REAL-TIME PARTICLE SIMULATION IN THE BLENDER GAME ENGINE WITH OPENCL

IAN JOHNSON & GORDON ERLEBACHER

INTRODUCTION

The goal of this project is to produce interactive scientific visualizations that can be used in educational games. We use the computational power of OpenCL to enable features in the Blender Game Engine that would otherwise not be possible in real-time. By adding an interactive particle system to the game engine, we set the stage to demonstrate many interesting scientific phenomena (molecular dynamics, fluid dynamics, statistics) with the added benefit of real-time special effects for games in general.

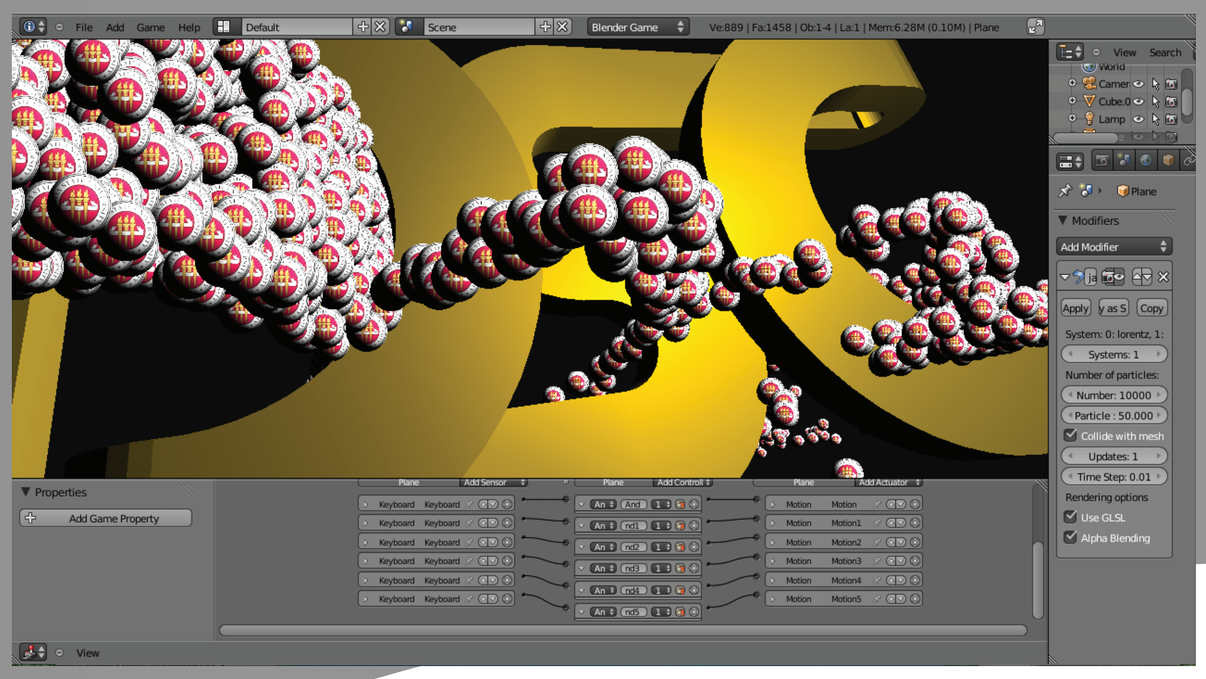


FIGURE 1: 10,000 textured particles colliding with 4,000 triangles at 60fps using the NVIDIA GTX480

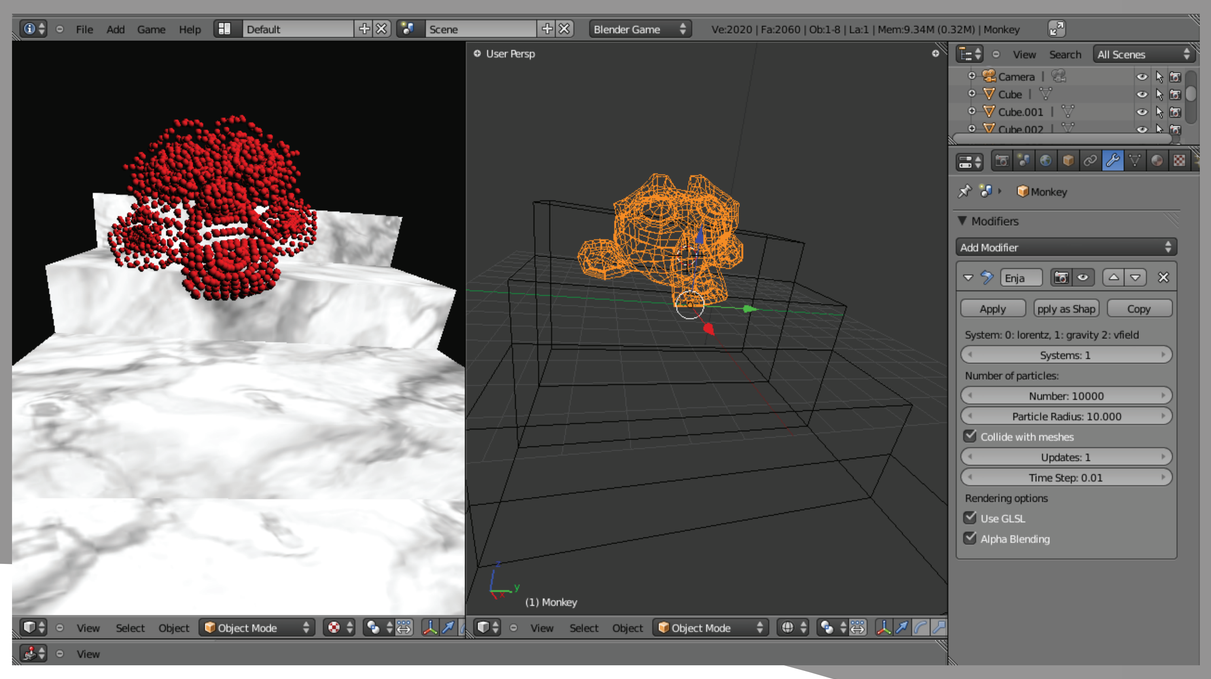


FIGURE 2: 10,000 particles emitted from a Blender object at over 60fps using the NVIDIA GTX480

IMPLEMENTATION

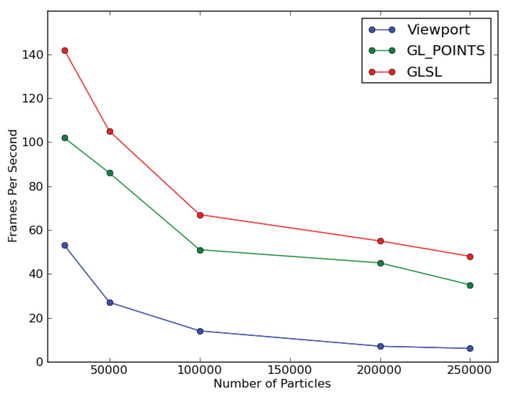
Blender is a very powerful 3D modeling and game development framework used by millions of people across the globe. In the department of Scientific Computing at Florida State University, we teach our Introduction to Game and Simulator design course using the Blender Game Engine due to its large online community, cross-platform support and rich set of features.

OpenCL is an open, royalty-free standard for cross- platform parallel programming of modern processors. Currently it is supported by the two largest GPU chip makers for general purpose GPU computing (NVIDIA and AMD), and is also backed by many of the largest companies in the computing industry.

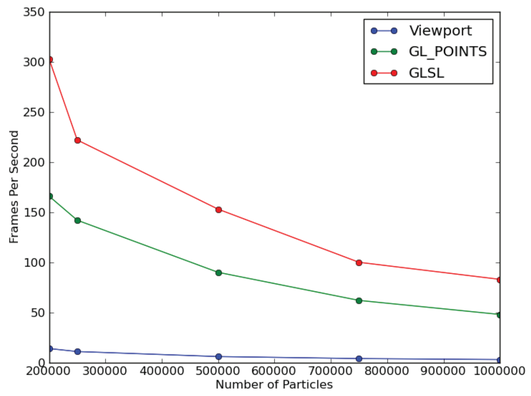
Blender is written C with the Game Engine written in C++, using OpenGL for rendering. We set out to make a C++ Particle System library that could be developed and tested standalone, but could easily be linked against and called from Blender. We utilize the OpenCL C++ bindings provided by Khronos to setup and call our OpenCL kernels.

CAPABILITIES

The following graphs show the speed of Blender's non-interactive built-in particle system labeled as viewport, compared with our OpenCL particles rendered with gl_points and with a simple GLSL shader.



9400M



GTX 480

We test our work on a MacBook Pro laptop with an NVIDIA 9400M running Mac OS X 10.6, We push the limits of our library with an NVIDIA GTX480 running Ubuntu 10.04

FEATURES

At this time, we immerse the particles in some velocity field and update their positions with a first order Runge-Kutta method. We have also implemented collision detection with mesh objects that exist in the Blender scene by harnessing existing data structures. The collisions can also affect other aspects of the system such as the colors of the particles or their lifetime.

The particles are rendered using GLSL shaders using GL_POINT_SPRITES and can either be shown as colored simulated spheres or textured from an image.

We utilize OpenCL context sharing with OpenGL for efficient use of GPU memory. We store bounding boxes of Blender objects in local memory on the GPU for faster collision calculations.

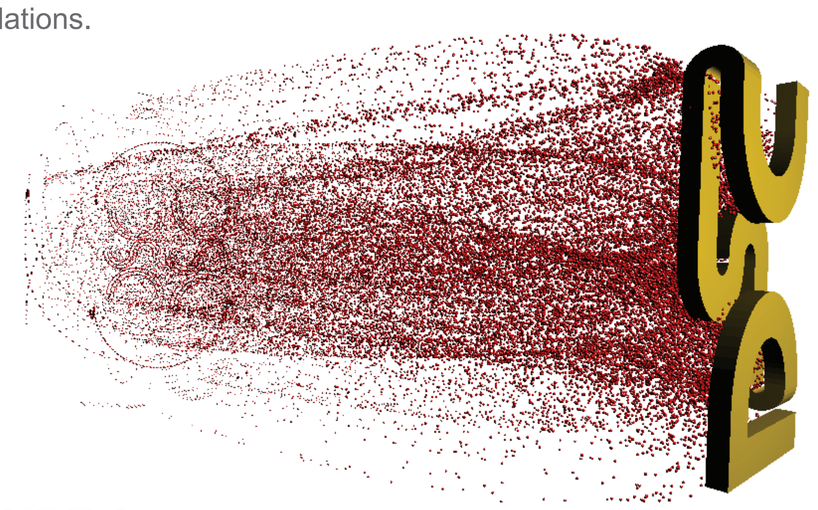


FIGURE 3: 100,000 particles at 70fps using NVIDIA 9400M

FUTURE WORK

Our current focus is on developing more sophisticated systems with particle-particle collisions for more interesting physics. With this capability, we are considering an OpenCL SPH implementation for fluid dynamics. As we develop we add more to our custom Blender interface and it is our hope that we can work with the Blender community for improved software integration and code dissemination.

ACKNOWLEDGEMENTS

Many thanks to Evan Bollig for his guidance and GPU expertise and Mitchell Stokes for sharing his Blender knowledge. Thanks also to Martin Lindelöf and Vu Thai for their design talent in contributing to this poster. A special thanks to the Blender Open Source community who have been very supportive of our efforts.