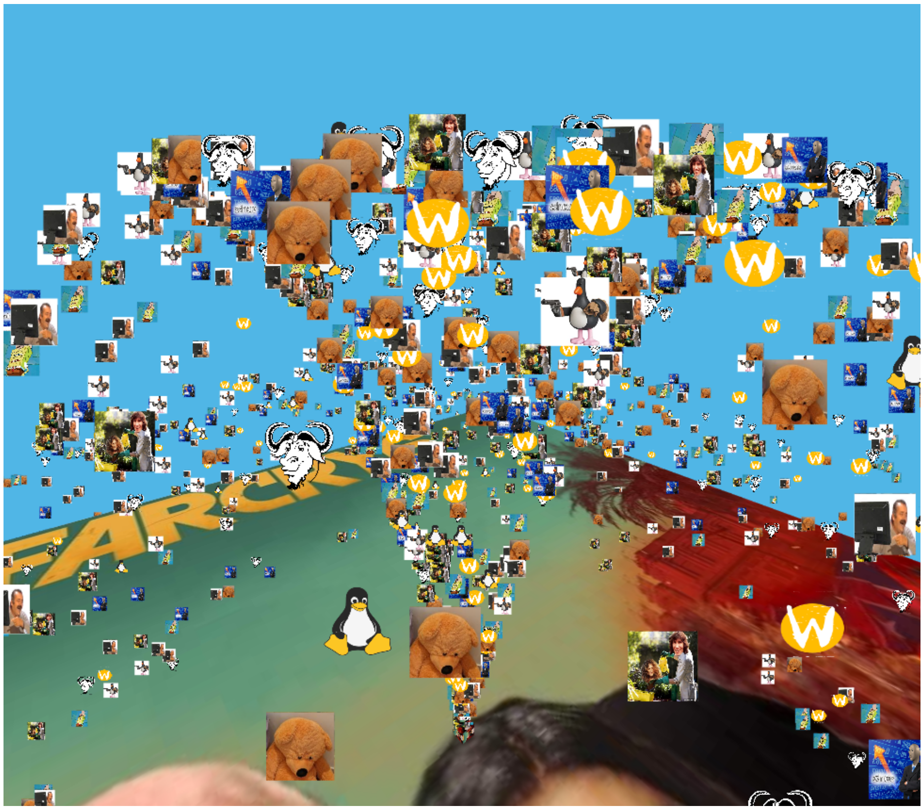
COSC 3P98 Assignment 3

Brett Terpstra - 6920201 - bt19ex@brocku.ca

April 4, 2023



Abstract

Particle systems are the cornerstone of all modern game engines as they allow effects such as smoke which are otherwise impossible with static meshes. Ergo a good particle system should be flexible while maintaining stable performance. This document serves as an informal report on the design and implementaion of two particle systems; a basic but flexible engine and a high performance extendable "modern" GPU powered particle system.

Table Of Contents

Introduction

1.1 Description

Over the course of working on this assignment I began to wonder how far I could push my hardware and chapter 2 ”Performance Mode” will go furher into this. As for the main assignment requirements, they are met simply by running the CMake project. On my hardware the simple particle fountain can reach 30k particles and features sorted transparency.

1.1.1 Extra Features

- Ordered Alpha Blending
- ”Spray” Mode (12)
- Textured Particles/Plane/Cube (16)
- Particles with Different Textures (18)
- Extra Feature - ”Performance Mode” (23)

1.1.2 Missing Features

Random spin mode was left out intentionally for two reasons. One, I specifically designed the particle strucutre to fit in 32 bytes, half the width of a cache line. Two, the potential marks was not worth disturbing the particle data structure and further altering the particle system. There is likely little benefit to ensuring the particles fit nicely inside a cache line as most of the CPU time is spent on OpenGL API calls. See chapter 2 ”Performance Mode” for more information.

1.2 Building

As of writing this report, I have yet to build and test on Windows. The Visual Studio project will build without issues, however, since this assignment was primarily designed and tested on Debain 12 ”Bookworm” (Linux 6.1.0-6-amd64) using AMD/Intel hardware (Mesa 22.3.6), I reccomend using CMake.

1.2.1 Caveats

The assignment makes use of a non-standard OpenGL extention during texture loading. ”GL_TEXTURE_MAX_ANISOTROPY_EXT” should work on all modern Intel/AMD/Nvidia hardware, if it doesn’t work on your hardware consider removing the line from texture.h and high_perf.cpp

1.2.2 Build Commands

```
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE=Release ../
make -j 16
./assign3
```

Figure 1.1: Linux build commands.

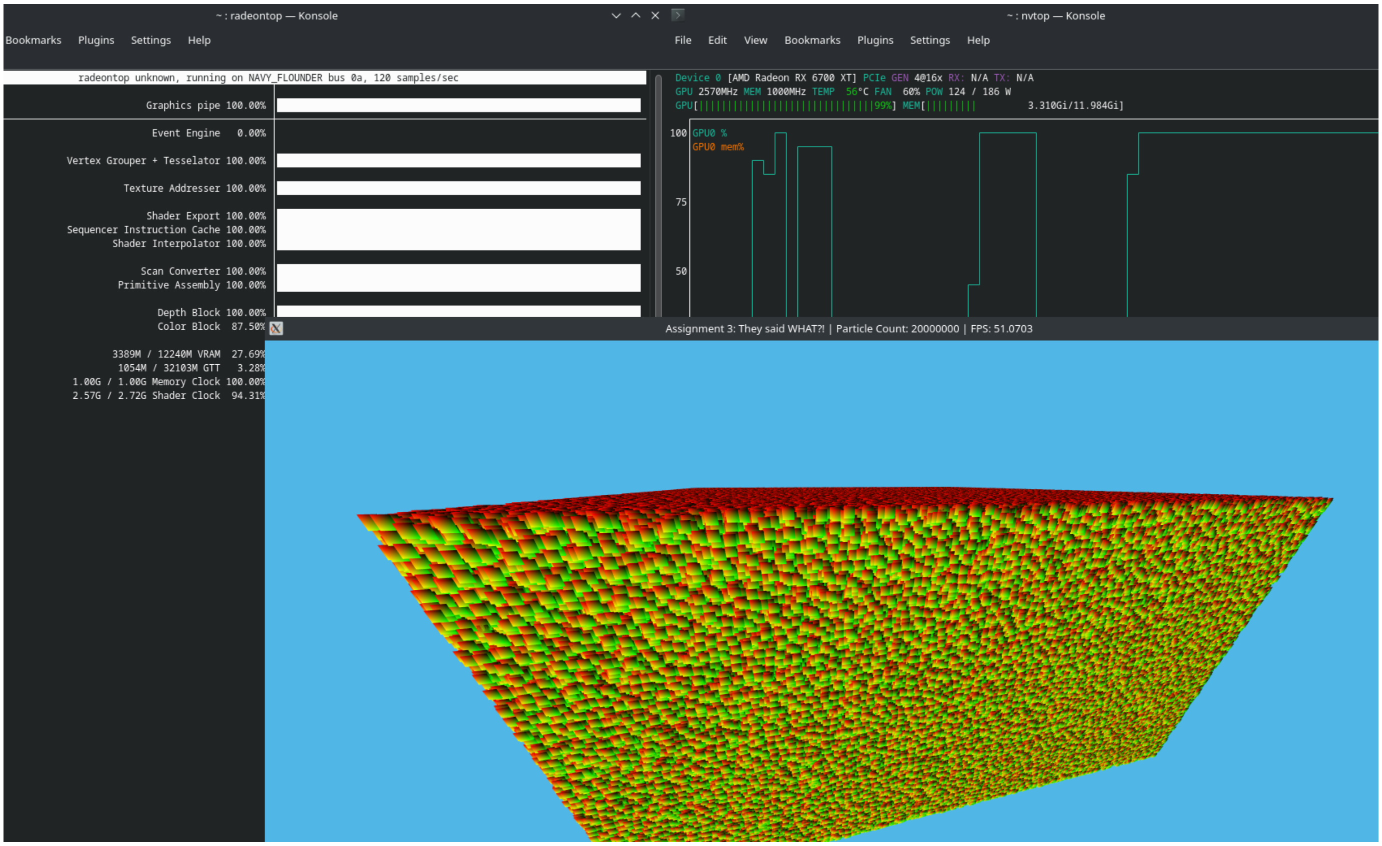
1.3 Usage

Keybindings and usage instructions are printed at program startup.

Performance Mode

2.0.1 Design

The high performance mode is the result of a weekend hack-a-ton where I wanted to see how easy it would be to implement a million particle+ renderer. The rendering engine itself can handle around 20 million particles at about 60fps (Figure 2.1).



```
            radeontop unknown, running on NAVY_FLOUNDER bus 0a, 120 samples/sec

                      Graphics pipe 100.00%

                       Event Engine   0.00%

        Vertex Grouper + Tesselator 100.00%

                  Texture Addresser 100.00%

                      Shader Export 100.00%
        Sequencer Instruction Cache 100.00%
                Shader Interpolator 100.00%

                     Scan Converter 100.00%
                 Primitive Assembly 100.00%

                        Depth Block 100.00%
                        Color Block  87.50%

              3389M / 12240M VRAM  27.69%
               1054M / 32103M GTT   3.28%
          1.00G / 1.00G Memory Clock 100.00%
          2.57G / 2.72G Shader Clock  94.31%
```

```
Device 0 [AMD Radeon RX 6700 XT] PCIe GEN 4@16x RX: N/A TX: N/A
GPU 2570MHz MEM 1000MHz TEMP  56°C FAN  60% POW 124 / 186 W
GPU[||||||||||||||||||||||||||||||99%] MEM[|||||||||        3.310Gi/11.984Gi]
```

Figure 2.1: 20 million particles distributed in a 50x25x50 cube with load monitors