Study on Dynamic Surface Visualization Simulation Based on GPU

Shinan Zhang¹, Jun Pan*²

(Army Armored Force Academy bengbu campus, Bengbu, Anhui Province, 233050)
*zxhtianyi@vip.qq.com

Abstract: In the visual simulation of battlefield geographic environment, the realistic water surface effect can increase the reality of battlefield and enhance the immersion of training personnel, but the fluctuation of water body and the optical characteristics of water surface will also lead to the increase of algorithm complexity and the increase of data quantity, which will affect the fluency of the system. With the improvement of computer graphics hardware performance, in order to reduce the pressure of CPU, many complicated graphics computing has been transferred from CPU to GPU gradually. This chapter’s dynamic surface real-time rendering algorithm will make full use of the programmable characteristics of GPU and powerful computing power, the surface reflection refraction texture sampling, bump mapping, texture movement and other operations are put into the GPU, in order to ensure the surface fidelity, improve the rendering speed.

1. Introduction
In the visual simulation of battlefield geographic environment, the realistic water surface effect can increase the reality of battlefield and enhance the immersion of training personnel, but the fluctuation of water body and the optical characteristics of water surface will also lead to the increase of algorithm complexity and the increase of data quantity, which will affect the fluency of the system. At present, the research on dynamic water surface at home and abroad shows that it is difficult to accurately describe the characteristics of dynamic water surface in real time, so researchers are always looking for the balance between real-time and real-time. With the improvement of computer graphics hardware performance, in order to reduce the pressure of CPU, many complicated graphics computing has been transferred from CPU to GPU gradually. This chapter's dynamic surface real-time rendering algorithm will make full use of the programmable characteristics of GPU and powerful computing power, the surface reflection refraction texture sampling, bump mapping, texture movement and other operations are put into the GPU, in order to ensure the surface fidelity, improve the rendering speed.

2. Optical properties of water surface
In terms of the appearance of the water surface, the optical properties of the water surface play a dominant role. It is because of the interaction of light and water that the special properties of water, a transparent fluid, are fully demonstrated. Therefore, in the visual simulation of water surface, in order to make the effect of water surface appear as real as possible, it is necessary to study and realize the optical effect of water surface. Therefore, this section will analyze the reflection, refraction, and Fresnal effects of light.
2.1 Reflection and refraction of light

In the real environment, when light propagates in two different media, the reflection and refraction of light will occur at the interface of the medium due to the different density of the medium.

1. Reflection

The reflection of light on the water surface has a great effect on the final effect of the water surface, especially when the point of view is far away from the water surface or when the range of the water surface is large (e.g. the sea surface), the view on the water surface is almost caused by the reflection of light on the earth and the sky. For example, the reflection of the sky and the shore on the lake, the shimmering of the water and the bright moon on the sea are all caused by the reflection of light.

The principle of reflection of light: When the incident light $I$ reaches the water surface, a part of the reflection light $R$ from the water surface enters the human eye, resulting in the illusion that the light is coming into the human eye from the incident light $I$ 'under the water surface. The reflection light $R$ is symmetrical with the normal $N$ of the incident light $I$ on the water surface. Assuming that $I$, $R$ and $N$ are all unit vectors, according to the law of reflection of light:

$$R = I - 2N(I \cdot N)$$

According to the formula (1), the reflection light can be easily obtained when the incident light and normal vector are known, and the color produced by the reflection effect can be obtained.

2. Refraction

The effect of refraction is obvious in shallow water or when the view point is close to the water surface. On the one hand, because water is a transparent liquid, the more shallow the water, the more obvious the refraction effect, and when the water is deeper, light may be blocked by impurities in the water and can not reach the bottom, so there will be no refraction phenomenon. On the other hand, the refraction effect is also closely related to the position of the view point. The closer the view point is to the water surface, the clearer the underwater scene is, the more obvious the refraction effect is, and when the view point is far away from the water surface, the refraction effect will weaken or even no refraction phenomenon will occur.

Assuming that theta $i$ is the angle of incidence and theta $t$ is the angle of refraction, assuming that the refractive index of light in air is $n_1$ and that of light in water is $n_2$ (where the refractive index of light in air is 1.0003 and that of light in water is 1.3333), the ray follows the formula (2) in the course of refraction:

$$n_1 \cdot \sin(i) = n_2 \cdot \sin(t)$$

Therefore, it is possible to calculate the direction of propagation of light after refraction on the surface of water according to the angle of incidence of light and the refractive index of light in air and water. In reality, we see that the refraction and reflection of the water surface depends to a large extent on the position of the view. When the angle of incidence is close to 90 degrees and the line of sight is close to the water surface, the reflection of the water surface is the strongest, but when the angle of incidence becomes small, the reflection effect becomes weak, and the refraction effect becomes stronger gradually, when the line of sight is perpendicular to the water surface, the refraction effect is the strongest. This is the Fresnel phenomenon.

2.2 Fresnel rule

Reflection and refraction occur when light passes through the interface of two transparent media. The direction of propagation of light after reflection and refraction can be obtained by formula (1) and formula (2). The proportion of refracted light and reflected light in incident light follows the Fresnel rule derived from electromagnetic wave theory. The relation between the Fresnel coefficient (i.e. the ratio of reflected light) and the incidence angle theta $i$ and the refraction angle theta $t$ can be described as follows:
Rewrite the formula (2) to:

$$ F = \left( \frac{\sin(\theta_i - \theta'_i)}{\sin(\theta_i + \theta'_i)} \right)^2 + \left( \frac{\tan(\theta_i - \theta'_i)}{\tan(\theta_i + \theta'_i)} \right)^2 $$

(3)

After the formula (3) is substituted, the Fresnel coefficient when the incidence angle is $\theta_i$ can be calculated. When light is not absorbed, the transmission coefficient $T = 1 - F$.

Although the refractive index of water surface is affected by many factors, such as water temperature, wavelength of incident light and trace elements in water, these factors are almost negligible in computer simulation of refraction and reflection. Therefore, a simplified formula can be used to calculate the Fresnel coefficient:

$$ F = \frac{(n_1 - n_2)^2}{(n_1 + n_2)^2} + (1 - \frac{(n_1 - n_2)^2}{(n_1 + n_2)^2})^2 \cos(\theta_i) $$

(5)

By using the Fresnel coefficient, the ratio of reflected light to refracted light of a pixel in the water surface drawn at a given moment can be obtained, and the accurate color value of the pixel can be obtained by sampling the reflected and refracted texture. By calculating the color of each pixel of the water surface in the GPU, the complete water surface effect can be drawn.

3. Sky Scene Simulation

Reflections on the sky, terrain, and ground are reflected in every corner of the water, especially in large areas of the water, which are basically reflections of the sky scene, so this section analyzes the rendering of the sky scene.

3.1 Construction of Sky Model

(1) The calculation model is simple and easy to realize real-time simulation;

(2) Skyball needs to deform the original texture image to ensure a better mapping effect, and skybox does not have these problems;

(3) The sky ball needs to match with the terrain, when the terrain size is large, the sky ball also needs to make corresponding magnification, at this time the sky texture precision is high, otherwise the sky rendering image will be very blurred, the effect is very poor. However, the high precision of the sky texture image will affect the rendering efficiency of the graphics engine. And the center of the sky box can move with the view point, do not need to adjust the proportion of the geometric model to match the terrain, as long as the six sky pictures that make up the sky box to meet the basic requirements of the texture can render high-quality sky effect;

(4) The Sky Box can be quickly implemented using Cube Mapping technology, and it can be used to realize the reflection effect of the sky background around the water surface on the water surface.

3.2 Drawing of Sky Scenes

In order to produce the real sky effect quickly, this paper uses the cube mapping technique to draw the sky scene. With the rapid development of graphics hardware, the current mainstream graphics processors support cube mapping technology. As shown in Figure(1), in cube mapping, six two-dimensional texture maps are stitched together according to a certain spatial relationship to form a texture cube, a point on the surface of the cube and a vector in a three-dimensional space. Real-time rendering environment mapping requires real-time rendering of the scene into the environment mapping, which will greatly increase the rendering overhead. In this paper, we use the static cube mapping method, the specific method is to read the pre-prepared static environment mapping from the file, when creating the environment texture can be directly loaded from the hard disk file, using the
static cube mapping although it will reduce a certain degree of authenticity. But it can realize fast rendering and save rendering cost.

Figure (1) Cubic Mapping

The cube mapping technique is used to realize the reflection effect of large-scale water surface to the sky scene. It combines six square textures into a cube mapping space, so that the normal vector of the grid points on the water surface can return a three-dimensional texture coordinate from the mapping space, and then find the three-dimensional texture mapping map through the texture coordinate, then the reflection texture of the sky scene can be obtained.

4. Dynamic Surface Rapid Rendering Based on Convex Mapping and Texture Moving Technology

Based on the analysis of the optical properties of water surface and the comprehensive consideration of the real-time and realistic scene in the whole battlefield geographical environment, this paper will use only two triangles to construct the surface model, make use of the powerful vertex and pixel computing ability of GPU, get the reflection and refraction texture of water surface by projection mapping, and generate the dynamic surface by concave-convex mapping and texture movement.

4.1 Dynamic surface model analysis

A still water surface scene can only reflect the state of the water surface at a certain time. In order to show the dynamic water surface effect, besides the optical characteristics of the water surface, the influence of the movement of water on the water surface should be considered. Because the dynamic water surface is usually the result of the interaction of many uncertain factors, the movement of water is hardly regular. And a small change at a point in the water's surface can directly affect the appearance of the entire water. Therefore, it is necessary to construct a real and reasonable dynamic water surface model in order to render the real water surface effect.

Generally, in the process of water surface rendering, a triangular mesh with a certain resolution is used to construct a static water surface model, and then a specific mathematical model is established to simulate the wave effect of water surface. These methods can be divided into four categories: geometric model-based method, physical model-based method, spectrum analysis-based method and noise source-based method. These methods have their own characteristics, but they all update the position of each vertex of the water surface grid in real-time according to the corresponding
mathematical model in the vertex shader, so as to realize the effect of high and low fluctuation of the water surface. Therefore, the resolution of the grid largely determines the display effect of the water surface. If the mesh density is higher, the larger the area, the more realistic the simulated water surface, the more delicate the water surface effect, the wider the range. But at the same time, the computation amount of GPU will be multiplied, which will affect the real-time performance of the whole system. If the density of the grid is reduced, the spacing between the grids is enlarged, the burden of GPU is reduced and the rendering speed is increased, the rendering effect is not true. It is difficult to find a balance between reality and real-time.

In this paper, a new surface construction method is used. The static surface model consists of only two triangles. In the vertex shader, only two triangles’ vertices need to be updated. However, since the water surface is composed of two triangles, it is impossible to use the method of updating the vertex position to simulate the fluctuation of the water surface after the reflection and refraction of the water surface. In order to obtain the dynamic surface effect, this paper firstly uses the concave-convex map to map the surface, and then generates the static ripples, and then moves the static texture according to the wind direction to realize the dynamic effect of the surface.

4.2 Realization of reflection and refraction
In reality, the color of the water depends on the surrounding environment. The water surface is similar to a mirror, allowing us to see the reflection of the surrounding scene, while the water surface is partially transparent, we can also see the scene below the water surface. So the final color of the water surface is a combination of reflected and refracted colors. The principle and calculation method of water surface reflection and refraction are not complicated, but if all objects in the scene are involved in the calculation of water surface reflection and refraction, there will be a lot of iteration and recursion, which will greatly reduce the rendering efficiency and affect the real-time display of the scene. In order to reduce the amount of computation and improve the rendering speed of water surface, this paper firstly adopts the texture mapping technology, which takes the water surface as the cutting surface, and renders the scene (except the water surface itself) into refraction and reflection texture in real time, and then uses the characteristics of GPU texture calculation and rendering to add the disturbance vector to the sampling coordinate in the pixel shader. The texture is sampled.

1. Reflection
In rendering the reflection texture, we first take the water surface as the reference surface and flip the view image below the water surface so that the direction vector I' of the view point and the direction vector I of the incident light are symmetrical about the water surface. Then the water surface is used as the cutting plane to judge the relation between the object and the water surface, and the mirror image is drawn for the object which is completely above the water surface; for the object which is partly above the water surface, only the part above the water surface is drawn; all the objects under the water surface do not have to be drawn because they do not reflect with the water surface. Finally, the object is rendered as a reflection texture. As shown in Figure (2), the view on the left is the scene seen on the surface of the water, and the reflection texture on the right is the scene.
We usually divide the reflection of water surface into two situations: when the view point is close to the water surface or when the range of water surface is small, the change of the view point has a great influence on the reflection of water surface, in which case we need to draw a complete reflection texture; when the view point is far away from the water surface or the range of water surface is large (such as the sea surface), the change of the view point has a little influence on the reflection of water surface, which can almost be ignored.

For the first case, we need to calculate the view-related reflection texture in real time, calculate the environment and sky around the water surface, and then map the reflection texture from 2D screen space to texture space. Its core HLSL code is as follows:

```plaintext
float2 ProjectedReflectTexCoords;
ProjectedReflectTexCoords.x = Input.ReflectionMapSamplingPos.x / Input.ReflectionMapSamplingPos.w / 2.0f + 0.5f;
ProjectedReflectTexCoords.y = -Input.ReflectionMapSamplingPos.y / Input.ReflectionMapSamplingPos.w / 2.0f + 0.5f;
float4 reflectiveColor = tex2D(ReflectionSampler, ProjectedReflectTexCoords);
```

In the second case, you don't need to consider the position of the point of view, just calculate the sky texture. Since the sky box used in this article serves as the sky model, the cube mapping technique can be used to sample the sky texture directly. Its core HLSL code is as follows:

```plaintext
// Sampling the sky box using a cube map
float3 eyeVector = normalize(Input.WorldPosition - gCameraPos);
float3 reflection = reflect(eyeVector, normalW);
float4 reflectiveColor = texCUBE(CubeTextureSampler, reflection);
```

2. Refraction

The refraction of water surface is caused by the change of direction after light transmission. The method of producing the refraction texture is basically the same as the reflection texture. However, because the refraction is caused by the light from underwater objects coming into the human eye, there is no need for mirror image processing when drawing the refraction texture. In clipping operations, only the underwater scene is retained, so the normal direction of the water surface needs to be reversed when the refraction texture is drawn. As shown in Figure (3), the view on the left is the scene seen on the surface of the water, and the refraction texture on the right is the scene.

![Figure (3) Refractive Texture](image)

The sampling method of refraction texture is similar to that of reflection texture, but due to the deviation of light caused by refraction, the underwater object looks shallower than the actual one, so it is necessary to compress its y coordinate properly when sampling. Part of the HLSL code for refraction texture sampling is as follows:

```plaintext
float2 ProjectedRefrTexCoords;
ProjectedRefrTexCoords.x = Input.RefractionMapSamplingPos.x / Input.RefractionMapSamplingPos.w / 2.0f + 0.5f;
// Compression of y coordinates
ProjectedRefrTexCoords.y =
```
-Input.RefractionMapSamplingPos.y/\text{Input.RefractionMapSamplingPos.w}/2.5f + 0.5f;
float4 refractiveColor = tex2D(RefractionSampler, ProjectedRefrTexCoords);

4.3 Convex Mapping and Texture Mapping

After sampling the reflection and refraction texture of the water surface, we will see clear reflection and refraction of the water surface, but now the water surface looks like a mirror, in order to make the water surface look more real, we need to add some ripples. Since two triangles are used to simulate the water surface and there is no vertex information similar to grid in the plane, it is not effective to use mathematical model (such as sine or cosine function) to adjust the position of the water surface vertex in the vertex shader. In this paper, the concave-convex texture mapping technique is used to add a perturbation to the texture coordinates of each pixel on the water surface, and the sampling position of each pixel is changed slightly to generate the ripple effect on the water surface.

To do a bump mapping, you need to generate a bump texture. Concave-convex textures can be transformed by pre-drawn light and dark images through software. The concave and convex textures are better for detail processing, but not ideal for the overall structure. Another method is to use a gray graph to store the information of true water surface height. First, two orthogonal tangent vectors T1 and T2 at the point P (x, y) are obtained according to the gray graph, and the reference formula (6) and formula (7) are obtained:

\[ T_1 = \{1, 0, \text{gray}[x + 1, y] - \text{gray}[x - 1, y]\} \]  
\[ T_2 = \{0, 1, \text{gray}[x, y + 1] - \text{gray}[x, y - 1]\} \]  

\text{Gary}[x, y] is the gray information at point P (x, y). Then the normal vector T of point P (x, y) can be obtained from T1 and T2.

\[ T = T_1 \times T_2 \]  

![Figure (4) Convex Texture](image)

According to the formula (8), the normal vector of each point of grayscale graph is calculated to generate concave-convex texture. As shown in Figure (4), each pixel of a concave-convex texture contains three values that correspond to three components of a normal vector. For example, if a pixel's normal vector points to the top, its X and Z components are 0, Y components are 1, the corresponding bump texture is red and green, and blue is 1, so the entire texture is blue. However, because the water surface is not flat, the normal vectors of each pixel are not the same, and the red and green channels contain information about the size of the normal deviation from the X and Z axes. So the larger the red and green components, the farther away the normal from the upward direction and the more uneven the water surface, we use these two components to disturb the refraction and reflection textures. In addition, since the normal vector is a normalized vector, the range of values is [-1, 1], and the range of
color values for concave and convex textures is [0,1]. So after sampling, you need to zoom in and out of the resulting texture value. The main HLSL code for bump mapping is as follows:

```csh"
// Declare Convex Texture Sampler
Texture xWaterBumpMap;
sampler WaterBumpMapSampler = sampler_state
{
    texture = <xWaterBumpMap>;
    magfilter = LINEAR;
    minfilter = LINEAR;
    mipfilter = LINEAR;
    AddressU = mirror;
    AddressV = mirror;
};

// Sampling bump texture in pixel shader
float4 bumpColor = tex2D(BumpMapSampler, Input.BumpMapSamplingPos);

// Sampling texture scaling and migration
float2 perturbation = gWaveHeight*(bumpColor.rg - 0.5f)*2.0f;
// Modify the reflection texture sampling code to add disturbance to the sampling coordinates
float4 reflectiveColor =
    tex2D(ReflectionSampler, ProjectedReflectTexCoords + perturbation);
// Modify the refraction texture sampling code to add disturbance to the sampling coordinates
float4 refractiveColor =
    tex2D(RefractionSampler, ProjectedRefrTexCoords + perturbation);
```

So far, the surface has generated a ripple effect, but this ripple is a still image generated by a bump texture mapping. In order for the surface to have a dynamic effect, it is necessary to move the position of the texture coordinates. The direction of movement is determined by the wind direction, and the speed of movement is determined by the wind speed. As shown in Figure (5), if the Y axis is in the direction of the texture movement, then the movement vector $V_{move}$ of any point A in the texture can be derived from the following formula:

$$V_{move} = (A_x, A_y + \text{windforce})$$

Figure (5) Texture Moving Direction

Ax and Ay are the projections of texture coordinate point A on X and Y axes respectively, and windforce is the wind speed.
Part of the texture movement HLSL code is as follows:

```hlsln
// Wind direction, that is, the Y axis in Fig.1-5
float3 windDir = normalize(xWindDirection);
// X-axis positive direction
float3 perpDir = cross(xWindDirection, float3(0,1,0));
// Get Texture Moving Vector
float ydot = dot(inTex, xWindDirection.xz);
float xdot = dot(inTex, perpDir.xz);
float2 moveVector = float2(xdot, ydot);
moveVector.y += xTime*xWindForce;
```

4.4 Fresnel Mixture

After the reflection and refraction texture sampling and bump mapping of the water surface are completed, the color after the sampling is mixed with Fresnel to achieve the real water surface effect. The key step of mixing is to calculate the Fresnel coefficient, which can be calculated according to the formula (1-5). The refractive index of light in water and air is known, and the formula (1-5) can be simplified to:

\[
f_{\text{fresnel}} = r + (1 - r) \cdot \text{pow}(1.0 - \text{dot}(-\text{eyeVector}, \text{normal}), 5.0)
\]  

Among them, the air-to-water reflection coefficient \( r \) is about 0.02037. The color after Fresnel mixing can be obtained by the lerp () function:

\[
\text{combinedColor} = \text{lerp}(\text{reflectiveColor}, \text{refractiveColor}, \text{fresnelTerm})
\]

4.5 Water depth effect

Observe underwater scene, general water is shallow place, the color of the water itself is lighter, refraction phenomenon is also more obvious, can clearly see underwater scene, the deeper the water underwater scene more blurred. When the water is very deep, the color of the water is very strong, and the refraction of the water surface disappears. In the case of this phenomenon, the concentration of water color can be set according to the depth of the water. For shallow water surface, a certain proportion of water color is used to mix with the color of Fresnel. For deeper water surface (such as sea surface), the refraction texture can be ignored and the water color can be mixed with the reflection texture.

4.6 Mirror high light

In real life, it is common to see the sparkling effect on the water surface, mainly because the reflective effect is very strong in a specific water surface area, which is called mirror highlight. This is also an essential special effect in surface visualization simulation. In this paper, a high power calculation is used to obtain a ray with a similarity greater than 99% (i.e., the reflection light and the view vector point multiplication result are greater than 0.95). The main HLSL codes for specular reflection are as follows:

```hlsln
// Traversing all light sources
for(int i=0; i < gTotalLights; i++)
{
    // Processing only available light sources
    if(gLights[i].enabled&&!gLights[i].lightType==0)
    {
        gLightDirection=gLights[i].direction;
    }
    // Add the specular reflection color of the water surface
    float3 reflectionVector = -reflect(gLightDirection, normalVector);
    float specular = pow(dot(normalize(reflectionVector), normalize(eyeVector)), 256);
    combinedColor.rgb += specular;
```
The results of the final water surface visualization simulation are as follows:

Figure (6) Water surface effect

Fig.(7) sea surface effect

Figure (8) Mirror Highlight Effect
5. Conclusion
In this chapter, the visual simulation of water surface is studied. Based on the analysis of optical properties and dynamic water surface model, the color of water surface is obtained by drawing reflection and refraction texture, and the dynamic water surface ripple is generated by concave-convex mapping and texture movement. Finally, Fresnel is mixed and the water depth effect and mirror high light are added to produce the final dynamic water surface effect. The water surface effect realized by this method is fast and real-time, because it does not increase the burden of rendering geometric elements in scene rendering. It is suitable for large-scale 3D scene. When the view is not parallel to the water surface, it can present a real dynamic water surface effect.

Reference
[1] AUBEL A. Anatomically-based human body deformations [D]. Université de Marne-la-Vallée, France, 2002.
[2] ZHOU X J, ZHAO Z X. The skin deformation of a 3D virtual human[J]. International Journal of Automation and Computing, 2009, 6(4): 344-50.
[3] Alexandre Santos Lobão A. Beginning XNA 3.0 Game Programming From Novice to Professional [M]. Apress, 2009:305-319.
[4] Method in Real-time Skeletal Character Animation[J]. International Journal of Virtual Reality, 2011, 10(3): 25.