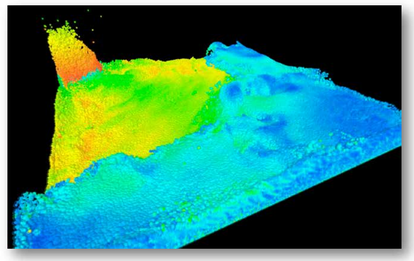
Fluidix
Particle Simulation Software

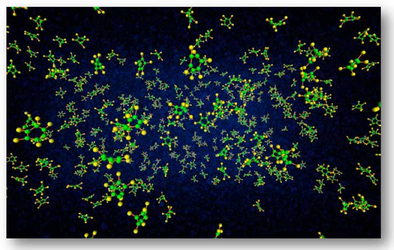
OneZero Software
www.onezero.ca

Adam MacDonald
Brendan Wood
Laura Kaderly
Fredericton, NB, Canada
adam.macdonald@onezero.ca
January 2014

Fluidix: GPU-based general particle simulation library

Combines state of the art performance with complete application independence

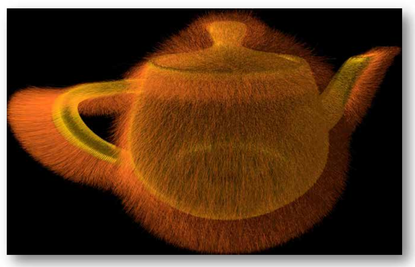




For personal, academic, and commercial research projects

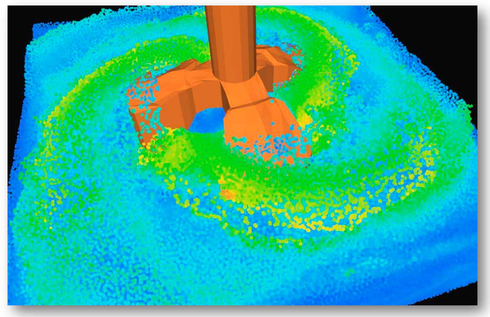
Supports 30+ million particles and dynamic, deformable polygon mesh objects

Intuitive development platform requiring only basic programming skills

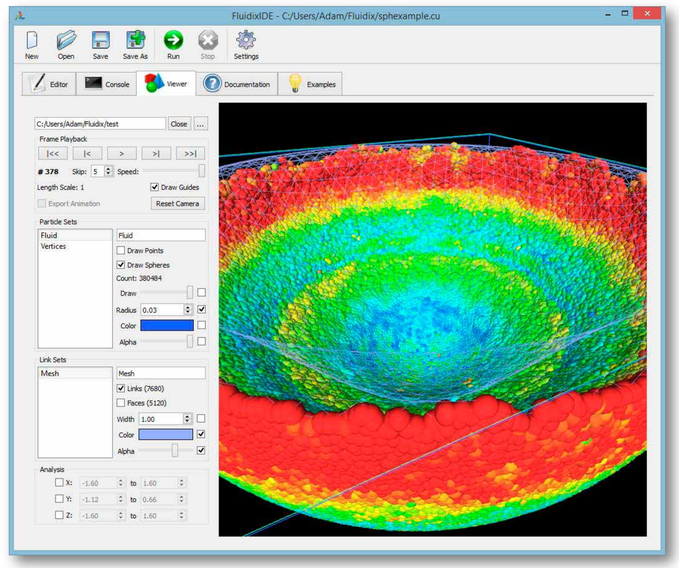


Runs on any system that supports CUDA

- **Smoothed Particle Hydrodynamics** for fluid and gas flow
- **Dissipative Particle Dynamics** for biological systems
- **Molecular Dynamics** for chemical and physical models
- **Visual Effects** for smoke, fluid, and particle simulation
- **Real-Time Interactivity** for more immersive user experiences
- **Polygon Meshes** for deformable, dynamic surfaces
- **Customized Interactions** for complete application creativity
- **High Performance** for iterative development and productivity



A lightweight integrated development environment makes it easy to learn, design, run, execute, and visualize simulations.



An intuitive C++ API using abstracted template classes makes it possible to build a multi-component system in minutes.

```
#include "fluidix.h"

struct Global {
    float range;
} g;

struct Particle {
    xyz r, v, f;
    float color;
};

FUNC_EACH(initialize,
    p.r = make_xyz_uniform() * 100;
)

FUNC_PAIR(repel,
    xyz f = 100 * u * (1 – dr / g.range);
    addVector(p1.f, f);
    addVector(p2.f, -f);
)

FUNC_SURFACE(soft_collision,
    if (dr > 1) dr = 1;
    p.f += 100 * u * dr;
)

FUNC_EACH(integrate,
    p.v += p.f * 0.01f;
    p.r += p.v * 0.01f;
    p.f = make_xyz(0, 0, 0);
    p.color = xyz_len(p.v);
)

int main() {

    Fluidix<> *fx = new Fluidix<>(&g);

    g.range = 2.0f;

    // initialize one million particles
    int fluid = fx->createParticleSet(1000000);
    fx->runEach(initialize(), fluid);

    // add a 3D model
    int2 shape = fx->importModel("shape.stl");

    for (int i = 0; i < 100; i++) {

        // run each interaction
        fx->runPair(repel(), fluid, fluid, g.range);
        fx->runSurface(soft_collision(), shape.y, fluid);
        fx->runEach(integrate(), fluid);

        // output frame for visualization
        fx->outputFrame("example");

    }
    delete fx;
}
```

Particle: a generic data element with local variables, part of one of many resizable sets, representing:

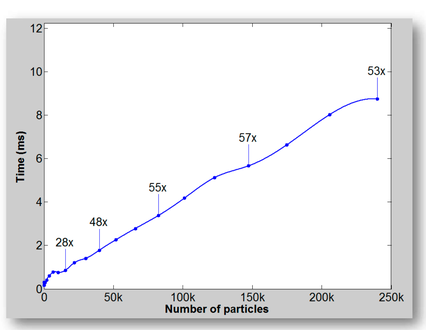
- **Fluid particle** with position, velocity, acceleration, density, etc.
- **Vertex** of a deformable or rigid 3D mesh
- **Grid point** for general computation and analysis

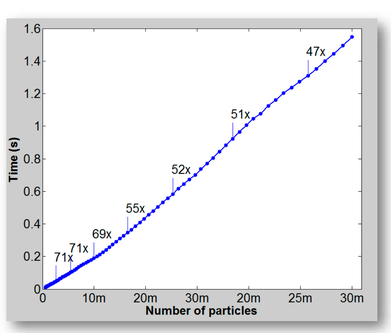
Interaction: a search algorithm which visits particles and executes custom code in parallel, such as:

- **Each particle in a set** for integration, general purpose
- **Pairs of nearby particles** for fluid simulation
- **Links of paired particles** for springs or internal mesh forces
- **Particles in a closed mesh** for soft penetration collisions
- **Particles crossing a surface** for hard boundary collisions

An original binary tree-based algorithm finds **particle-particle** and **particle-triangle** pairs for fluid and mesh collisions.

- Efficient for uniform and sparse particle distributions
- Supports multiple particle sets with no bounding box limits
- Supports meshes with millions of triangles
- Real-time capability of 250k particles at 100 fps
- Scientific capability of 20 million particles per second
- Typical 50x GPU to CPU (single core) speedup





Benchmarks performed using a GTX 780 and Core i7-4770K (single-thread)

Fluidix is free for non-commercial use at **www.onezero.ca**