UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

OpenGL Framework

author

Dragoş-Andrei Bobu

session: iunie, 2024

scientific coordinator

Conf. Dr. Varlan Cosmin

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

OpenGL Framework

Dragoş-Andrei Bobu

Sesiunea: iunie, 2024

Coordonator științific

Conf. Dr. Varlan Cosmin

Abstract

Overview: The developed application features an intuitive OpenGL interface that allows users to interact with vectorial math operations, render graphics primitives, and manage game objects. The interface is designed to be user-friendly, catering to both beginners and advanced users.

Contents

Chapter 1

Introduction

1.1 Motivation

I chose this thesis project because of my extended knowledge around the subject and because i thought i could combine all of my past research into this.

Before i even started writing, i already had experience working in the following fields and i will briefly present my computer science background

1.1.1 Background

The following sections highlight pieces of my work that are relevant to this thesis. Each included project is significant due to specific implementations that are directly or indirectly connected to this paper.

Computer Graphics Experience

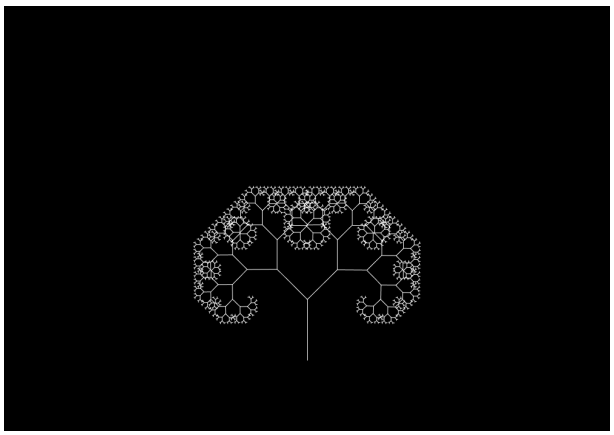

Figure 1.1: Fractal Tree Visualization (usage of p5.js primitive functions in a recursive manner)
showcase

- Supershape Rendering Techniques

- Function Visualizer in OpenGL
- Complex Function Visualizer

Game Development Experience

- 3D Open World Environment Development
- Studies of Vector Movements in Unity

in game development

my game development studies had mostly been around 3D open-world games. Being fascinated by rockstar's grand theft auto series i want to build something similar. i always wondered how cj was able to move in all the directions and calculating how there would be way too many paths to generate all the possible outcomes. so there should be smarter ways to do movement. And there is. Using vectors. My unity projects were mostly about perfectioning the 3D vector movement. Something that i also tried to implement in the opengl framework.

1.1.2 Goal

I challenged myself to dig deeper into Game Development. I wanted to understand what makes all the pretty images move. i already had somewhat of an understanding of how the frames have to be processed independently and displayed in a fastly manner in order to trick the brain. but i wanted to go deeper then that.

i already understood how to do certain simple tasks in unity, but i was so fascinated of the "transform.position = Vector2.One * scalar" command that i wanted to create a similar environment.

the intention of this project is to act as a foundation for a possible game engine that i will continue to deveop in the future.

a game engine is no easy task, there are many running parts and each of them must be SOLID.

1.2 Literature Review

in order to achieve this project i have went through multiple pieces of literature.

the ones i used most extensively are:

1.2.1 Books

OpenGL SuperBible

SuperBible provided a thorough introduction to OpenGL, detailing its functions and capabilities. This resource was instrumental in understanding the core principles of rendering and shading, which are fundamental to the development of any graphics application. By following the examples and exercises in this book, I was able to implement efficient rendering pipelines and gain a deep understanding of shader programming.

Computer Graphics (Donald Hearn)

Donald Hearn's Computer Graphics offered a comprehensive overview of graphics primitives and the algorithms used to draw them. The book's clear explanations of line drawing algorithms, polygon filling techniques, and transformations were particularly beneficial. Implementing these algorithms in my application allowed me to create accurate and efficient rendering routines.

Mathematics for Game Development (Christopher Tremblay)

Christopher Tremblay's book on mathematics for game development provided a solid foundation in vectorial math, which is crucial for tasks such as collision detection and physics simulation. The detailed explanations of vector operations, matrix transformations, and geometric algorithms were directly applied in the development of the vector math library in my application.

C++ (Bjarne Stroustrup)

Bjarne Stroustrup's definitive guide to C++ significantly improved my programming skills, enabling me to write efficient, robust, and maintainable code. The book's coverage of advanced C++ features, such as templates, polymorphism, and the Standard Template Library (STL), was particularly valuable in structuring my application and optimizing:want performance.

1.2.2 Papers

ml agents that can communicate to one another

comp graphics projects from stanford

...

1.3 Theorethical overview

1.3.1 Electronics

1.3.2 Math

1.3.3 Computer Graphics

1.3.4 Game Development

1.3.5 Coding Practices

Chapter 2

Computer Graphics

2.1 Math Engine

2.1.1 vector.cpp

the Vector3 class supports operations such as addition, subtraction, dot product, and cross product, enabling users to perform complex calculations with ease.

2.1.2 random.cpp

Generating true-randomness is one of the biggest computer science challenges.

In my project i needed a way the user to be able to get a ”insert formal specs here for non-deterministic random” vector3 and color variable.

Lukily, i have stumbled upon this random-generator function and was able to implement it. Now, in the framework there is a function that returns a different each call random variable and also the seed is randomized so it differes between individual launches of the application.

2.2 Renderer Engine

2.2.1 primitives

OpenGL and WebGL are quite similar. P5.js is built on top of WebGL and offers the end-user (besides many more) a set of convenient functions for simple tasks like drawing the background, drawing a square, a circle, choosing stroke width and color.

besides these functions i have thought of implementing classes for each of the base geometrical shapes. This implementation will come in handy when integrating with the game engine stuff (see chapter3:Game Development, section GameObjects). For example, each renderer component will use one of the primitive classes (square, circle, box, sphere) to display itself on the screen.

2.2.2 Transformations

As we've already mentioned in the mathematics chapter, transformations are a big deal when dealing with game development. They basically give fluidity. Also, they can be a tough challenge to overcome, especially because it requires the mathematical framework to be able to do complex calculations and as quickly as possible.

This topic is brought up multiple times in this paper and in this chapter we will focus on the specifics on how primitives transform.

types of transformation:

- translation
- rotation
- scaling
- skewing
- warping
- ...

fun fact: did u know that u can achieve a rotation by doing multiple skew transformations in a row?

2.2.3 color.cpp

2.3 Collision Detection

Chapter 3

Game Development

3.1 Game Engine

3.1.1 GameObjects

3.1.2 Components

3.2 Collision Detection

Chapter 4

Machine Learning

4.1 Probabilities

4.2 Scipy compatibility

Concluzii

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nunc mattis enim ut tellus elementum sagittis vitae et. Placerat in egestas erat imperdiet sed euismod. Urna id volutpat lacus laoreet non curabitur gravida. Blandit turpis cursus in hac habitasse platea. Eget nunc lobortis mattis aliquam faucibus. Est pellentesque elit ullamcorper dignissim cras tincidunt lobortis feugiat. Viverra maecenas accumsan lacus vel facilisis volutpat est. Non odio euismod lacinia at quis risus sed vulputate odio. Consequat ac felis donec et odio pellentesque diam volutpat commodo. Etiam sit amet nisl purus in. Tortor condimentum lacinia quis vel eros donec. Phasellus egestas tellus rutrum tellus pellentesque eu tincidunt. Aliquam id diam maecenas ultricies mi eget mauris pharetra. Enim eu turpis egestas pretium.

Bibliografie

- Author1, *Book1*, 2018
- Author2, *Boook2*, 2017