SIMULATION OF PRECIPITATION AND SNOW COVER IN THE JASKIER IMAGE GENERATOR

Abstract. The article presents methods of implementing atmospheric precipitation and snow cover using techniques of modern computer graphics. Methods applied in the Jaskier image generator and problems of image continuity in multiple display monitors are discussed. The results obtained are discussed in the summary.

Keywords: JASKIER simulator, Jaskier IG, simulation of atmospheric precipitation, 3d graphics, image processing, particle effects.

1. INTRODUCTION

In the case of computer graphics in its broad meaning and wide application (games, simulations, video), there is no single method that would provide satisfactory visual effect of a phenomenon or of a physical process. Usually various techniques are applied and the results of these are combined at the final stage. This is also the case with the Jaskier image generator of the JASKIER simulator [1], where every weather phenomenon is accomplished using at least three techniques. These mostly include particle effects, dynamic materials (snow cover, diaphragm) and post processing. The graphic effects described below were developed at OBRUM and applied in the JASKIER simulator designed for training in operating the ROSOMAK wheeled armoured vehicle [2].

2. DYNAMIC SNOW COVER

Dynamic snow cover with the procedural effect of destroying vehicles is one of the more complex system implemented on the Jaskier IG. While the process itself, which comprises adding snow texture to the materials\(^1\) on the scene, seems to be a simple task, implementing this process in the visualization engine used required more practical experience in design work.

Every material may have up to a dozen or more textures, and it is not always clear which texture is responsible for colour, and which, for instance, for normal mapping [3] – texture indexing or the lack of it depends on engine architecture. The method applied requires information about colour textures. It therefore proved necessary to manually index textures in every material. For this purpose a variable indicating the colour texture was created in the materials. The advantage of this solution was better control over the materials and the ability to indicate materials which should have a snow cover, and those which should not.

In simple terms, the dynamic snow cover is an additional material that has a colour texture (texture index derived from the "parent" material), a snow texture and a texture responsible for destruction animation (scorching with alpha channel). Fig. 1 shows the effect of snow cover for various values of snowfall.

\(^1\) Every object has a material which describes the shell of the object. The material may comprise a plurality of textures (e.g. colour texture, reflected light texture) and may be associated with a particular shader by means of which more complex material modelling is effected (e.g. parallax mapping [4]).
Fig. 1. Dynamic snow cover over vegetation (example)

Fig. 2 shows snow cover with a procedural destruction effect. The effect of destruction is implemented on the material that is responsible for the snow cover, therefore it enables gradual removal of the snow texture with advancing deterioration of the vehicle (temperature rise simulation). Additionally use is made of vertex deformation (vertex shading [4]) and scorch texture with alpha channel.

Fig. 2. The effect of vehicle deterioration illustrated with Bergenpanzer 2 vehicle

3. PARTICLE EFFECTS

In the case of an image generator with combined display monitors (side by side) served by multiple instances\(^2\) of the same application, it is important that the movement and appearance of particles is not defined at random, i.e. the image is seamless (continuous). Fig. 3 shows the effect of a snow storm generated using fixed parameters, which ensures identical effects in various instances (processes). Fig. 4 shows the same effect generated using parameters with random values.

---

\(^2\) Instance of a program – that is an independent occurrence of a process in computer memory (one application can have multiple instances in a system).
4. WEATHER PHENOMENA ON THE VEHICLE WINDSHIELD

Diaphragm, that is the vehicle windshield, should generate an image different from the one seen by a free observer located outside the vehicle. In order to simulate raindrops and snowflakes on the windshield, a material with a specially developed shader\(^3\) was prepared (Fig. 5).

In addition to the effects mentioned, the shader may also simulate the action of a windshield wiper. In this case time synchronization between the instances of the application is crucial, and this is provided by the parameter defining the simulation step transferred by a simulator. In each instance the shader receives the same value, which facilitates the synchronization of the movement of the wipers on all display monitors (the same principle applies to all effects that change with time).

---

3 Shader languages have no global variables. In every frame of a rendered scene, shader is effected in a manner in which it was designed (model). However, initial parameters may be assigned to it. Such a parameter may be transferred to the shader program from outside, e.g. from the native system code (environment), and it may dynamically affect the results generated by the shader.
Michał BUGAŁA

5. CLOUD COVER SIMULATION

Most of the graphic engines are provided with integrated cloud generators (Fig. 6). The use of a commercially available solution significantly facilitated the implementation of atmospheric clouds, which were supplemented with a particle system to render lower layers of clouds.

Fig. 6. Example of clouds generated in Unreal Engine
6. POST-PROCESSING

Besides the lack of randomness and the presence of the parameter providing time synchronization, it is also important that visual effects, including post-processing, do not form stitched edges and provide seamless results, i.e. the effect generated on one display monitor does not produce different results on the second display monitor. Such effects of image processing as tone mapping, vignetting and lens flares must not participate in the final rendering (Fig. 7).

![Fig. 7. Sandstorm on five display monitors (lack of seam and edge effects yields image continuity)](image1)

Post-processing, that is modification of the image upon rendering the whole scene (frame), enables enhancing visualization with additional effects. Depending on the desired weather conditions, these might include: water droplets running down, frost, sand (sand storm), etc. Fig. 8 presents the phenomenon of refraction which was utilized in the Jaskier IG.

![Fig. 8. Refraction used for simulating frost in the top part of the image and animated water droplets over the entire image](image2)
7. FINAL EFFECT

Figs. 9 and 10 show examples of screenshots from the Jaskier IG image generator, illustrating the combined methods (described above) for various weather conditions.

Fig. 9. Snowfall simulation with dynamic snow cover

Fig. 10. Rainfall simulation
8. CONCLUSIONS

Techniques for simulating precipitation and snow cover, developed at the OBRUM’s Centre for Military Simulations, have greatly advanced the work of the graphics team due to their procedural nature. It was not necessary to create additional materials for different seasons and there was no need for modelling of destroyed objects – animation of object deterioration is performed in real time by the developed material shader.

The use of shader technology enables troubleless transfer of the developed techniques to any graphical environment, and thereby the use thereof in games, in other simulation systems or in civilian training systems. Moreover, all of these techniques have been developed for multi-screen projection (seamless stitching of the image) on different computer processes (time synchronization between instances of the application).

9. REFERENCES