WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

PROCEDURAL FOLIAGE GENERATION

Prof. Ila Savant[1],Advait Kelkar[2], Mayur Dahibhate[3], Shantanu Jagtap[4]

[#]Department of Computer Engineering, Marathwada Mitra Mandal's College of Engineering, Pune
[1]ilasavant@mmcoe.edu.in
[2]advaitkelkar2018.comp@mmcoe.edu.in
[3]mayurdahibhate2018.comp@mmcoe.edu.in
[4]shantanujagtap2018.comp@mmcoe.edu.in

ABSTRACT

Procedural foliage generation refers to the generation of flora features, through the use of algorithms, with minimal input required from the user. In the process of game development, generating foliage is often an important part of the game development process. Traditional generation methods are often too time-consuming, especially with larger foliage variety. On the other hand, procedural methods that generate foliage automatically often do not have much user control over the output. We explore the usage of Lindenmayer systems in the creation of tree assets.

Keywords: *Procedural Generation, Computer Graphics, Lindenmayer Systems, Foliage Generation*

1. INTRODUCTION

Procedural Foliage Generator is an application that generates semi-realistic foliage based on user input. The generator uses the algorithm programmed and adds in some random elements based on the rules specified to generate a digital approximation of real foliage seen in nature. The generator includes features like varying trunk thickness, the presence of leaves at the appropriate locations, as well the ability to edit the ability to rotate the generated object on all axes to observe the product.

The main objective is to reduce the burden on the artists and 3D modelers who have to spend a considerable amount of time handcrafting the foliage assets. The generator can be incorporated into the programming pipelines of game development studios to speed up the asset generation process. In the Unity version of the generator, a 3D modeler can add more elements to make the generated product more realistic, thus saving time as well as creating more believable foliage that would hold up to the scrutiny of the consumers

The goal is to provide game developers with a handy tool that will accelerate the workflow while still maintaining the quality observed in assets modeled by hand

Lots of computer applications are required to generate a variety of assets in abundance. Instead of hard coding such assets, programmers can take motivation from the randomness of nature. With some treatment of mathematics and theory of computation, such randomness can be reasonably recreated with computers. Procedural generation aims to generate a variety of assets given parameters and production rules. In this project, Procedural generation is used for generating foliage. Particularly this is one of the features required frequently. Such features would take a lot of disk space if hardcoded. Procedural generation allows the developers to create assets at run time and provide immersive and refreshing experiences to the users. This family of algorithms can be applied to a lot of other domains as well

To develop a procedural foliage generation system that can take input from the user through the web interface as well as through the Unity application interface. The output will be generated instantaneously and can be tweaked as much as wanted by the user.

2. LITERATURE SURVEY

This section includes a brief overview of existing frameworks used for implementing procedural generation. The following frameworks are open source projects which help developers to create immersive and interactive computer graphics applications.

A. ***P5.js:***

p5.js is a JavaScript library, with a focus of making coding accessible and inclusive for artists, designers, educators, beginners, and anyone else. p5.js is free and open-source. p5.js has a very useful drawing functionality which we are going to use in our 2D model for generation of foliage.

However, you're not limited to your drawing canvas. You can think of your whole browser page as your canvas, including HTML5 objects for text, input, video, webcam, and sound.

B. ***Unity Engine:***

Unity is a real-time 3D development platform that lets artists, designers, and developers to come together and create amazing and exciting experiences. You can work on a system with any Operating System. It supports both 2D and 3D graphics and scripting through C# and also includes an amazing feature of drag-and-drop functionality. Developed by Unity technologies, it is a cross-platform game engine. It was used to design simulations and video games for consoles, computers, and mobile devices in its early days.

3. OBJECTIVES

The main objective is to reduce the burden on the artists and 3D modelers who have to spend a considerable amount of time handcrafting the foliage assets. The generator can be incorporated into the programming pipelines of game development studios to speed up the asset generation process. In the Unity version of the generator, a 3D modeler can add more elements to make the generated product more realistic, thus saving time as well as creating more believable foliage that would hold up to the scrutiny of the consumers. The goal is to provide game developers with a handy tool that will accelerate the workflow while still maintaining the quality observed in assets modeled by hand.

4. PROPOSED SYSTEM

Two versions of the procedural foliage generator are planned: a web version that can run in the browser, and a full-fledged Unity application that makes use of all the features provided by the game engine.

A. Web Browser Version:

To give potential users a glimpse of the real product, we have planned to implement a version that runs in the browser, which eliminates the need for expensive hardware. Potential users can simply go to the URL and see a completely interactive demo that showcases the salient features of the foliage generator. To make this version browser independent, we have chosen to use the JavaScript library p5.js, which is a special purpose library designed and developed for graphics applications. This version offers multiple features, such as the ability to alter the width and height of the generated plant, the ability to alter the thickness of the branches, a functionality that allows the user to change the color of generated branches and leaves, and a few presets to help the user see what we envision the final generated plants to look like.

(Fig. 1: Tree Generated On Web Application)

B. Unity Version:

Users who are interested in the complete version of the product can opt for this offering. To use this version, the user must have a computer that satisfies the minimum requirements of Unity Game Engine, and the latest version of Unity installed on their system. Additionally, the user needs a code editor in case they wish to make some changes to the source code. Everything is set up in the Unity Project File, the only thing the user needs to do is press the play button and the generator will start. Once started, a basic version of the tree will be displayed

(Fig.2: Generator Start-Up Screen)

Using the UI provided, the user can alter the parameters, which will be updated in real-time and the changes will be reflected immediately. The following functionalities are provided:

- *Tree:* This feature enables the user to get started on the generation quickly by providing some presets that have been created beforehand for convenience.
- *Iterations*: This feature is related to the number of recursions the algorithm will make, thereby increasing or decreasing the size of the tree, also affecting the number of branches the tree will have.
- *Angle:* feature alters the angle of the branches with each other, giving rise to more sloping or gently sloping branches as per the user's requirement.
- *Length:* The function alters the length of the branches, allowing the user to create longer or shorter branches as per the requirement. This length is applied to all branches of the tree.
- *Width:* The function alters the width of the branches, allowing the user to create thicker or thinner branches as per the requirement. This width is applied to all branches of the tree.

(Fig.3: Web Browser Starter Screen)

(Fig. 4: Basic Tree Generated)

- *Variance:* This is one of the most important features provided since it controls the degree of variation, or "randomness" in the algorithm. Low values will generate trees that are similar to each other, while high values will introduce high randomness, generating that will differ much from the previous ones.

- *Generate New Tree:* This is where the real generation takes place. On pressing this button, a new tree will be generated every time, taking into account the parameters set by the user. Due to the rules of procedural generation, each tree will be different from the one generated before, giving unique results with each click of the button.

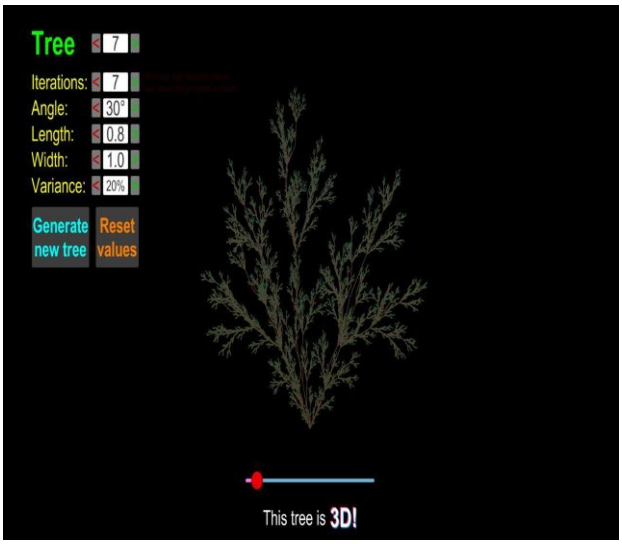
- *Reset Values:* Clicking this button will reset all parameter values to default values supplied by the program.

- *3D Rotation Slider:* Some presets are 3 dimensional, and this slider allows the user to rotate the tree on the Y-axis to see how it will look from all directions.

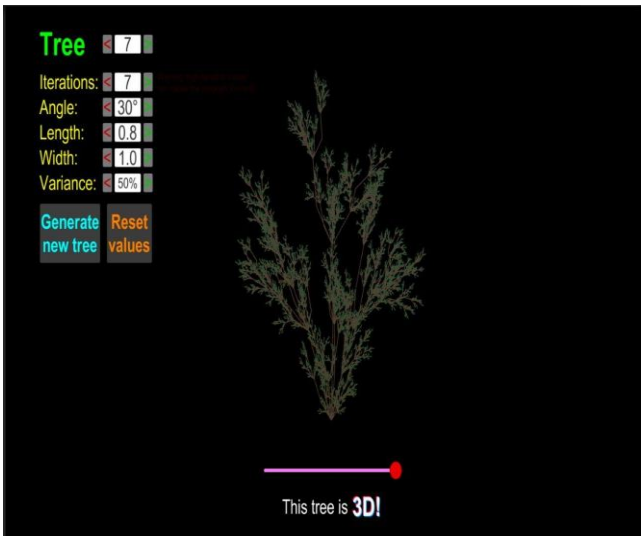
- Overall UI impressions: The functionalities provided by the UI allow the user to use the application to generate a wide variety of trees and tweak their appearance in real-time.

(Fig. 5: Complex Tree Generated)

After the user has played with the parameters provided by the application, below results are some of the results produced by the procedural generation rules.



(Fig. 6: Complex 3D Tree[1])



(Fig. 7: Complex 3D Tree[2])

5. CONCLUSIONS

The project report proposes a Procedural Foliage Generated that could potentially be developed into a large-scale, commercial product. We conducted a comprehensive literature survey to find the most appropriate algorithms, game engines, and libraries and also studied various research papers that have already attempted to make similar systems. During this semester, we identified various functional and non-functional requirements of the system along with the risks and constraints that we might face during the development of the application. We also designed various UML diagrams that would facilitate the implementation of the project.

Acknowledgment

It gives us great pleasure in presenting the project report on PROCEDURAL FOLIAGE GENERATION.

We would like to take this opportunity to thank our internal guide, Prof. Ila Savant, for giving us all the help and guidance we needed. We are also grateful to Prof. Dr. Kalpana Sunil Thakre, Head of Computer Engineering Department, MMCOE for her encouragement and help.

REFERENCES

[1] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. Mesh optimization. In Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, pages 19–26. 1993

[2] Hoppe, H. Progressive meshes. Computer Graphics (SIGGRAPH ’96 Proceedings), pages 99–108, 1996

[3] Xia, J., and Varshney, A. Dynamic view-dependent simplification for polygonal models. In Visualization ’96 Proceedings, IEEE, pp. 327–334. 1996

[4] Hoppe, H., Smooth view-dependent level-of-detail control and its application to terrain rendering, Proc. of IEEE Visualization ’98, pp 35-42; 516, 1998.

[5] “Algorithmic Botany: Publications,” BMV Publications May 2007, University of Calgary http://algorithmicbotany.org/papers/

[6] Lorenz, W. “Fractals and Fractal Architecture”, Fractals and Fractal Architecture, May 2007 http://www.iemar.tuwien.ac.at/modul23/Fractals/subpages/62City