Chanhaeng Lee

Seoul · South Korea · lch32111@gmail.com / github.com/lch32111

Background

I am a game engine programmer interested in computer graphics. I started my programming career to be a graphics programmer because I wanted to make my own world. I worked at Com2us as a game engine programmer, where I mainly focused on rendering architecture and made a variety of progresses on the game engine. I know how to be patient to achieve what I want and like to challenge complex problems.

Education

Master of Computer Science 2022.09–

Concordia University, Montreal, Canada

I am currently a full time graduate student for my master's degree of comptuer science in Concordia University, supervised by Dr. Tiberiu Popa. My research topic is cloth simulation with deep learning.

- 2022 Fall : TA for COMP 352 Data Structures and Algorithms.

Bachelor of Business Administration 2012–2018

Chonnam National University, Gwangju, South Korea
GPA : 4.0 / 4.5

I majored in **Business Administration**, and minored in **Computer Science** from 2016. I studied Computer Graphics at Northumbria University in the UK as an exchange student for one semester in 2017. I served in the military from 2013 to 2015
My passion for Computer Graphics led me to take more lectures beyond my total required credits for graduation, which contributed to my better understandings in Mathematics and Computer Graphics. These are my class lists I took for Computer Science study :

- 2016 2nd : Computer System Architecture (A+) / Data Structures (B+)
- 2017 1st : C Programming and Practice (A+) / Operating System (A+) / Logic Circuits Design (A+) / Software Application Project (A+) / Database Systems (A+)
- 2017 2nd : 3D Graphics Programming (A+) / Artificial Intelligence and Affective Computing (A)
- 2018 1st : Digital Image Processing (A+) / Linear Algeba 1 and Laboratory (A+) / Introduction to Geometry (B+) / Mathematics 1 (B+)

Work Experience

LG AI Research 2022.05–2022.08

Freelancer Software Engineer

I Implemented the application that a user can view chemical files, edit rendering results, and get labeled data for the results in order to generate a number of training datasets for the company's machine learning project. The program is based on one famouse chemical rendering library. The program targets cross-platforms including Windows and Ubuntu. The project is built with the cmake build system and C++.

- Implemented the command line interface program for Windows and Ubuntu that can generate loads of training datasets in parallel.
- Implemented the GUI program for Windows that a user can edit rendering results and get labeled data with custom rendering options. The program also support multi-threaded processing of chemical files and multiple window rendering. DirectX 11 is used for rendering.
- Analyzed and customized one of the famous chemical rendering libraries to generate training datasets

COM2US 2019.03–2021.06

Game Engine Programmer

I implemented a renderer in a game engine and other systems for the engine. This project targeted cross-platforms including Windows, Mac, Android, and iOS. The project was built with the cmake build system and C++ 11. I mainly handled not only various aspects related to rendering, but also useful facilities for the engine to make it work robustly.

- Implemented **an integrated Graphics API** using OpenGL ES, Vulkan, Metal.
- Implemented **a material/shader system** which enables the engine to do the shader permutation and a user to do **shader programming** on the engine. The user shader programming feature is inspired by Godot engine. This system mainly uses glslang for shader code conversion.
- Implemented **a forward shading renderer**, which renders game scenes and editor viewports. The renderer also supports **multiple window rendering** for the editor. The renderer architecture refers to the mesh drawing pipeline from Unreal Engine 4 (UE4)
- Co-planned **an asset system** of the game engine, and implemented the core part of the asset system.
- Co-planned **a serialization system** of game engine files.
- Implemented **an automation test framework** inspired by UE4 by using Socket API to communicate test requests/results between processes on a local computer.

PROJECTS

PERSONAL GAME ENGINE (GITHUB, YOUTUBE) 2018–2021

This is a toy game engine project, where I studied computer graphics and physics. I implemented **a range of graphics features** such as deferred shading, lighting, shadow cast, post-processing (High Dynamic Range, Tone Mapping, Bloom, Screen Space Ambient Occlusion), terrain rendering with Perlin noise, and so on. As for the physics part, I implemented **a simple physics engine** which detects collisions and resolves contacts. I studied **Rigidbody Dynamics** and **Sequential Impulse Constraint Solver** for the physics engine. After this work, I implemented **a simple multi-threaded ray tracer** with a bounding volume hierarchy.

GJK COLLISION DETECTION (GITHUB) 2019

I studied and implemented a frequent-used algorithm (Gilbert-Johnson-Keerthi distance algorithm, GJK) in collision detection with a Game Developer Conference(GDC) presentation by Erin Catto, Box2D developer.

PS4 MINI GAME (YOUTUBE) 2017

This was a submission for 3D Graphics class when I studied in the UK. I developed a simple game with basic graphics and physics features on a PlayStation4 hardware.

MONSTER AI (GITHUB) 2017

This was a submission for Artificial Intelligence and Affective Computing class when I studied in the UK. I gathered training data sets by playing my 2D defense game and trained AI with the data through a neural network using multi-layer perceptron.

2D DEFENSE GAME (GITHUB, YOUTUBE) 2017

I developed a 2D Tile-based defense game for my first programming project by using SDL2 (Simple DirectMedia Layer) library which handles low level functions of a system.

SCHOLARSHIP

MIRAE ASSET SCHOLARSHIP FOR INTERNATIONAL STUDENTS 2017

I received a scholarship of KRW 7,000,000 (roughly CAD 7,500) for my exchange student program from Park Hyunjoo Foundation of Mirae Asset Company.

Skills

Programming

Language : C++, C, GLSL, HLSL, Python
Graphics API : OpenGL, Vulkan, Metal, DirectX
Build Tools : CMake
IDE : Visual Studio, XCode.

Communication

Korean, English