Cloth Simulation

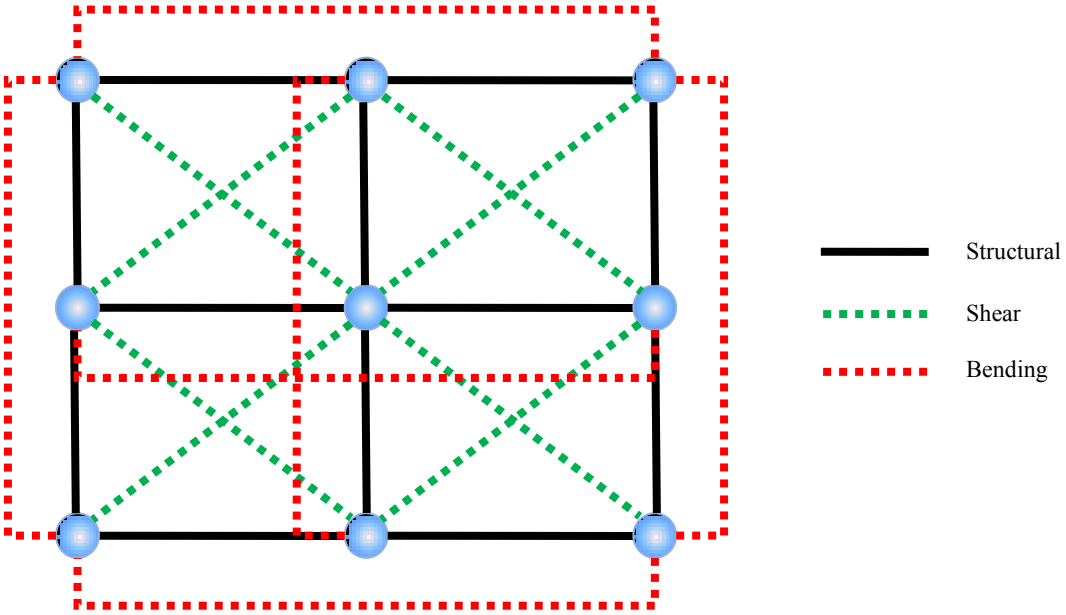
Cloth simulation is a subject of significant interest to the Computer Graphics (CG) community. Modelling and simulation of a perfect cloth has been an extensive topic of research for years, to add realism in the virtual CG world of games, movies, etc. Many methods have been used to simulate a perfect cloth that is reacts correctly to forces applied, character movements, and display the various properties of cloth like folds and bends. Among other methods, Mass-Spring model is widely used over complex energy models as it provides the ease of implementation. Its efficiency in handling external forces and fast responses gives it an edge when compared to other models.

Mass-spring model

Mass-spring model has been extensively used in computer graphics for modelling various deformable objects. The cloth model is represented by a grid of particles of known mass connected by a series of spring-dampers. Each spring's rest length is set to the original length of an edge. *This helps to keep two particles at a fixed distance as required by the model property*

There are three types of springs which contribute towards the cloth structure:

- Structural Springs- these springs connect the cloth mesh in a rectangular grid. Structural springs handle extension and compression.
- Shear Springs – these springs connect the adjacent particles diagonally and handle the shear stresses.
- Bend Springs – these springs are connected vertically and horizontally to every other particle; and help in imparting the bending properties of a cloth.



The mass particles are present at the grid vertices and are connected to neighbours using the above springs. The varying properties of a spring give different cloth behaviors. When external forces are applied to the specific vertices of the cloth mesh, the vertices will displace relative to the other vertices of the cloth. This displacement induces spring forces which impart forces to the adjacent vertices and reactive forces back to the initial vertex. This causes more displacements resulting in more springs forces and so on.
Most types of cloths have a strong resistance to stretch, where as they fold over very easily when subject to forces. Mathematically, this behavior can be expressed as a partial differential equation of the following form:

$$x = M^{-1}\left(-\frac{\partial E}{\partial x} + F\right)$$

Here, *x* and *M* represent state and mass distribution of cloth. *E* is the internal cloth energy dependant solely on its current state. *F* represents all the external forces acting on the cloth which can be gravity, wind, etc.

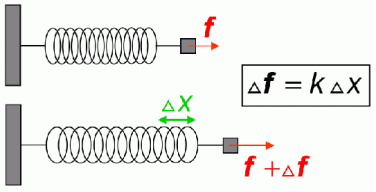
The simulation of cloth comes down to realistic movement of mass particles subject to forces which can be internal or external.

Internal forces –

1) *Spring Forces* - the tension in connected springs gives rise to internal forces. The spring forces are determined by the Hooke's law of elasticity:

$$F_s = -kx$$

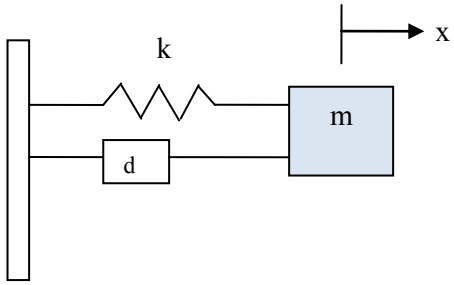
where, k is the spring constant and x is the displacement in the spring from its rest length.



F_s can be further expanded to define the spring forces that pull or push two mass particles at a desired rest length of *d as:*

$$F_s = -k(|x| - d)(x/|x|)$$

where, $|x|$ is the distance between the two points connected by spring. And $x/|x|$ is the unit direction vector between the two mass particles.



The force acts as a restoring force, and is hence negative (the direction of the restoring force is opposite to that of the displacement).

2) *Damping* - The component of internal force that brings the spring back to its rest state is damping. In the absence of damping force the springs oscillate to infinity. The damping force acts against the velocity of the mass particles to slow them down until they come back to rest state. The damping force is defined as follows:

$$F_d = -bv$$

where, b is the damping coefficient and v is the relative velocity between two mass particle connected by a spring.

Thus spring forces acting on each mass particle is given by,

$$F_i = F_s + F_d = -k\text{x} - bv$$

And $F_i = -k(|x| - d)(x/|x|) - bv$ for two particles connected by a spring

External Forces- the simulation can involve various external forces as per requirements. It basically consists of gravity and wind, to simulate main physical forces acting on a cloth.

Gravity is implemented using Newton's second law of motion, $F_g = mg$

where, m is the mass and g is the constant of freefall acceleration.

Wind is simulated by applying the wind force on mesh faces instead of individual particles, in the desired direction of wind. For this, the normal of the face is multiplied with the wind direction to calculate the force acting on the triangular face of the mesh.

Adding all the internal and external forces gives a beautiful simulation of a mass spring model cloth.

Once all the forces are applied on cloth particles, the state of the mass particles changes. The state of the mass particle is defined by its position and velocity. We are using the RK4 integrator to calculate the mass particle state values i.e. the position and the velocity depending on the forces acting.

RK4 Integrator or the Runge–Kutta method is an iterative method for the approximation of solutions of differential equations.

Runge-Kutta methods are described by Conte & de Boor as, "methods to obtain greater accuracy than Euler's method, and at the same time avoid the need for higher derivatives, by evaluating the function f(x, t) at selected points on each sub-interval" [3].

These methods are based on taking intervals during the time step to approximate a more accurate answer. RK4 is described as a fourth order since it takes more time intervals in between time steps.

The standard *fourth-order Runge-Kutta* method takes the form: [4]

$$k_1 = hf(x_n, y_n),$$
$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + k_1/2\right),$$
$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + k_2/2\right),$$
$$k_4 = hf(x_n + h, y_n + k_3),$$
$$y_{n+1} = y_{n+} + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5).$$

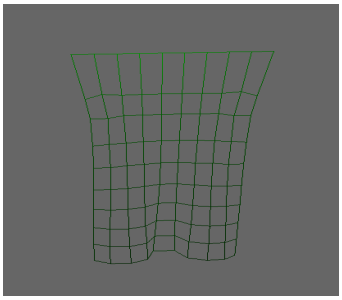
IMPLEMENTATION

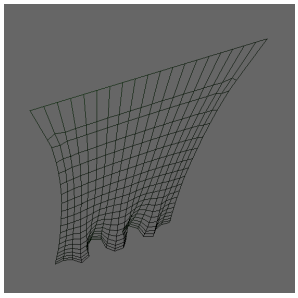
Algorithm: The cloth is updated in the simulation using the following algorithm –

1. calculate spring forces
2. apply gravity
3. update the state (position and velocity) of mass particles, using RK4 integrator
4. draw the cloth
5. check collisions
    a. if there is collision, update the cloth using RK4 integrator
    b. repeat step 3

The simulation checks for collision with two rigid bodies - a sphere and a table. For sphere mesh,

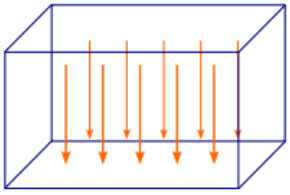
Initial implementation: involved creating a simple grid structure of springs and experimenting with the internal spring forces to create a stable cloth mesh. The structure was tested with external gravity force. Following are the screenshots of initial implementation:-





Final Implementation: has been integrated with RK4 to calculate the state of the mass particle with respect to the changes in internal and external forces. The interaction of the cloth with rigid bodies has been implemented using collision detection. The mass particles joining together the cloth springs are assumed to be spheres for implementing the algorithms. Forces like gravity and wind have been added to give realism to simulation.

For applying gravity to cloth: - each cloth particle is affected by the gravity force vector to make the simulation realistic. The gravity acts in the negative y-axis direction with a force F_g.

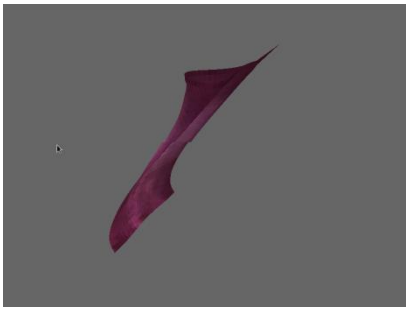


F_g = mg, where 'm' is the mass of the particle and 'g' is the gravity vector in –y direction.

For applying wind force to cloth: - for applying the wind force on cloth, sphere-plane collision algorithm has been applied using the following algorithm:

- The cloth's mass particles are stored in a vector in the format of a triangular mesh.
- Iterate through each triangle of the cloth

- If there is collision, the mass particle position $P(t) \in triangle\ ABC(t)$ of mass particles.
- Calculate its current normal $N(t)$ and normalize it: $N(t).normalize(\ )$;
- Calculate the wind force by multiplying the normal with the direction of wind.
- Update the force vector of the each particle constituting the triangle with the calculated wind force.



For collision between cloth and sphere: - I started with Sphere-Sphere collision detection algorithm for collision between cloth and sphere. Two spheres *A* and *B* are defined as having a centre and a radius, r1 and r2. They collide when the distance between the two centers is less than the sum of their radii: ||A-B|| < (r1+r2)

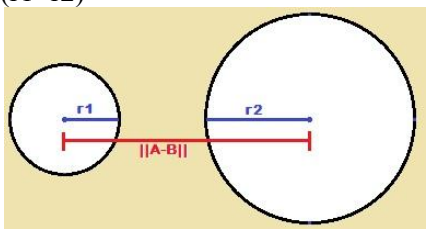
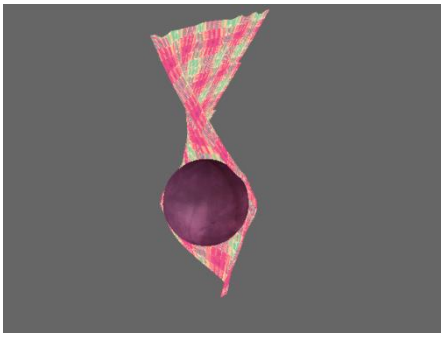


Fig 4 - Two circles *A* and *B* are defined as having a centre and a radius. They collide when the distance between the two centers is less than the sum of their radii:
||A-B|| < (r1+r2)
$||A-B||^{*2} < (r1+r2)^{*2}$
Amount of penetration, p = ||A-B|| – (r1+r2)
So, ||A-B|| – (r1+r2) < 0
Normalized Vector, N = A-B) / ||A-B||, between the two centers.
Penetrating Vector, P = N*p
The two main parts of collision detection are –

- Detecting whether a collision has happened or not
- If yes, what should be the response to the collision?

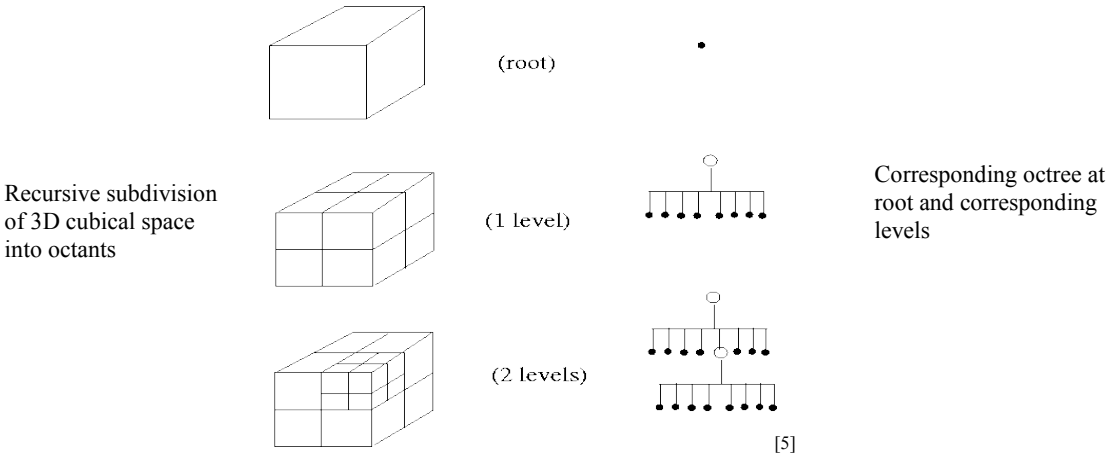
In our case, we iterate through all the mass particles and compare the distance between the mass particles and the sphere. The minimum distance for a collision between them is the radius of sphere, as the mass particles do not have any radius. So the algorithm reduces to point sphere collision. An offset has been added so make sure that the cloth does not penetrate the sphere.



For collision between cloth and table: - in the simulation the cloth is dropped from a height and drops on a table. For this we iterate through the mass particles and check the distance between the cloth particle and the table's plane. If it is less than the table height, we consider a successful collision between the

cloth and the table. The cloth settles on the table and the extra cloth that does not lie on the table plane, gives a nice fall around the edges due to the effect of gravity.

For self-collision between cloth particles: -



[5]

The algorithm starts with an empty tree, containing one leaf node with no items. We iterate the mass particle array to check for collisions between mass particles. In the event of self collision i.e. the cloth folds, we reduce the overhead of calculations by dividing the cloth mesh into octants. The mass particles present in different octants are not compared with the mass particles present in different octant. The calculations are concentrated solely on the octant where the collision occurs.

PROGRAMMING

OpenGL – OpenGL stands for Open Graphics Library. It is the graphics API (Application Programming Interface) which provides the programmer an interface to the graphics hardware. The main advantage of OpenGL is its compatibility with multiple platforms like Windows, Linux, Mac OSX and portable devices. It is and open standard i.e. many companies can contribute to its development (does not mean that it is an open source).

Qt Creator - is a cross-platform C++ IDE (integrated development environment), part of the Qt SDK. It includes:

- visual debugger
- integrated GUI layout
- forms designer

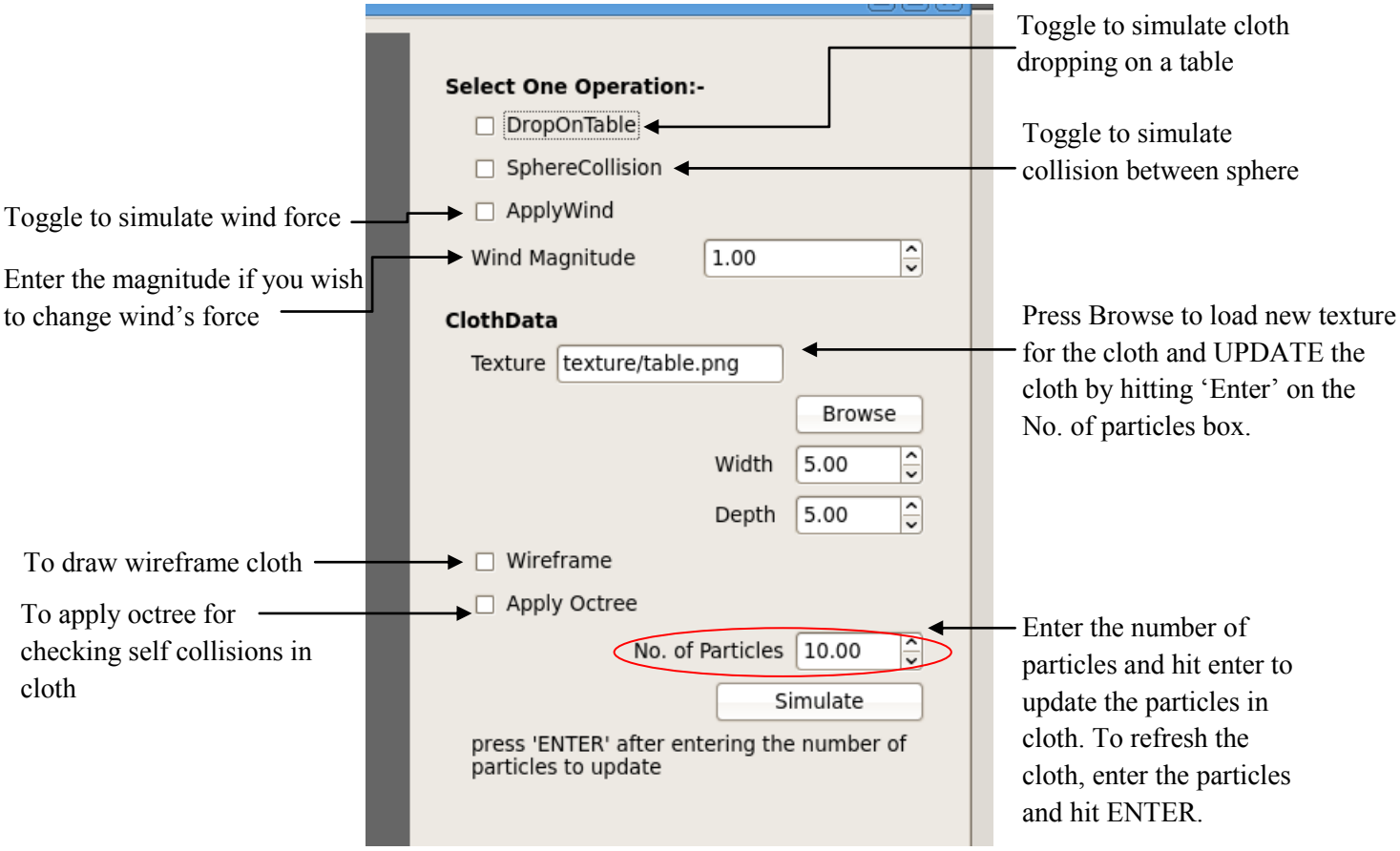
C++ - object oriented programming language (OOPS)

NGL – NCCA graphics library, authored by Jon Macey.

Referenced Code – Jon Macey's NGL Demos – RK4 integrator and Octree Abstract.

HOW TO USE:

I have tried providing a good user interface to the user for running the simulation but it does not do justice to the programming. It does let the user get a brief glimps of the possibilities that can be achieved using the implementation.



CONCLUSION:

Overall the implementation of cloth simulation was successful and provided a platform to explore the fundamentals of the concept. The project introduced me to the mass-spring model which is the basis for many other simulation systems. I started with no prior knowledge of mass-spring model, or RK4 integrator; and have successfully implemented a stable mass spring model and integrated it with RK4 to update the cloth's state. Generating a stable mass-spring model to behave like a cloth was a challenging task, but the results have been fruitful in the end.

Future Improvements –

- Flexible User interface
- Improve collision detection and responses with imported meshes
- Implementing triangle edge-edge detection algorithm for improving self-collisions detection in cloth.
- Mouse interaction with cloth

REFERENCES:

[1] Parent, R. 2008. Computer Animation: Algorithms and Techniques. 2nd Edition. Massachusetts : Morgan Kaufmann.

[2] Fielder, G. 2006. Spring Physics. Gafferon Games. Available from http://gafferongames.com/game-physics/spring-physics/ Last Accessed 08 May 2013

[3] EBERHARDT et al. A Fast, Flexible Particle System Model for Cloth Draping. IEEE Computer Graphics and Applications, 16(5): (September 1996), 52–59

[4] Fitzpatrick, R. 2006. Runge-Kutta methods. Available from: http://farside.ph.utexas.edu/teaching/329/lectures/node35.html Last Accessed 08 May 2013.

[5] Baert J. Ray / Octree traversal. Available from: http://www.forceflow.be/2012/04/20/ray-octree-traversal/ Last Accessed 08 May 2013.