RUI Xiao-ping[1], SONG Xian-feng[1], JU Yi-wen[1], LU Jin[2],DING Zhen[1]

University of Chinese Academy of Sciences, Beijing 100049, China (1)
China Institute of Water Resources and Hydropower Research, Beijing 100038, China (2)

Flow Adaptive Simulation of Dynamic Water Body Based on River Velocity Field

Abstract. *Simulation of natural water in river flow is a key issue in implementation of Digital Basin. This paper presents a method of water flow visualization which can adapt for the channel of the river based on the velocity field. Aiming at simulating large area river flow, first, Level of details(Lods) are constructed for river velocity fields according to the spatial relationship between view point and the targets. Second, it calculates the velocity fields of steady water flow in real-time using a novel quick Poisson-disk boundary sampling method, and then, Simplified velocity fields are adopted for driving and restrict the sprite textures so as to make them meet Poisson-disk distribution along the river channel. Finally, it blends and render sprite textures and gets real water flow effects. This method uses GLSL（OpenGL Shading Language）shaders to render the dynamic water flow, the shaders use GPU programmable rendering pipeline for graphics calculations, it reduces the real-time computation time of CPU and improves the overall efficiency of the algorithm. An experiment is carried out over Three Gorges Reservoir basin, it suggests that the method achieved a promising results, and the proposed method can be effectively used in simulation of large scale dynamic flowing water.*

Streszczenie. *W artykule przedstawiono metodę wizualizacji przepływu wody w korycie rzecznym, biorący pod uwagę rozkład prędkości wody na dużym obszarze. W pierwszym kroku określony został stopień ilości detali w badanym obszarze, następnie szczegółowo obliczono pola określonych prędkości w oparciu o rozkład Poisson'a, po czym zaadaptowano uproszczone pola prędkości do opisu tzw. sprite'ów. Finalnie, na podstawie wspomnianych analiz otrzymano rzeczywisty obraz przepływu cieczy. Przeprowadzono badania weryfikacyjne proponowanej metody symulacji przepływu. (**Adaptacyjna symulacja przepływu wody w rzece – rozkład prędkości wody**).*

Keywords: velocity field of river, Poisson-disk distribution, sprite texture, GLSL shader
Słowa kluczowe: rozkład prędkości wody w rzece, rozkład Poisson'a, sprite, GLSL.

1. Introduction

Digital River Basin (DRB), as an information infrastructure for watershed science, has attracted increasing attentions over the past years. The DRB supports advanced data acquisition, storage, management, integration and visualization as well as other computing and information processing services via Internet on the river basin scale [1-4]. An important aspect of constructing DRB is the use of digital hydrological and geographic information platform to simulate water systems in whole basin. Visualization technology is integrated in the DRB framework to support the watershed science since it help people get the quantity, distribution, spatial structure and tendency of uncertainty of geo-spatial data and information of watershed intuitively. The DRB provides scientific decision support for the water pollution control, flood mitigation and selection of dam reservoir site through dynamic, real-time, three-dimensional simulation. Real-time water simulation is a key part of DRB visualization system. It's not only important in the engineering applications but also in computer graphics research field. Although 3D scene technology has been widely used in many fields, most of these 3D systems are adopted to display static scene. Currently, the function of dynamic fluid flow simulation is not satisfactory in practical applications, and there are still a lot of research topics should be carried out in simulating dynamical fluid flow.

Water flow simulation in real time has been widely studied in recent years. Most of studies only simulated the real time water flow from two perspectives: fluid velocity and fluid surface [5-8]. In general, velocity simulation is applied to describe large scale river basin, it focuses on studying physical river flow simulation. Fluid surface simulation is adopted to describe realistic small scale river appearance and it focuses on simulating water surface, water wave and environment effective to water(such as light, sky and so on) [9-12]. This study focuses on larger scale river simulation, so we are more interested with the fluid velocity simulation. Generally, there are two kinds of methods to simulate the large scale water flow, one is based on particle system and the other is based on procedural method.

Particles system has been widely applied to track river surfaces in the past decade [13-16]. The method utilized 'particles' as basic units to simulate water surface, and the 'particles' were tracked throughout the volume of the deforming liquid or solid. Although particle system is an alternative way to implement the interaction between the coarsely 3D model and individual water molecules, the limitation of the method is the time consuming computation due to the handle of huge number of particles in real-time visualization. Improved algorithms of particle systems (e.g. Smoothed Particle Hydrodynamics) are also not satisfactory in practical applications due to incorporate sufficient fluid incompressibility for simulating a real-time process.

In general, procedural method computes the velocity locally over a whole domain first, and then it simulates the river according to the real river conditions. This method can be applied in simulating the river flexibly and it has been widely applied in real time water simulations. The hydrodynamic simulations also produced vector data in the form of velocity fields for channel and overland flows. Vector data were visualized using multiple representations including glyphs and particle. Perlin's eponymous noise function(1985) has been widely used in practice to generate random velocity fields; however, it can not handle boundaries issue. Lamorlette and Foster (2002) adopted procedural methods including a Fourier-synthesized turbulence model to animate flames, and also provided some good arguments on why procedural methods can be preferable to fluid simulation. CHENNEY (2004) proposed a new technique named flow tiles to represent velocity fields and the tiles can be constructed to meet a variety of boundary conditions. But this method uses square tiles on a regular grid which result in tiles being mapped with low distortion. Patel and Taylor (2005) introduced a divergence-free velocity field-"fast simulation noise", which is similar to Perlin noise and can create realistic unbounded flow fields with high efficiency. The disadvantages of the method are that it can not handle the dead spots (points of zero velocity) and is not suitable for arbitrary solid boundaries. Funck et al. (2006) introduced the divergence-free velocity fields for shape deformation with a slightly different construction. While it is not clear how does this approach handle boundaries, it could in principle also be used for our application. BRIDSON (2007) proposed a solution to

impose boundary conditions to a velocity field based on procedural noise. But it does not work with complex channel confined flows with branches and obstacles. Qizhi Yu and Fabrice Neyret (2009) presented an interactive simulation algorithm of realistic flowing fluids over large area water body. The method relies on two key issues: the local computation of the velocity field of a steady flow given boundary conditions, and the advection of small scale details on a fluid. It did not applied for simulating large area river. Consequently, the accuracy and performance still need to be evaluated by real river simulating practice. In general, physics-based model of the water surface should capture the scene of water movement in a reasonable time with the minimum computational cost, otherwise, the effect can be sensed by the viewers because either the scene is not realistic or too slow to be rendered in real time.

This study investigates the method of constructing the river velocity field model using fluid dynamics principle. Based on the approach introduced by Qizhi Yu et.al, it adopts quick poisson-disk boundary sample method to generate Poisson distribution of velocity fields, with the combination of LoDs (Level of Details) technology, an improved method is proposed to simulate large scale fluid efficiently. The sprite texture rendering method satisfied with Poisson-disc distribution and the method that river velocity field drive sprite texture move along the river channel adaptively are also introduced in this paper.

2. Velocity Field Constructing For River

2.1. Basic Idea of River Velocity Field

Building the flow velocity field model is a basic step of river simulating. Velocity field of river is the velocity vector distribution of all points in the channel at a specified time [17-21]. The velocity field can be used to describe the flow of river at any time if we can get the velocity field of river. Consequently, we can use graphic technology and geographic information system technology to simulate the natural fluid flow based on procedural model.

The river channel data used in this paper is stream network with boundaries. The river stream network data contains the spatial distribution of the water body, the connection between the water body and the volume of flow of all water body as well. The river can be visualized in 3D space directly by using the river stream network.

This paper assumes the fluid water is incompressible and its flow pattern is plane fluid, the differential equation of the streamline is $-vdx+udy=0$, for the plane flow of incompressible fluid, the incompressible fluid velocity field satisfies the continuity equation $\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$, then there is a function $\psi(x,y)$, which satisfies:

$$d\psi=\frac{\partial\psi}{\partial x}dx+\frac{\partial\psi}{\partial y}dy=-vdx+udy=0$$

$$u=\frac{\partial\psi}{\partial y}$$

$$v=-\frac{\partial\psi}{\partial x}$$

where $\psi(x,y)$ is the flow function of velocity field, u and v are velocity distribution of fluid at different directions respectively. They are the flow velocity along the X axis and Y-axis component in Cartesian coordinates plane. The velocity of fluid can be represented as formula (1).

$$V=\sqrt{u^2+v^2}\,\frac{u+v}{|u+v|} \tag{1}$$

For plane flow of incompressible fluid, unit thickness volume flow of any curves between two stream lines is equal to the stream function difference between two stream lines, it does not have correlativity with the streamlined shape. Thus $Q=\psi_{left}-\psi_{right}$, Where Q is the unit thickness volume flow of any curves between two stream lines, ψ_{left} and ψ_{right} are stream functions on the left and right border of river respectively.

The data of river stream network used this paper contains the volume flow information. The stream function of any points at the same stream line is a constant, $d\psi=0$ on the stream line, it means $\psi=C$ (C is a constant), therefore, each stream line corresponds to a constant value. If an initial value is assigned to stream function of all the river boundaries, we can calculate the values of stream functions at the other boundaries. Then stream function interpolating can be carried out based on all the stream functions of the boundaries, and the stream function over the river channel can be calculated.

2.2. Interpolation of Stream Function

Assumed P in Fig.1 is a point in river channel, the stream function can be calculated by following method. Given a circle with radius s at center point P can be identified, and channel boundaries which intersected with the circle can be found as B_i, and d_i is the distance between point P to B_i, ψ_i is stream function at boundary B_i, linear interpolation can be conducted by equation (2) :

$$\psi(P)=\frac{\sum w(d_i)\psi_i}{\sum w(d_i)}$$

$$w(d)=\begin{cases}d\cdot f(1-d/s), & 0<d\le s\\ 0, & s<d\end{cases} \tag{2}$$

where w(d) is the weight function, $f(t)=6t^5-15t^4+10t^3$ and second derivatives of f is continuous, so this function can ensure the stream function ψ is continuous at the boundary channel, we can obtain stream function over the whole river through Equation(2), and the velocity spatial distribution of fluid can be calculated at any different location at a specified time by formula(1). Velocity field of the river can be obtained by this method.

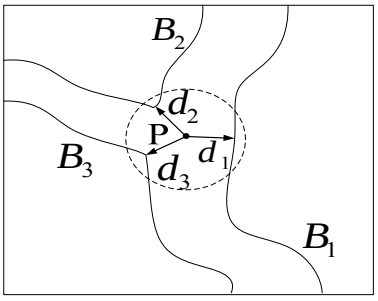

Fig.1. Interpolation of stream function

3. Channel Adaptive Simulation Method for River Water

In recent years, researchers often use triangular strips to simulate water flow while the textures changed dynamically. With the expansion of the water scope, the number of triangles will increase significantly and the program efficiency will also be decreased overall. In addition, the water beating will occur inevitably while changing the textures dynamically. This paper proposes a method to simulate the water flow which uses river velocity field to drive the sprite textures. Sprite texture is a kind of texture which is composed of an independent point with texture slices. Sprite texture used in graphical rendering will reduce the number of triangles, and improve the program efficiency significantly. In our method, points with sprite textures constrainedly move in the river channel driven by velocity field. We can update the activity state of sprite textures according to the relationship between river boundaries and sprite textures, the water surface can be reconstructed by individual active texture through mixing and rendering process, and the dynamic effect of water flow can be applied to the whole river channel.

In the visible extent of 3D scene, point set must be distributed uniformly because it carries texture with fixed block size. Points of uniform distribution should have a reasonable density, the low density will lead to empty holes, and the high density points will cost too much CPU time during real time processing. This paper uses a Poisson-disk distribution method for point sampling. Comparing with the traditional simple random distribution method, this method not only maintains the random of sampling well, it also makes the sampling points to be distributed more uniform. The method regulates the sampling interval distance effectively, so a better result of point distribution can be achieved. This paper assumes the minimum distance among points is d, the radius of sprite texture is r (screen coordinates), in order to prevent empty holes in the water flow, r must be equal or greater than d. Point moves constantly in 3D scene based on the model of river velocity field, The point set are inserted and deleted dynamically to make sure they are always follow the Poisson-disk distribution. Sampling area for Poisson-disk distribution should be the entire visible area of river on the screen rather than the visible area of the river because the shape of river channel is irregular. So the sampling area is a square and this sampling method is more simple and faster. The points located outside of the river boundaries do not be handled in the operation of moving, deleting and rendering, so the high efficiency can be achieved in the fluid simulate system.

3.1. A Quick Poisson-disk Boundary Sampling Method

As mentioned above, Poisson-disk distribution method was adopted to obtain the texture distributed uniformly. The issue is, traditional Poisson-disk distribution method is very complex and could not run real time sampling computation on PC. Therefore, this paper proposes an improved method which named quick Poisson-disk boundary sampling method for Poisson-disk sampling, the work flow of the method is shown in Fig.2: 1) A Poisson disk is generated through random sampling in the polygon regions, radius of the Poisson disk is r, and 2).Implementing sample selection on the boundaries of Poisson disk, but not samples at the rest of the entire region. In Fig.2, the solid line is the border that can be sampled and the dotted line is the boundary that can not be sampled. It can generate Poisson disk when sampling on the boundaries which can be sampled, set the boundaries parts that the Poisson disk boundary falls into the other Poisson disks to be non-sampled border, according to this approach, Sampling until the entire region is completed. This algorithm can maintain the characteristics of the Poisson distribution well and is easy to implement in programming.

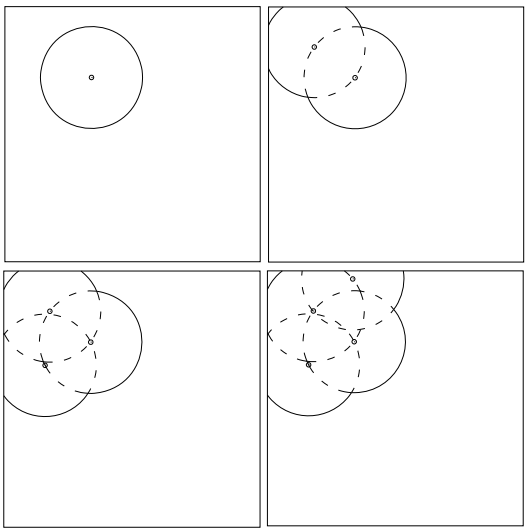

Fig.2. The sampling process of Poisson-disk distribution

In this paper, an extended angle is applied to determine the non-sampled border of Poisson-disk, as Fig.3 shows, assume that we are now sampling at random on the border of Poisson-disk on point A, point B is adjacent to point A, the polar coordinates that point B relative to of point A are (d,θ), we can get the extent of angle for point A which could not be sampled is $(\theta-\cos^{-1}(\frac{d}{2r}),\theta+\cos^{-1}(\frac{d}{2r}))$. The extent of angle that can be sampled remove the extent of angle that can not be sampled is the new extent of angle for point A that can be sampled.

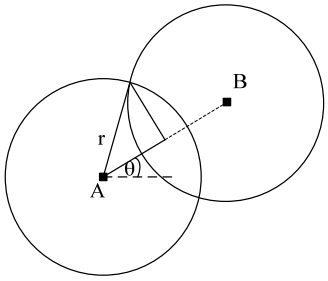

Fig.3. Calculation of sample boundary

The sampling steps at the Poisson-disk boundaries are described as follows and Fig.4 shows the work flow for sampling at the Poisson-disk.

(1) generate point set P through random sampling and point set H that could not be sampled

(2) if P is null then get to step (8); Otherwise select a point A randomly from set P, sampling at the border of point A, determine the extent of angle of A that can be sampled set as *angle Range*.

(3) find out all points that the distance to point A is less than 2r and form a new point set Q

(4) calculate the polar coordinates (d,θ) for points in Q relative to point A looply and calculate the extent angle that could not be sampled set as cutAngleRange, thus, the new extent angle that can be sampled for A now equals to *angle Range-cut Angle Range.*

(5) if *angle Range*=0, then remove A from P and add it to H, go to step(2), otherwise, go to next step

(6) sample on the sample-boundary of A at random and generate poisson-disk C, add C to set P
(7) calculate the extent of angle of A that can be sampled, if *angle Range*=0, the remove A from P and add to H, go to step(2), otherwise, go to step(6)
(8) return back to set H and stop sampling.
The time complexity of this method is O(N) and can be satisfy most sampling computing applications in real time to generate enough points.

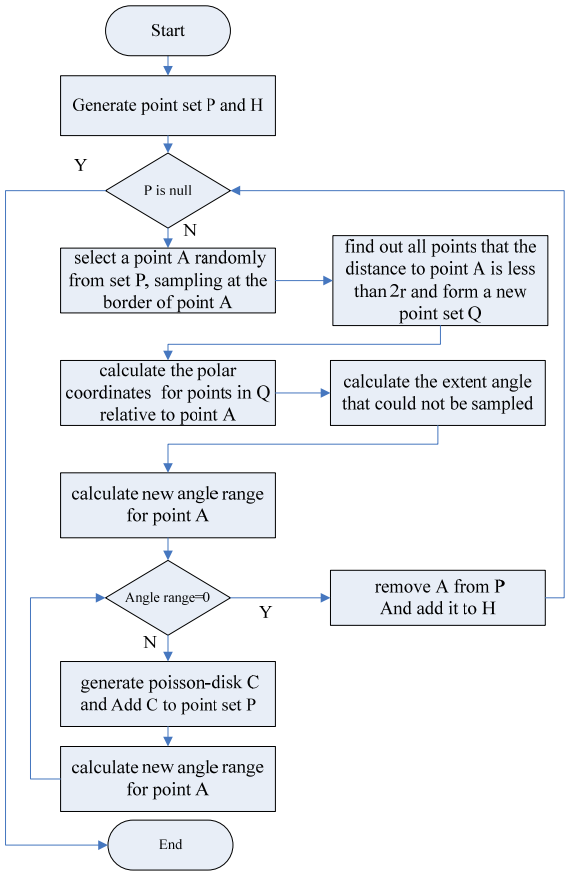

Fig.4. Work flow for sampling at the Poisson-disk

3.2. Move, Delete and Insert of Point

Point set can be handled after reasonable number of points being generated by *quick Poisson-disk boundary sampling* method. The velocity of any points can be calculated in the river channel through formula(1). Assume the world coordinate of a point is P, the world coordinate after elapsed time dt is $P+V(P)dt$, V is the velocity function of the water flow, and it can drive the points moving along the river flow direction in real time. The moving directions of the points are also adapted to the river boundary.

While the points are moving along the river flow, some points may move out of the view and become invisible. It should stop rendering the invisible points and delete them from memory of computer. The points in the visible extent are divided into 2 types: active points and inactive points. If the distance of one point to other points is very close after moving and less than the minimum points distance d, the point is updated as inactive point so as to maintain points be Poisson-disk distributed. The inactive points should also be deleted. In order to avoid hollow hole temporary in water flow due to points deleting, fading out rendering model is adopted to remove the inactive points, and this method reserves a better visible effect. The points will be deleted from computer memory when they are invisible in 3D scene and formula (3) shows the blending factor used for rendering.

After moving or deleting of points, empty holes maybe raise among the rest points, so new points should be generated based on these rest points. The points should be inserted into the middle of the remained points so as to fill the empty holes and all the points should be distributed in a Poisson-disk form overall. As mentioned in previous section, *quick Poisson-disk boundary sampling* method is applied in this study to select points, it inserts new point when the distance between points is less than the minimum distance d, and generate new poisson-disk distribution. To prevent the local contrast when rendering the water flow caused by inserting points suddenly, the method inserts the point into 3D river channel by gradually rendering. Formula(4) shows the blending factor used to render.

$$b_{out}(t) = b_{in}(t_1)\max(t_1 - t + T, 0)/T \tag{3}$$

$$b_{in}(t) = \min(t - t_0, T)/T \tag{4}$$

where t_0 is the time that point generated, t_1 is the time that points are inactive , T is the time required that point getting into or out gradually.

3.3. River Flow Visualization Based on Sprite Textures

In general, a point only carries color information with texture. Unlike the point in conventional rendering method, sprite point carries texture information. The utilization of sprite points in objectives simulation instead of polygons reduces the number of vertices obviously and improves the rendering efficiency for 3D system. The rendering method of sprite point is similar with the conventional rendering method, and the only difference is that texture coordinate replacement mode is assigned for the point sprite. Every sprite point should carry a specified texture while scene rendering. The radius of sprite texture in the screen coordinate is r, but the size of sprite textures is not same size in the world coordinate system. The size of sprite point is related to the distance between the sprite point and the view point. It needs too many textures if specific texture is assigned to each sprite point, and it will cost a lot of computer resources. In this paper, a square texture is adopted as reference texture, and a part of the texture is selected randomly from the reference texture while generating a new point, the point sprite texture is related with the size of the sprite texture and forms a sprite texture.

In order to construct a continuous complete water surface using the sprite textures, the sprite textures must be mixed, the blending factor is defined as follows:

$$b(x,t) = b_x(x)b_t(t) \tag{5}$$

where

$$b_x(x) = \|x - x_0\|/r \tag{6}$$

$$b_t(t) = \begin{cases} b_{in}(t) & t < t_1 \\ b_{out}(t) & other \end{cases} \tag{7}$$

In formula (6), x_0 is the center of sprite texture.

3.4. Build Lods for River Velocity Fields

Compare with the polygon based the conventional methods, the use of point sprite texture reduce the number of vertices and improve the rendering efficiency significantly. Point sprite texture rendering is similar with the conventional point methods, the difference is that the

texture coordinate replacement mode for point sprite and a specific texture must be assigned while scene rendering. In the world coordinate system, the number of sprite texture is uncertain and it is related to the distance between the point of view and the point with texture. In general, it uses Lods (levels of details) model to enhance the efficiency of specific texture rendering and the specific practices in the following two aspects.

First, it removes the invisible river area and the parts beyond view frustum don't need to be drawn in current frame. It obtains the intersection area between view frustum and the terrain according to the coordinate of view point and the direction of sight line, and also determines the visible area of the view point. The back of the terrain and the invisible river obscured by the line of sight will not be drawn.

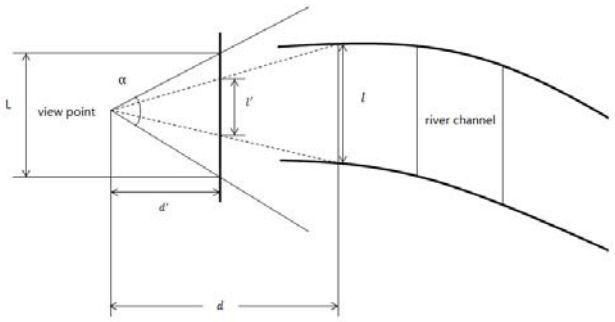

Fig. 5. The relationship between view point and river channel

Then LoD is constructed based on the spatial relations between the viewpoint and scene, different drawing strategies is defined at different levels. The relationship between view point and river channel is shown in Fig. 5.

α is the extent of the view point, d 'is the distance from the view point to screen, d is the distance between view point and the river in the world coordinate, L is the length of projection plane, l is the width of the real river at one point, l' is projection width of river on the screen. Equation (8) and (9) can be derived according to the geometric relationship.

$$d' = \frac{L/2}{\tan\frac{\partial}{2}} \tag{8}$$

$$l' = \frac{d'}{d} \times l = \frac{l \times L}{2d * \tan\frac{\partial}{2}} \tag{9}$$

Divide the river into different segments along the flow channel according to the longest width of the river, the layout coefficient of point sprite texture for each segment is set as h, the layout coefficient is applied to control the number of point sprite texture on each river segment. h is proportional to the radius of the point sprite texture, the number of the point sprite texture is l'/h at the place where the distance from the view point is d in world coordinate(as formula 10 shows).

$$n = \frac{l'}{h} = \frac{1}{h} \cdot \frac{l \times L}{2d \cdot \tan\frac{\partial}{2}} \tag{10}$$

Equation(10) suggests that the greater length the river project on the screen (ie the river closer to the screen) is, the more point sprite texture should be laid on the river, and the river be further away from the screen just put less point sprite texture on it.

3.5. Rendering of Water Surface

The bump texture mapping algorithm is utilized to implement the water fluid rendering. Bump mapping is a texture-blending method which creates the scene of a complex texture on primitives. Although standard texture blending works well for smooth surfaces such as a wood floor, it can't be used in realistic rough surfaces modeling. Bump mapping provides a texture which contains depth information in the form of values indicating high and low spots on the surface [23-27]. Bump mapping creates a per-pixel texture coordinate perturbation of specula or diffuse environment maps by taking the specification of the bump map contour using delta values. Fig.5 shows the bump texture used in this paper.

Fluctuations in the surface rendering uses programmable GPU rendering pipeline and applies GLSL to write GLSL shaders for bump texture rendering. This paper uses two rendering shaders: vertex shader and fragment shader. The vertex shader is mainly utilized to implement the vertex coordinates transformation, calculate line of sight vector, light position, vertex reflection vector, refraction vector and texture coordinates. The fragment shader is mainly used to select texture, calculate light intensity and obtain the final fragment color.

Usually, Fresnel effect happened in water flow simulating very frequently due to the extreme angle between the observer and the water body: small angle will lead to the more significant reflection, and large angle value will make obvious refraction effects. Therefore, reflection, refraction, color and hybrid surface color of the water should be calculated according to the angle between the observer and the water in scene rendering. Generally, the calculation of Fresnel is complex, and the following calculation method often used in practical applications.

FresnelRatio = F + (1.0-F)×
pow((1.0-dot(*eyeDir*,*normal*)),5.0)

where F is the reflection coefficient of air to the surface and its value is 0.020037, *eyeDir* is the line of sight, *normal* is normal vector between the light and the surface at their intersection point, the color value of water can be obtained as follow:

finalColor=waterColor + reflectionColor×
FresnelRatio+ refractionColor×（*1- FresnelRatio*）

where *finalColor* is the final color value of water, *watercolor* is the initial color of water, *reflectionColor* is the surface reflection color, *refractionColor* is refraction color of water. The steps to achieve the vertex shade are as follows.

(1) Calculate the vertex location in the world coordinate system through the transformation

(2) Calculate the line of sight vector, light position, and normalize the vector

(3) Calculate the reflection and refraction vectors of the vertices and normalize the vectors

(4) Computing texture coordinates.

The steps to achieve the fragment shader are as follows.

(1) Calculate texture samples at the direction of reflection and refraction respectively

(2) Computing the environment light, scattered light and mirror light

(3) Obtain the final fragment color.

Vertex and fragment shaders are utilized together and there are five variables passed from the vertex shader to fragment shader when rendering. Fig.6 shows the texture used for simulating water surface and Fig.7 shows the rendering results of water surface with texture:

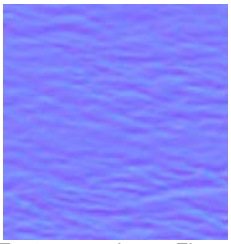

Fig.6. Texture used

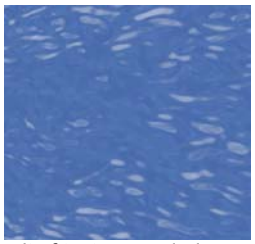

Fig.7. Result of texture rendering

Fig.8. Visualization of flowing water

4. Case Study

In this paper, Three Gorges Reservoir basin water in China is selected as study site for the experiment. The three-gorge project is one of the largest hydraulic projects all over the world and it attracted attentions from many people and organizations in China and other countries. On the one hand, it provides comprehensive contribution of flood control, electric power generation and river shipping; On the other hand, it influences the eco-environment in the reservoir area which needs more assessments profoundly. So the water surface simulation of three-gorge dam area becomes an important topic in the study and management. Based on the simulation of three-gorge river flow, researchers may study the related issues from an explicit way which provides further knowledge for decision makers and the public as well.

The configuration of computer used in the simulation is: Windows XP SP3 system, P4 CPU, 2G RAM, ATI graphics cards whose memory is 256MB. QT, OSG(Open Scene Graph) and GLSL are combined together to achieve the effect of real-time dynamic water flow. DEM and image data used in the system mount to 5GB and we designed a dynamic block scheduling. The simulated water surface is the Three Gorges Reservoir area Yangtze River water. The start point is Cuntan in Chongqing city and the end point is the Three Gorges Dam and the length of simulated river is 645km. We divided the study river into 33 segments according to the width of the river and the width of every segment is nearly equal. Then we evaluate the number of sprite points using different texture radius, tab. 1 shows the result, PN1 is the sprite point number obtained by the area of the river dividing the area of the sprite point, PN2 is the sprite point number obtained based on the random poisson-disk sample method and PN3 is the number obtained by our quick poisson-disk boundary sample method. We can see that even when the radius is great, a lot of sprite points are still needs, therefore it will impact on the rendering efficiency.

Radius(Km)	PN1	PN2	PN3
0.001	453859	454299	453753
0.005	90772	90859	90677
0.01	45386	45427	45351
0.02	22692	22724	22625
0.03	15129	15149	15892
0.04	11346	11355	11323

So, we divide the distance between 3D objects at visible area and view point into 5 classes which are very near, near, middle, far, and very far, and the corresponding radius is 0.001, 0.01,0.05 and 0.1 and 0.5 km, the number of sprite points when roaming is just about 2500 and the fps of rendering is larger than 50. Fig.8 shows some flow visualization water bodies in 3D scene and people can roam smoothly.

5. Conclusions

Water flow simulation is a critical topic in generating realistic three-dimensional scene of river basin. It is useful in hydrological studies, water ecological protection, water pollution control and other fields. Dynamical simulation of water flow is also key technique in DBR. This paper focuses on the 3D simulation of natural water flow, it attempts to propose a new algorithm for water simulation, a real-time dynamic flow effects is obtained according to the water features. The main achievements of this study are :

(1) According to the characteristics of water flow, dynamic effect of river water flow is obtained by applying the river velocity field to drive the sprite texture moving adapt to the river channel.

(2) Stream function of any point can be acquired according to the interpolation of boundary stream function. Flow speed at any point in the river can be calculated using the fluid mechanics principles.

(3) A quick Poisson disk boundary sampling algorithm is proposed. The algorithm provides a promising way to generate sprite points which satisfy the Poisson distribution, and it also can be implemented in programming easily. The time complexity of this algorithm has linear relationship with the extent of water flow. In addition, it only generate samples at visible area, which reduces the number of sprite points as well as improve the efficiency of real-time sampling. Quick Poisson disk boundary sampling algorithm generates points with evenly distribute sprite texture, which reduce redundant rendering. Combining with LoDs technique, high rendering efficiency can be achieved.

(4) The shaders are defined by GLSL language and programmable GPU rendering pipeline is utilized for data processing. It means the approach blends and renders the moving sprite textures using GPU, and uses programmable shading pipeline to process the data, it will reduce the real-time operation amount of CPU which release CPU from large complex computations and calculate using GPU for graphics rendering. Consequently, high quality and efficiency of scene rendering can be achieved by concurrent computation of CPU and the GPU.

Experiment suggests that the proposed approach is a promising solution for adaptive flow of river water simulation. The proposed method adopts river velocity field to drive sprite texture movement and obtains water flow effects, combining with the flow field, it is an easy-to-use method for other specific applications such as spread and diffusion simulation of pollution. In this paper, the effect of island in water area is not considered in the simulation, in some cases, the distribution of island in water body probably affects the movement of waver surface significantly. Consequently, the velocity filed of river with island will be explored in future studies.

Acknowledgments.

This work is supported by the National Science and Technology Major Project of China (Nos.2009ZX07104-006 and 2011ZX05039-004) and the National Natural Science Foundation of China (No.40901191).

REFERENCES

[1] BRIDSON R., HOURIHAM J., NORDENSTAM M.. Curl-noise for procedural fluid flow. ACM Transactions on Graphics. 2007.

[2] Cani, M.-P. and Desbrun, M. 1997. Animation of deformable models using implicit surfaces. IEEE Transactions on Visualization and Computer Graphics. 3(1997), No. 1, 39–50.

[3] CHEN, J., AND LOBO, N.. Toward interactive-rate simulation of fluids with moving obstacles using the navier-stokes equations. Computer Graphics and Image Processing 57(1994), 107–116.

[4] CHEN, S., JOHNSON, D., AND RAAD, P.. Velocity boundary conditions for the simulation of free surface fluid flow. J. Comp. Phys. 116(1995), 262–276.

[5] CHEN, J., LOBO, N.,Charles H.. Real-Time Fluid Simulation in a Dynamic Virtual Environment. IEEE Computer Graphics and Applications.17(1997), No. 3, 52-61

[6] CHENNEY 2004]CHENNEY S., Flow tiles. In Symposium on Computer Animation (2004), 233–242.

[7] D.Enright, S.Marschner, and R.Fedkiw, Animation and Rendering of Complex Water Surfaces. ACM Transactions on Graphics (Proceedings of SIGGRAPH), 2002, 736-744.

[8] Desbrun, M. and Gascuel, M.-P.. Animating soft substances with implicit surfaces. In the Proceedings of SIGGRAPH 95.1995, 287–290.

[9] Desbrun, M. and Cani, M.-P.. Smoothed particles: A new paradigm for animating highly deformable bodies. In Computer Animation and Simulation 1996. 61–76.

[10] Dunbar D. & G.Humphreys. A spatial data structure for fast poisson-disk sample generation. ACM Transactions on Graphics, 25(2006), No. 3, 503–508.

[11] Foster, Nick, and Dimitri Metaxas. Realistic Animation of Liquids, Graphics Models and Image Processing. 58(1996), No. 5, 471-483.

[12] Wolfram von Funck , Holger Theisel , Hans-Peter Seidel. divergence-free velocity fields for shape deformation. ACM Transactions on Graphics-TOG, 25(2006), No. 3, 1118-1125.

[13] Kass, Michael, and Gavin Miller. Rapid, Stable, Fluid Dynamics for Computer Graphics. Siggraph.24(1990), No. 4, 49-55.

[14] Kawai, M., Hirota, K., Kuroyanagi, S., "Development of a real-time fluid simulator for an interactive virtual environment: Improvement of density feedback in SPH", Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on, 14(2009), 1701 - 1706.

[15] LAMORLETTE, A., AND FOSTER, N. 2002. Structural modeling of flames for a production environment. In Proc. ACMSIGGRAPH, 729–735.

[16] M¨uller, M., Charypar, D., and Gross, M. 2003. Particle-based fluid simulation for interactive applications. In the ACM SIGGRAPH 2003 Symposium on Computer Animation. 2003, 154–159.

[17] PERLIN, K.. An image synthesizer. In Proc. ACM SIGGRAPH, 1985, 287–296.

[18] PATEL, M., AND TAYLOR, N. 2005. Simple divergence-free fields for artistic simulation. journal of graphics tools. 10(2005), No. 4, 49–60.

[19] Premo˘ze, S., Tasdizen, T., Bigler, J., Lefohn, A., and Whitaker, R. 2003. Particle-based simulation of fluids. Computer Graphics Forum 22, 3 (Sept.), 401–410.

[20] Robert Bridson, Ronald Fedkiw, and Matthias Muller-Fischer. Fluid simulation:SIGGRAPH 2006 course notes. In SIGGRAPH '06: ACM SIGGRAPH 2006.Courses, New York, NY, USA. ACM Press.(2006), 1-87.

[21] Stam, Jos. Stable Fluids. Proceedings of Siggraph 1999. New York: ACM Siggraph, 1999, 121-127.

[22] Stora, D., Agliati, P.-O., Cani, M.-P., Neyret, F., and Gascuel, J.-D. 1999. Animating lava flows. In Graphics Interface 99(1999), 203–210.

[23] Terzopoulos, D., Platt, J., and Fleischer, K.. Heating and melting deformable models (from goop to glop). In Graphics Interface 1989. 219–226.

[24] Tonnesen D., "Modeling Liquids and Solids Using Thermal Particles," Proc. Graphics Interface, Canadian Information Processing Soc., Toronto, 1991, 255-262.

[25] Qizhi Yu,Fabrice Neyret, Eric Bruneton. Spectrum-preserving texture advection for animated fluids: report of INRIA. INRIA, 2009.

[26]Qiuwen Zhang, Cheng Wang and Ryosuke Shibasaki.Distributed Modeling of Hydrologic System Based on Digital River Basin. Environmental Informatics Archives, 3 (2005), 92-97.

[27] Zhu, Y. and Bridson, R. 2005. Animating sand as a fluid. ACM Trans. Graph. 24(2005),No. 3, 965–972.

Authors: *RUI Xiao-ping, SONG Xian-feng, JU Yi-wen and DING Zhen are with University of Chinese Academy of Sciences, Beijing, China. LU Jin is with* China Institute of Water Resources and Hydropower Research, Beijing, China.

(E-mail: ruixp@yahoo.com.cn; xfsong@ucas.ac.cn; juyw03@163.com; lujin@iwhr.com; 11782294@qq.com)