously checks up the endpoints of the branches

by using vector formulae.

The `time()` functions calculates the operation time of these functions and the duration of the CPU process. It is invoked to properly calculate the timing of drawing fractal tree in the system.

Fig. 4.2 depicts the outcome of the HTML file in a browser. If we zoom into the depth of the leaf in the generated tree, we will find the same fractal pattern all along. It's the basic characteristic of fractal and it proves the successful run of the implemented program.

### 4.3 Drawing Fractal in OpenGL

A C++ based OpenGL program is written in CodeBlocks IDE in Microsoft Windows environment to draw similar fractal tree drawn in web browser in Fig. 4.2.

Along with the functions already implemented in JavaScript, some new methods are also defined in the program such as `reshape()`, `display()`, `maketree()` etc. The `maketree()` function repeatedly draw the branches on the tree trunks. The trunks are drawn into the `init()` function [20], same as HTML5. The `reshape()` function is invoked into the `maketree()` function in order to reduce or extend the size of each tree brunch.

To properly draw the tree on the window, the `display()` function plays a significant role. It utilizes both `COLOR_BUFFER_BIT` and `DEPTH_BUFFER_BIT` [21] to flush the drawing window pane at the beginning of the program. The resultant tree drawn by OpenGL can be seen in Fig. 4.3.

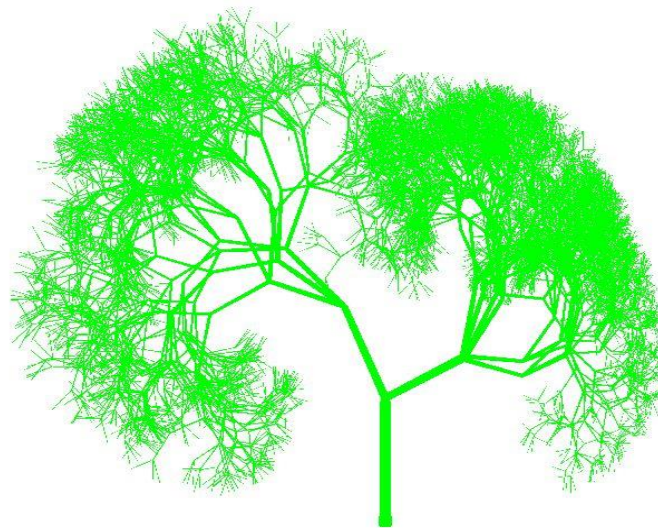


Fig 4.3: Fractal Tree drawn by OpenGL.

As the generated image shows the repeated nature of fractal patterns, it intuitively proves the effectiveness and correctness of the OpenGL program.



## CHAPTER V

### Experimental Analysis

#### Introduction

In this chapter, we evaluate the performance of the implemented programs. The performance comparison of HTML5 and OpenGL over rendering fractals are briefly discussed in this chapter.

#### 5.1 Experimental Setup

The project was implemented on a desktop computer with the following configuration

Intel(R) Core™ i5-4130 CPU @ 3.40 GHz

- RAM 8 GB
- CodeBlocks 17.12
- OpenGL Glut

#### 5.2 Experimental Results

This study consists of two major portions: study of HTML5 and standard OpenGL features and performing rendering of fractal tree on both technology. Outcomes and expected results of this study are discussed in this section.

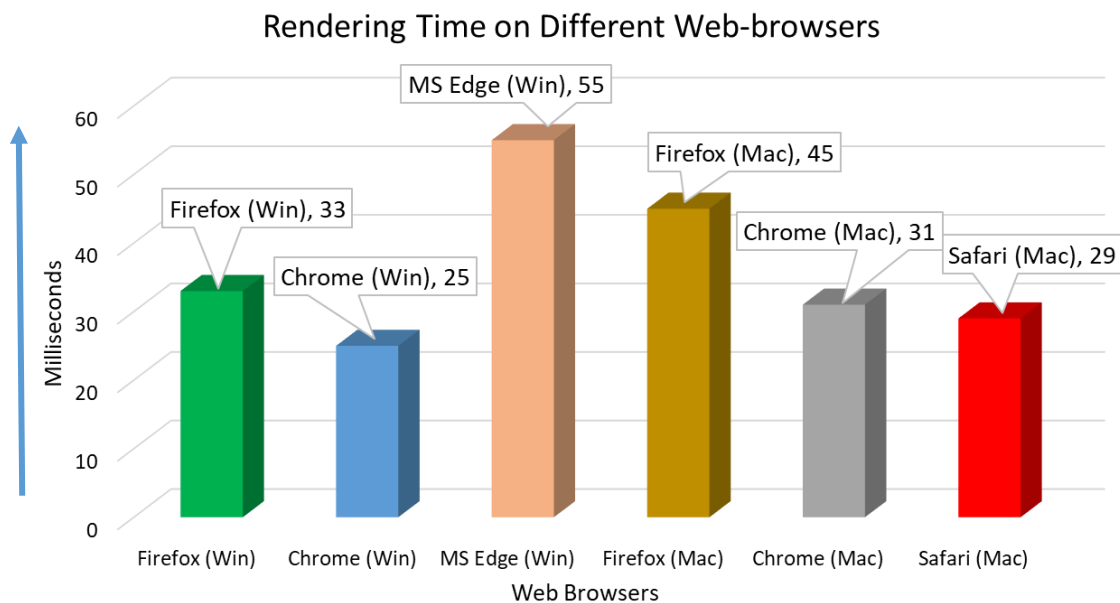


Fig 5.1: Rendering Time of fractal tree on multiple web browsers.

In Fig. 5.1, the comparison of rendering time of HTML5 on different web-browsers can be seen. If we run the HTML5 program in Firefox browser in windows, the corresponding rendering time is 33ms. Chrome in windows perform relatively better than other web-browsers yielding a rendering time of about 25ms. On Mac, both the Firefox and Chrome requires a little bit more time to render fractal trees [22]. Safari performs comparatively better with respect to Chrome and Firefox in Mac. The slowest performer among the web-browsers is Microsoft Edge with a rendering time of 55ms.

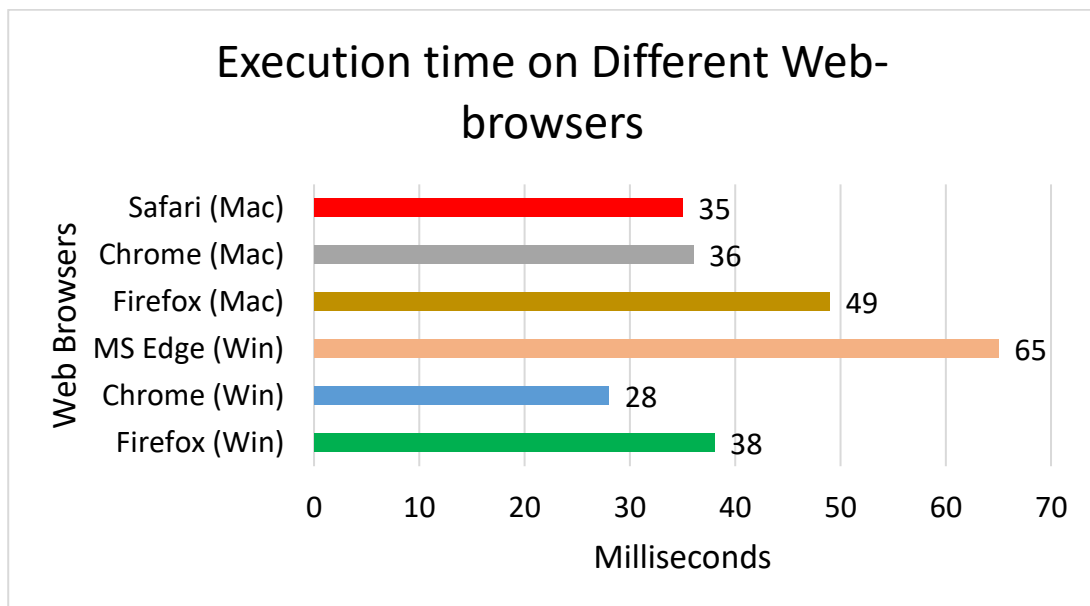


Fig 5.2: Comparison of execution time on different web-browsers.

In Fig. 5.2, the execution time of HTML files takes a little jump from the rendering time on different web-browsers. The total execution time is calculated from the beginning of the rendering process to the end of the program. The results show that Chrome on windows takes the lead with 28ms, 3ms extra to the rendering time. On the other hand, Safari and Chrome, both performs similarly in Mac environment with 35ms and 36ms respectively [23]. Again, the slowest of them all is Microsoft Edge browser with a 65ms execution time.

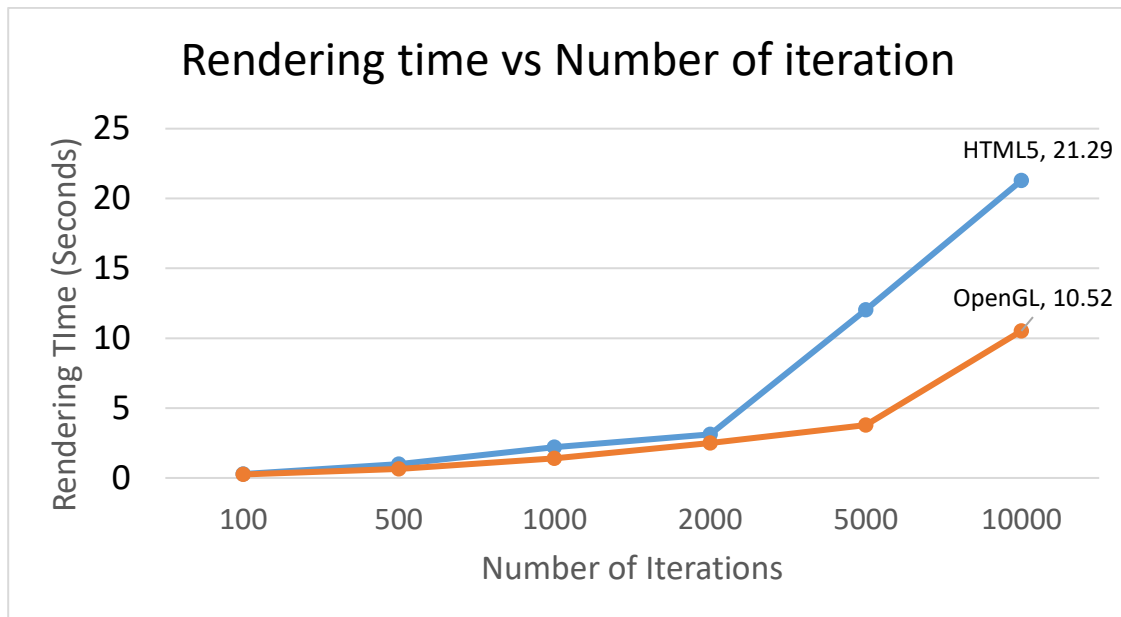


Fig 5.3: Comparison of HTML5 and OpenGL on rendering time with respect to number of iterations in fractal trees.

The rendering time of HTML5 and OpenGL differs with the increasing number of iterations performed in building fractal trees which can be observed in Fig. 5.3. For 100 iterations, both HTML5 and OpenGL performs similarly with a rendering time of 0.28s & 0.25s respectively [23]. As the number of iterations increases rapidly, the rendering time required by the HTML5 increases. Whereas, OpenGL performs faster with respect to increasing number of iterations. Till 2000 iteration, HTML5 maintained a healthy competition with OpenGL needing 3.12s and 2.5s respectively. After 5000 iterations, HTML5 lags behind the functional capability of OpenGL with a huge margin.

The web programmers or Graphic intensive program developers or game developers face a hard time to choose the right tool for their right task. The results of both HTML5 and OpenGL shows that, in any platform, let that be web browser-based application or desktop-based application development, if the amount of graphical drawing is heavy, OpenGL performs better than HTML5. Whereas, HTML performs well in lightweight drawing or rendering applications, if the sole target of the developer is to develop any light weight web app or canvas-based application. Both of these technologies provide numerous opportunities to the application developers.

## CHAPTER VI

### Conclusions

#### 6.1 Conclusions

A simple HTML5 web page is implemented along with a C ++ based command line program is also implemented to provide OpenGL support in rendering fractal trees in both of these programs. The proposed method shows that HTML5 and OpenGL both performs significantly well in case of rendering fractals. The comparison in case of rendering time shows that OpenGL performs quite well in desktop platforms as it has the power of GPUs to render things much faster. On the other hand, HTML5 shines in case of web-based rendering where the rendering only relies on the resources available to web browsers. The two platforms vary significantly but it also shows the strength of HTML5 in rendering the fractals on the web whereas OpenGL relies on desktop platform. The study can be further extended in case of rendering complex 3D graphical structures such as 3D fractals, 3D human models etc.

#### 6.2 Future Work

Use of HTML5 and OpenGL is a growing field. The internet is booming with graphical content. As the number of content increases, the use of the technology becomes eminent. Some directions for further research include:

- How HTML5 can be applied into rendering more heavy graphical contents.
- How to utilize canvas more efficiently to generate graphical contents.
- Reduction of rendering time of HTML5 and OpenGL.
- The study can be further extended in case of rendering complex 3D graphical structures such as 3D fractals, 3D human models etc.

## References

1. "Know HTML5," May 5, 2018. [Online]. [www.w3schools.com/html/html5\\_intro.asp](http://www.w3schools.com/html/html5_intro.asp). [Accessed: July. 3, 2018]
2. HTML5.2: editor's draft" [Online]. September 2017, <http://w3c.github.io/html/>. [Accessed: 11 May, 2018]
3. "The OpenGL Book" [Online]. [www.openglbook.com/chapter-0-preface-what-is-opengl.html](http://www.openglbook.com/chapter-0-preface-what-is-opengl.html) [Accessed: 11 May, 2018]
4. "Fractal drawing of Ferns" [Online]. <https://ayoqq.org/explore/fractal-drawing-fern/>[Accessed: 11 May, 2018]
5. Mikael Waerner, "3D Graphics Technologies for Web Applications: An Evaluation from the Perspective of a Real World Application", *Master thesis in information coding*, Linkoping Institute of Technology, 2012
6. P. Lubbers, B. Albers, and F. Salim., "Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development". *Apress*, 2010.
7. "HTML5" – Wikipedia. [Online]. Available: <http://en.wikipedia.org/wiki/HTML5>. [Accessed: Nov 24, 2017]
8. R. Miao, J. Song and Y. Zhu, "3D geographic scenes visualization based on WebGL," *2017 6th International Conference on Agro-Geoinformatics, Fairfax, VA, 2017*, pp. 1-6.
9. Min, Qiusha, Zhifeng Wang, and Neng Liu. "An Evaluation of HTML5 and WebGL for Medical Imaging Applications." *Journal of healthcare engineering 2018* (2018).
10. Ratha, Ashis Kumar, ShibaniSahu, and Priya Meher. "HTML5 in Web Development: A New Approach." *International Research Journal of Engineering and Technology (IRJET)*, Vol 5, Issue 1 (2018).
11. Mowery, Keaton, and HovavShacham. "Pixel perfect: Fingerprinting canvas in HTML5." *Proceedings of W2SP* (2012): 1-12.
12. "What is OpenGL." 2013. [Online]. <https://www.opengl.org/about/> [Accessed: 18 May, 2018]
13. Jagannathan, AK, S Suresh, VG Venkatarahaman, and SR Milton. "A Canvas- Based Presentation Tool Using Scalable Vector Graphics." *Technology for Education (T4E), 2012 IEEE Fourth International Conference on Technology for Education 2012* (2012): 149-152. Digital.

14. "Scalable Vector Graphics (SVG) 1.0 Specification." W3 Organization. 4 Sept 2001. [Online] <http://www.w3.org/TR/SVG10/> [Accessed: 4 June, 2018]
15. "The Khronos Group. WebGL - OpenGL ES 2.0 for the [Online]." 2012. Available: <http://www.khronos.org/webgl/> [Accessed: Nov 24, 2017]
16. "What are the differences between WebGL and OpenGL?" [Online] <https://www.quora.com/What-are-the-differences-between-WebGL-and-OpenGL> [Accessed: 5 May, 2018]
17. E. Pinto, G. Amador, and A. Gomes. "A graphics library for delivering 3D contents on web browsers." In *Digital Content, Multimedia Technology and its Application (IDC), 2010 6th International Conference on*, pages 109-114, Aug. 2010.
18. "Definitions of fractals " [Online]<http://www.math.union.edu/research/fractaltrees/FractalTreesDefs.html>[Accessed: Nov 24, 2017]
19. "Chapter 8. Fractals"[Online]<https://natureofcode.com/book/chapter-8-fractals/>[Accessed: 4 June, 2018]
20. Belmonte, Nicolas Garcia. "Learning OpenGL." 2013. [Online]. <http://www.senchalabs.org/philog/> [Accessed: May 11, 2018]
21. Kristian Sons, Felix Klein, Dmitri Rubinstein, SergiyByelozyorov, and Philipp Slusallek. "XML3D: interactive 3D graphics for the web." In *Proceedings of the 15th International Conference on Web 3D Technology, Web3D '10*, pages 175-184, New York, NY, USA, 2010. ACM.
22. Lee, Geonhee, Seunghyun Lee, and Soonchul Kwon. "A Study on Loading Speed of Web Browser for 3D Object." *Welcome Remarks* (2018): 55.
23. MehbubaZerin Khan and M. M. A. Hashem, "A Comparison between HTML5 and OpenGL in Rendering Fractal", *International Conference on Electrical, Computer and Communication Engineering (ECCE 2019)*, Cox's Bazar,07-09 February, 2019.

## Appendix -A

### Program Codes

#### HTML5 Codes:

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas"></canvas>

/*Some basic CSS*/
* {margin: 0; padding: 0;}
/*To remove the scrollers*/
#canvas {display: block;}

<script>
window.onload = function(){
    var canvas = document.getElementById("canvas");
    var ctx = canvas.getContext("2d");
    //Lets resize the canvas to occupy the full page
    var W = window.innerWidth;
    var H = window.innerHeight;
    canvas.width = W;
    canvas.height = H;

    //Some variables
    var length, divergence, reduction, line_width, start_points = [];

    init();

    function init()
    {
        //filling the canvas white
```

```
ctx.fillStyle = "white";
ctx.fillRect(0, 0, W, H);

//Lets draw the trunk of the tree
//lets randomise the variables
//length of the trunk - 100-150
length = 100 + Math.round(Math.random()*50);
//angle at which branches will diverge - 10-60
divergence = 10 + Math.round(Math.random()*50);
//Every branch will be 0.75times of the previous one - 0.5-0.75
//with 2 decimal points
reduction = Math.round(50 + Math.random()*20)/100;
//width of the branch/trunk
line_width = 10;

//This is the end point of the trunk, from where branches will diverge
var trunk = {x: W/2, y: length+50, angle: 90};
//It becomes the start point for branches
start_points = []; //empty the start points on every init();
start_points.push(trunk);

//Y coordinates go positive downwards, hence they are inverted by deducting it
//from the canvas height = H
ctx.beginPath();
ctx.moveTo(trunk.x, H-50);
ctx.lineTo(trunk.x, H-trunk.y);
ctx.strokeStyle = "brown";
ctx.lineWidth = line_width;
ctx.stroke();

branches();
}
```



```

//Lets draw the branches now
function branches()
{
    //reducing line_width and length
lengthlength = length * reduction;
line_widthline_width = line_width * reduction;
ctx.lineWidth = line_width;

    var new_start_points = [];
ctx.beginPath();
    for(var i = 0; i<start_points.length; i++)
    {
        var sp = start_points[i];
        //2 branches will come out of every start point. Hence there will be
        //2 end points. There is a difference in the divergence.
        var ep1 = get_endpoint(sp.x, sp.y, sp.angle+divergence, length);
        var ep2 = get_endpoint(sp.x, sp.y, sp.angle-divergence, length);

        //drawing the branches now
ctx.moveTo(sp.x, H-sp.y);
ctx.lineTo(ep1.x, H-ep1.y);
ctx.moveTo(sp.x, H-sp.y);
ctx.lineTo(ep2.x, H-ep2.y);

        //Time to make this function recursive to draw more branches
        ep1.angle = sp.angle+divergence;
        ep2.angle = sp.angle-divergence;

new_start_points.push(ep1);
new_start_points.push(ep2);
    }

    //Lets add some more color
    if(length < 10) ctx.strokeStyle = "green";

```

```
        else ctx.strokeStyle = "brown";
ctx.stroke();
start_points = new_start_points;
    //recursive call - only if length is more than 2.
    //Else it will fall in an long loop
    if(length > 2) setTimeout(branches, 50);

}

function get_endpoint(x, y, a, length)
{
    //This function will calculate the end points based on simple vectors
    //http://physics.about.com/od/mathematics/a/VectorMath.htm
    //You can read about basic vectors from this link
    var epx = x + length * Math.cos(a*Math.PI/180);
    var epy = y + length * Math.sin(a*Math.PI/180);
    return {x: epx, y: epy};
}

}

</script>
</body>
</html>
```

**OpenGL Code:**

```
#include <stdio.h>
#include <stdlib.h>
#include <float.h>
#include <math.h>
#include <time.h>
#include <iostream>
using namespace std;
#include <GL/glut.h>

GLuintmakeaTree;
float x,y,z;

void makeCylinder(float height, float base)
{
    GLUquadric *obj = gluNewQuadric();
    //gluQuadricDrawStyle(obj, GLU_LINE);
    glColor3f(0, 1,0);
    glPushMatrix();
    glRotatef(-90, 1.0,0.0,0.0);
    gluCylinder(obj, base,base-(0.2*base), height, 20,20);
    glPopMatrix();
    glutSwapBuffers();
}

void makeTree(float height, float base)
{
    float angle;
    makeCylinder(height, base);
    glTranslatef(0.0, height, 0.0);
    height -= height*.2;
    base-= base*0.3;
```

```
for(int a= 0; a<3; a++)
{
    //angle = 45;
    angle = rand()%60+20;
    if(angle >59)

        angle = -(rand()%60+20);
    if (height>.7)
    {
glPushMatrix();
glRotatef(angle,1,0.0,1);
makeTree(height,base);
glPopMatrix();
    }
}
void init(void)
{
glClearColor(1.0,1.0,1.0,1.0);
glShadeModel(GL_SMOOTH);
glEnable(GL_DEPTH_TEST);
makeaTree=glGenLists(2);
glNewList(makeaTree, GL_COMPILE);
makeTree(5,0.2);
glEndList();
}

void display()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glPushMatrix();
glRotatef(x,1.0,0.0,0.0);
glRotatef(y,0.0,1.0,0.0);
```

```
glRotatef(z,0.0,0.0,1.0);
glCallList(makeaTree);
glPopMatrix();
glutSwapBuffers();
}
void reshape(int w, int h)
{
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(30.0, (GLfloat) w/(GLfloat) h, 0.001, 1000.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0,-8.0,-50.0);
}
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowSize (1200, 800);
glutInitWindowPosition(0,0);
glutCreateWindow("3D Tree Using Recursion");
init();
glutReshapeFunc(reshape);
// glutKeyboardFunc(keyboard);
glutDisplayFunc(display);
glutMainLoop();
}
```