# Real-time Particle Interaction with Relief-Surfaces

Ryan Bergeron
University of Maryland Baltimore County

**Abstract**

In this paper we present a method for simulating erosion of relief surfaces. Our method uses graphics hardware to simulate particle interaction with relief surfaces in real-time. Our system is designed to allow users to simulate the aging of surfaces from various environmental elements.

## 1 Introduction

Time is a constant with an inevitable affect on all things. Various surfaces can be exposed to numerous environmental conditions including sunlight, rain, snow and wind. These naturally occurring circumstances act synergistically with time, leaving an indication of their presence. A solid surface such as nickel may show erosion marks from rain and water, lines from prolonged exposure to wind and etching from acid rain. These surface aging signs are important environmental indicators for the presence of pollution and natural weathering.

Over the lifetime of a surface exposure to elements create what is known as deposition processes. This development manifests itself in the form of household dust clumps, mounts of dirt, pollen on a wind shield, even snow on our lawns. These are all visible signs of deposition over a surface's lifetime. Although the accumulation of a different substance on a surface may change its original properties, its existence is too ephemeral to affect it in a manner that chronologically ages it. Deposition processes, like the accumulation of pollen or snow, are important in understanding the trends of specific environmental fluctuations such as global wind patterns. Since it is possible to measure the magnitude of different erosion and deposition processes, helpful guidelines for creating new surfaces and materials can be devised by understanding how they will react to the aging effects encountered throughout their lifetime. Unfortunately, measuring these effects is very time consuming, resulting in months or even years of diligent observation to witness any change in the surface.

In response to the difficulties experienced in recording and documenting deposition processes, this paper proposes the creation of a system that would allow us to simulate the aging affects of different weathering processes by incorporating current real-time surface representation techniques with a real-time particle system that will create a simple weathered surface.

## 2 Previous Works

This project was based off of three key elements, Surface Deformation, Particle Systems, and Surface Weathering. In the remainder of this section we will explain methods that have been created relating to our project.

### 2.1 Surface Deformation

We need a way to change the structure of a surface. Most current methods of changing the structure of a surface require geometry editing. Surfaces can be created through triangles connected into meshes of differing complexity [Borouchaki et al. 1997]. Larboulette and Cani created a simulator to create wrinkled surfaces with geometry [2004]. The difficulty in creating a realistic surface through straight geometry is the need for high polygon count models.

Another way of creating a realistic surface is through manipulating the surface pixels to create a pseudo distorted surface. Bump mapping is a method that creates a complicated surface by manipulating the surface normals per pixel [Blinn 1978]. Displacement mapping is a similar method to create complex surface structures [Kautz and Seidel 2001][Hirche et al. 2004]. Relief mapping is a real-time method for rendering height-displacement surfaces [Oliveira et al. 2000][Oliveira et al. 2005]. This method uses a height specified at each pixel to create realistic surfaces. We have chosen to create our surface with relief mapping for two reasons. First, relief mapping creates a detailed surface structure that is less view dependant that both displacement and bump mapping. Secondly, our particle system we have chosen to implement is in similar space and will require less effort to compute intersections.

Now that we know how to create surfaces, we need to know how to deform a surface. Paquette et al. used the collision of geometry and then changed the location of the vertices that were in the collision zone in their hail simulator [2001]. Summer and O'Brien deformed the heights in a height map then created the geometry based on the modified heights [1999]. The way they modify the height field is similar to our system. However in our system, there will be no geometry created since we will be working on a relief surface.

### 2.2 Particle System

Simulating environmental elements, we have chosen to use a particle system. Particle systems were first introduced by Reeves as a method for creating fuzzy objects [1983]. After particle systems were introduced, researchers wanted to be able to work with particles more realistically, and added functionality such as particle-particle, particle-object collisions [Reynolds 1987]. More recent efforts to create more manageable particle systems have turned to graphics hardware for accelerated render times. Knott and van den Doel used hardware to detect collisions in particle systems [2003]. The UberFlow system uses textures to hold both position and velocity [Kipfer et al. 2004]. During rendering it uses a vertex shader to assign positions to particles and a fragment shader to update the position and velocity of a particle in texture space. This system handles all types of particle collisions. Our project will use the same manner of

controlling particle motion in texture space. Other system have focused on managing large particle systems in hardware [Kolb et al. 2004]. These systems use the same method of creation and movement of particles as UberFlow but are scaled for larger particle sets.

### 2.3 Surface Weathering

Aging a surface requires methods to simulate weather on a surface. Our system is looking to erode a complicated surface. Erosion models have been created for runoff at a geometry level. Musgrave et al. wrote a system that handled run off or water to create a eroded surface [1989]. This is similar to our work, but is on a large scale erosion model. We are more concerned with how the element will interact with the surface. "Flow and Changes in Appearance" created a system that treats particles as water drops and managed the chemical interaction of water and surfaces [Dorsey et al. 1996]. This is similar to what we have planned to do, but for this project we are more concerned with particle-relief surface interaction and not the chemical properties leading to the change in surfaces. Dorsey as worked on geometry based slice model for creating weathered stone surfaces [Dorsey et al. 1999]. This is a geometry model based aging of a surface. Dorsey wrote another paper about rendering wet materials that focused on the visual effects water has on viewing surfaces [Dorsey et al. 1999].

### 2.4 Other Related Works

Oka et al. presented work on manipulating surfaces while keeping textures meaningful and correctly aligned on that surface [1987]. This is relevant to previous methods because when editing geometry, a surface will still need to maintain the information mapped to the vertexes.

Since the relief mapping method uses a linear ray trace from the users view into the relief-space, it is appropriate to include work in ray tracing. Current methods of ray tracing have moved onto hardware accelerated ray tracing of a scene. The GPU has been used for this performance gain [Purcell et al. 2002]. The latest movement in real-time ray tracing is to create a RPU or ray tracing processor unit so that a user can use ray tracing that is built into the hardware [Woop et al. 2005].

## 3. Method

Our method has been broken down into six steps. Steps one through three are particle related. Then we age the surface, followed by rendering steps five and six.

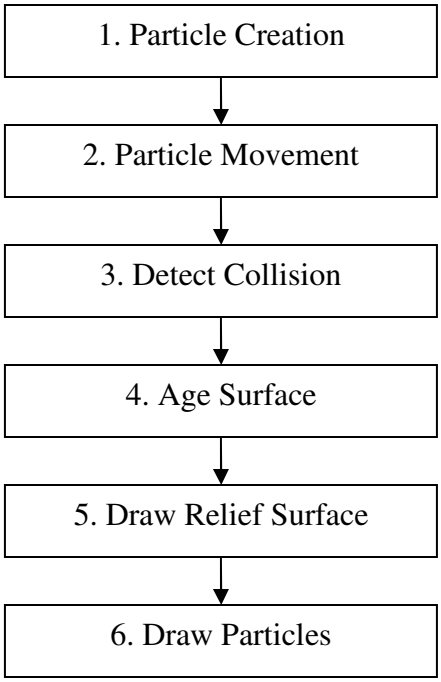


Figure 1: This is the flow chart of our system that occur per-frame before rendering the final scene.

### 3.1 Particle Creation

Our particle system begins when a particle in world space hits the relief surface (Figure 4a). From there, particles are added to the system at the location of impact. Because this specific process was not in this project's scope, it will be further defined in future work. Our system manages the particle when it's ready to move into relief space. The particle's position and age is represented as color in the particle position texture of our system (Figure 3). A particles velocity will be stored in the velocity vector texture. The particle is assigned a texture value that corresponds to both these textures. This is the general layout of the UberFlow particle system.

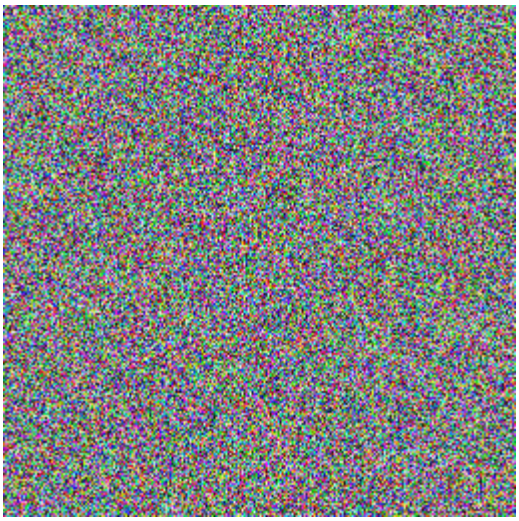

Figure 2: This is a sample initial particle position texture. Each texel can be assigned to a particle.

### 3.2 Particle Movement

Now that we have our particles locations and velocity vectors, we perform need to move out particles location in the relief space. To compute the new location of a particle, we use a simple physics calculation.

$$P_f(s,t) = P_o(s,t) + V_o(s,t)T + \frac{1}{2}gT^2$$

Po(s,t) is the x, y, z position of the particle stored in our texture that contains particle positions. Figure 4c is a trace of a particle in the relief texture space. Vo(s,t) is the velocity vector stored in our texture that holds the velocity vectors. T is the age of our particle in the scene which is stored in the alpha channel of our position texture. We then have to update our particle velocity vector such that,

$$V_f(s,t) = V_o(s,t)T + gT$$

Once we have moved the particle, we need to age it as well.

$$T_f(s,t).a = T_o(s,t) + .1$$

$$P_f(s,t).a = T_f(s,t).a$$

$$T_o(s,t) = P_o(s,t).a$$

If a particle has reached a certain age, we will kill the particle so that a new particle can be created. We will need to copy the output from the frame buffer into the new velocity vector texture. To save on execution time, we used a frame buffer that will allow us to swap render targets faster and have their resulting outputs stored directly to textures. This work is done in a fragment shader.

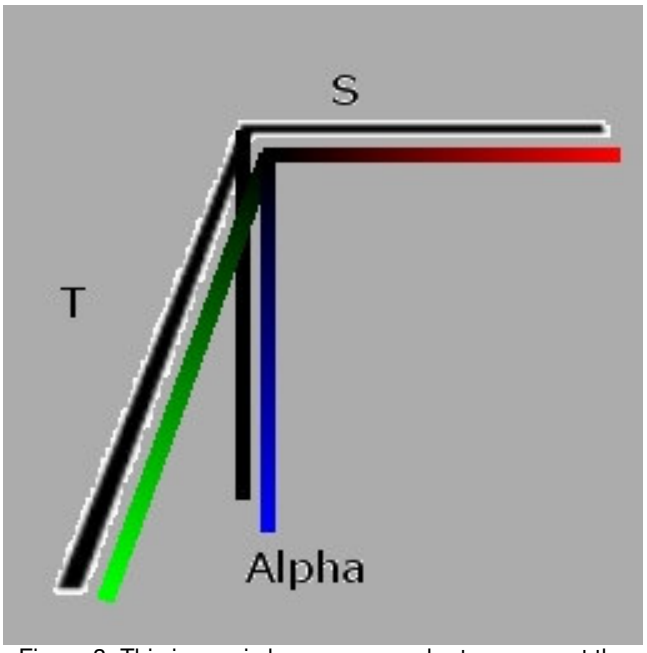


Figure 3: This image is how we use color to represent the position of a particle in relief space. Relief mapping uses the alpha value to determine the depth of the surface and our system uses blue for the depth of a particle. S and T refer to the position of a particle on screen in our relief space (X,Y) and alpha is the depth.

### 3.3 Detect Collision

Once we have moved our particles to their new positions, we need to compare their location to the depth map off the relief surface. To do this we compare if the z coordinate at an (x,y) position of the particle against the Depth(x,y) of the relief map. If z is greater than or equal to the depth of the relief surface then the particle has collided with the relief surface. Once a particle has collided with we will set its age to a terminal value so that we know at what places particles have collided with the relief surface. The frame buffer object then stores the new texel values into an update position texture. This is the final step in our fragment shader outlined in our Particle Movement Section.

### 3.4 Age Surface

In the scope of this project we were concerned with creating impact zone on the relief surface at locations where a particle has collided. To accomplish this, we will look through our particle position texture for ages with the terminal value we set. A particle that has hit the relief surface, we will increase depth of that texel value by a scalar value. We have written a fragment shader that is run over a relief surface. This scalar value in our results is the setting to maximum depth. We copy the output from the frame buffer into a new depth map for the relief surface.

### 3.5 Draw Relief Surface

Once all the particle moving and aging of surface has been finished, we will draw our relief surface in view space using the same method as described in “Real-Time on Arbitrary

Polygonal Surfaces" using our new depth map that we created in our aging surface step [Oliveira et al 2005].

### 3.6 Draw Particles

The final step of our system is to draw particles. In this stage of rendering, we will draw our particles. Given a list of particles in our scene, we have constructed a vertex shader that will assign the position in world space the values of the particles in relief-texture space. This will require use to convert the x, y, z position from relief-texture space to world space. Any particles that have hit the relief surface and have the age terminal value will not be drawn. This is will allow us to see particles moving in the view space as if it were in the relief space.

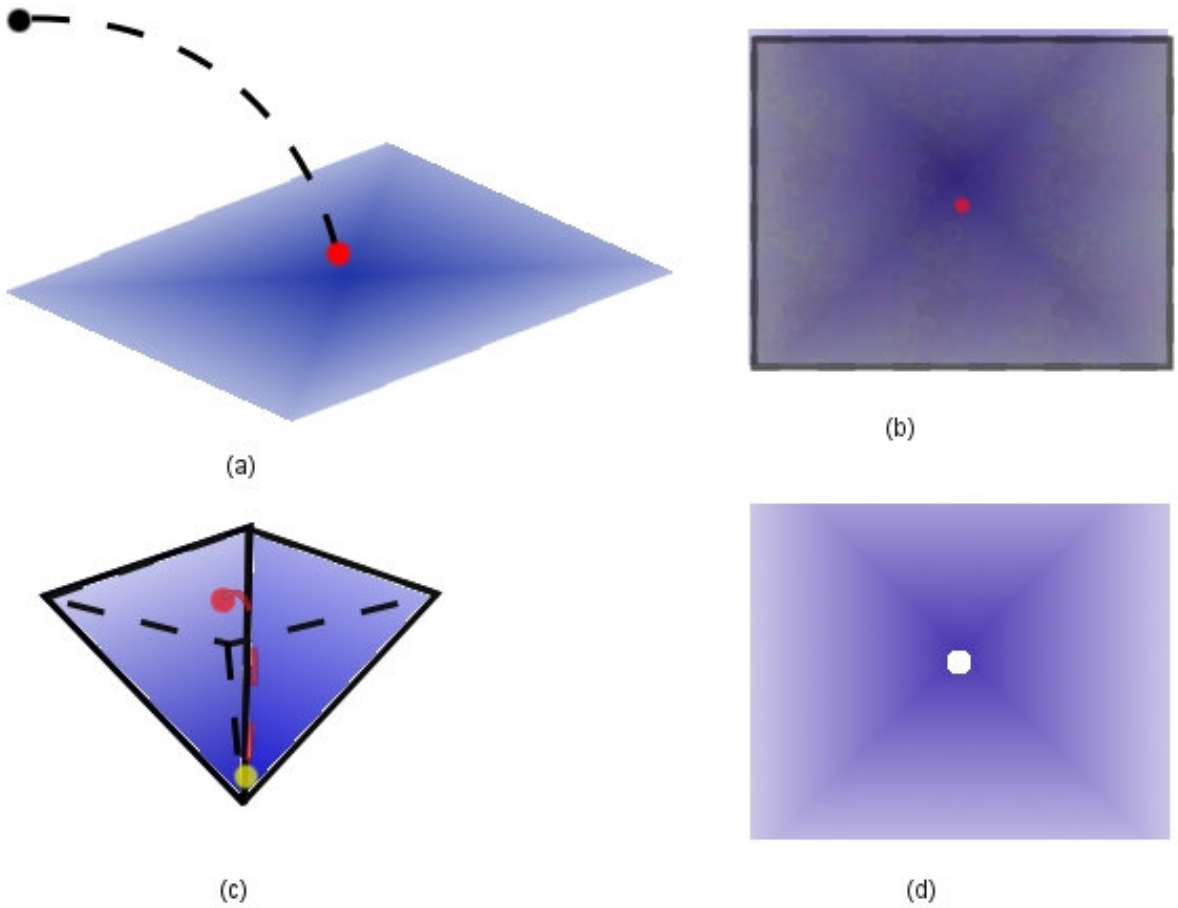


Figure 4: (a) Refers to a particle moving in world space before it intersects with our relief mapped surface. (b) The red is the location in texture space at which the particle hits our polygon. This is where our particle system begins. (c) In this image we track the movement of our particle inside the relief surface. The depth in relief space is related to the transparency of the blue. The yellow refers to the location of impact. (d) This is the result of a particle hitting the relief surface.

### 4. Results

We have implemented the relief texture mapping code as described in section 3.5 Draw Relief Surface. The fragment shader was written as laid out in Oliveira's work [2005]. We added a vertex shader that will pass information into the fragment shader as well as set up the tangent and binormal vectors.

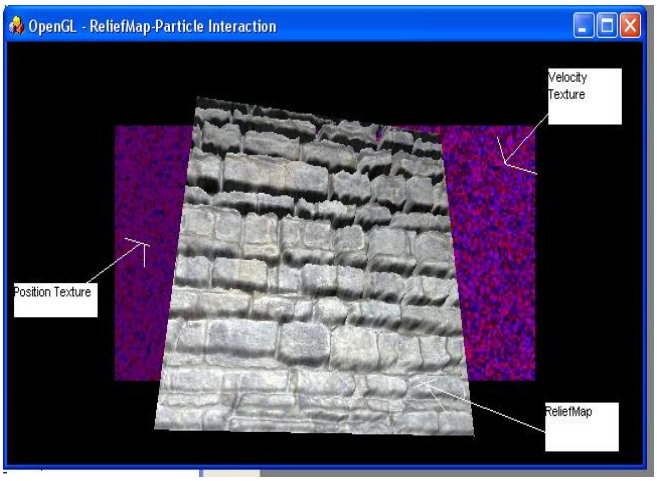


Figure 5: This shows three stages of our output. The relief surface is indicated with the reliefMap label. The Velocity Texture label is the output of Particle Movement faze, with the new velocity vector. Position Texture is the output of the particle position texture.

We implemented our Particle Movement method using one frame buffer object writing off of sample code found at http://www.gpgpu.org/developer/. We used their framebufferobject and renderbuffer object code along with our fragment shaders and created updated particle position and velocity vector textures per frame.

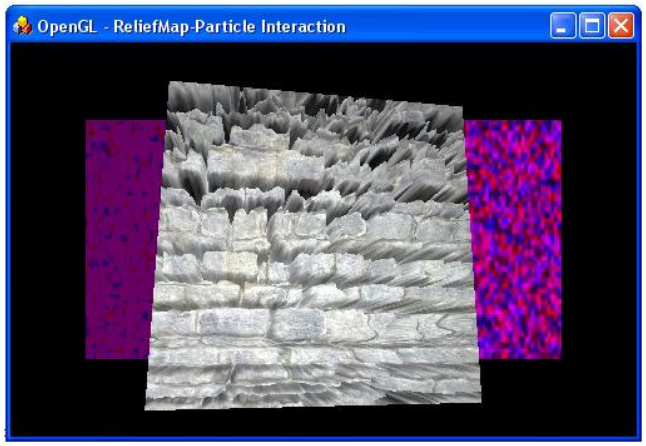


Figure 6: This is the same image in as in Figure 5 but one time step in the future. We can see both the particle position and the velocity vector texture have changed..

In Figure 6 if you will notice that the amount of green in each texel has decreased in the green channel from Figure 5. This is what is expected from our updating of our two textures since the green channel is related to our Y position and gravity reduces the Y position and velocity in a negative factor.

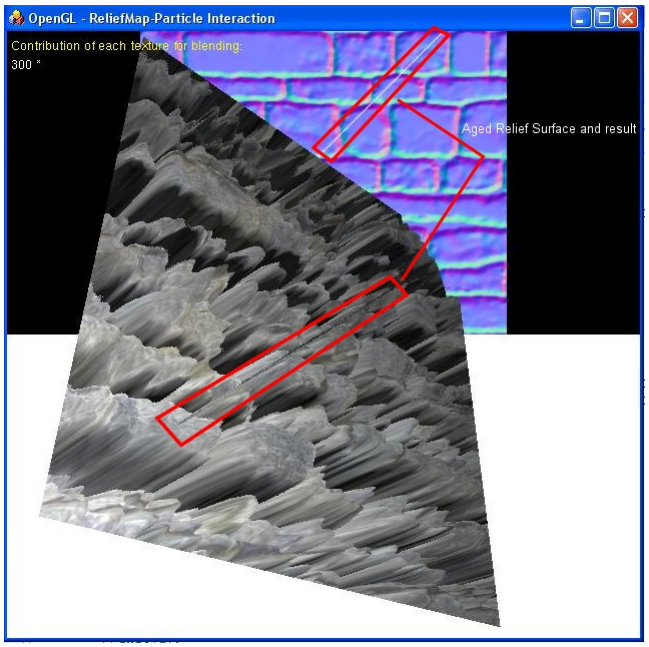


Figure 7: This is image shows both the relief map after aging and the resulting relief surface. The changes are noted in the red boxes.

In Figure 7 you can see that we age our surface by assigning the color of the texel to white and alpha value to 1. We do this so that we can watch the collision zones and see the change in the relief surface immediately. The lower red box is where the relief surface has changed. The speed at which this occurs Is interactive, but we need to control it at a frame per frame basis so that we can see the results of this project. We believe that we will be able to attain real-time results in later work.

Our method for drawing the particles did not work as planned, and we were not able to draw the particles using the vertex shader. We believe that adding a method to our Draw Relief Surface shader to allow for drawing a particle into our relief space.

## 5. Future Works

In our particle creation section we said that we needed a method to convert the position and velocity of a particle in world space to a particle in our relief space or the polygon.

Our aging surface performance is directly dependant on the number of particles in our relief space. Since we look at every texel in our relief map and need to find any particle that hit our surface, a large number of particles would result in slow run times.

Another area of future work would be applying erosion models to create more realistic visual surface interaction. This will involve changing the relief map color values so that the normals reflect how the surface was changed.

We mentioned in the results section needing a method to allow us to draw a particle that is in relief space. This still needs to be done.

## 6. References


J. F. Blinn. “Simulation of wrinkled surfaces”. In Proceedings of SIGGRAPH’78, pages 286–292, 1978.

Borouchaki, H., Hecht, F., and Frey, P. J. “Mesh Gradation Control.” *In Proceedings of 6th International Meshing Roundtable,* Sandia National Labs (oct 1997), pp. 131-141.

J. Dorsey, A. Edelman, J. Legakis, H. W. Jensen, and H. K. Pedersen. Modeling and Rendering of Weathered Stone. In Computer Graphics, SIGGRAPH ’99 Proceedings, pages 225–234. Los Angeles, CA, August 1999.

J. Dorsey , H. W. Jensen, J. Legakis. “Rendering of Wet Materials.” In Eurographics Rendering Workshop, (1999).

Dorsey, J., Pedersen, H. K., and Hanrahan, P. 1996. “Flow and changes in appearance.” In *Proceedings of the 23rd Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '96. ACM Press, New York, NY, 411-420.

Hirche, J., Ehlert, A., Guthe, S., and Doggett, M. 2004. Hardware accelerated per-pixel displacement mapping. In *Proceedings of the 2004 Conference on Graphics interface* (London, Ontario, Canada, May 17 - 19, 2004). ACM International Conference Proceeding Series, vol. 62. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, 153-158.

Jan Kautz and Hans-Peter Seidel. “Hardware accelerated displacement mapping for image based rendereing.” In Graphics Interface, pp 61-70, (2001).

D. Knott, K. van den Doel, D. K. Pai. “Particle System Collision Detection using Graphics Hardware.” ACM SIGGRAPH, p 1, (2003).

C. Larboulette and M. Cani. “Real-Time Dynamic Wrinkles.” In Proceedings of Computer Graphics Inter-
national’04, IEEE Computer Society Press, 522–525.

F. K. Musgrave, C. E. Kolb, R. S. Mace. “The Synthesis and Rendering of Eroded Fractal Terrains”, ACM SIGGRAPH ’89 Computer Graphics, Volume 23, (1989)

Oka, M., Tsutsui, K., Ohba, A., Kurauchi, Y., and Tago, T. 1987. “Real-time manipulation of texture-mapped surfaces.” In *Proceedings of the 14th Annual Conference on Computer Graphics and interactive Techniques* M. C. Stone, Ed. SIGGRAPH '87. ACM Press, New York, NY, 181-188.

Oliveira, M. M., Policarpo, F., and Comba, J. L. 2005. “Real-time relief mapping on arbitrary polygonal surfaces.” *ACM Trans. Graph.* 24, 3 (Jul. 2005), 935-935.

Oliveira, M. M., Bishop, G., and McAllister, D. 2000. “Relief texture mapping.” In *Proceedings of the 27th Annual Conference on Computer Graphics and interactive Techniques* International Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., New York, NY, 359-368.

E. Paquette, P. Poulin, G. Drettakis. “Surface aging by impacts.” Canadian Information Processing Society, pp 175-182, (2001).

Reynolds, C. W. 1987. “Flocks, herds and schools: A distributed behavioral model.” In *Proceedings of the 14th Annual Conference on Computer Graphics and interactive Techniques* M. C. Stone, Ed. SIGGRAPH '87. ACM Press, New York, NY, 25-34.

R. Sumner, J. O’Brien, and J. Hodgins. “Animating Sand, Mud and Snow.” *In Proceedings of Graphics Interface*, pages 125–132. Canadian Information Processing Society, 1998.

Woop, S., Schmittler, J., and Slusallek, P. “2005. RPU: a programmable ray processing unit for realtime ray tracing.” *ACM Trans. Graph.* 24, 3 (Jul. 2005), 434-444.