Amateur Techniques of Particle Based Physics Modeling

Justin Dierking

Abstract

I'm presenting a project that provides a new process in which geometrical primitives (triangular surfaces, linear segments) interact with lagrangian particle fields using projected shadowing particles. I've achieved realistic small scale behavior of interacting dynamic objects and particle fields. Simulations are constructed and visualized using a very simple simplex format and manipulation toolkit based out of the Matlab programming language. Subsequently, this toolkit interfaces to the fisx.exe engine which achieves a simulation's enumerative computation.

Categories and Subjects: Matlab Toolkit – Primitives – Particle to Particle Dynamics – Particle to Face Dynamics – Particle to Cylinder Dynamics – Cylinder to Cylinder Dynamics – Fisx Engine Operation – Advanced Processing Schemes – Simulations

Introduction

Particle based physics modeling has been employed in many applications; some of which have been aimed at realistically simulating the dynamics of compressible and incompressible fluids. Such applications most prominently include special effects for the entertainment industry. More recently, with rise of sufficient processing power with contemporary home computers these days, particle based physics modeling have made their debut in pc and console games. The purpose of this project is to effectively demonstrate a means of implementing reactive geometric primitives that dynamically interact with particle fields as well as other such primitives using a process called projected particle shadowing.

Projected shadow particles allow surfaces and segments to realistically interact with complex particle fields and still conserve momentum and mass. This is achieved by distributing force to particles that are coupled to the vertices of geometric primitives via a shadowing particle. This shadowing particle is projected tangent to the primitive's normal and surface or axis. In effect, coupling particles to a primitive invokes the conservation of mass, momentum, and angular momentum.

To realize this concept of particle interaction, a physics rendering engine, manipulation toolkit, and visualization system has been developed. The rendering engine "fisx.exe" was developed using Microsoft Visual Studio .NET while the manipulation toolkit and visualization program has been developed in Matlab 2006a. These three components interact by passing and simplex data. This simplex encapsulates a protocol for storing, restoring, and manipulating simulations.

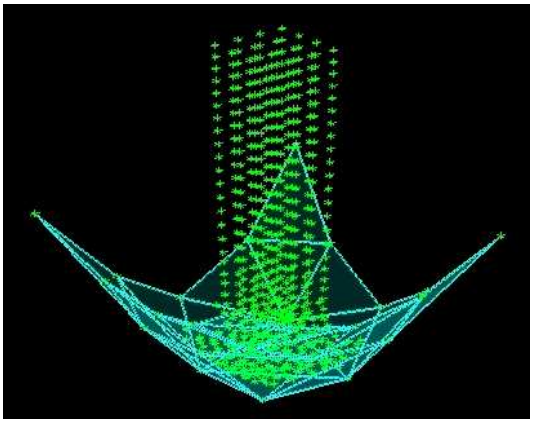


Figure 1: *Flow catching simulation illustrating dynamic interaction with particle flow and blanket composed of interlinked cylinders, particles, and faces.*

Primitives

The fisx engine's functionality allows for a simulation dealing with three types of primitives: spheroid particles ("part"), cylinders ("bumper"), and triangular faces ("container").

A particle is the most basic primitive in the fisx engine. It interacts with other particles and also serves as the dynamic foundation for the cylinder, container, and binder primitives. The properties of mass and momentum all reside within the particles themselves.

A cylinder primitive is a quadric which is governed by two particles. By coupling the quadric to these particles, the quadric exhibits a mass and momentum. The distance between the particles, as well as the difference in the particles' masses and radii determine the quadric's shape, and center of mass. Any force enacted on the cylinder is distributed through the shadowing particle to the physical particles that the quadric is coupled to. The distribution is weighted by the relative location of the shadowing particle with respect to the particles that the quadric is coupled to.

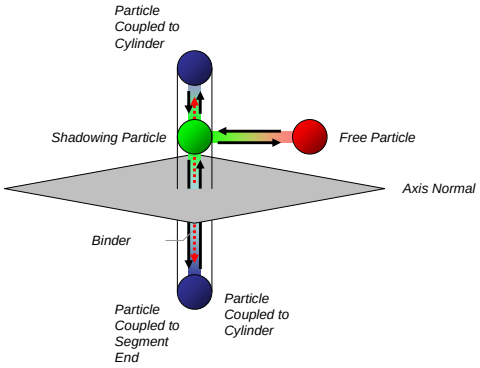


Figure 2: *Cylinder illustration indicating dynamics of components and force distribution theory.*

A face primitive is a triangular face that is coupled to three particles. Like the cylinder, the face's mass, momentum, and width are determined by the particles' masses and radii. The force that is enacted on the face is then weighted and distributed the same way as the cylinder with the obvious exception of a third coupled particle.

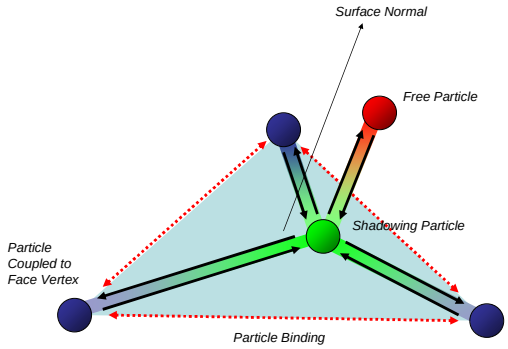


Figure 3: *Face illustration indicating dynamics of components and force distribution.*

With these three primitives, it is possible to construct and incorporate complex structures within a simulation.

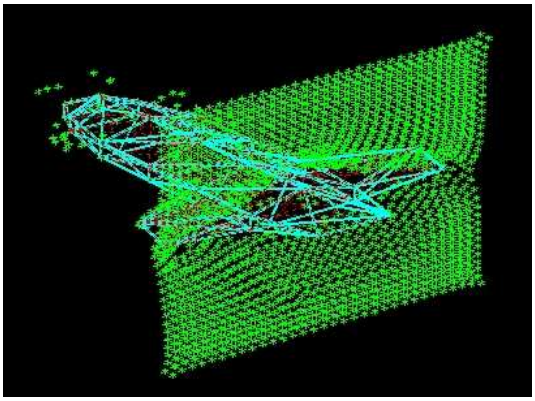


Figure 4: *Pegasus model flying through a field of 2048 particles.*

Dynamics

Fisx utilizes several dynamics in order to process inter-primitive interactions. These dynamics are classified as particle binders, particle to particle, cylinder to cylinder, cylinder to particle, and particle to face. At a fundamental level, all of these dynamics are based on the particle to particle dynamic. This is realized through the use of projected shadowing particles for the dynamics that deal with the face and cylinder primitives.

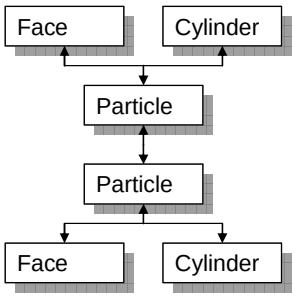


Figure 5: *Dynamics hierarchy of primitive interaction.*

Particle Binders

A Particle binder serves as a dampened spring which couples two particles together. It is commonly used with faces and cylinders to maintain their volumetric attributes. The spring has a variable rest distance. By default, this distance is taken as the distance separating the particles when the particle binder is first created. A binder is stored within the particle object as a particle assignment and rest distance.

$$u = \frac{A_p - B_p}{\|A_p - B_p\|}$$

Unit vector of displacement.

$$s = \|A_p - B_p\| - R$$

Spring function where R is the rest distance.

$$p = \lim_{t \to 0} \frac{\sqrt{(A_p + A_v t) \cdot (B_p + B_v t)} - \sqrt{A_p \cdot B_p}}{t}$$

Derivative of spring compression…T is set as .000001 for the limit's enumerative implementation within Fisx.

$$f = C_p(s - C_v p)u$$

Implemented spring function coupled with a dampening function to minimize resonance. Magnitude of spring and damper are scaled by the constants C_p and C_v.

Particle to Particle Dynamics

A "particle to particle" dynamic couples two particles by localizing a function along the vector of the difference in position amongst the two particles. Fisx employs four functions for this dynamic: repulsion, attraction, viscosity, and thermal couplings.

Rigid Body Repulsion Dynamic:

$$V = \frac{A_p - B_p}{\|A_p - B_p\|}$$

Force transfer unit vector.

$$m_p = \|A_p - B_p\|$$

Relative displacement magnitude.

$$m_v = \|A_v - B_v\|$$

Relative velocity magnitude.

$$m_a = \|A_a - B_a\|$$

Relative acceleration magnitude.

$$u_{A_v} = \frac{A_v}{m_v}$$

Unit vector of particle A's velocity.

$$u_{A_a} = \frac{A_a}{m_a}$$

Unit vector of particle A's acceleration.

$$u_{B_v} = \frac{B_v}{m_v}$$

Unit vector of particle B's velocity.

$$u_{B_a} = \frac{B_a}{m_a}$$

Unit vector of particle B's acceleration.

$$d = A_r + B_r$$

Sum of both particles' radii.

$$l_{A_v} = u_{A_v}\sqrt{(V \cdot A_v)}$$

Collinear component of A's velocity with respect to relative position where the magnitude

$$i_0 = \begin{Bmatrix} m_p \geq d, 0 \\ m_p < d, \frac{d - m_p}{m_p} \end{Bmatrix}$$

Logical interlock function of the inequality of the magnitude of relative position and the sum of the particles' radii coupled to particle intrusion function.

$$i_1 = \begin{Bmatrix} m_p \leq m_p + C_t m_v, 0 \\ m_p > m_p + C_t m_v, 1 \end{Bmatrix}$$

Logical interlock function of the inequality of the magnitude of relative position and the magnitude of relative position after the particles have traveled for Ct amount of time.

$$i_2 = \begin{Bmatrix} m_v \leq m_v + C_t m_a, 0 \\ m_v > m_v + C_t m_a, 1 \end{Bmatrix}$$

Logical interlock function of the inequality of the magnitude of relative velocity and the magnitude of relative acceleration after the particles have traveled for Ct amount of time.

$$F_v = \frac{\|A_m u_{A_v}\sqrt{(V \cdot A_v)} - B_m u_{B_v}\sqrt{(V \cdot B_v)}\|}{2}$$

Impulse required to cancel velocity component.

$$F_a = \frac{\|A_m{}^2 u_{A_a}\sqrt{(V \cdot A_a)} - B_m{}^2 u_{B_a}\sqrt{(V \cdot B_a)}\|}{2}$$

Impulse required to cancel acceleration component.

$$f = C_s V i_0 + V i_1 F_v + V i_2 F_a$$

The sum of a static impulse that is scaled by Cs and the impulse required to equalize the particles' difference in momentum with respect to the particles' relative position and difference in mass.

Attraction Dynamic for Fluids:

$$l = \begin{cases} \|A_p - B_p\| \leq S_R \text{ and } A \neq B, 1 \\ \|A_p - B_p\| > S_R \text{ or } A = B, 0 \end{cases}$$

Logical interlock for particle sampling where Sr is the sampling radius.

$$D_A = \sum_{B=1}^{\#p} l\left(1 - \frac{2\|A_p - B_p\|}{S_R}\right)^2$$

Pseudo density sampling function where the density is the summation of range dependent quadratic function.

$$f = C_s(D_r - D_A)\frac{A_p - B_p}{\|A_p - B_p\|}$$

Force to be applied to other particle where the force consists of a scaled difference between a rest density Dr and an immediate density Da. This force used to stabilize the particle density that is with in a particle's scope.

Thermal Coupling Dynamic:

$$i = \begin{cases} \|A_p - B_p\| < 2(A_r + B_r), 1 \\ \|A_p - B_p\| \geq 2(A_r + B_r), 0 \end{cases}$$

Logical interlock used to discriminate particles within twice the range of the two particles' combined radii.

$$T = i\left(\frac{B_{tcp}}{A_{tcp}}\right)\left(\frac{A_{ter} + B_{ter}}{2}\right)(B_t - A_t)$$

Function to determine a particles undergone thermal load where tcp is a particle's thermal capacity, ter is a particle's thermal exchange rate, and i is the logical interlock.

Viscosity Dynamic:

$$r = \|A_p - B_p\| - (A_r + B_r)$$

The distance separating the particles minus the distance of their combined radii.

$$u_p = \frac{A_p - B_p}{\|A_p - B_p\|}$$

Unit vector of relative position.

$$d = \lim_{C_t \to 0} \frac{\|(A_p + A_v C_t - B_p - B_v C_t\| - \|A_p - B_p\|}{C_t}$$

Derivative of particle distance using a time step of Ct to facilitate an enumerative approximation.

$$f = u_p C_s d\theta\left(\frac{A_r + B_r}{2} - r\right)^2$$

Applied force based on the collinear relative velocity component with respect to the relative position vector.

Within the Fisx engine, these four dynamics are applied to all the neighboring isolated and projected shadowing particles.

Particle to Face Dynamics

The particle to face interaction allows discrete particles to interact with triangular faces. Each triangular face has its vertices coupled to three particles. These particles allow each face to have a mass and momentum. The process involves using a unit vector of the triangle's normal to project a shadowing particle that is tangent to the triangle's plane. If the shadow particle falls within the area of the triangle (barycentric coordinate test), then a particle to particle interaction will be processed.

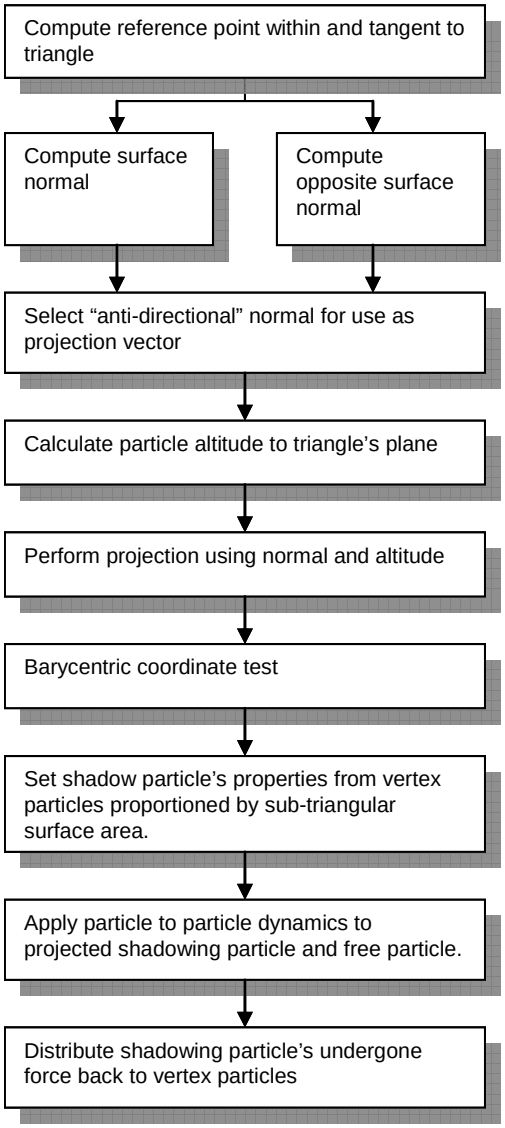


Figure 6: *Particle to face flow of operations.*

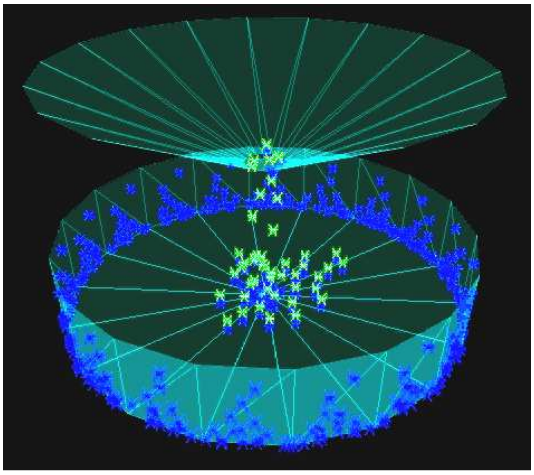


Figure 7: *Illustration showing interaction of basic particle field (green) to the faces' projected shadowing particles (blue).*

To compute a surface normal, a cross product of the triangle's face is calculated using two intersecting vectors that connect the vertices of the triangle. An inverted normal is also calculated given that fact the face will interact with particles on both sides of the plane that the triangle lies on. For each particle the face interacts with, the normal that traverses toward the particle in question is the normal that will be used.

With a surface normal now in hand, an altitude for the particle question is now calculated. This altitude is the distance that separates the particle to the plane the triangle lies on. This is achieved by calculating a difference vector between one of the face's vertices and the particle and then multiplying this vector against the normal. The dot product of the resulting vector is the altitude.

With a surface normal and altitude, it is now possible to project a shadowing particle onto the face's plane. This is achieved by adding the normal's unit vector multiplied against the altitude.

With a shadowing particle, a barycentric coordinate test is used to determine if the projected particle lies with the face's triangle. If it does, then a particle to particle interaction will be applied between the shadowing particle and the particle in question. Whatever force that the shadowing particle incurs is distributed back to the face's vertices.

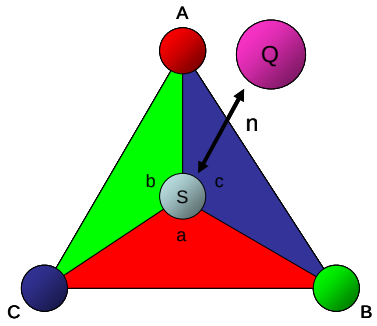


Figure 8: *Comprehensive face to particle interaction.*

Projected Shadowing Particle:

$$a = \frac{\sqrt{(Q_p - A_p) \cdot (v_0 \times v_1)}}{\|v_0 \times v_1\|}$$

Calculate altitude of particle in question Q with particle associated with vertex A.

$$n = \left\{ \frac{v_0 \times v_1}{\|v_0 \times v_1\|} \; or \; \frac{v_1 \times v_0}{\|v_0 \times v_1\|} \right\}$$

Calculate unit vector of the face's normal with respect to which side of the triangles plane that Q lies on.

$$S_p = Q_p + an$$

Create projected shadowing particle S by adding the altitude that is multiplied against the normal's unit vector.

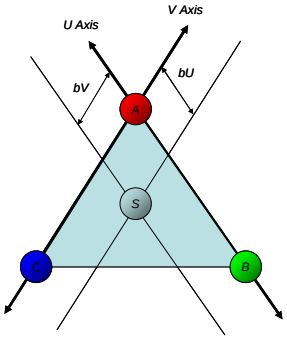


Figure 9: *Illustration of barycentric coordinates.*

Barycentric Coordinate Test:

$$v_0 = C_p - A_p$$
$$v_1 = B_p - A_p$$
$$v_2 = S_p - A_p$$

Calculate vectors required to perform the test from the face's vertices and S.

$$m_{00} = v_0 \cdot v_0$$
$$m_{01} = v_0 \cdot v_1$$
$$m_{02} = v_0 \cdot v_2$$
$$m_{11} = v_1 \cdot v_1$$
$$m_{12} = v_1 \cdot v_2$$

Calculate magnitudes.

$$i_D = m_{00} m_{11} - m_{01} m_{01}$$

Calculate inverse denominator.

$$b_U = \frac{m_{11} m_{02} - m_{01} m_{12}}{i_D}$$
$$b_V = \frac{m_{00} m_{12} - m_{01} m_{02}}{i_D}$$

Calculate barycentric coordinates.

$$W_t = \begin{cases} (b_u > 0) \; and \; (b_v > 0) \; and \; (b_u + b_v < 1), 1 \\ (b_u \leq 0) \; or \; (b_v \leq 0) \; or \; (b_u + b_v \geq 1), 0 \end{cases}$$

Implement barycentric test…If the barycentric coordinates are both greater than or equal to 0 and their sum is less than or equal to 1 then the projected shadowing particle lies on or within the triangle.

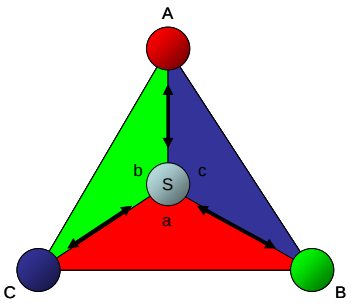


Figure 10: *Illustration displaying processes of weighted force feed back and interpolation of radii and velocity.*

Force Feed Back and Component Interpolation:

$$w_A = \frac{a}{a+b+c}$$

$$w_B = \frac{b}{a+b+c}$$

$$w_C = \frac{c}{a+b+c}$$

Weight functions that are used to interpolate the face's vertex particles' and radii.

$$S_r = w_A A_r + w_B B_r + w_C C_r$$

Sum of weighted radii and velocity components of the face's vertex particles.

$$A_f = A_f + w_A S_f$$
$$B_f = B_f + w_B S_f$$
$$C_f = C_f + w_C S_f$$

Force feedback from the projected shadowing particle is added to the vertex particles' undergone force.

Particle to Cylinder Dynamics

Particle to cylinder interaction functions the same way as particle to face interaction where a shadowing particle is used to interact with another particle via a simple particle to particle interaction. Each end of a cylinder is coupled to a particle. A particle to cylinder interaction differs by projecting a shadowing particle along a one-dimensional axis rather than a two-dimensional plane.

Similar to a face, this axis is defined by a vector between the two particles that make up the ends of a cylinder. Again like a face, these particles give the cylinder a mass and momentum.

To achieve a shadowing particle, a two-dimensional right triangle is constructed using the positions of the particles that make of the cylinder as well as the particle that is interacting with the cylinder itself. With the sides of the triangle known, the interior angles are calculated using trigonometric functions. A right triangle is now placed within this typically irregular triangle using two of its vertices. The triangles remaining vertex is the position for the projected shadowing particle.

At this point the process of interpolating the velocity and radii of the cylinder and applying force feed back to the cylinders associated particles is exactly the same as the process employed in the particle to face dynamic.

$$\vec{A} = \|C_p - B_p\|$$
$$\vec{B} = \|C_p - A_p\|$$
$$\vec{C} = \|A_p - B_p\|$$

Calculate length of all sides of the triangle from the magnitudes of the differencing vectors between the triangle's vertices.

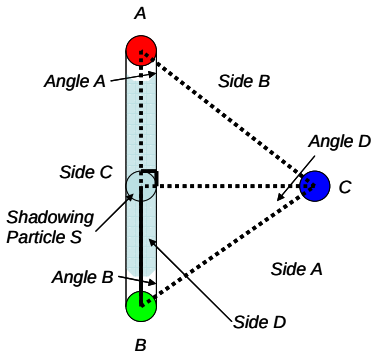


Figure 11: *Illustration showing components utilized to project a shadowing particle onto a cylinder that lies on its principle axis.*

$$\angle A = \cos^{-1}\frac{\vec{B}^2 + \vec{C}^2 - \vec{A}^2}{2\vec{B}^2\vec{C}^2}$$
$$\angle B = \cos^{-1}\frac{\vec{A}^2 + \vec{C}^2 - \vec{B}^2}{2\vec{A}^2\vec{C}^2}$$
$$\angle D = 90 - \angle B$$

Calculate angles A and B from the available sides and angle D with the assumption that a right triangle is being formed within the outside irregular triangle.

$$\vec{D} = \vec{A}\sin(\angle D)$$

Calculate Side D.

$$I = \begin{Bmatrix} (\angle B \le 90°)\ and\ (\angle A \le 90°), 1 \\ (\angle B > 90°)\ or\ (\angle A > 90°), 0 \end{Bmatrix}$$

Logical interlock…If angles A and B are less than or equal to 90 degrees then a shadowing particle is projected and a particle to particle interaction is attempted.

$$S_p = B_p\frac{\vec{C} - \vec{D}}{\vec{C}} + A_p\frac{\vec{D}}{\vec{C}}$$
$$S_r = B_r\frac{\vec{C} - \vec{D}}{\vec{C}} + A_r\frac{\vec{D}}{\vec{C}}$$

Set shadowing particle's position, and radius as a linear interpolation.

$$A_f = A_f + S_f\frac{\vec{D}}{\vec{C}}$$
$$B_f = B_f + S_f\frac{\vec{C} - \vec{D}}{\vec{C}}$$

Particle force feed back.

Cylinder to Cylinder Dynamics

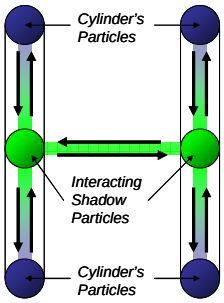


Figure 12: *Basic illustration of interacting cylinders.*

A Cylinder to Cylinder interaction involves the interaction of two projected shadowing particles. Each shadowing particle is a linear interpolation between the particles that are coupled to the cylinders. A search algorithm is used to isolate the interpolation weights for both cylinders that will bring the shadowing particles to the closest distance to each other.

$$C_p(t) = \int .98^t \, \mathrm{SLPFLD}\left(C_p(t)\right) dt + I_p$$

Differential equation used to solve for the optimal interpolation settings for which to place the projected shadowing particles with. Cp is the cursor position, and Ip is the initial cursor position.

This search algorithm is based on a first order differential equation where an "interpolation setting" cursor is progressively adjusted with respect to an integral that consists of a dependent slope field function multiplied against an independent decay function. The slope field is a normalized derivative of the distance that separates the projected shadowing particles with respected to the interpolations of each cylinder. The decay function consists of a constant that is raised to the power of the value of the differential equation's independent variable. As the differential equation is integrated, the magnitude to which the cursor is adjusted decreases logarithmically. The direction that the cursor moves is determined by the slope field at the cursor's present location.

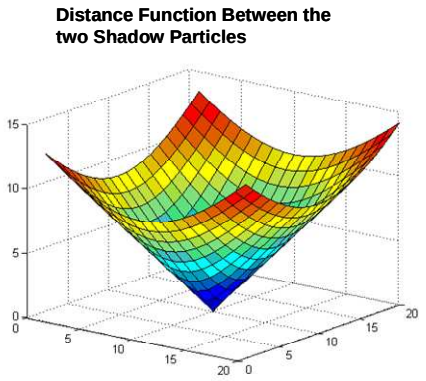


Figure 13: *Distance function…The x and y axis represent the interpolation degree that is used to obtain a shadowing particle for each cylinder while the z axis indicates the distance separating the shadow particles.*

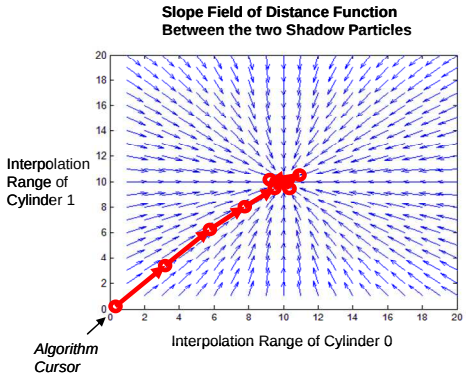


Figure 14: *Derivative of the distance function indicated with quiver arrows. The progressing cursor is outlined in red.*

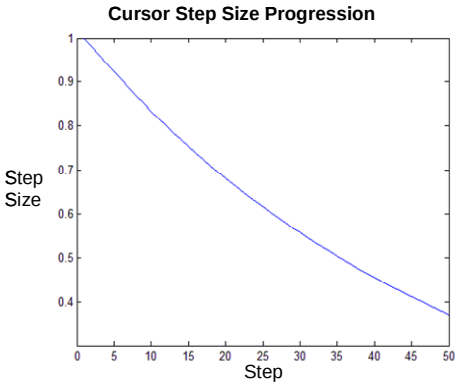


Figure 15: *Cursor adjustment magnitude.*

Matlab Toolkit

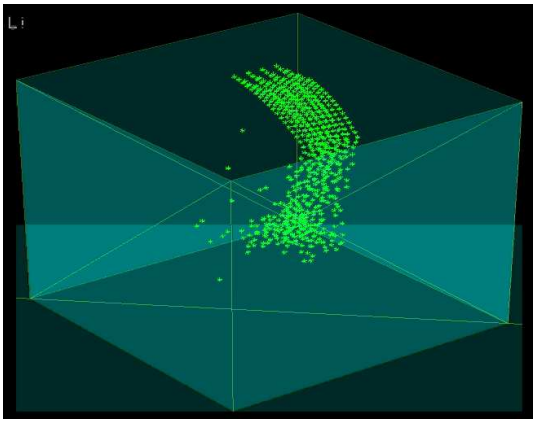


Figure 16: *Water fall simulation using an external Matlab function to implement a 4x4 particle emitter.*

The Matlab toolkit consists of a collection of scripts that directly and indirectly interface with Fisx.exe and its data simplex. These scripts include a load simplex, save simplex, fuse simplex, optimize simplex, particle attribute manipulation, and stock geometry simplex generators. The load simplex and save simplex scripts translate simplex data from the ASCII simplex that Fisx.exe generates and processes to a Matlab structure and vice versa. The optimizing simplex script removes any redundancies that may be present in a simplex's binders and cylinders. The particle attribute scripts modify a particle's position, radius, velocity, mass, radius, type, color, and etc. To initially build a simplex, there are several scripts that generate stock simplexes which are basic particle, container, and cylinder constructs. Finally, the render.m script actually visualizes a simplex.

To process a simulation, the Fisx.exe, load simplex and render scripts are placed into a loop that continues for however number of frames are specified or until the loop is manually terminated. In more complex simulations, high level functions are needed to implement particle emitters and particle dependent vectored impulses. In these cases the simplex is processed a second time from within the Matlab code to implement such functions since they cannot be directly implemented by the Fisx.exe engine itself.

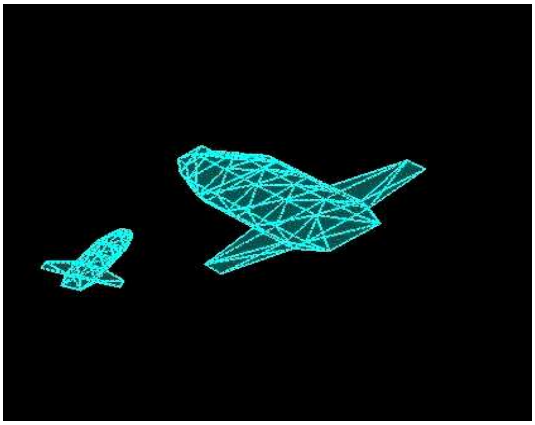


Figure 17: *Pegasus model flyover using external Matlab function to implement particle dependent vectored impulse for propulsion.*

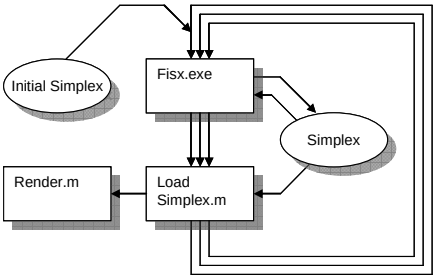


Figure 18: *Flow of operations for a simple simulation that only utilizes the Fisx.exe functionality to process a simulation.*

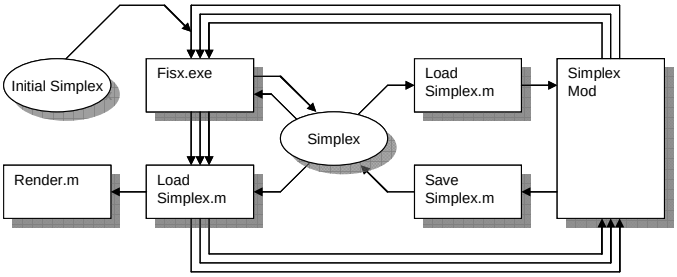


Figure 19: *Flow of operations for a more complex simulation that utilizes the Fisx.exe functionality in conjunction with high level functions implemented from Matlab to support dynamics such as particle emitters and particle dependent vectored impulses.*

Fisx Engine

The Fisx engine is responsible for processing and integrating the simulation that is encoded within the simplex file. Once the simulation has been integrated for a fixed amount of time, the simulation is re-encoded into the simplex file at which time the Matlab code will visualize the simplex and start the next pass of the loop to the Fisx engine.

To integrate a simulation, Fisx executes several processes in succession to realize this task. First, the simulation is loaded from the simplex file that is specified in the executable's input arguments. Once the simulation is loaded, it is placed into processing loop that will execute the task of integrating the simulation. When the loop terminates, the simulation is overwritten to the before mentioned simplex file, and the Fisx executable is terminated.

The processing loop that facilitates the actual integration consists of a four step process. Through every pass of the loop, each particle's undergone force is calculated. Second a variable time step is determined. Third, the simulation is integrated for the time step that was just specified. Fourth, the time step that was selected for this pass is added to a running total that is used to control the loop. Once the running total is equal to 1, the loop is terminated.

To specify a time step, Fisx uses the reciprocals of the differencing magnitudes of the vector extremes for the particles' velocity and undergone force. The smallest reciprocal of the two differencing magnitudes is used as the integration time step for the current pass of the loop. In the terminal cases where the time will cause the running total to be greater than 1, the time step is reduced so that the running total will be exactly 1.

Integrating the simulation in this way has the effect of minimizing the tunneling error that is inherent when complex interactive systems are being discretely integrated through enumerative processes.

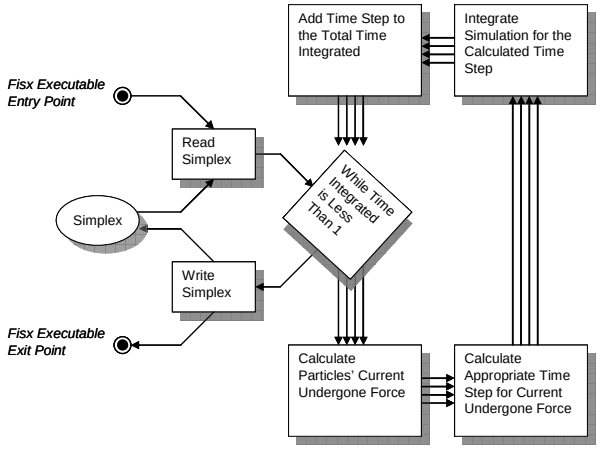


Figure 20: *Flow of operations of the Fisx engine (single scope).*

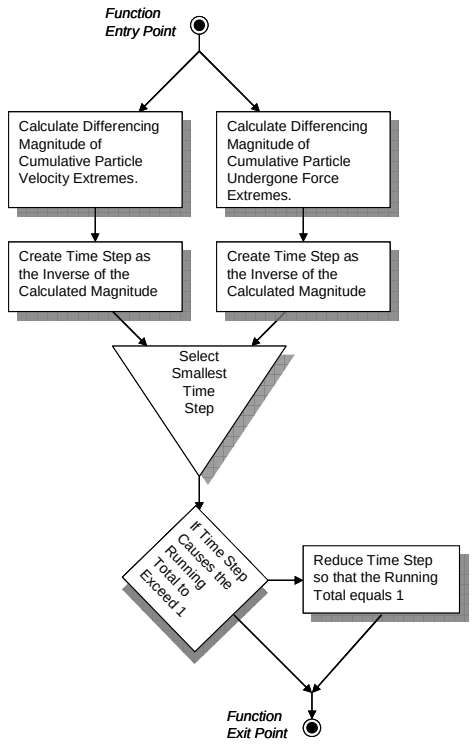


Figure 21: *Flow of operations for selecting a time step.*

Advanced Processing Schemes

There are two schemes that Fisx currently uses to evaluate the undergone force that non-static particles accumulate throughout the course of a simulation. The first scheme to be implemented is a single scope simulation where all particles are dynamically visible to all other particles. The second scheme is a multiple scope approach where only the particle's immediate surrounding particles are dynamically visible.

Each scheme has its own advantages and disadvantages. A single scope processing scheme has the advantage of maintaining consistent particle referencing. In other words, the index number associated with a particular particle does not change over the course of a simulation. A multiple scope processing scheme works by parsing an original simulation into multiple simulations as elements of a mesh. These are small enough to be completely stored within the L2 cache of a processor thus maximizing the processor's internal cache hit rate and therefore drastically improving processing efficiency. The disadvantage to this scheme is that the indexes that particles are identified with change as they traverse from element to element. Another draw back to this scheme is the fact that is utilizes more memory. Since each element within the mesh incorporates a certain amount of spatial redundancy with adjoining elements, the same particle may be injected into multiple adjoining elements in the mesh. It is vitally important to remember that no one particle may be active in multiple sub-simulations of the mesh. If this were to occur, the simulation's simplex would erroneously expand thus causing an inequality of momentum and force within the simulation.

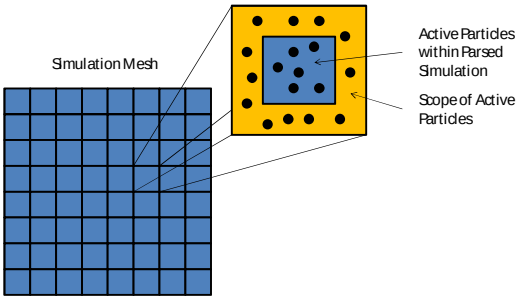


Figure 22: *Illustration of basic mesh and simulation parsing theory for this processing scheme. The blue section of the enlarged box indicates the particles that are to be designated as active while the amber section designates them as static.*

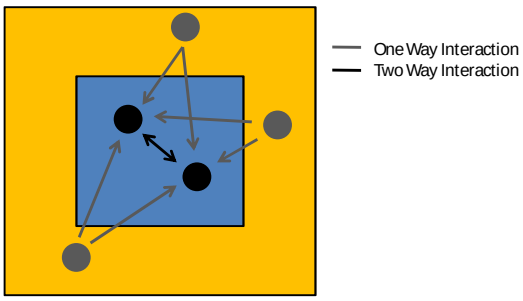
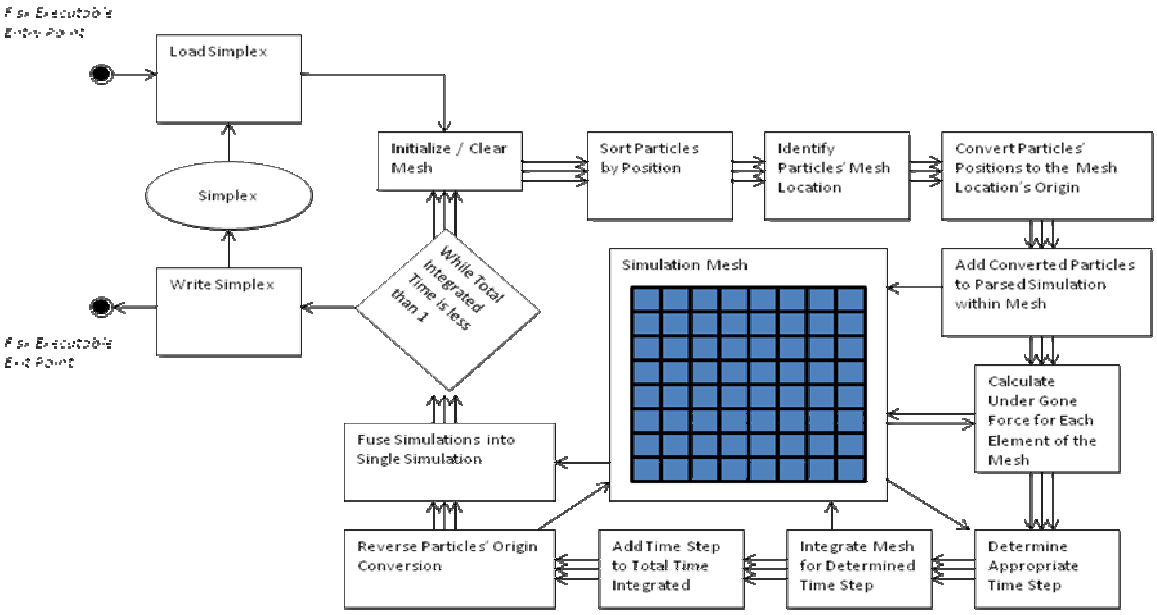


Figure 23: *Illustration showing relationships between active(black) and inactive(grey) particles and how they interact within a sub-simulation of the mesh. Only the active particles can accumulate undergone force.*

Figure 24: *Illustration depicting the flow of operations of the multiple- scope processing scheme.*



Simulations

To test the Fisx engine and verify the development of various components, a variety of simulations were devised and executed. These simulations not only serve to verify and benchmark the operation of the engine's capability, but they also demonstrate the potential uses and applications of this program.

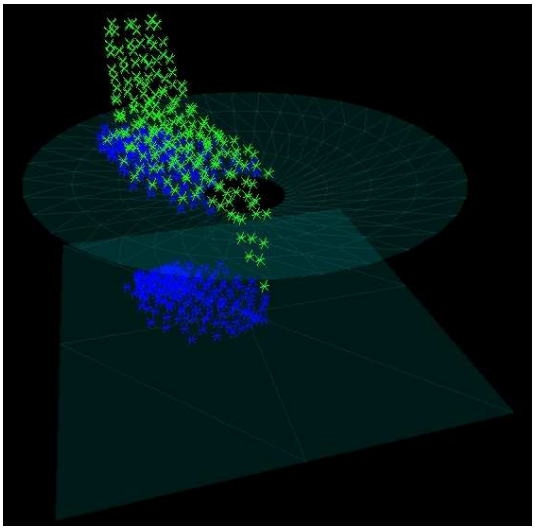


Figure 25: *First successful use of shadowing particles as the dynamic mechanism for faces. Simulation consists of approximately 200 particles and a funnel. The shadowing particles are shown in blue.*

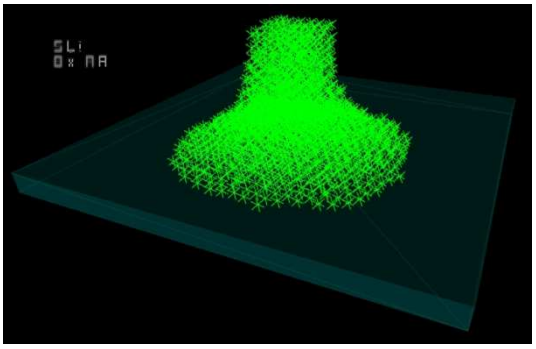


Figure 26: *2500 particle viscosity test using a 5x5 particle emitter and a "tray" geometry.*

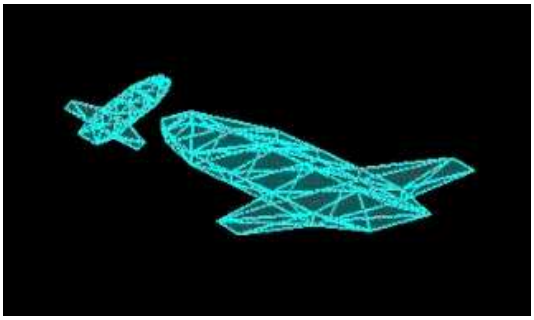


Figure 27: *First practical use of particle binders and cylinders in a collision between two Pegasus models.*

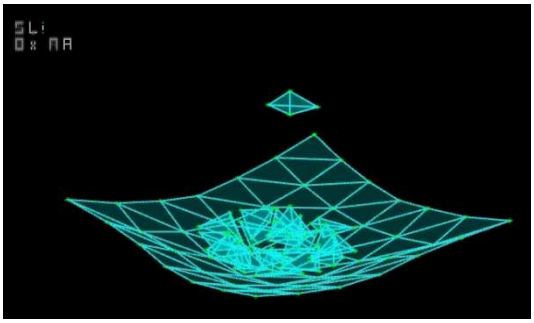


Figure 28: *Simulation incorporating a deformable surface and tetrahedron emitter.*

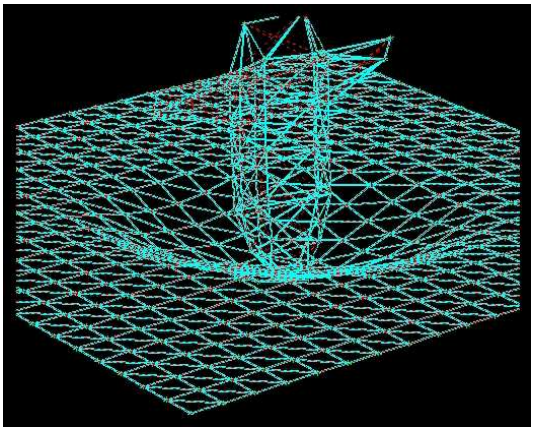


Figure 29: *Test of customizable binders in a simulation consisting of 1300 binders, 883 cylinders, and 443 particles.*

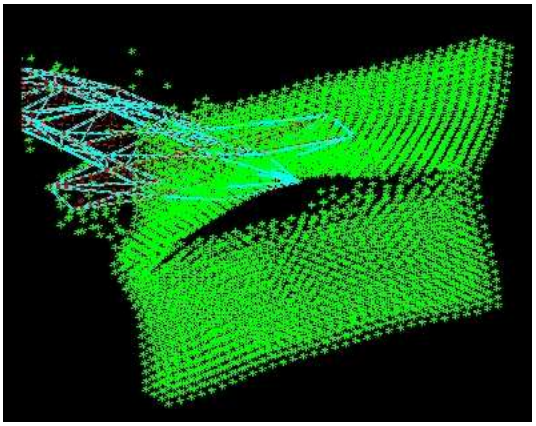


Figure 30: *Simulation of the Pegasus model being propelled through a particle field. 4129 particles were used in this simulation.*

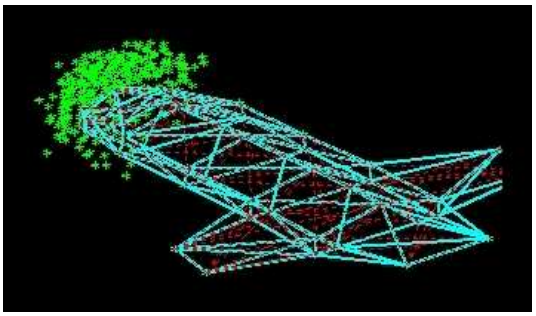


Figure 31: *High energy collision of the Pegasus model and a particle field.*

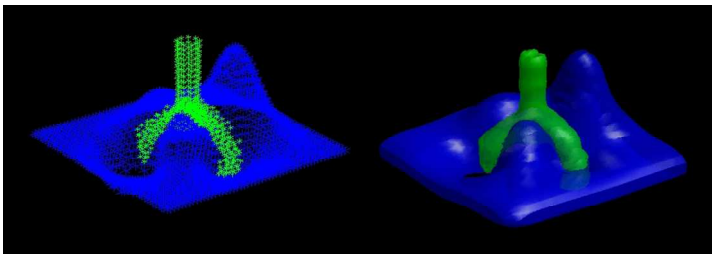


Figure 32: *Terrain test using the multiple scope processing approach.4500 particles were used in this simulation. On the right, Matlab's surface extraction capability used to render the exterior surfaces of the fluid(green) and the terrain(blue).*

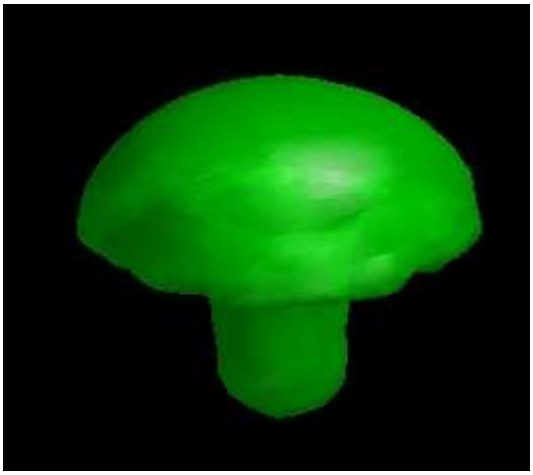
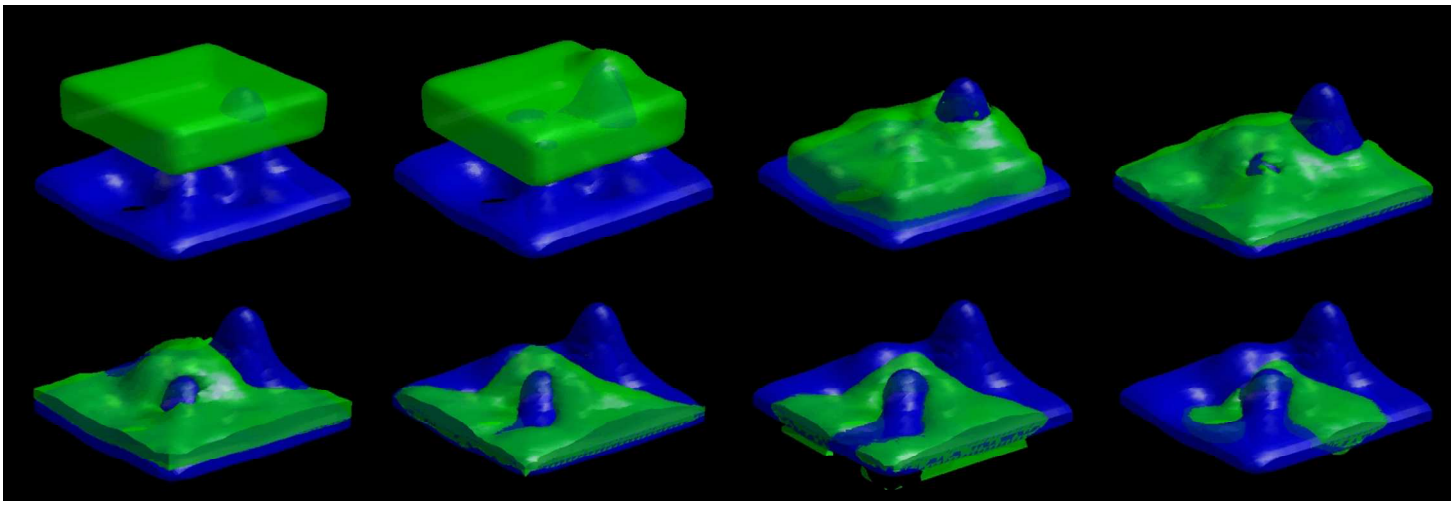


Figure 33: *Fluid jet realizing a mushroom effect under the presence of gravity. 3200 Particles were used in this multiple scope simulation. Again, Matlab's surface extraction capability was used.*

Figure 34: *Simulation of a 9000 particle terrain flood test. The Matlab logo geometry was used to generate the terrain.*



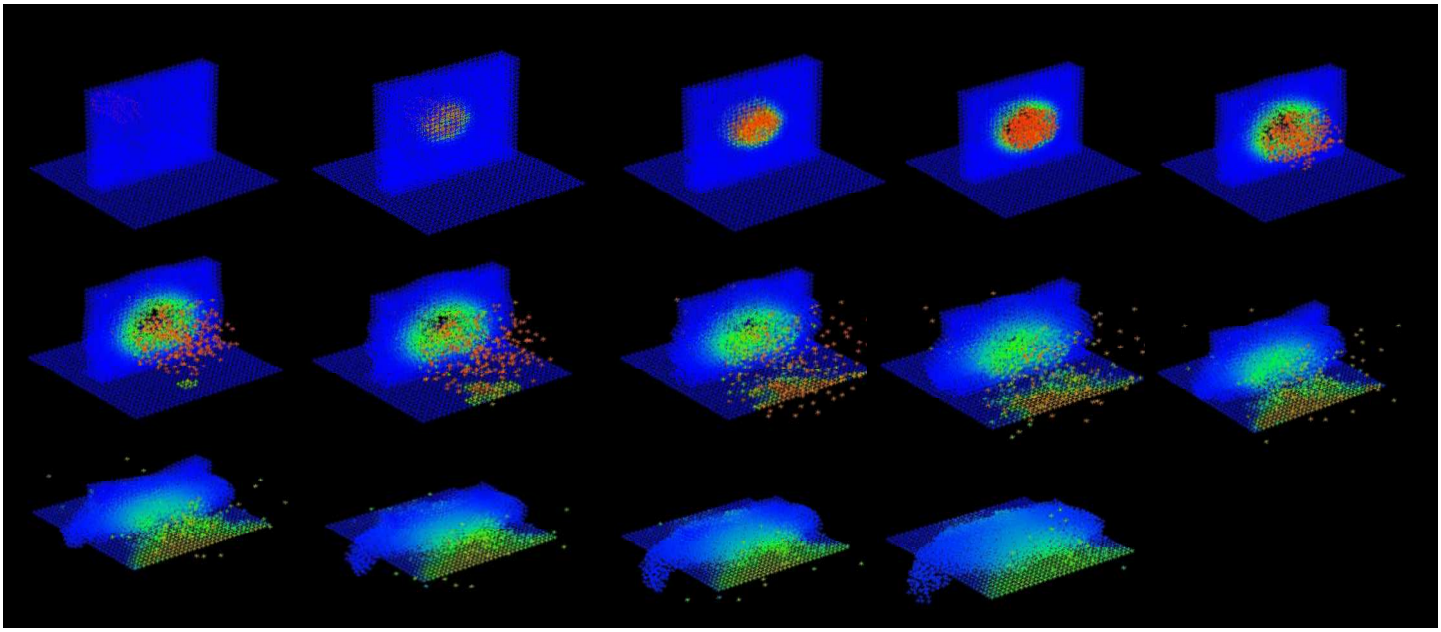


Figure 35: *Thermal dynamic wall penetration using 4200 particles. Particles use simple, proximity controlled thermal couplings. A particle's temperature is used to control its viscosity, as a particle heats up (RED) its viscosity drops.*

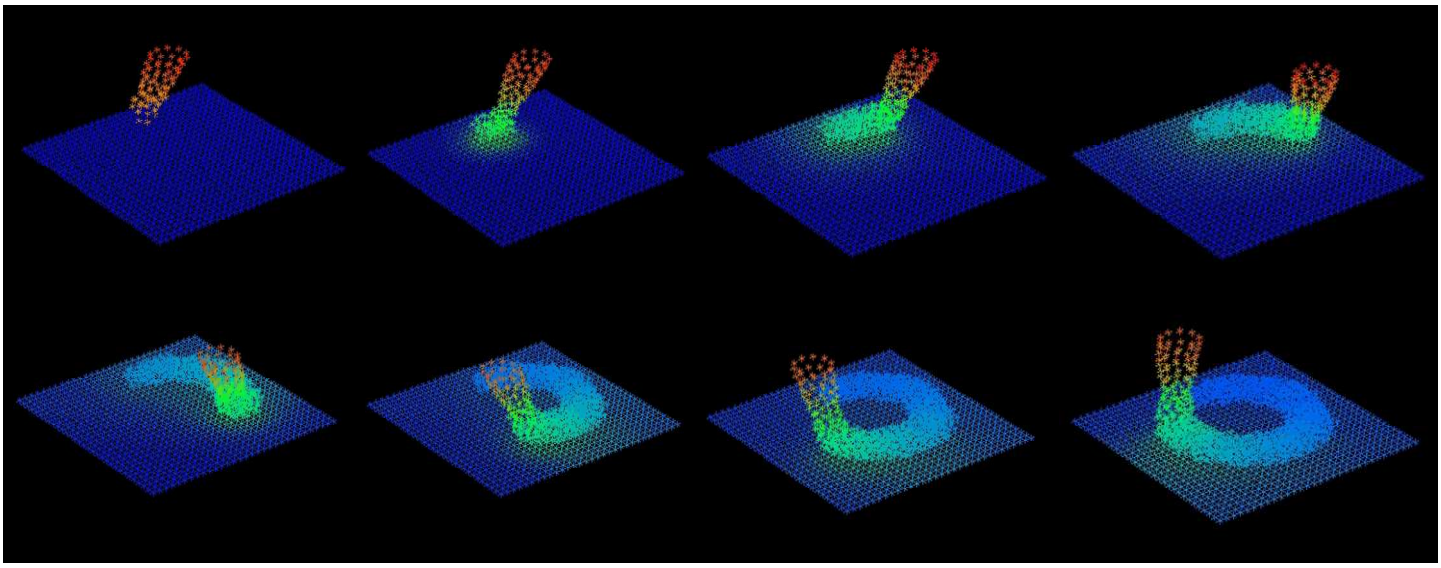


Figure 36: *Substrate deposition utilizing a circular particle emitter that randomizes its orientation along its principle axis. As the flow cools on the substrate, the particles' viscosities are raised in proportion to their temperature.*

Conclusion

I have presented a method of simulation that utilizes simple particle to particle interactions to faciliate applications that incorporate interactive particle fields with interactive geometries to model the behavior of various types fluids, deformable containers, and deformable structures.

The main focuses of development for this project are:

1. Particle to particle interactive dynamics to include rigid body repulsion, viscoelastic behavior (force diffusion), pseudo-density governed attraction, simple thermal couplings, and particle binders.
2. Projecting shadowing particles along vectors and surfaces that use ray projections and barycentric coordinate tests to facilitate dynamic cylinders and surfaces.
3. A toolkit for constructing, manipulating and rendering Fisx simplexes in Matlab.
4. A non-exhaustive method of processing that deals with parsing a simplex into a mesh to maximize cache efficiency.
5. Dynamic integration scheme that regulates time step duration based on the differencing magnitudes of velocity and acceleration.

The application of this method was intended for a vast variety of tasks. Throughout the development of this project, many types of simulations were attempted and evaluated in an effort to expand Fisx's capability and realism as a physics rendering program.

There are many ways to improve Fisx. Because Matlab is used to manage and visualize a simulation, a huge overhead due to file transfer is accumulated drastically reducing the over all effiency. In large, high density, symmetrical simulations, resonance issues occur due to the rigid body implementation because of the integration step-ahead that is used to estimate a particles next undergone acceleration and velocity. The simplex remains stable but Fisx uses an extremely small time step.

I believe that the use of Matlab combined with Fisx's simplicity and versatility should prove useful for many non real time applications.

References

Clavet S., Beaudoin P., Poulin P.: *Particle-Based Viscoelastic Fluid Simulation.* In *Eurographics /ACM SIGGRAPH Symposium on Computer Animation (2005).*

Hoffmann C.: *Robustness in Geometric Computations.:* In *Eurographics /ACM SIGGRAPH Symposium on Computer Animation (2003).*

Premoze S., Tasdizen T., Bigler J., Lefohn A., Whitaker R.: *Particle-Based Simulation of Fluids.* In *Eurographics /ACM SIGGRAPH Symposium on Computer Animation (2003).*

Muller M., Charypar D., Gross M.: *Particle-Based Fluid Simulation for Interactive Applications.* In *Eurographics /ACM SIGGRAPH Symposium on Computer Animation (2003).*

Muller M., Solenthaler B., Keiser R., Gross M.: *Particle-Based Fluid-Fluid Simulation Interaction.* In *Eurographics /ACM SIGGRAPH Symposium on Computer Animation (2005).*

Wikipedia: *Dot Product.* In www.wikipedia.com.

Wikipedia: *Cross Product.* In www.wikipedia.com.

Mathworks: *Point in Triangle Test - Barycentric Coordinate Technique.* In *Realtime Collision Detection* at www.mathworks.com.

Ericson C.: *Memory Optimization.* At *Sony Computer Entertainment, Santa Monica.*

Ericson C.: *Physics for Games Programmers, Numerical Robustness for Geometric Calculation.* In *Game Developers Conference, Sony Computer Entertainment, Santa Monica.*

Funkhouser T.: *Ray Casting, COS 426, Fall 2000.* At *Princeton University.*