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.

Game Engines

Building your own game engine is the standard practice when it comes to large in-house development 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.

Smaller companies often turn to open-source software as a solution. There are several providers in the market, such as Unity, Unreal, and Godot. Other notable names include P5.js, Processing, and even content creators like The Coding Train, Sebastian Lague, and Code Bullet, as well as electronics expert Ben Eater.

If you're not familiar with these references, don't worry. The following sections will provide more in-depth insights into each of them.

Overview

As I have already previously mentioned, this is a standardized practice that seems to return profit on investment. Large part of the returnment is based around security, bussines integrity and competitive edge.

Not many of those closed-source software had ever seen the light of end-users' eyes. Some get leaked sometimes. (Figure:)

On the other side of the spectrum live the restricted source software and the open sourse software.

In the following i will like us to analyse some of the following game engine environments and draw out relevant particularities

Machine Learning

Machine Learning is a fun/handy technology to integrate.

is eye candy, everyone implements it into anything. Let's see how they did it.

What other reasons could there be for wanting to implement it?

Dialogue is important in simulators.

In game design, it's a known fact that the npcs are there to guide the player, are the pawns in a chess game. They are an immovable objects that determines the max speed of the unstoppable force.

This static elements could be greatly improved. **Not only gamers would benefit from dialogue improvements!**

NPCs are there to guide the player and are the projection of 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. Dialogue trees are an really good solution but 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

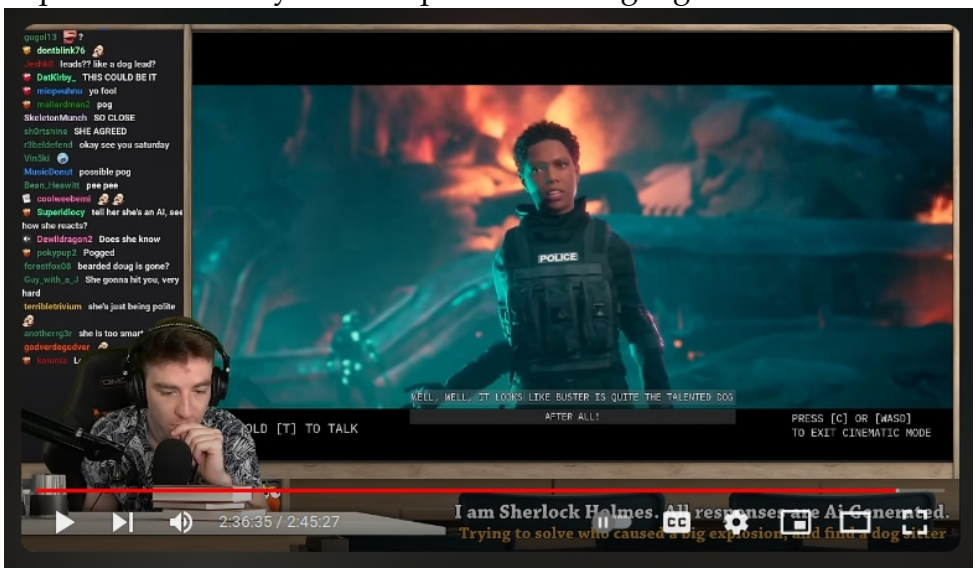
choices.

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 2 other papers that use machine learning to improve npc dialogue and interaction.

Simulating Human Interaction

”Inworld Origins is a technical demo developed to demonstrate the power of generative AI-powered NPCs in video games. It showcases advanced NPC behavior and dialogue driven by artificial intelligence, leading to more immersive and personalized gaming experiences. The demo is set in a neo-noir sci-fi crime scene in the aftermath of an explosion in the city of Metropolis.” source:google

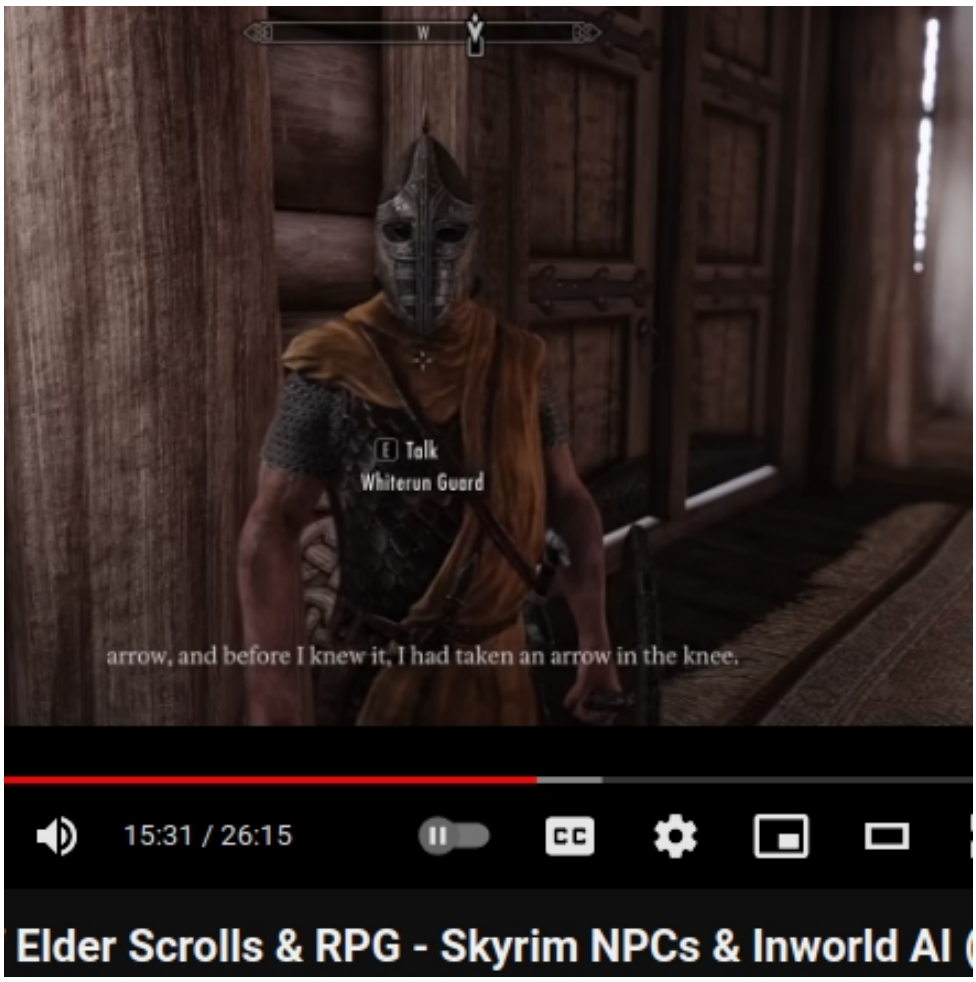


Screenshots of one demo using their openai abstraction layer.

This team even offers multiple solutions for implementing such agents in popular environments such as Unreal.

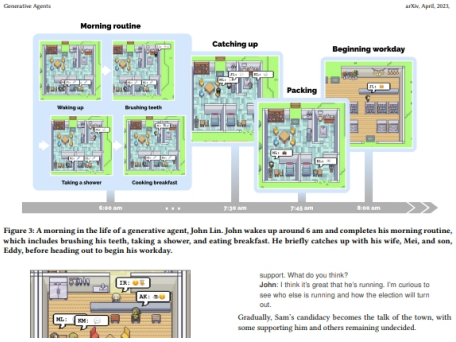
Something that this project and the following one have in common is the presence of microphone input for live dialogue.

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



Ai interacting to Ai

One paper that i found beyond fascinating was Interactive Simulacra of Human Behavior.



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.

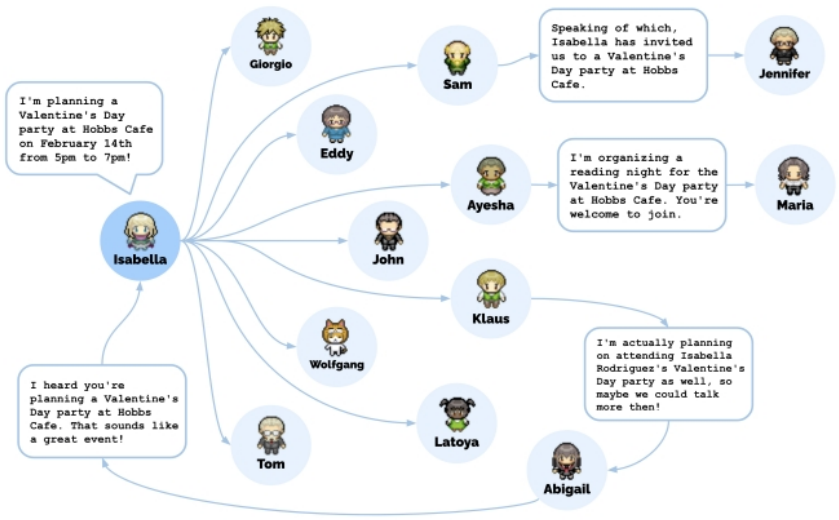
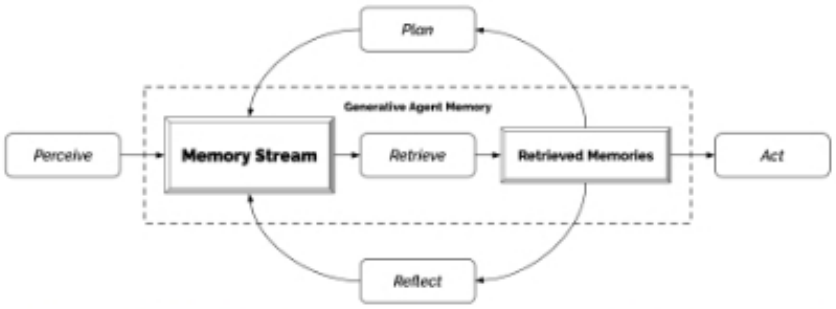


Figure 9: The diffusion path for Isabella Rodriguez's Valentine's Day party. A total of 12 agents heard about the party at Hobbs Cafe by the end of the simulation.

The following goes more in detail over the implementatation design for one agent:



igure 5: Our generative agent architecture. Agents perceive their environment, and all perceptions are saved in a com ensive record of the agent's experiences called the memory stream. Based on their perceptions, the architecture retri :levant memories, then uses those retrieved actions to determine an action. These retrieved memories are also used to f nger-term plans, and to create higher-level reflections, which are both entered into the memory stream for future use.

This design grants access to managing memories by long-time storage of relevant info.

Out-performing Human Performance

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.

groundbreaking chess engine.

Origins of AlphaZero

AlphaZero emerged from the research labs of DeepMind, a subsidiary of Alphabet Inc. (Google's parent company), in 2017. Building upon the success of AlphaGo, which had conquered the ancient game of Go, AlphaZero aimed to push the boundaries of what was possible in the realm of chess.

Reinforcement Learning: A Paradigm Shift

What sets AlphaZero apart from traditional chess engines is its approach to learning. While conventional engines rely on carefully crafted heuristics and exhaustive search algorithms, AlphaZero takes a different path. It utilizes reinforcement learning, a form of machine learning where the program learns by playing against itself repeatedly, improving through trial and error.

From Zero to Hero: AlphaZero's Journey

In its infancy, AlphaZero knew nothing about chess beyond the basic rules. But through millions of self-play games, it began to develop a deep understanding of the game's principles, gradually refining its strategies and tactics. What emerged was nothing short of extraordinary – a chess engine that played with a style and creativity previously unseen in computer chess.

eady to Play Chess?

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

This mathematical framework is very important because it assures that the phigs requirements can be fulfiled and how to.

Homogenous Representation

Intuitively, for storing a 3D point one might think about using a vector of length 3 (maybe call them x, y, z) and have a good day. Well, this datastructure is good *enough* for most cases. There is, of course, one little edge case in one of the things we must be able to do with this DataStructure that would benefit of storing a 3D point as a 4D Vector (x, y, z, w).

Please do not be afraid of the w. Since w is defined by a pretty straight-forward formula.

w=1, if point and w=0, if arrow

Vector Multiplication

Matrix Multiplication

Coordinate Systems

There are two main coordinate systems:

- Cartesian Coordinates
- Polar Coordinates

Formulas for converting from one to another:

- from Polar to Cartesian

  - cos(x)
  - sin(x)

- from Cartesian to Polar
  - x
  - y

In the following, we will take a look over the possible operation that are extensively used:

In most of the cases, the equasions are really easy.

Translation

Scalation

Rotation

Euler-Lock

And it would've been all so easy if it weren't for you!

Euler Lock is a problem that ocurs when we try to use euler coordinates in rotation aplications. This problem can become extremily dangerous when solving robotics solutions where you can't afford to ???

Quaternions

For eliminating the euler-lock problem, quaternions are used. Quaternions are ??

Formulas:

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)

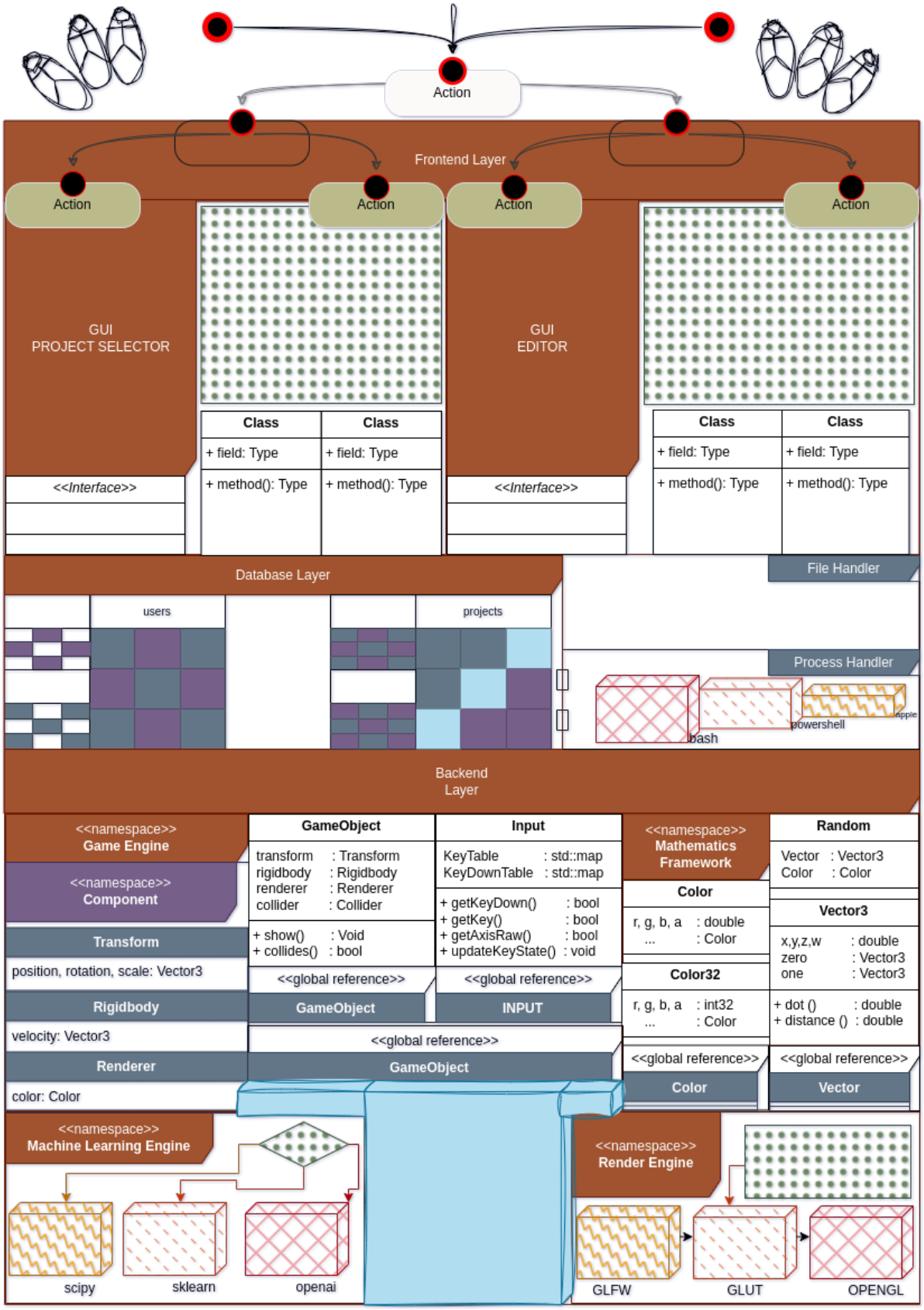


Figure 1: Arch

The backend

The Backend is wrote mostly in C++ and offers the user namespaces to interact with the opengl inner-workings of the engine.

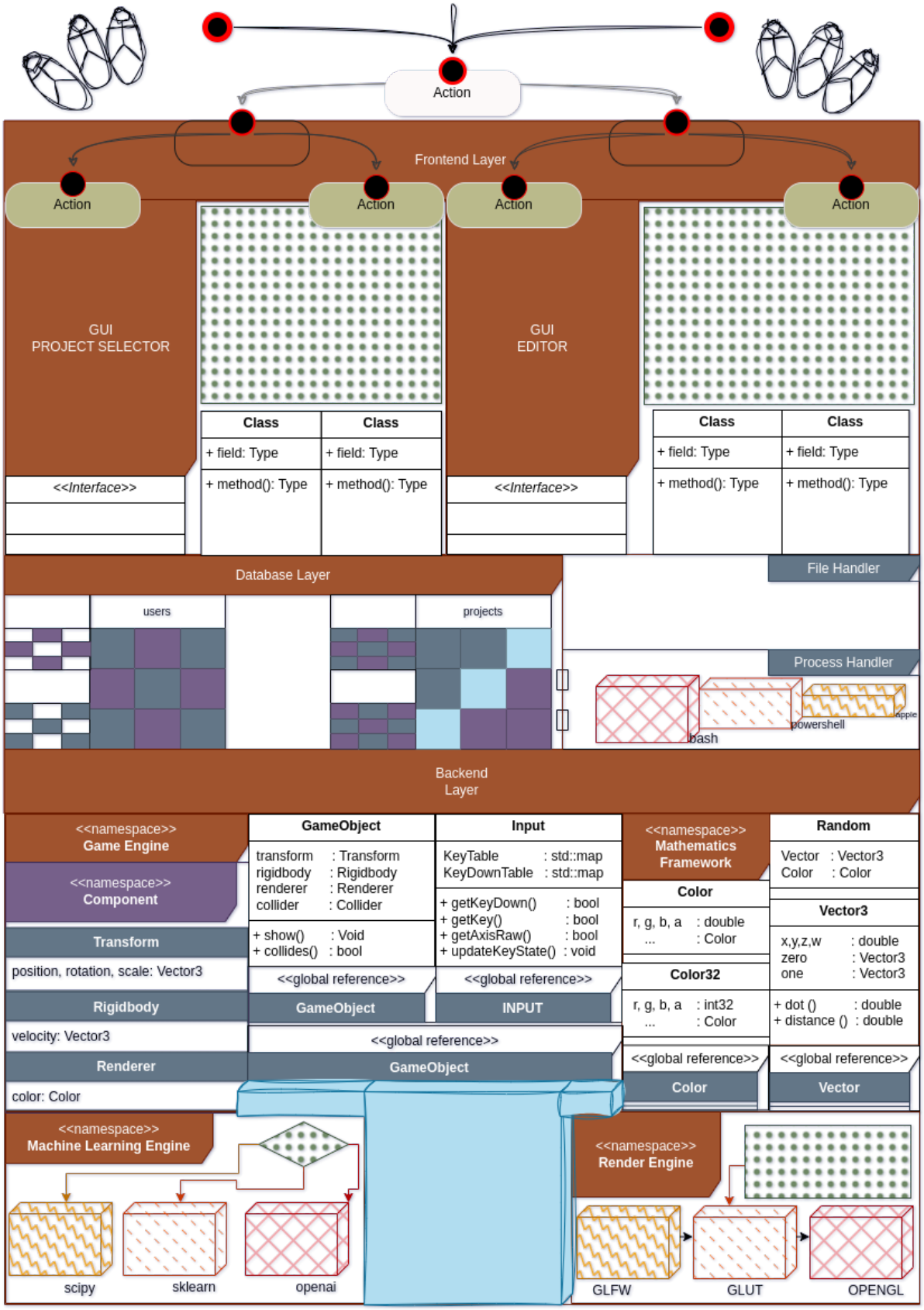


Figure 2: Arch

The Frontend

The frontend is wrote mostly in Python with the help of PyQt, a python wrapper for the industry standard qt.

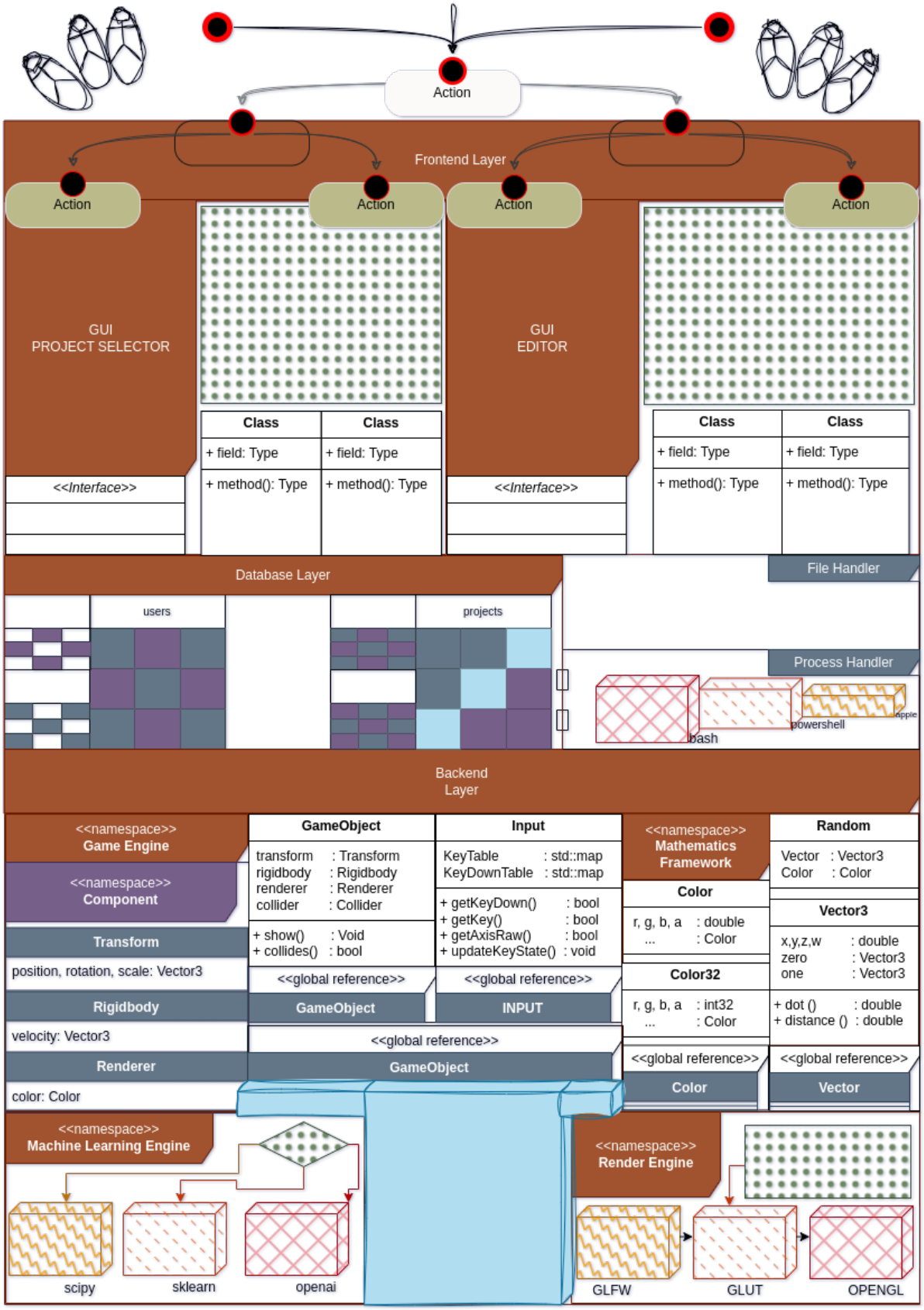


Figure 3: Arch

Additional Tools

Additional tools has been developed already, tools that improves the programmer's workflow. Such example is a web scraper for data collecting. Docs for each major component.

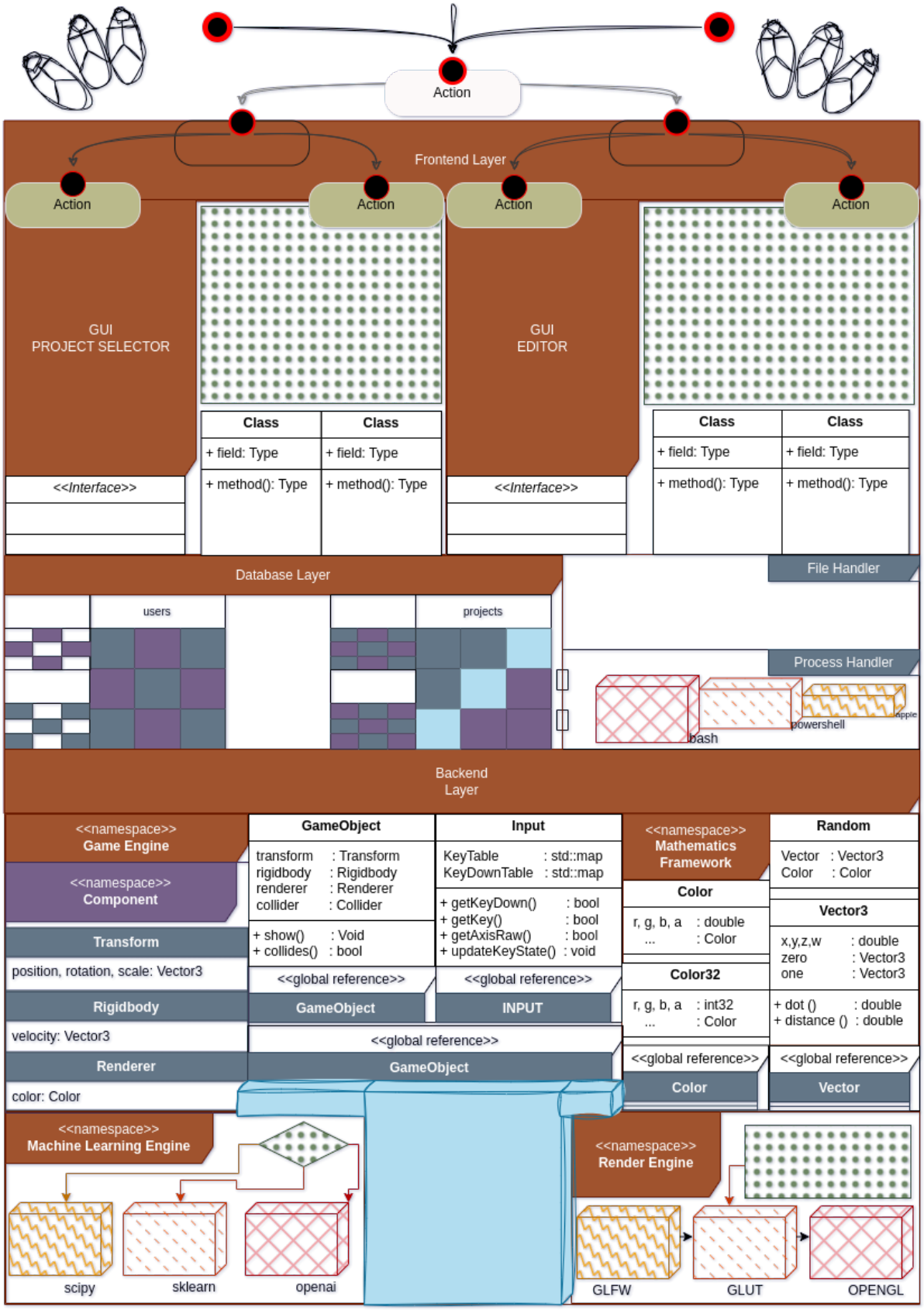


Figure 4: Arch

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