Particle Based Weathering System

MSc CAVE
Computer Generated Imagery Techniques
Bournemouth University
NCCA

Martin Davies

November 2014

Abstract

Image synthesising techniques tend to create very clean scenes. This is not reminiscent of reality where objects that are exposed to the environment change appearance and form over time. This is a process known as weathering. A desired outcome of computer graphics is to imitate what is seen in the real world. Therefore, creating weathered objects is a large but relatively unexplored research area. This project attempts to implement a well known solution to this problem using the notion of γ-tons. These are propagating particles that interact with the scene by depositing or transferring dirt and humidity attributes. The input of the system is an string, a file path to a .obj file. The output is a texture map that can be applied to the object. A preview window will also be available. In this document, related work is investigated and the implementation method is explained.

Contents

Introduction

Objects exposed to the environment change in both appearance and form. This process is known as weathering. Weathering causes the properties of the object to alter. Some factors that can result in material alteration are: dirt, rusting (oxidation), water flow, lichen growth, fire and impacts etc. Photorealistic computer generated images attempt to fool audiences into believing that they are real. Weathering modelling is key to creating realistic scenes as we are exposed to weathered objects everyday. Therefore an audience will not be convinced if weathering is not added. Previously artists were subject to tedious texture creation[7]. This process was labour intensive and regularly resulted in textures breaking when applied to a model. An automated solution was required. Many researchers based their investigations around multilayering of textures and BRDFs [6] [1]. Other research concentrated on automated solutions. Many used particle systems [3] [10] [7] [17].

This project attempts to implement Chen et al's [3] solution for generating weathering effects. It will incorporated γ-tons to imitate the weather. γ-tons are very versatile, it is possible to imitate water droplets in the atmosphere, flowing water, scratches, impacts etc. by using them. Therefore, a system based on Chen et al's research would be very versatile but could also be very expensive. This is due to γ-ton processing, since a large amount of γ-tons will result in high computation time. A solution to this is parallelism using a GPU or, on a larger scale, a render farm[10].

The material being weathered has to be considered. For many materials, especially metals, the weathering process can be destructive [11]. Destructive corrosion exposes the base layer of metal. Therefore, corrosion is dependent on the amount of water that gains access. Other forms of corrosion exist, for instance; when copper is exposed to the weather it forms a patina [6]. Patina's are a protective layer of corroded metal; they do not allow water to access the base layer of metal. Therefore, the corrosion rate reduces after a patina is formed whereas corrosion rates increase in destructive forms. This project concentrates on the aesthetics of weathering, therefore accuracy to nature is not important but it is considered.

Related Work

2.1 Weathering

Blinn developed a way to create dusty surfaces using light reflecting off particles which were contained in a cloud [2]. It proved to be inspiration for weathering effects after being applied to create Saturn's rings.

The main inspiration behind this project is the work of Chen et al [3]. The system they produced used a concept called γ-ton tracing. A γ-ton is a particle containing two components, motion probabilities and carrier attributes. Motion probabilities define the motion of the γ-ton and must be included when one is created. There are three types of motion a γ-ton can perform: Straight, Parabolic and Flowing (*Ks, Kp, Kf*), see figure 2.1, and must be $0 \leq K \leq 1$.

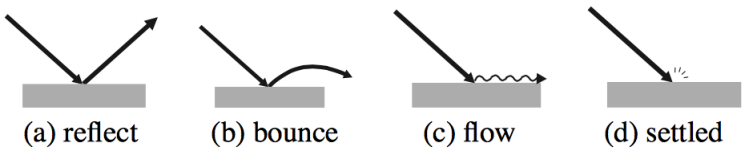

Figure 2.1: Types of motion. Image from [3]

Ks is the probability it follows a straight path. *Kp* is the probability that it follows a parabolic path. *Kf* is the probability that it will slide along the surface of an object. Chen's system also had a method to check whether a γ-ton had settled. This is calculated by subtracting the motion probabilities from 1.

$$ProbabilityOfSettling = (1 - Ks - Kp - Kf) \tag{2.1}$$

Carrier attributes contain weathering properties of the γ-ton. A γ-ton can have any number of these, but usually they are tailored to fit the type of weathering that is going to be produced. For instance, a γ-ton may have a humidity carrier attribute that results in a rust effect.

γ-tons are assigned an emitter that shoots it into the scene. Chen et al describes three types of sources : Point, Area and Environment. Point emits γ-tons directly form the object surface. It is used to create paths of weathered area, e.g. rust from a leaky pipe. Area distributes γ-tons throughout the area of a shape and emitted towards the scene. Environment emits γ-tons all around the scene. Chen et al uses a hemispherical environment emitter.

Object surfaces are described using surfels. Surfels are a primitive that holds two attributes, γ-reflectance and material property, of the material at the surface point which they exist. The surfels that make up the object density are stored in a kd-tree [14]. γ-reflectance ($\triangle s$, $\triangle p$, $\triangle f$) responds to γ-ton collisions by deteriorating γ-ton motion probabilities. This must be included when a surfel is defined. Material properties contain the surface properties of the object. Values can be added or remove from these to manipulate the surface attributes. When a collision occurs between a γ-ton and a surface two things occur. Firstly, new motion probabilities are calculated, Chen et al used Russian Roulette [20] which are calculated using:

$$Ks' = max(Ks - \triangle s, 0) \tag{2.2}$$

$$Kp' = max(Kp - \triangle p, 0) \tag{2.3}$$

$$Kf' = max(Kf + max(Kp - \triangle p, 0) - \triangle f, 0) \tag{2.4}$$

γ-transport also occurs. This is where propagating γ-tons pick up and deposit substances from surfels it interacts with. Done by using :

$$a \leftarrow a + b \cdot k \tag{2.5}$$

to calculate the transfer of material properties. Where a are the material properties of a surfel, b are the carrier attributes of the γ-ton and k is a scalar weight.

Günther et al [10] developed Chen et al's [3] work by accelerating γ-ton processing using the GPU. Also, they manipulated the textures directly instead of using surfels. Gu et al [9] also used Chen et al's [3] research to develop time varied BRDFs for a variety of different materials. These materials were placed in a camera dome, similar to an environment source, and images were taken of materials in different stages of weathering. Following Blinn there is similar work preceding that of Chen et al's. Dorsey et al created a weathering system that used water droplets to transport dirt and corrosion [7]. Water droplets were single particles that were controlled by a number of parameters. The amount of water was controlled by using absorption and evaporation. Desbenoit et al developed a system for lichen growth [5]. Using regions, which can be drawn on by the artist, for lichen seed dispersion, where they attempt to settle and grow lichen.

Using multi-layering Dorsey created a system that could create a patina for copper [6]. This method used operators to define how each layer interacts when exposed to an environment. This method is good for producing patinas but not for destructive corrosion. Merillou et al expanded on Dorsey's work to develop a system that applies destructive corrosion to an object [11].

As well as dirt and corrosion, object deformation plays a part in the weathering process. Paquette et al developed a system that uses impacts to age objects by deformation [17]. By creating hundreds of impacts between the object they want to age and the impacting object per second; they created an interactive and artist friendly system. Bosch et al created a system to add scratches to a surface by improving on their preceding system [1]. Scratches were created using a combination of textures and BRDFs. The texture defines the location and direction of the scratch. The BRDF calculates the light reflection at each scratch point; calculated using the microgeometry of the scratch.

Before an artist friendly system was created artists would meticulously create weathering textures for their scenes [6] [7]. This method gave good end results but was time consuming and textures regularly broke when applied to an object. Therefore, an automated solution was desirable.

2.2 Texture Synthesis

Texturing is key to computer graphics. This system will synthesise a texture based on interactions between γ-tons and surfels. It will accomplish this by compositing a dirt map, a humidity map and a displacement map together. The maps will be created using a γ-ton map. Porter and Duff introduced the compositing of digital images to computer graphics [19]. They proposed that image synthesis be comprised of a number of modules and combined together later. This is a process used throughout the animation and visual effects industry today.

2.3 Particle Systems

Seminal work by Reeves resulted in the first particle system [21]. It was used to create the genesis bomb sequence in Star Trek II: Wrath of Khan[12]. A particle is a primitive that is assigned behavioural attributes. For example, position, lifetime, acceleration, direction etc. By manipulating these attributes particle behaviour can be altered. Once the lifetime reaches 0, a particle exits the scene or a particle displays odd behaviour it can be removed from the simulation. Particles were a breakthrough in computer graphics and have been used to model fire [21], grass [22], smoke [23], snow [24] and much more.

Implementation Method

3.1 Simulation Iterations

The system is iterative[3]. A class is included to control the system[4], it initialises other classes when their part of the iteration begins. See figure 3.1.

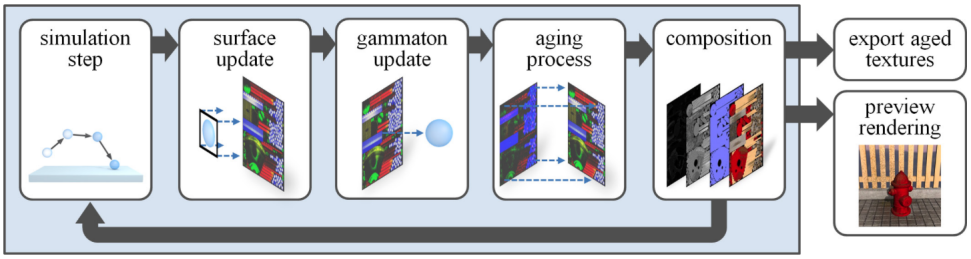

Figure 3.1: Actions per iteration.Image is from [10]

3.2 Object

The system will have a default object that will load if a wavefront .obj file is not supplied to it. This default object will either be a cube or a sphere. In the user interface a file path to an .obj can be entered. This will be stored as a string. NGL has a module that loads .obj's. It gives us access to the vertices and the faces of the object and will be used in this system.

3.3 Object Density

Once an object has been loaded a density map is created; formed of a user defined number of surfels.

3.3.1 Surfels

The system will use points to represent Surfels. They contain information about object surface material properties and are organised in a kd-tree [14]. These are emitted from a bounding box and attach to the point where they collide on the object surface. This creates a representation of the surface which γ-tons can interact with. A surfel is made up of γ-reflectance and material properties. An example structure of a surfel is:

```
//γ-reflectance properties
float △s; // deterioration rate for Ks
float △p; // deterioration rate for Kp
float △f; // deterioration rate for Kf

//material properties
float sh; //humidity
float sd; //dirt
...
```

When a γ-ton intersects the object the nearest surfel can be found by traversing the kd-tree. Once this is found the γ-reflectance properties of the nearest surfel are used to manipulate the behaviour of the intersecting γ-ton. Transfer of material properties will also occur between the surfel and γ-ton.

3.3.2 Bounding Box

This emitter is a cube shaped environment emitter that surrounds the object in the scene. Surfels are added to the inside faces of the bounding box. Since a representation of density is required, surfels need to distributed evenly across the inside face of the bounding box. To do this Halton Sequences [18] can be used. Halton sequence's is in the family of quasi-random sequences. It takes a base prime and an range of numbers to split between. For a square in 3D three Halton Sequences are combined for the X,Y and Z coordinates of each point. Once the surfels are distributed they are emitted towards the object. Collision with the object generates a density map.

The larger amount of surfels the better the end result will be; higher amounts of surfels will result in a higher computation times. The user should be restricted to a minimum and maximum surfel number. This will ensure a good effect is achieved at acceptable computation times. This emitter works as demonstrated in figure 3.2

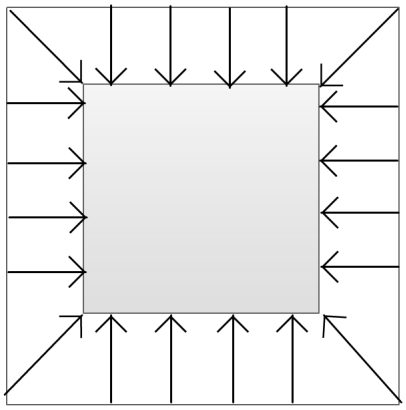

Figure 3.2: Bounding box emission.

3.3.3 Density Map

Each surfel can be drawn creating a point cloud of the object. This can be used to visualise the weathering simulation. By mapping these points to (u, v) space it creates a texture of surfels that can be manipulated after γ-ton interactions. This is the initial γ-ton map.

3.4 Emitters

There will be three emitters in the system. Two will be environment emitters and one will be an area emitter. Initially a point emitter will not be implemented; the code will be designed in such a way that adding one will be trivial. The bounding box emitter has been explained previously.

3.4.1 Spherical

This is a constricting environment emitter that surrounds the whole scene. γ-tons will be added to the sphere at each of the vertices, when the emitter is constructed. Therefore, someγ-tons will have the same origin. This is not a problem as all the γ-tons will have different motion probabilities, which are dynamic as they interact with the scene. A sphere is used as objects can be confined within them easily. Also a good distribution of γ-tons can be acheived. The provisional code to construct the sphere is:

```
//define PI
const float PI = 3.14159265358979323846264338327950288419 7;

//Sphere Centre is origin
const ngl::Vec3 centre (0,0,0);
const float radius = 2.0;

for (float phi = 0.0; phi < (m_pnum/100) * PI; phi += PI/10.0)
{
   for (float theta = 0.0; theta < PI; theta += PI/10.0)
         {
            ngl::Vec3 point;
            point[0] = radius * cos(phi) * sin(theta) + centre[0];
            point[1] = radius * sin(phi) * sin(theta) + centre[1];
            point[2] = radius * cos(theta) + centre[2];
            m_eP.push_back(point);
         }
}
```

Where m_eP is a vector of γ-ton positions. The γ-tons will behaviour similarly to figure 3.3

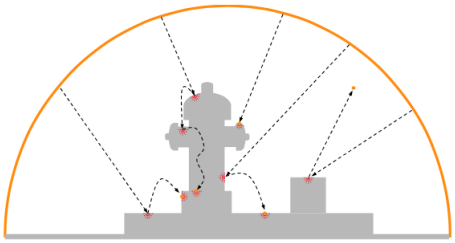

Figure 3.3: Hemispherical emission. Image is from [3]

3.4.2 Vertical

This is a rectangular area emitter that is placed above the scene. The first four γ-tons are assigned to the vertices of the square. The Y value is constant, as they all have to be at equal heights, and the X and Z values are calculated using the radius of the Hemisphere emitter. Once the initial vertices have been put in place Halton Sequences [18] are used to distribute the remaining γ-tons. Each γ-ton will be emitted downwards towards the scene, similar to rainfall. γ-tons produced by this emitter will have a Ks probability of 1, Kp and Kf will be 0. This is because all γ-tons emitted from here have to travel straight. A vertical emitter will work as shown in figure 3.4

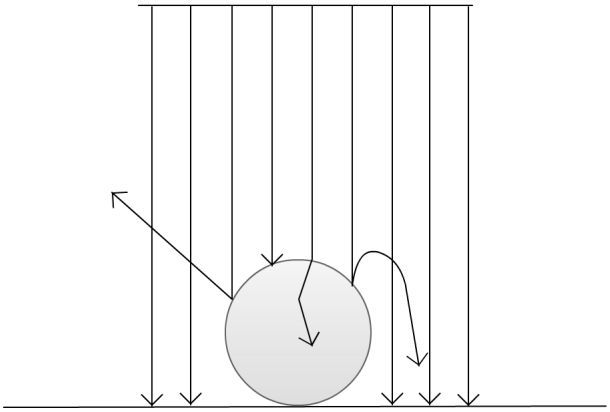

Figure 3.4: Vertical Emission.

3.5 γ-tons

The biggest part of the system is γ-ton processing. The amount to construct is user defined and stored as an integer by the emitter. Once constructed they propagate the scene based on their motion probabilities and interact with objects they collide with. When all γ-tons have settled, run out of energy or exited the scene the iteration ends and the γ-ton map can be updated.

3.5.1 Construction

The number of γ-tons is used in a loop that adds instances of the Particle Class [4] into a vector. Therefore there is easy access to all of the γ-tons associated with the emitter. The basic structure of a γ-ton is:

```
//γ-ton motion probabilities
float Ks; // Probability of straight line motion
float Kp; // Probability of parabolic motion
float Kf; // Probability of flowing motion

//material properties
float sh; //humidity
float sd; //dirt
...
```

.

The construction in this system will be slightly more complex. The attributes of γ-tons in this system are:

- Position - Current position in 3D space.
- Ks - Straight motion probability.
- Kp - Parabolic motion probability.
- Kf - Flowing motion probability.
- dirt - Material attribute.
- humidity - Material attribute.
- probability - Probability of settling. Used in Russian Roulette [20].
- intersect - Boolean for intersection with object.
- energy - If energy = 0 remove or settle.

3.5.2 Propagation

This system incorporates straight, parabolic and flowing motion. To implement the straight motion parametric lines and linear interpolation will be used. Using the γ-ton position and the origin, as a start and end point, a parameter, t, can be used to interpolate along the line. At each t a check for collisions can be performed. The following equation represents this:

$$NewPosition = (CurrentPosition * (1 - t) + origin * t) \tag{3.1}$$

When a γ-ton reflects and end position will not be known. However, the starting position and direction will be known. The following equation can be used in this case:

$$NewMotion = CurrentPosition + (t * NewDirection) \tag{3.2}$$

Parabolic motion is straight motion under the influence of gravity. The implementation will increase and decrease the Y value by the influence of gravity for each t . Gravity's influence must be $0 \leq$ gravity strength ≤ 1. Initially the gravity strength will be constant, features to allow the user to select gravity strength may be added later. The following equation represents parabolic motion:

$$NewYPosition = CurrentYPosition * GravityStrength \tag{3.3}$$

$$CurrentPosition = [XPosition, NewYPosition, ZPosition] \tag{3.4}$$

$$NewPosition = (CurrentPosition * (1 - t) + origin * t) \tag{3.5}$$

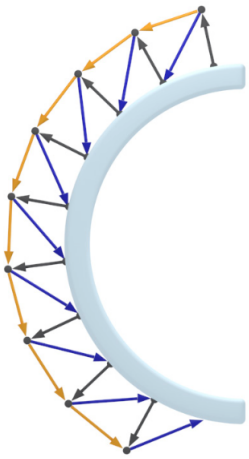

Figure 3.5: Bounding sphere reflections for flowing γ-tons. Image is from [10]

Flowing motion can be implemented using a series of reflections [10]. It requires a bounding sphere around the object. When a γ-ton flows along the surface it will reflect between the surface and the bounding sphere. See figure 3.5.

This method is compute intensive, and easier solution is to use straight motion along the tangent at the point of intersection. This method will be implemented in the system.

3.5.3 γ-reflectance / γ-transport

After a γ-ton interacts with a surfel, the surfel γ-reflectance properties are used to decide how the γ-ton should react. γ-reflectance properties deteriorate the γ-ton motion probabilities. Once this is completed Russian Roulette [20] is used to decide whether the γ-ton settles. Based on real Russian Roulette where odds of survival are reduced per round. The probability of settling, a value $0 \leq$ probability ≤ 1 and stored in the γ-ton attributes, is supplied to the Russian Roulette algorithm. It is used as the current rate of survival. If the γ-ton survives then its probability is increased using :

$$W_{new} = W_{old} \frac{1}{(1-p)} \tag{3.6}$$

Where W is the statistical weight of the γ-ton and p is the survival rate.

γ-transport is the transfer of material properties. This can be done by adding a proportion of γ-ton material properties to the surfel material properties and vice versa, accomplished using equation 2.5.

3.6 Collider

The collider checks for collisions and processes them. Checks occur at each step in γ-ton propagation. Compute time of checks will be high as each γ-ton will have a number of checks per iteration. To reduce the compute time parallelism can be used, this is explained later. To check for collisions techniques from ray tracing, primarily ray/triangle intersection, are used. The processing stage consists of conversion of collision data for map generation.

3.6.1 Ray/Triangle Intersection

Parametric lines are used for motion, therefore processes described by Möller and Trumbore[13] will be used to check for γ-ton/object intersections. This method assumes that the object has been triangulated. A check will be required to ensure that the object is in the correct format. If the correct format is not supplied, the object will have to be triangulated. Often quadrilaterals are used to construct object meshes. Therefore, polygonal triangulation can be used to convert meshes. This is explained in [25].

3.6.2 Processing Collisions

Each of the collision found are stored into a list. The processing stage consists of traversing the list and updating the γ-ton map ready for the next iteration. At the last iteration the γ-ton map is updated and used to generate textures.

3.7 Maps

Output of the system is a texture. The texture is generated by compositing a displacement map, humidity map and dirt map.

3.7.1 Displacement Map

Some γ-tons may cause dents, destructive corrosion or scratches. The γ-ton map can be used to select ones that cause deformation. By projecting selected γ-tons into (u, v) space a displacement map can be created.

3.7.2 Humidity and Dirt Map

To add rust γ-tons, with high humidity, are selected from the γ-ton map. Projecting this into (u, v) space creates a texture for rust. The colour is selected by drawing on previously handmade textures. The same process for humidity is performed for dirt.

3.7.3 Compositing Maps

Ideas are drawn from [19] but techniques described are drawn from image synthesis. Each texture will be considered from top left to top right until the bottom right is reached. Information on pixel colour at each point will be sent to the final texture. If there are competing pixel colours, at a point, the humidity/dirt colour will always win over the displacement. Each texture will also be available separately. Allowing the user to apply the textures in a 3D software package, e.g. Maya.

3.7.4 Applying Texture

The system will render a preview of the weathering effect using OpenGL. To apply the texture synthesised by the system OpenGL has glBindTexture(). This allows us to load the texture into the scene and assign it to a vertex using glTexCoord().

3.8 GPU Acceleration

With time permitting the solution could be parallelised where γ-ton processing can be spread over a number of cores in the GPU [10]. A SIMD (Single Instruction Multiple Data) structure is suitable for γ-ton processing. Each core is assigned a number of γ-tons to process. Speed increases should be noticed by adding this functionality into the system. By using the GPU nicer effects can be generated or the same effects but faster.

Conclusion

4.1 Conclusion

This document has presented related work and an implementation method for a particle based weathering system. This project has gathered a lot of inspiration from the automated system developed by Chen et al[3]. It attempts to implement these ideas and extend them. By using γ-ton tracing the user can produce aesthetically pleasing, although not accurate, weathering effects for their scene. Chen et al's [3] system has been groundbreaking by making the generation of effects such as "stain bleeding" trivial. The original system did not allow for interactive visual feedback from the simulation. This implementation will have a preview of the weathering effect developed using OpenGL. The preview will allow users to set some parameters, such as the amount of γ-tons to emit from each emitter, the strength of gravity, the levels of humidity and dirt per γ-ton etc. It will also attempt to gain a speed up in γ-ton tracing by using parallelism techniques. Lastly a point emitter will not be implemented but the source code will be easily extensible so it can be added in the future.

4.2 Future Considerations

The core system has been explained. The simulation section is more understood than the texture synthesis section. Therefore more research will be required in this area before implementation. The ideas proposed will work but will be very compute intensive. To reduce the compute time an alternative method will have to be derived.

γ-ton processing is also compute intensive. A solution to achieve speed up will be required. The solution provided in the previous section will provide efficient speed up. This can be implemented using OpenMP [15] or OpenMPI [16].

Bibliography

[1] Carles Bosch et al, "A Physically-Based Model for Rendering Realistic Scratches", *Computer Graphics Forum*, Volume 23 2004, 361-370.

[2] James F. Blinn, "Light reflection functions for simulation of clouds and dusty surfaces", *In Proceedings SIGGRAPH '82*, SIGGRAPH 1982, 21-29.

[3] Yanyun Chen et al, "Visual Simulation of Weathering By γ-ton Tracing", *ACM Transactions on Graphics*, SIGGRAPH 2005, 1127-1133.

[4] Martin Davies, "Particle Based Weathering System Design", unpublished, 2014.

[5] Brett Desbenoit, Eric Galin and Samir Akkouche, "Simulating and modeling lichen growth", *Computer Graphics Forum*, Volume 23, 2004, 341-350.

[6] Julie Dorsey and Pat Hanrahan, "Modeling and Rendering of Metallic Patinas", *In Proceedings SIGGRAPH '96*, SIGGRAPH 1996, 387-396.

[7] Julie Dorsey, Hans Køhlig Pedersen and Pat Hanrahan, "Flow and Changes in Appearance", *In Proceedings SIGGRAPH '96*, SIGGRAPH 1996, 411-420.

[8] Julie Dorsey et al, "Modeling and Rendering of Weathered Stone", *In Proceedings SIGGRAPH '99*, SIGGRAPH 1999, 225-234.

[9] Jinwei Gu et al, "Time-Varying Surface Appearance: Acquistion, Modeling and Rendering", *ACM Transactions on Graphics*, SIGGRAPH 2006, 762-771.

[10] Tobias Günther, Kai Rohmer and Thorsten Gorsch, "GPU-accelerated Interactive Material Aging", *In Proceedings VMV*, Vision, Modelling Visualization 2012, 63-70.

[11] Stephane Merillou, Jean-Michel Dischler and Djamchid Ghazanfarpour, "Corrosion: Simulating and Rendering", *In Proceedings Graphics Interface '01*, GI 2001, 167-174.

[12] Director : Nicholas Meyer, Star Trek II: Wrath of Khan, 1982. USA: Paramount Pictures.

[13] Tomas Möller and Ben Trumbore, "Fast, Minimum Storage Ray/Triangle Intersection", *J. Graph. Tools*, Volume 2, 1997, 21-28.

[14] Andrew W. Moore. (1991). *An introductory tutorial on kd-trees* [Online]. Available : http://tinyurl.com/cja9o9

[15] OpenMP Architecture Review Board. *The OpenMP API Specification for Parallel Programming* [Online]. Available : http://openmp.org/wp/

[16] The OpenMPI Project (2004). *OpenMPI : Open Source High Performance Computing* [Online]. Available : http://www.open-mpi.org

[17] Eric Paquette, Pierre Poulin and George Drettakis, "Surface Aging by Impacts", *In Proceedings Graphics Interface*, GI 2001, 175-182.

[18] Dan Phaneuf. (2012). *Halton Sequences* [Online]. Available : http://tinyurl.com/njj46ns

[19] Thomas Porter and Tom Duff, "Compositing Digital Images", *In Proceedings SIGGRAPH '84*, SIGGRAPH 1984, 253-259.

[20] Magdi Ragheb. (2013, 17 March). *Russian Roulette and Particle Splitting* [Online]. Available : http://tinyurl.com/qde6t7r

[21] William T. Reeves, "Particle Systems - a Technique for Modeling a Class of Fuzzy Objects", *ACM Transactions on Graphics*, SIGGRAPH 1983, 91-108.

[22] William T. Reeves and Ricki Blau, "Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems", *In Proceedings SIGGRAPH '85*, SIGGRAPH 1985, 313-322.

[23] Andrew Selle, Nick Rasmussen, Ronald Fediw, "A vortex particle method for smoke, water and explosions", *ACM Transactions on Graphics*, SIGGRAPH 2005, 910-914.

[24] Alexey Stomakhin et al, "A material point method for snow simulation", *ACM Transactions on Graphics*, SIGGRAPH 2013, Article 102, 1-10.

[25] Subhash Suri. *Polygon Triangulation* [Online]. Available : https://www.cs.ucsb.edu/ suri/cs235/Triangulation.pdf