Journal of Theoretical and Applied Information Technology
10th April 2013. Vol. 50 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SNOW SIMULATION IN TRANSPARENT FORM BASED ON DIRECTX

[1]SONG XING-SHEN, [1]QUAN JI-CHENG, [1]ZHAO XIU-YING, [2]WANG YU
[1] Dept. of Aeronautics &Astronautics Intelligence, Aviation University of Air Force, Changchun Jilin 130022
[2] Dept. of Military Simulation Technology Institute, Changchun Jilin 130022
E-mail: [1]songxingshen@126.com, [2]querryj@sina.com, [3]zxy780345@hotmail.com, [4]wang2566968@126.com

ABSTRACT

Simulating the scenery of climate environment can consumedly enhance the reality of 3D GIS, which has been widely used in computer games, movies and visual simulation system, but until now there hasn't been one effective solution to realize simulating climate environment when cruising the terrain in 3D GIS. In this paper, a simple method of climate environment emulating in transparent form based on DirectX is promoted. The basic concept of this method is imitating snowing by applying particle system, then updating the form using the graphics rendered and vitrifying the form utilizing Windows API. The result of the simulation shows that the method in this paper is simple to implement and can satisfy the requirement of real-time and reality.

Keywords: *Particle System, Snow Simulation, Transparent Form*

1. INTRODUCTION

With the development of computer technology, the three-dimensional geographic information system function also gradually expands and improves, because of its powerful multidimensional information displaying, analysis and processing capacity, the system has a general application prospect. As its most prominent feature -- providing users a 3D realistic feeling, thus requiring the system can interact to realize terrain walkthrough and visual simulation. However, in many domestic and foreign excellent 3D GIS, such as Skyline, Google Earth, which can display terrain and object on it smoothly, but for climate environment simulation remains yet to achieve. Because these systems don't open the corresponding API for users to get an access to modify its 3D scene, so using the form to achieve a transparent climate environment simulation, and superimposing on these systems becomes a kind of feasible solution.

Windows GDI(Graphics Device Interface) has an efficient image processing ability, but it can only support a two-dimensional drawing, in order to improve the visual effects in 3D, we adopt the DirectX as scene rendering tool. Particle system is considered to be the most successful graphics generation algorithm for the simulation of irregular objects and movement, which gets widely use in rain, snow, fog and smoke simulation.

2. DIRECT3D

DirectX is a 3D game engine of Microsoft based on COM component, the latest version has been developed to DirectX11, Microsoft provides a software development kit(SDK) for programmers to design a wide variety of programs. Direct3D is a component of SDK for interactive media in real-time 3D graphics, which can be used for the rendering of 3D graphics and utilizing hardware acceleration, with its good hardware compatibility and friendly way for programming, Direct3D has been generally recognized and most 3D graphics cards provide Direct3D support. Its core structure is a 3D rendering engine, providing coordinate transformation, illumination transform and pixel raster processing [1].

Between Direct3D and graphics device, there is a hardware abstraction layer(HAL), providing device independence for Direct3D. HAL is provided by the equipment manufacturer , which can be a portion of device driver, or special interface and dynamic link library(DLL). Direct3D uses the interface to get access to the graphics device, so that it can bypass the GDI and directly operate the hardware, the application does not directly interact with the HAL, thus realizing the device independence. Fig.1 shows the relationship among Direct3D, GDI, HAL and hardware [2].

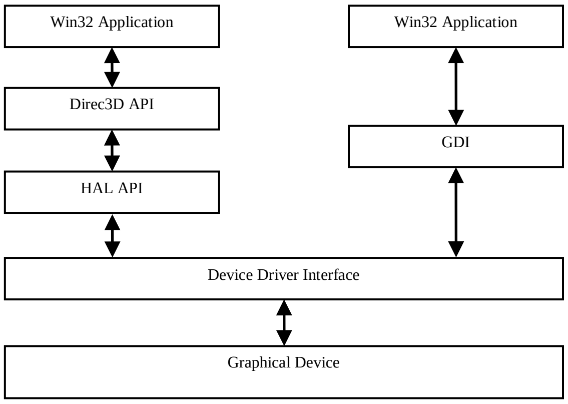

Fig.1 Relationship Between Direct3d And System

From Fig.1 we can also find out that, both Direct3D and GDI can utilize hardware devices through driver software, but the difference is that, by using HAL, Direct3D can fully get the benefit of different hardware, because the hardware producers provide HAL specially for their own devices which support hardware acceleration for programmers. GDI can offer such function because of unity. When programming, we can use functions provided by Direct3D to obtain the characteristic information of different hardware and utilize them.

3. PARTICLE SYSTEM

Scenes like flame, smoke, rain and snow, traditional method of computer graphics geometric modeling cannot be used due to their irregular, because plane and polygon are unable to express these objects, the appearance of particle system successfully solve this problem.

Particle system is first proposed in 1983 by Reeves [3], its basic idea is: using large, randomly distributed particles to represent the irregular fuzzy objects. Each particle has its shape, size, color, transparency, location, movement speed, direction, life cycle and other common properties. These particles are not static sets, they move as time goes by, in the process of change, we don't care single or local changes of particles, but the whole and global characteristics changes. New particles are produced, at the same time the old particles continuously death, showing the irregulation of objects. Therefore, the particle system can well simulate the irregular natural scenery.

3.1 Model analysis

As for virtual simulation, 3D GIS lay its focus on the observation and analysis of geography, snow simulation lies inferior to it, there is no need to completely simulate real snow in physical model, otherwise it will take up too much CPU time for calculation, inducing poor real-time performance, but also very complicated difficult to achieve. In this paper the physical properties of snow is simplified : ignoring the snow interactions between particles; using point model represent a snowflake; using texture mapping instead of complex model; regardless of the snowflake 's own rotation and acceleration.

According to the principle of particle system, an instant screen steps are set as shown below [4]:

(1)Define each particle’s initial attribute;

(2)Produce new particles in the particle source;

(3)Change the attribute value according to the former properties;

(4)Perform judgment of particle life value one by one;

(5)Remove those particles out of deadline;

(6)Draw all the remaining particles.

3.2 Model analysis

When we are trying to create a three-dimensional scene, we can improve computer performance by rendering some two-dimensional objects with three-dimensional visual effect. Billboard technology is put forward to realize such thought.

Generally speaking, billboard refers to some of the roadside bulletin board. When programming in Direct3D applications, we can create such a billboard by setting a fixed rectangle and using texture mapping methodology. The three-dimensional graphics expands its applications. Aiming at using fast rendering of images instead of 3D solid models, which preserving the visual effect while reducing the burden of graphic devices, it applies an image texture to a rectangular primitive and fixes it to a point, drive this primitive to rotate as the observer perspective changes, by keep the normal vertical to the screen make it faces towards observer all the time. Furthermore, part of the billboard can become transparent according to requirement of programmers ,so that objects we don’t want to see become invisible[5].

Billboard is widely used in computer games and three-dimensional visual systems, such as rendering trees and cloud. This paper use the billboard technology to map the snow texture to particles, which improves the rendering speed and increases the sense of reality. According to human visual imaging principle, the distant snowflake looks small and blurry while near snowflake looks clear. Therefore we uses two snowflake textures, as shown in Fig.2.

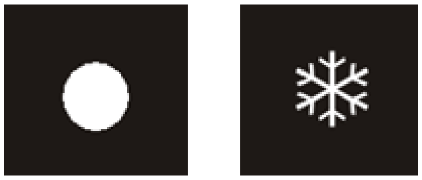

Fig.2 Texture Of Snowflake

3.3 Model analysis

According to the previous analysis, we design the data structure of particles as follows:

```
struct ParticleAttribute
{
D3DXVECTOR3 _position;
D3DXVECTOR3 _velocity;
D3DXVECTOR3 _acceleration;
float       _lifeTime;
float       _age;
D3DXCOLOR  _color;
bool        _isAlive;
};
```

Each particle contains its position and velocity, acceleration vector, life cycle, the life value, color and identifier judging whether the particle is out of deadline or not. A bounding box as snowing area is set, restricted by the bottom left and up right two points.

Each particle is produced in the top of bounding box. From the physical movement of snow particles, particles in the vertical direction are driven by their own gravities and air resistance effect, similar to uniform motion to fall, in the horizontal direction are driven by wind, whose direction may change at any time, therefore, the motion of the particle can be regarded as a composition of uniform motion in the vertical direction and variable motion in the horizontal direction [6]. Steps of generation of new particle are in the following:

```
void AddParticle(ParticleAttribute *attribute)
{
attribute->_isAlive = true;
//set a random position of the particle in the top of bounding box
D3D::SetRandomVector(&attribute->_position,&_boundingBox._min,&_boundingBox._max);
Attribute->_position.y=_boundingBox._max.y;
//velocity vector of snow particle
attribute->_velocity.x = D3D::SetRandomFloat(0.0f, 1.0f) * -3.0f;
attribute->_acceleration.x = D3D::SetRandomFloat(0.0f, 1.0f) * -1.0f;
attribute->_velocity.y = D3D::SetRandomFloat(0.0f, 1.0f) * -10.0f;
attribute->_velocity.z = 0.0f;
}
```

For every frame of the rendering process, record the data of position, speed and acceleration of each particle of the last frame, the current position of the snow particle is calculated by Eq.1.

$$\vec{P}_i = \vec{P}_{i-1} + \vec{v}_{down} \cdot (t_i - t_{i-1}) + \vec{v}_{hi} \cdot (t_i - t_{i-1}) + \frac{1}{2}\vec{a}_{hi} \cdot (t_i - t_{i-1})^2 \quad (1)$$

Among which, $\vec{P}_{i-1}$ is the position in last frame, $\vec{v}_{down}$ is vertical velocity, $\vec{v}_{hi}$ is the horizontal velocity, $\vec{a}_{hi}$ is current level acceleration.

At the same time, determine whether snow particle is moving out of the box one by one, if true the identifier is set to false and render the remaining particles.

Before rendering a vertex buffer is filled first, then the buffer is sent into the graphics hardware for rendering, at the same time we lock another vertex buffer, begin to prepare the desired vertex for the next frame. The benefits of such steps are keeping CPU and GPU efficiently utilized.

As found in this experiment, the number of particles will directly affect the simulation speed and quality: the more particles are generated, the larger the snow is and vice versa. Small amount is fast to render, but the visual effect is not obvious, however, large amount may lead to screen blocked by snow and rendering speed dramatically slow down. Especially when the particle amount excesses 10000,human eyes can realize the hysteresis obviously, which is unacceptable. Table 1 shows the relationship between the amount of particles and frame rate.

Table 1. Frame Rate And Particle Amount

Particle amount	***Framerate/(s^{-1})***
100	92.11
200	88.30
400	84.65
800	76.33
1000	68.38
2000	48.14
4000	34.74
8000	20.62
10000	18.46
20000	8.29

While simulating, we can inform the system to asynchronously response to keyboard or mouse operation, realize roaming in the scene of snow, improve the sense of reality. Fig.3 is a common form of snow simulation.

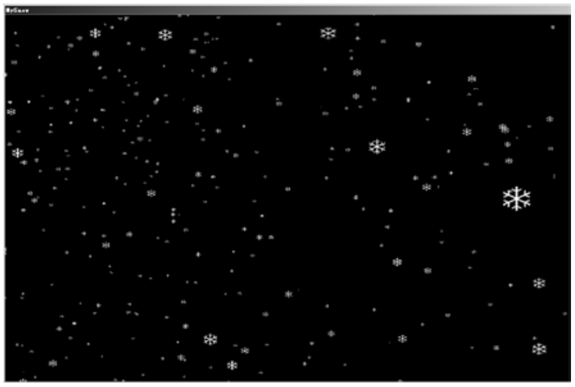

Fig.3 Snow Simulation

4. REALIZATION OF TRANSPARENT FORM

Windows API provides a function UpdateLayeredWindow(), which is the foundation of transparent form, the function is used to update the window of its location, size, shape, content and transparency. Thus, in each frame updating the form content after rendering can realize transparency.

4.1. Specific implementation

When creating a new form, first of all set up the extended window style values for WS_EX_TRANSPARENT, WS_EX_LAYERED, so as to make the form when covering other forms does not affect their responses to operation, only when the form is declared as layered window, can it be transparent or semi-transparent [7], the function of SetWindowPos() can set form on the top of desktop. The following shows implementing codes:

SetWindowLong(g_hMainWnd,GWL_EXSTYLE, WS_EX_TRANSPARENT|WS_EX_LAYERED);

SetWindowPos(g_hMainWnd,HWND_TOPMOST,0,0,WIDTH,HEIGHT,SWP_SHOWWINDOW);

Next, create a Render Target before rendering, on which all the graphics will be drawn, completing the rendering, an Offscreen Surface is created as a carrier of image converted from graphics. In order to achieve a transparent form, a bitmap is needed for the alpha blending. Finally the handle of the bitmap is transited to a device context (DC) by UpdateLayeredWindow(). Implementation methods are as follows:

Next, create a Render Target before rendering, on which all the graphics will be drawn, completing the rendering, an Offscreen Surface is created as a carrier of image converted from graphics. In order to achieve a transparent form, a bitmap is needed for the alpha blending. Finally the handle of the bitmap is transited to a device context (DC) by UpdateLayeredWindow(). Implementation methods are as follows:

Device->CreateRenderTarget(Width,Height, D3DFMT_A8R8G8B8,D3DMULTISAMPLE_NONE,0, false, &d3dSurface, NULL)

MySnow.Display(Device, g_hMainWnd);

//Create Offscreen Surface

Device->CreateOffscreenPlainSurface(Width,Height,D3DFMT_A8R8G8B8,

D3DPOOL_SYSTEMMEM, &offscreenSurface, NULL);

//Pass the content of Render Target on to Offscreen Surface

Device->GetRenderTargetData(d3dSurface, offscreenSurface);

bitmap.Create(Width, Height);

//copy the pixels to bitmap

memcpy(bitmap.GetBits(), lockedRect.pBits,4 Width * Height);*

UpdateLayeredWindow(g_hMainWnd, hdcWnd, &ptWinPos, &szWin, bitmap.GetSafeHdc(), &ptSrc, 0, &stBlend, ULW_ALPHA);

The testing computer is equipped with NVIDIA Geforce GTX580 graphics card, screen resolution is 1920*1080 and the particle number is 8000. As a result, the frame rate can only be maintained at about 21 FPS. After research, the step of copying to bitmap occupy too much time, resulting in decreased efficiency.

4.2. Further Improvement

The particles are generated in the world coordinate system, after projection transformation, only part of them falls in the view frustum. But the system still update all the particles regardless of their visibilities, thus occupy many resources, especially when the particle number is very large, there is an obvious delay.

To improve this, the particle generation is defined in screen coordinate system, only rendering visible particles, thus can solve the efficiency problem, but it can easily cause inconsistent phenomenon when the viewpoint moves, due to viewpoint change frequently on the terrain, so the improvement scheme is limiting the number of particles and reset the dead particles for another life cycle [9].

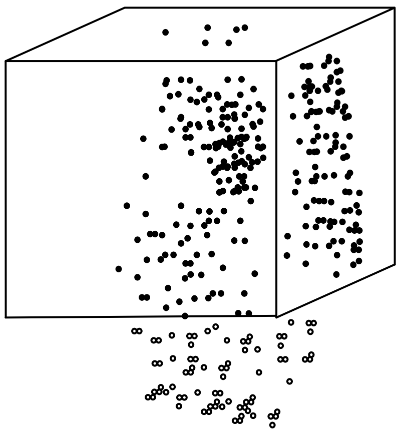

Fig.4 Snowflake And The Bounding Box

A particle moves outside of the current bounding box after a period of time, changing from a visible particle to a completely invisible particle. As designed in previous system, particles that moves outside of the bounding box will be judged to death and removed from the system, at the same time new particles are created at the top of the bounding box. Such operations of releasing and allocating memories in computer iteratively seem a little clumsy. To improve this, when a particle moves outside, we don't remove it immediately, but reset its properties and relocate it to the top of the bounding box, thus no memory changes occur and particle system gets updated still. As Fig.4 shows, the cube represents the bounding box, black points represent the snowflake inside and white points represent the snowflake that move outside [9,10].

The research also finds that, there is redundancy between snow rendering and form updating, so a multi-threaded operation is put up, while copying bitmap to update the form, next frame is being rendered simultaneously. Still one thing to be noted that Render Target is iteratively used for rendering and duplication, a critical region should be set up to avoid corruption in program.

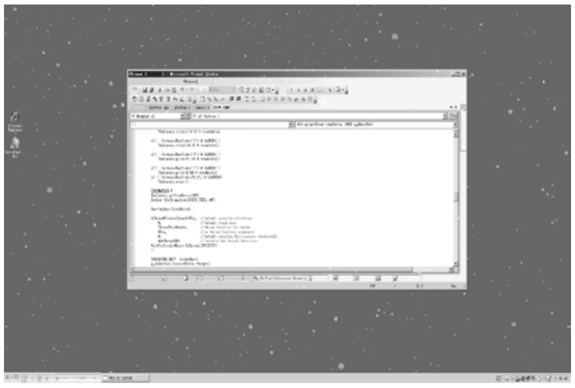

Fig.5 Snow Simulation In Transparent Form

Implementation codes are as follows:

HANDLE hThread= CreateThread(NULL, 0, ThreadProcRender, NULL, 0, &dwThreadID);

//Create a new thread, wait until rendering is finished, then update the form

WaitForSingleObject(hThread,INFINITE);

EnterCriticalSection(&CriticalSection);

g_psysSurface->LockRect(&lockedRect, &rectSurface, D3DLOCK_READONLY);

*memcpy(g_dcSurface.GetBits(), lockedRect.pBits, 4 * Width * Height);*

UpdateLayeredWindow(g_hMainWnd,hdcWnd,&ptWinPos,&szWin,g_dcSurface.GetSafeHdc(),&ptSrc,0,&stBlend,ULW_ALPHA);

LeaveCriticalSection(&CriticalSection);

After the improvement, frame rate retains at around 32 FPS, which generally meet the requirements of visual continuity, Fig.5 is the final result.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5. CONCLUSION

After the test of covering the transparent window of snow simulation on Google Earth, the window could response to operation and work smoothly; although a sudden mouse drag will produce a certain delay, but basically meet the visual effect of the continuity requirement. However, as the particle model constraints, snow particles do not rotate and collide, the next step of the research will focus on resolving the model design flaws.

REFRENCES:

[1] Hao Jinliang, Chen Lei, Lou Gaoming: Cloud simulation based on DirectX. Computer Technology and Development. Vol. 19 (2009), p.195

[2] Liao Xuejun, Wang Rongfeng: Digital Battlefield Visualization and Employment. (National Defense Publications, Beijing, China 2010).

[3] Reeves W T.: Particle System: a Technique for Modeling a Class of Fuzzy Object, Vol. 17 (1983), p.359.

[4] MSDN Library Visual Studio 9.0 release on http://www.msdn.microsoft.com

[5] Zhang Qian. Realistic shape modeling and real-time rendering of snow scene. Clem: submitted to Ynashan University (2009)

[6] He Liang, Ba Lideng. Dynamic simulation of snowscape based on particle system. submitted to Journal of Northwest University. Vol. 40(2010), p.603

[7] Antonio Olvarez, Gustavo Martonez Romero, J Correa-Basurto.Trends, Performance and general operation of the graphic processing unit. Journal of Theoretical and Applied Information Technology.Vol.7(2009), p.129

[8] R. Shankar Naik, K. ChandraSekhar, K. Vaisakh. Adaptive PSO Based optimal fuzzy controller for AGC equipped with SMES and SPSS. Journal of Theoretical and Applied Information Technology.Vol.7(2009), p.8

[9] Zhuo Minhui, Wang Weiming, Zhou Jingjing. Routing optimization for forces based on traffic matrix. Journal of Theoretical and Applied Information Technology.Vol.44(2012), p.7

[10] Takeshita D ,Shin OTA ,et al . Particle2based Visual Simulation of Explosive Flames[C].Proceedings of the 11th Pacific Conference on Computer Graphics and Applications. USA, New York: IEEE, 2003 :p.482