REAL-TIME FLUID SIMULATION WITH OPENCL

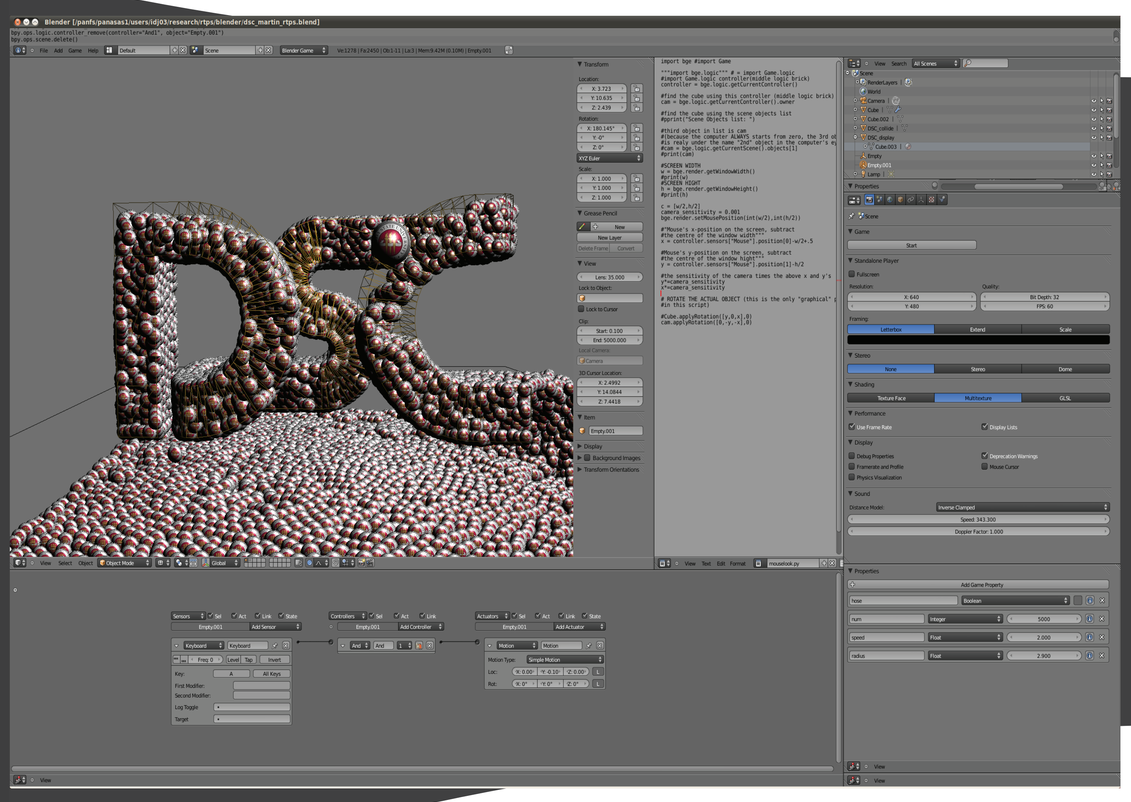
DEPARTMENT OF Scientific COMPUTING

IAN JOHNSON & GORDON ERLEBACHER
ANDREW YOUNG & MYRNA MERCED

FLORIDA STATE UNIVERSITY 1851 VIRES ARTES MORES

INTRODUCTION

The goal of this project is to create interactive tools for scientifically oriented educational games. Enabling students to interact with visualizations of physical phenomenon can aide in the understanding of complex concepts. To this end we have implemented the Smoothed Particle Hydrodnamics method for fluid simulation. We use OpenCL to enable features in the Blender software that would otherwise not be possible in real-time. We leverage existing features for game creation and the powerful consumer hardware used for playing games to bring computational science



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.

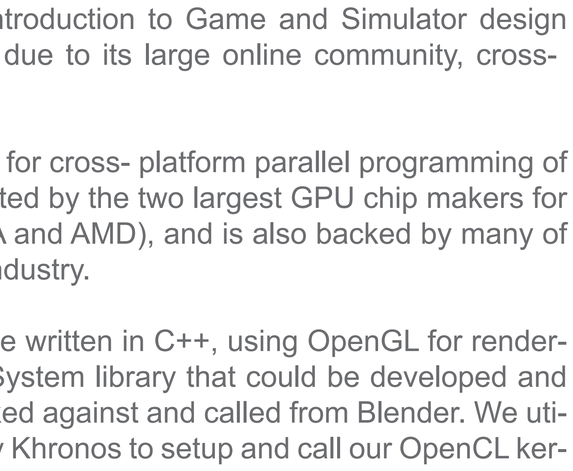
Blender is written C with the Game Engine written in C++, using OpenGL for rendering. We set out to make a C++ Particle System library that could be developed and tested standalone, but could easily be linked against and called from Blender. We utilize the OpenCL C++ bindings provided by Khronos to setup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

USER INTERFACE

We expose many of our simulation parameters to the game creator in the extensible Blender graphical user interface. The user can determine the number of particles used to simulate the fluid, the time step and many physical parameters such as the gas constant or gravity.

The user can use Blender models to construct the domain as well as arbitrary geometry for collision.

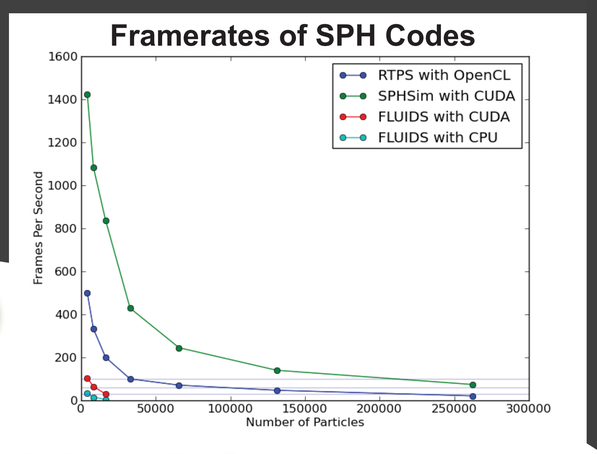
Particle emitters and colliding objects can be controlled through Blender's Logic Brick interface or scripted with Python.



PERFORMANCE

In order to make compelling educational games real-time performance is critical. The criteria for real-time video games is usually measured as 60 frames per second, with 30 frames per second being acceptable but not ideal. This means that our simulation must take no longer than 30 milliseconds to compute an iteration, and preferably under 10 milliseconds. This performance is possible because we take advantage of the

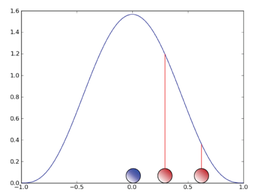
The following graphs show the performance of our project as compared with other open source fluid simulators using the same method. Our project is shown as the blue line and achieves real-time performance with over 100,000 particles and maintains 30fps with over 250,000 particles. We are currently outperformed by a CUDA implementation which we hope to outperform after implementing some known optimization techniques.
The timings were performed on a GTX480 without collisions.



SMOOTHED PARTICLE HYDRODYNAMICS

The Smoothed Particle Hydrodynamics (SPH) method is a meshfree Lagrangian particle method for solving the Navier-Stokes equations. It is highly parallel in nature since the equations applied to each particle can be done simultaneosly.
The essential idea behind SPH is the technique of integral approximation of the fluid at each particle using a smoothing kernel as shown in the figure on the right.
This formulation allows for large deformations but makes dealing with boundary conditions difficult.
Since the position of each particle is updated based on its neighbors it is important to have an efficient method for calculating a particle's neighbors.

$$\rho_i = \sum_j m_j W(y_j - y_i)$$



FUTURE WORK

Our top priority for future work is improving the interaction between our library and Blender, providing more features for educational game designers. We plan to improve the user interface as well as creating new ways of emitting and deleting particles.
We have several optimizations we would like to implement including an improved neighbor searching algorithm to speed up particle interaction.
We would like to improve the accuracy of our simulation as well as the accuracy of collisions. In the future we hope to add rigid body interaction.

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.