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 the extended knowledge i obtained during my computer science bachelors in the fields of computer graphics and machine learning and because of my personal interest in the field of game development.

My goal was to apply some of those newly aquired concepts in a project that would benefit my long-term game development research.

The way i've decided to apply these concepts is by building a graphics framework compatible with some of the simpler machine learning solutions.

1.2 Goal

1.2.1 What makes a piece of software a game engine?

Wikipedia "A game engine is a software framework primarily designed for the development of video games and generally includes relevant libraries (...). The core functionality typically provided by a game engine may include a rendering engine ("renderer") for 2D or 3D graphics, a physics engine or collision detection (and collision response), sound, scripting, artificial intelligence, (...) "

So, in order to satisfy this wikipedia definition, in order for a piece of software *P* to be considered a game engine, *P*has to satisfy the following:

- *P* must be a software framework

- *P* should include relevant libraries
- *P* should include a suite of engines

is *P* a software framework?

Wikipedia "A software framework is an abstraction in which software, providing generic functionality, can be selectively changed by additional user-written code. (...) It provides a standard way to build and deploy applications and is a universal, reusable software environment (...) to facilitate the development of software applications, products and solutions. "

- Generic functionality that can be selectively adapted based on user's code.
- provides a standard way of building and deploying applications

1.2.2 What does it mean to be machine learning compatible?

1.2.3 What is a game engine?

Maybe it feels irrelevant to talk about game engines when the project title says graphics engine, but in reality the graphics engine is one of the core components of a game engine.

A graphics engine is complex task in itself, besides the renderer itself, it requires a math backend and ways for fast inter-process communication.

These being said, let's formally define a game engine!

Paper Providing frameworks for reusability and separation of concerns is key to software development today.

Licenses:

- Closed source with closed access
- Closed source with open access
- Open Source

Closed source game engines are usually a sign that the project is owned by a big company with a big number of employees and generous funding.

It is unusual for a game company to provide access to its game engine to the end-users. This could be exploited into an competitive disadvantage. Although, once in a while, because of leaks or because of reverse-engineering some parts of the ecosystem are revealed to the public.

In the following i will insert some snapshots of Rockstar's RAGE engine that had older versions reverse engineered and the newer ones leaked.

As you can see, the environments offer statistics relevant to development.

A closer inspection on figure [??] shows this line of code: "" that would indicate that ...

The two options are either building an in-house framework

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

1.2.4 What should a game engine do?

In this subsection i will describe some of the tools that i would like my game engine to be able to offer. I will ilustrate this by showcasing some projects from my personal repertoir that had been built in a variety of environments and highlight the relevant tools that each environment offered me.

Computer Graphics Experience

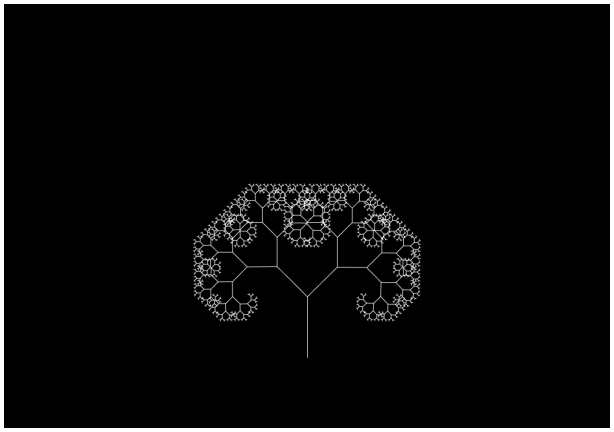

Figure 1.1: Fractal Tree Visualization (usage of p5.js primitive functions in a recursive manner)
showcase

- 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.

1.2.5 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.

1.3 Literature Review

In this section i will explore other's solutions to the problem i was trying to solve.

Unfortunately, there aren't many studies regarding "Machine Learning Compatible Game Engines" so i had to broaden my search.

1.3.1 Simulating Human Interaction

One paper that i found extremily fascinating was TITLE by AUTHOR. They created an environment that allowed ml agents to communicate to one another. One of the most exciting outcomes was that one agent organised a birthday party and proceeded to invite other ml agents to the party. In the following i will briefly go over the implementatation design for one agent:

Another paper that highlights machine-learning agents interacting in human-like behaviour is TITLE by AUTHOR. This team even offers multiple solutions for implementing such agents in popular environments such as Unity or ??.

One popular demo of their plugin? is the game GAMENAME. Game that illustrates a scenario where the player is a detective and has to figure out a case, with the added twist that comunicating with any of the non-playable-characters (NPCs) is made through the microphone and with openai dialogue.

There is also a mod for the popular game Skyrim that allows the player to have fluid dialogue with any in-game character.

1.3.2 Out-performing Human Performance

popular youtuber Code Bullet has a series where he ”solves” games using AI models. He usually uses neural-networks for his solutions. One recent such video is where he programmed a JUMP KING ml.

There are chess bots being developed that use machine learning in an attempt to ”solve” the game of chess. So it is clear to say there is is a lot of incentivise towards acomodating machine learning algorithms into games.

1.3.3 Bibliography Review

in order to achieve this project i have went through multiple pieces of literature. the ones i used most extensively are:

1.3.4 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.

1.3.5 Papers

ml agents that can communicate to one another

comp graphics projects from stanford

...

1.4 from leds to opengl

1.4.1 Why does my computer need a graphics card?

There is this extremily interesting video that showcases how easily one can communicate through the VGA protocol with any display. (BEN EATER)

There are X many pins on a VGA port. From which, X1 are GROUND, X2 are VCC and 4? are used for actually displaying.

The algorithm loop is fairly straight-forward: The monitor expects data through Y pin every Z miliseconds. Every piece of data is considered to be a pixel on the grid, if there is data sent through W pin, the monitor knows to skip to the next line.

The setup revolves around telling the monitor the display resolution, refresh rate and bitmap? of the colors.

Roughly speaking, an led can receive anywhere from 0V to 3.3V (of course this can get more complicated depending on the color of the led). This is electronics-language but computer programs dont work with volts, they work with variables. A translation convention is needed.

This 0V-3.3V range can be devided into however many steps but 255 steps is the most popular and widely accepted one. Fun Fact: old apple/MSDOS/... computers are using 2Bit colors. That meaning that any led can be either completly turned on or off.

Also old computers were using 1Led/Pixel and now we use 3LEDs/Pixel.

1.4.2 Colors and Vectors

So we need a data-structure that can hold 3 255bit values: r, g and b. Throw in an extra one for opacity and make it a vector.

This data-structure should be used repetedly and continuosuly because there are many pixels in a line and there are many lines in a display matrix.

This color class is inherited from the vector class because it makes some calculations easier.

This Vector3 Class is the class that is used regulately

1.4.3 Computer Graphics

1.4.4 Game Development

1.4.5 Coding Practices

1.5 from opengl to end-users

This is the space in the rendering pipeline that my project desires to occupies.

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