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

Academic Overview

Introduction

In the following i will present to you my work of the past year. Work where i attempt to combine multiple computer sciences.

Chapter 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.

Chapter 2

Goals

The goal is to build a game engine that has machine learning solutions already integrated. In the following i will present the benefits of this goal.

2.1 Benefits of building your own game engine

2.1.1 Standard industry practice

it is common for big companies to build their own in-house game engines and then develop their games on it. advantages: provides competitive edge, security, integrity ... disadvantages: cost, team special for that.

it is common for smaller sized companies to develop their games/projects on already existing game engines advantages: already existing reources and docs, community, disadvantages: dificult to come up with unique style.

On the following, i want us to analyse some of the game engines that there are. and draw out relevant particularities of each of them.

for this i have chosen 1 Open Source graphics framework (p5.js), 1 closed software game engine (RAGE) and 1 restricted game engine (UNITY).

RAGE

even though this is a closed project and unaccesible to the public, over the years different screenshots and code snippets had been leaked and/or reverse-engineered and i would like us to take a look at some of the more expressive ones.

Unity

Even though unity's source code is not accesible to the public, the engine is completly free to use for any individual*.

P5.js

This graphics engine is completly free and open-source

2.1.2 Educational purposes

i strongly believe that building a game engine had massively improved my abilities.

2.2 Benefits of integrating machine learning with gaming

ml is the new and fancy cool shiny thing that shows promising numbers and gets ppl hyped and everyone loves it and it must be implemented into everything that exists.

game development is no exception.

2.2.1 Simulating Human Interaction

NPCs are important in games.

NPCs are there to guide the player and are the projection the game designers into the game world.

Because of this, is is really important that npcs have fluid dialogue and dont break the illusion of choice too easily.

Current solutions imply using dialogue trees.

they can still feel rough on the edges. and the illusion can be broken easily when u have to decide from a set of predefined dialogue choices.

the imersiveness of games could greatly improve if ml were to be implemented on top of this already existing dialogue tree solution.

Such solutions have already been experimented with, in the following i will present the findings of 3 other papers that use machine learning to improve npc dialogue and

interaction. Two of the following are solutions for human-to-ai dialogue and one of them simulates ai-to-ai.

Ai interacting to Ai

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:

Ai interacting to humans

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.

2.2.2 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.

Field Study

Field Study - Introduction

The Field Study chapters will present a brief explication of the moving parts that are involved in computer graphics rendering.

First chapter is the one with more focus on embedded-systems concepts Second chapter goes through graphics abstraction layers Third chapter presents an overview of the popular solutions offered to end-users.

Real-World use cases

Student Projects - Stanford Computer Graphics Course

LINK

Academic Research

IMAGES

Field Study - Hardware

This chapter brings focus towards embedded systems by being a brief description of the process of getting the LEDs on our screens to display whatever the computer's video board decides to render.

The connection between screen and motherboard

Proof of concept using Arduino

TINKERCAD IMAGE

Proof of concept using embedded circuits

BEN EATER VIDEO

videoboards

LINUS TECH TIPS OLD VIDEO MANUFATEURS PCBs

Field Study - Software

This chapter accepts the embedded solutions as they are and develops solutions that step forward. The usual philosophy is abstraction layers.

OPENGL - Motivation

There are a few options when choosing a graphics abstraction solution. The most popular in the game development industry are Vulkan and DirectX. DirectX is more appropriate when it comes to ... , while Vulkan profits from ... and performs better at ... .

The only disadvantage to both those solution is that neither of them is as documented as opengl. Also, opengl is more popular in the educational/academic space and felt like the more appropriate choice.

OPENGL - Features

OpenGL's extensive documentation comes with both positives and negatives. Being one of the oldest solution to this problem it had seen multiple refactoring stages throught the years, this is best observed when realising there is a new revision of the opengl superbible released every couple of years. Each presenting the usual "How-to" projects and also acting as an update journal.

In the following i will briefly talk about popular opengl features.

PHIGS Standard Compliant

Offers primitive support and for their attributes by using Rendering Algorithms that use the low-level per pixel drawing functions extensively.

Primitives

- Points and Lines
  - Width
  - Color
    - Flood-Fill Algorithms
  - Segments
- Circle-generating Algorithms
- Text Renderer

Operations

- Matrices
  - Homogeneous Representation
  - Matrix Multiplication
- Basic Transformations
  - Translation
  - Rotation
    - Euler Method
    - Euler Problem
    - Quaternions
  - Scaling

Other Graphics Abstraction Layers

Vulkan

directX

Field Study - Game Engines

Part I

Implementation

Implementation - Architecture

In case you haven't read the previous chapter i advise glimsing over the chapters' titles at least once because it would give better context on where my project lives in the graphics lifecycle.

My Application is a Graphics Abstraction Layer that imitates industry standards when it comes to procedures used for ease of learning. Besides being an easy-to-use beginner-friendly tool because of following industry standard solution in the c++ rendering backend engine. At it's core, this rendering engine is built on top of a vectorial mathematics engine.

The novelty that this project brings to the computer graphics world is the presence of a python Machine Learning backend that acts as an abstraction layer for simplifying the communication with services (openai, ollama) or with powerful machine learning libraries (scipy, skilearn)

The C++ backend

The Python backend

The C++ frontend programming interface

The Python frontend programming interface

Comparison between results

DVD bouncer

Pong, The Game

The Machine Learning Interface

interface for communicating with openai

interface for communicating with scipy

tool for web crawling

Technical Review

Speed

Memory

Chapter 3

Technical Review

In this chapter, we will inspect to what extent needs one piece of software to satisfy in order for it to be considered part of the collection containing 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 definition, a piece of software P can be considered a game engine, if and only if P satisfies the following:

- P is a software framework
  - "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. " source: Wikipedia
    - Generic functionality that can be selectively adapted based on user's code.
    - provides a standard way of building and deploying applications
- P includes a suite of relevant engines

Chapter 4

Conclusions

4.1 Summary of this paper

This has been a journey and after reading this paper you should have a view through the window of progress. There are still many to implement and properly integrate. There will be multiple update journals of this type posted on the following.

4.2 Future improvements

4.2.1 Packaged builds

The scope of this project is to one day make it as an AUR package and also a PyPi library.

Some work has already been made in this direction in the matter that one of the early builds is available by running 'pip install -i https://test.pypi.org/simple/ game-genie' in any terminal. Unfortunately, I had to interrupt pip support for until application grows into a more stable and mature form.

4.2.2

This project could just as well transform into an companion app for a

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 a 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

4.2.3 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

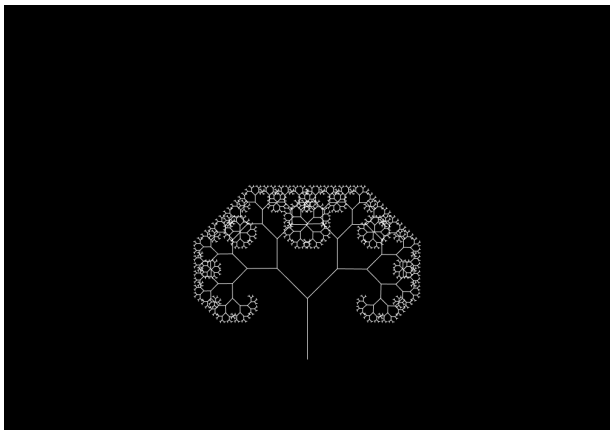

Figure 4.1: Fractal Tree Visualization (usage of p5.js primitive functions in a recursive manner)
showcase

- 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.

4.2.4 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.

4.3 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.

4.3.1 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.

4.3.2 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.

4.3.3 Bibliography Review

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

the ones i used most extensively are:

4.3.4 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.

4.3.5 Papers

ml agents that can communicate to one another

comp graphics projects from stanford

...

4.4 from leds to opengl

4.4.1 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.

4.4.2 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

4.4.3 Computer Graphics

4.4.4 Game Development

4.4.5 Coding Practices

4.5 from opengl to end-users

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

Chapter 5

Computer Graphics

5.1 Math Engine

5.1.1 vector.cpp

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

5.1.2 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.

5.2 Renderer Engine

5.2.1 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.

5.2.2 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?

5.2.3 color.cpp

5.3 Collision Detection

Chapter 6

Game Development

6.1 Game Engine

6.1.1 GameObjects

6.1.2 Components

6.2 Collision Detection

Chapter 7

Machine Learning

7.1 Probabilities

7.2 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