Interactive Weathering of Materials

Timothy Hoff April 26, 2007

1 Introduction

Texture mapping, the use of two-dimensional images to alter the materials of 3D models, is widely used in real-time computer graphics today. Texture mapping techniques such as specular, normal[1], and ambient occlusion[2], add a great deal of realism to 3D models and are viable for real-time applications. It is not commonly considered, however, how the material should be affected by the environment. Some modern video games use scripted changes in materials to simulate these effects. For example, Tomb Raider: Legend[3] makes the main character's clothes look soggy whenever it gets out of water, and the Colin McRae Rally games[4] show dirt accumulation on the bodies of cars as they drive off-road. While the images produced were high quality, they were not very realistic or interactive because they would look the same every time. This technique seeks to introduce a technique that combines the simulation of environmental effects such as rain, snow, and dirt with the real-time rendering of texture maps to simulate realistic weathering of materials.

2 Technique

The technique developed can be broken down into two main components: environmental effect simulation and material weathering.

2.1 Environmental Effect Simulation

The simulation of environmental effects (such as rain and dust) in this technique is accomplished using particle systems[5]. The particles are treated as points in space and are drawn using textures maps. The particles are emitted into the scene at a certain rate and persist until they either hit a model or grow older than their specified lifetime. Various forces can be applied to the particles as long as the position of each is known.

2.2 Material Weathering

The weathering of the material on each model in the scene can be broken down into two processes: collision detection and overlay rendering.

2.2.1 Collision Detection

The process of detecting a collision between a particle and the model incurs the majority of the cost of this technique. In the worst case, the collision detection algorithm runs in $O(mvp)$ time where m is the number of models in the scene, v is the number of vertices in the current model, and p is the number of particles in the scene.

The following pseudo-code describes the collision detection algorithm:

```
For each model M in the scene
{
  Initialize a list of collision texture coordinates L_C
  For each particle P in the scene
  {
    If P is outside the bounding box of M, continue
    For each vertex V_I in M
      If V_I is the closest vertex to P, store it as V
    If V is outside of M, continue
    Initialize a list of triangles L_T
    For each triangle T in M
      If T contains V, add it to L_T
    For each triangle T in L_T
    {
      Cast a ray from V in the normal direction to T
      If ray intersects T at barycentric coordinates C_B
      {
        Using C_B, interpolate the texture coordinates
          of T and add the result to L_C
        Remove P from the scene
      }
    }
  }
}
```

(Figure 1 – collision detection algorithm)

Once the lists of collision texture coordinates are generated, they are used to render the texture overlays.

2.2.2 Overlay Rendering

To help describe the overlay rendering process, the weathering material that is applied to the models in the scene must be defined. Six texture maps comprise the weathering material. The first two, diffuse and diffuse overlay, define the colors of the material. The next two, specular and specular overlay, define the regions that are shiny or matte. The last two, normal and normal overlay, perturb the normals of the model in certain regions to give the illusion of the extra geometry. The overlay textures are initialized to be completely transparent at the beginning of the simulation and further drawn upon. During the rendering process, the overlays are layered on top of their corresponding texture map to imitate weathering of the material. This material can be implemented using vertex and pixel shaders for real-time applications.

Along with the texture maps required by the material, three more are necessary for each type of environmental effect. These textures are prepared beforehand and will be drawn on the corresponding overlays of the material. As such, diffuse, specular, and normal texture maps are required for each type of environmental effect in the scene.

Given the texture coordinates output by the collision detection algorithm, the model painted with the material described above, and the environmental effect texture maps, the process of overlay rendering begins. Rendering each overlay proceeds in the same fashion. First, the overlay texture is set as the render target. Second, the environmental effect texture map corresponding to the current overlay is drawn, centered about each texture coordinate. At the user's request, if a texture is partially drawn out of bounds, it is drawn a second time wrapped around to the other side of the overlay. This ensures that the material is seamless when it is wrapped around a model. Finally, the render target is reset and the overlay is sent to the material.

3 Results

The simulation was written in C# using Microsoft's XNA Framework 1.0, and the material was written in HLSL. The results are based on the application running on a Pentium D PC running at 3.0 GHz with 2 GB of RAM and a GeForce 7900 GT GPU.

The models used in the test scenes have the following complexity:

Model	Vertices	Triangles
Ground	926	1584
Sphere	539	936
Bunny	1585	1000
Vase	5883	11008

(Figure 2 – model complexity)

The first test scene (shown in Figure 4) contained the ground, sphere, and bunny models (totaling 3,050 vertices) and a hose particle system spraying particles across the scene. The second scene (shown in Figure 5) contained the ground and vase models (totaling 6,809 vertices) and the same hose particle system as the first scene. The performance of both scenes at varying particle emission rates (in particles introduced per second) is shown below:

Scene/Particles	100	250	500	1000
1 (3,050 vert.)	427	402	4	2
2 (6,809 vert.)	442	47	2	1

(Figure 3 – simulation performance in frames per second at varying particle emission rates)

4 Conclusion

With the results of the performance of this technique in mind, it is safe to say that it is not ready for use in a real-time application with even a moderately high geometric complexity. The collision detection algorithm must be dramatically optimized for any large-scale scenes to be viable. Visually, however, the technique produces good results. The realism, however, is directly related to the quality of the environmental effect textures. Those produced for this implementation need more detail. Another limiting factor of this technique is the requirement that textures cannot be tiled. The UV coordinates of each vertex must be in the range [0..1]. This increases the memory requirement of the application in most cases. With some optimizations and added flexibility, however, this technique could become practical for video games and simulations in the near future. Even if this is not possible, it would be a step in the right direction for realism in interactive 3D graphics.

5 References

[1] "Normal mapping." Wikipedia, The Free Encyclopedia. 23 Apr 2007, 05:04 UTC. Wikimedia Foundation, Inc. 25 Apr 2007 <http://en.wikipedia.org/w/index.php?title=Normal_mapping&oldid=125074453>.

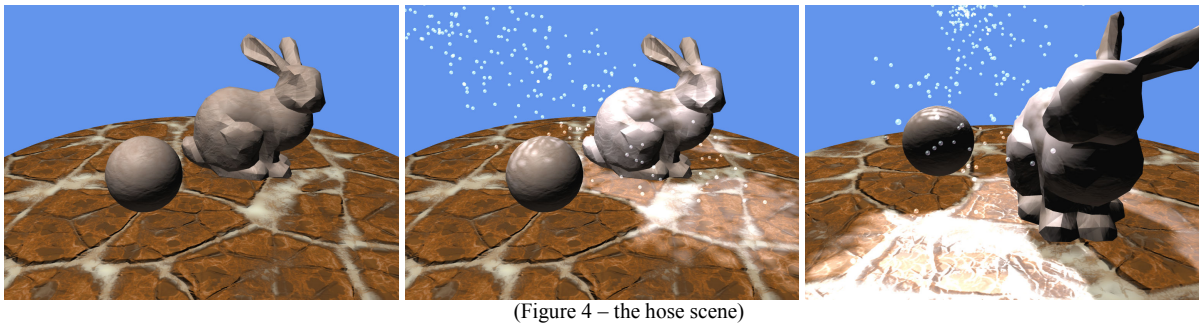
[2] "Ambient occlusion." Wikipedia, The Free Encyclopedia. 9 Jan 2007, 22:09 UTC. Wikimedia Foundation, Inc. 25 Apr 2007 <http://en.wikipedia.org/w/index.php?title=Ambient_occlusion&oldid=99632736>.

[3] "Lara Croft Tomb Raider: Legend." Wikipedia, The Free Encyclopedia. 23 Apr 2007, 18:56 UTC. Wikimedia Foundation, Inc. 25 Apr 2007 <http://en.wikipedia.org/w/index.php?title=Lara_Croft_Tomb_Raider:_Legend&oldid=125252015>.

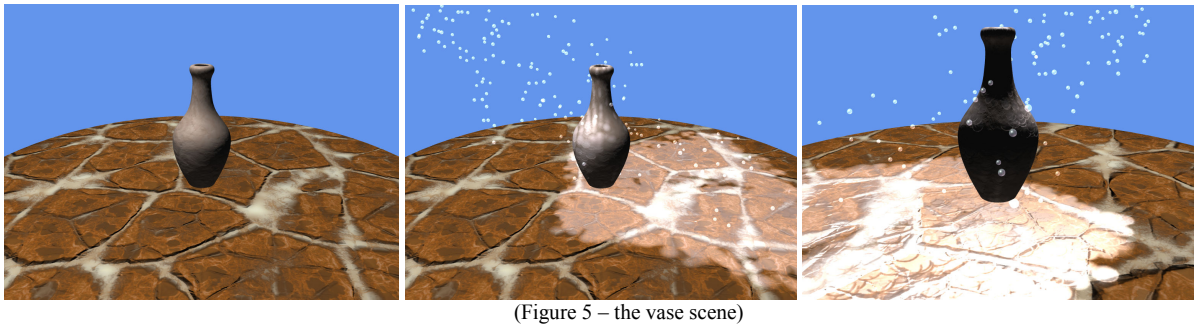
[4] "Colin McRae Rally." Wikipedia, The Free Encyclopedia. 19 Apr 2007, 15:05 UTC. Wikimedia Foundation, Inc. 25 Apr 2007 <http://en.wikipedia.org/w/index.php?title=Colin_McRae_Rally&oldid=124097599>.

[5] "Particle system." Wikipedia, The Free Encyclopedia. 18 Apr 2007, 03:00 UTC. Wikimedia Foundation, Inc. 25 Apr 2007 <http://en.wikipedia.org/w/index.php?title=Particle_system&oldid=123711893>.

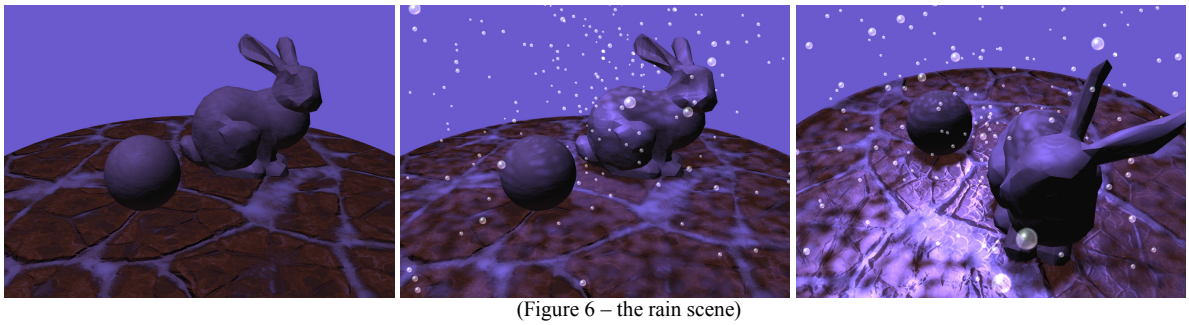
6 Screenshots



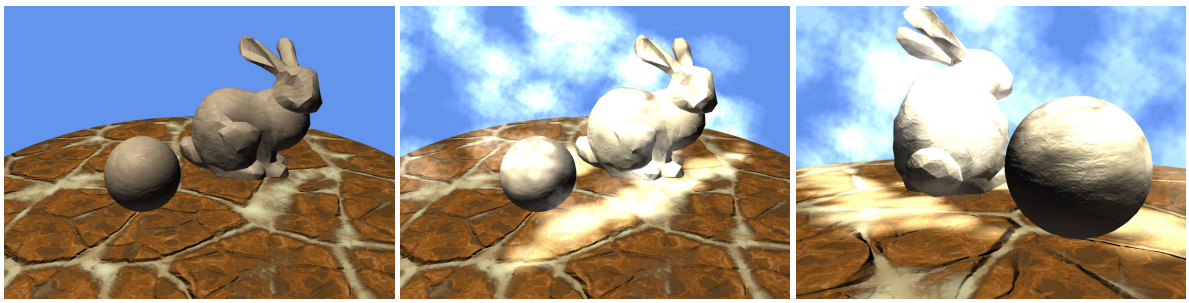
(Figure 4 – the hose scene)



(Figure 5 – the vase scene)



(Figure 6 – the rain scene)



(Figure 7 – the dust scene)