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

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

Part I

Overview

″University teaches you more than what to think. It teaches you how to think″
Stefan Pantiru

Introduction

What you are about to read is over 5 years of game engines research combined with my recently gained machine learning experience aquired as part of my bachelors programme.

Each part acts as an introduction to the one following it. You are now reading about the academic influence that lead to this extended study.

Of course, machine learning wasn't the only knowledge i gained during my bachelors study. I also take pride of myself for actively improving my Software Engineering and Object-Oriented-Programming knowledge, my System Design and Management skills and Computer Graphics and Computational Geometry understanding.

Besides the knowledges, i have gained much experience working with industry-standard tools. There were courses where the main goal was a specific programming language understanding. Courses like Python Programming, OOP, Python Programming, PLP and many many more. Courses like this made me an agile handyman that has his toolbox in order.

Motivation

In the previous i mentioned the toolbox that this university allowed me to organise for myself. In the following i will go over some of the tools i specifically tailered for this implementation.

The biggest motivator factor for persuing this project is a strong personal interest in the field of game development and computer graphics.

My goal became to apply some of those concepts aquired during my study in my personal research.

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

Goals

The goal is to build an Abstraction Layer that opens the inner workings of computer graphics and machine learning.

This Abstraction Layer **must** respect PHIGS standards, **must** be easy-to-learn for both intermediate and beginner end-users, **must** present shorter and more straightforward solution than existing solutions.

The project's theme is combining Machine Learning with the Game Makers' Environments and standards.

In the following i will present the benefits of this theme.

Benefits of Game Engines

Building your own game engine is the standard practice when it comes to big in-house teams. This of course, comes with both advantages and disadvantages. The advantages are spaced around the ideas of security and integrity. While the disadvantages are mostly cost-related.

The often solution for the smaller companies remains open-source software. There are a few providers on the market already, if names like Unity, Unreal, godot might sound familiar, then maybe also do names like P5.js, Processing or maybe even Coding Train, Sebastian Lague, Code Bullet all the way towards electronics field with Ben Eater.

It doesn't matter where in the previous enumeration you lost the references because the following will go more in-depth on the relevant discoveries of each of them.

Benefits of building your own game engine

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

Educational purposes

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

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.

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.

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.

Part II

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 developing 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

Part III

Implementation

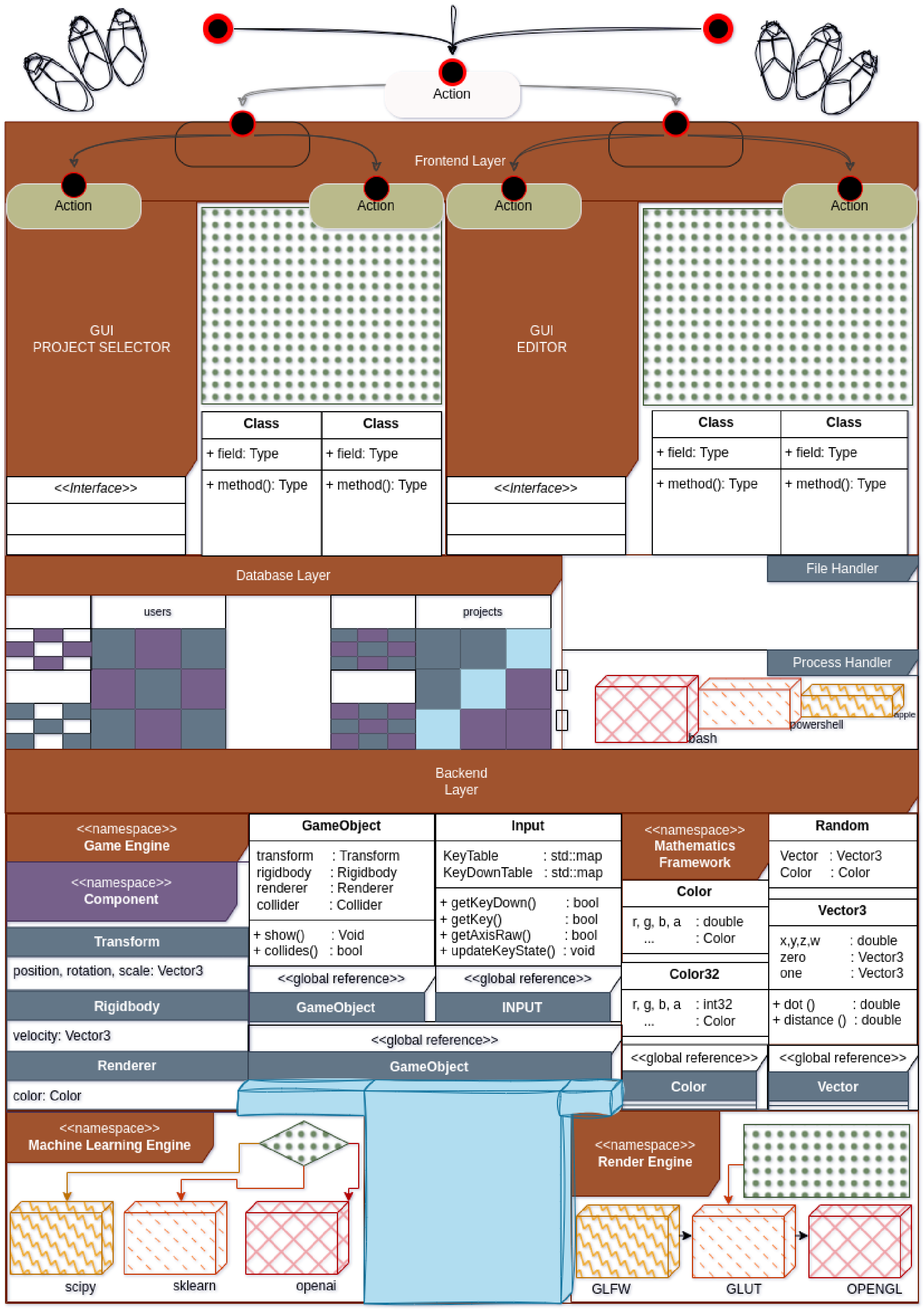
Mathematical Framework

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)



Action
Frontend Layer
Action
Action
Action
Action
GUI
PROJECT SELECTOR
GUI
EDITOR
Class
+ field: Type
+ method(): Type
<<Interface>>
Database Layer
users
projects
File Handler
Process Handler
bash
powershell
apple
Backend
Layer
<<namespace>>
Game Engine
<<namespace>>
Component
Transform
position, rotation, scale: Vector3
Rigidbody
velocity: Vector3
Renderer
color: Color
GameObject
transform : Transform
rigidbody : Rigidbody
renderer : Renderer
collider : Collider
+ show() : Void
+ collides() : bool
Input
KeyTable : std::map
KeyDownTable : std::map
+ getKeyDown() : bool
+ getKey() : bool
+ getAxisRaw() : bool
+ updateKeyState() : void
<<global reference>>
GameObject
INPUT
<<namespace>>
Mathematics
Framework
Color
r, g, b, a : double
... : Color
Color32
r, g, b, a : int32
... : Color
Random
Vector : Vector3
Color : Color
Vector3
x,y,z,w : double
zero : Vector3
one : Vector3
+ dot () : double
+ distance () : double
Vector
<<namespace>>
Machine Learning Engine
scipy
sklearn
openai
<<namespace>>
Render Engine
GLFW
GLUT
OPENGL

The C++ backend

The Python backend

The C++ frontend programming interface

The Python frontend programming interface

interface for communicating with openai

interface for communicating with scipy

tool for web crawling

Implementation - Showcase

Part IV

Results

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.

Speed

Memory

DVD bouncer

Pong, The Game

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

Future Improvements

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.

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.

Chapter 1

Bibliography

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