Research on Trajectory Algorithm of The Arrow Based on Game Engine

Wenfeng Hu
School of Computer Science
Communication University of China
Beijing, China
huwf@cuc.edu.cn

Taobo Gao, XiaoLong Fu
School of Computer Science
Communication University of China
Beijing, China
1121252070@qq.com

***Abstract*—Trajectory detection is a classic problem, which is often used to predict trajectories and optimize trajectory paths. In video games, trajectory algorithms are often used to implement the motion of three-dimensional objects such as bullets, shells, and arrows. However, archery games have poor simulation effect on the arrow trajectory. The motion of the arrow is not affected by the tension and the material of the bow. Arrow trajectory optimization is necessary to achieve a more realistic simulation. So, this paper uses Unity3D to simulate the trajectory of the arrow, and proposes an arrow trajectory algorithm that is influenced by the player's motion state, the material of the bow, initial velocity of the arrow, distance, gravity and air resistance.**

Keywords—Interpolation fitting; Trajectory algorithm; Unity3D; Simulation; Ballistic Algorithm

I. INTRODUCTION

Archery is a common gameplay in video games. For example, in "Dynasty Warriors 5", the arrow moves forward in a straight line; in “World of Warcraft”, the arrow follows the movement of the target and is not affected by gravity. In the above game, the movement of the arrow is relatively simple, and does not conform to the movement process of the arrow in actual life. It is need to further research and improve the algorithm.

In the archery process [1], first, the archer pulls the bowstring to give the arrow a forward initial force. According to Hooke's law, the initial force of the arrow is affected by the pull and the material of the bow. The greater the pulling force, the farther the arrow moves. Next, the arrow will move forward in the form of a parabola under the influence of gravity and air resistance.

The focus of this paper is on the factors affecting the arrow motion, and the arrow trajectory algorithm will be optimized to simulate a more realistic arrow motion process. The structure of this paper is to summarize several common ballistic algorithms in the game, then propose an optimized trajectory algorithm, and use the Unity3D game engine to create an educational game [2] to show the new arrow trajectory algorithm. The factors involved in this paper that affect the trajectory of the arrow are as follows:

(1) The magnitude of the tensile force. The initial velocity of the arrow is related to the strength of pulling the bow [3]. Under the constant elastic modulus of the bowstring, the initial velocity of the arrow is positively correlated with the elastic deformation of the bowstring, and the elastic deformation is affected by the magnitude of the tensile force. The greater the pull, the greater the initial speed of the arrow.

(2) The elastic modulus of the material of the bow [4]. The bow consists of a flexible bow arm and a flexible bowstring. When the accumulated force during the pulling process is released instantaneously, the arrow on the bowstring can be shot at a distant target. When the bow is pulled, the bow arm and the bow string will be elastically deformed. Bow arms and bowstrings with different elastic coefficients will cause different trajectories of the arrows.

(3) Arrow’s weight. According to Newton's second law, the initial velocity of the arrow is affected by the weight of the arrow.

(4) Direction and size of the wind. The wind can speed up the movement of the arrow or slow down the speed of the arrow.

In summary, this paper will put forward an arrow trajectory algorithm which is suitable for game development based on Unity3D game engine. With the algorithm, game developers can use this algorithm to create a more realistic archery system.

II. TECHNICAL FOUNDATION

A. *Unity3D game engine*

Unity3D is a game development engine developed by Unity Technologies that allows developers to create 3D games, architectural roaming, virtual reality, educational simulations, and more [5]. The Unity3D game engine uses Mono as a scripting engine virtual machine with C# or JavaScript as the game scripting language. The Unity3D interface is simple and easy to operate, allowing artists and programmers to work together in a unified environment. It is a user-friendly, powerful game engine.

B. Camera frustum

A frustum represents a prism whose head is intercepted, representing a viewable area or a projection. A frustum refers to a geometry in which a cone or pyramid is intercepted by two parallel planes between two parallel planes in geometry. The frustum of the camera is a geometry obtained by cutting a quadrangular pyramid, formed by a near clipping plane, a far clipping plane, and a field of view (FOV). As shown in Figure 1.

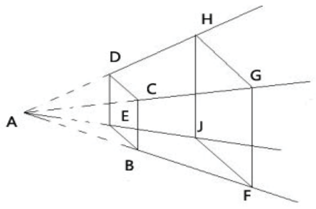

Fig.1 Frustum

The camera is located at the intersection of the extension lines of all the side edges of the frustum, which is point A in the figure. The near plane of the camera is the plane BCDE in the figure. The far plane of the camera is the plane FGHJ in the figure. In a game engine, FOV generally refers to the resulting intersection angle of the four surrounding cameras facing the camera. The smaller the FOV value, the narrower the field of view of the camera. When it is 0, the frustum will be a straight line. The 3D object in the game engine will be successfully rendered by the GPU and displayed on the screen in the frustum of the camera. And all three-dimensional objects on the axis of the frustum are rendered to the center of the screen, its screen coordinates are (screen width/2, screen height/2).

C. Projection Vector and Gram-Schmidt Orthogonalization

A projection vector is a problem in which a vector finds its projection vector on a plane that is not coplanar with it.

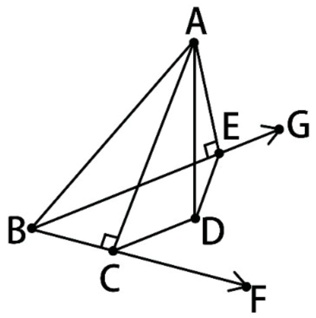

Fig.2 Projection vector

As shown in Figure 2, it is assumed that the vectors BA, BF, and BG are not collinear. The vector BC is the projection vector of vector BA on vector BF. The vector BE is the projection vector of the vector BA on the vector BG. The projection point of point A on plane BFG is point D, so vector BD is the projection vector of vector BA on plane BFG.

The vector AD is perpendicular to the plane BFG, and the vector BE belonging to the plane BFG, so the vector BE is perpendicular to the vector AD. The vector BE is the projection vector of the BA on the vector BG, so the vector BE is perpendicular to the vector AE. The vector AD and the vector AE intersect with the point A, so the vector BE is perpendicular to the plane ADE. The vector DE belongs to the plane ADE, so BE is perpendicular to the ED and the angle BED is a right angle. Similarly, the vector BC is perpendicular to the plane ACD and the angle BCD is a right angle.

Therefore, the sufficient and necessary condition is that the vector BD is the sum of the vector BE and the vector BC is that the plane BEDC is a parallelogram.

As can be seen from the above paragraph, a set of diagonals of the planar BEDC is a right angle. If the angle CBE is a right angle, it can be concluded that the BCDE is a rectangle. In other words, if the vector BC and the vector BE are a set of orthogonal bases, the sum of the projection vectors of the vector BA on the vector BC and the vector BE is the projection vector of the vector BA on the plane BFG.

Therefore, the problem of finding a projection vector of a vector on a plane that is not coplanar with it can be solved by finding two non-collinear vectors on this plane. If the two vectors are orthogonal, then separately the projection vectors of the two vectors on the two non-collinear vectors on this plane are then added and then obtained.

So, the problem of finding a projection vector of a vector on a plane that is not coplanar with it can be converted to find two non-collinear vectors on this plane. If the two vectors are orthogonal, the projection vectors of the vector on the two non-collinear vectors on this plane are respectively determined, and then the sum of the two vectors is the desired projection vector.

The Gram-Schmidt orthogonalization formula will be used to solve two non-orthogonal vectors. In linear algebra, if a set of vectors on the inner product space can form a subspace, then this set of vectors is called a base of this subspace. The solution process of Gram-Schmidt orthogonalization is shown in Figure 3.

$\boldsymbol{V}^n$: Inner product space with dimension n

$\boldsymbol{v} \in \boldsymbol{V}^n$: The elements of V^n can be vectors or functions, etc.

$\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle$: Inner product of <v1, v2>

$\text{span}\{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n\}$: Subspaces consisting of $v_1, v_2 \dots v_n$

$\text{proj}_{\boldsymbol{v}}\, \boldsymbol{u} = \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\langle \boldsymbol{v}, \boldsymbol{v} \rangle} \boldsymbol{v}$: u projection on v

$$
\begin{array}{ll}
\boldsymbol{\beta}_1 = \boldsymbol{v}_1, & \boldsymbol{\eta}_1 = \frac{\boldsymbol{\beta}_1}{\|\boldsymbol{\beta}_1\|} \\
\boldsymbol{\beta}_2 = \boldsymbol{v}_2 - \langle \boldsymbol{v}_2, \boldsymbol{\eta}_1 \rangle \boldsymbol{\eta}_1, & \boldsymbol{\eta}_2 = \frac{\boldsymbol{\beta}_2}{\|\boldsymbol{\beta}_2\|} \\
\boldsymbol{\beta}_3 = \boldsymbol{v}_3 - \langle \boldsymbol{v}_3, \boldsymbol{\eta}_1 \rangle \boldsymbol{\eta}_1 - \langle \boldsymbol{v}_3, \boldsymbol{\eta}_2 \rangle \boldsymbol{\eta}_2, & \boldsymbol{\eta}_3 = \frac{\boldsymbol{\beta}_3}{\|\boldsymbol{\beta}_3\|} \\
\vdots & \vdots \\
\boldsymbol{\beta}_n = \boldsymbol{v}_n - \sum_{i=1}^{n-1} \langle \boldsymbol{v}_n, \boldsymbol{\eta}_i \rangle \boldsymbol{\eta}_i, & \boldsymbol{\eta}_n = \frac{\boldsymbol{\beta}_n}{\|\boldsymbol{\beta}_n\|}
\end{array}
$$

Fig.3 Gram-Schmidt orthogonalization

Thus, a set of orthogonal bases {β¬1, ... β¬n} on span {v_1, ... v_n}, and corresponding standard orthogonal bases {η_1, ... η_n} are obtained.

Therefore, when finding a projection vector of a vector on a plane that is not coplanar with it, if two non-collinear vectors on this plane are not orthogonal, the Gram-Schmidt orthogonalization formula should be applied to obtain a vector. Group orthogonal substrates. Then, the projection vectors of each vector on each vector in the orthogonal base are respectively obtained, and finally the projection vectors are obtained.

D. Levels of Detail (LOD)

LOD [6] is a common method used by game engines to reduce the complexity of the scene and increase the rendering rate of the model. According to the rules of the near and far, the details of the objects that the human eye can see are different. In the video game, the level of detail refers to the different parts of the scene, using a different model detail [7]. The LOD technique builds a set of geometric models of varying degrees of detail for the same model. The degree of detail of the model is rendered according to the coordinate position of the model and the distance of the camera.

III. Overview of Trajectory Algorithm

A. Bullet Ballistic Algorithm

In the FPS game, the bullet moves from the muzzle toward the point at which the crosshair is aimed. In game development, Fire Point is the coordinate point generated by the bullet. The Fire Point is on the central axis of the frustum of the first-person camera. The forward axis of this point is collinear with the forward axis of the camera. The coordinate point that Fire Point renders on the screen is at the center of the screen and is also the position of the crosshair.

After the program listens for a mouse click event, it produces a ray at the Fire Point that is perpendicular to the camera's near plane. The hit point coordinates are obtained by this ray. The core code is as follows:

```
function GunShot ()
{
    firePointSetup();
    var instantiatedProjectile;
    instantiatedProjectile = Instantiate (Gun.bullet,
firePoint.position, firePoint.rotation);
    lastShot = Time.time; audio.clip =Gun.fireSound;
    audio.Play();
    GunMuzzleFlash();
    BroadcastMessage ("Fire",
SendMessageOptions.DontRequireReceiver);
}
```

B. Arrow trajectory algorithm with unknown target point

The mathematical model corresponding to this algorithm is the projectile motion [8] that does not consider air resistance. First, the velocity component of the arrow in the vertical direction and the velocity component on the horizontal plane are obtained by using the angle θ between the initial direction of the arrow and the vertical direction. Use t to represent the time of the arrow movement.

According to Newton's second law, the displacement of the arrow in the vertical direction after t is:

$$\mathrm{S}_{ver} = v_0 * \sin\theta * t + {}^{1}\!/_{2} * g * t * t\text{^}2 \qquad (1)$$

The displacement of the arrow on the horizontal direction after t is:

$$S_{hor} = v_0 * \cos\theta * t \qquad (2)$$

This displacement solves the projection direction of the arrow on the horizontal plane when it is shot. According to the above, the core code is as follows:

```
v1=Vector3.right;
v2=Vector.forward;
v3=this.transform.forward.normalized;
projectorvec=(v3.magnitude)*(v1.magnitude)*Mathf.Cos
        (Vector3.Angle(v3,v1))*v1+(v3.magnitude
        )*(n2.magnitude)*Mathf.Cos(Vector3.Angl
        e(v3,v2))*v2;
    projectorvecdir=projectorvec. normalized；
```

C. Arrow trajectory algorithm with target point

In some role-playing games, after the player selects the target point, the arrow is fired toward the coordinate point. This is the arrow trajectory algorithm with a fixed hit position.

The above algorithm is abstracted into a two-dimensional plane. Assume that the plane passing through the arrow starting point and the target point and perpendicular to the horizontal plane is recorded as the XOY plane.

But the 3D game world is a three-dimensional space, and the arrow that the player shoots out is not necessarily facing the target point. From the top view, the initial velocity direction of the arrow does not necessarily intersect the target point. So, the initial velocity of the arrow we studied on the XOY plane is only a component of the initial velocity of the arrow. This component is the projection vector of the initial velocity of the arrow on the XOY plane.

As shown in Fig.4, the arrow starts from point A, the initial velocity is vector AB, and the target point is D. The initial velocity value that is for the calculation is the projection vector AC of the vector AB on the vertical plane where the vector AD is located. Therefore, the problem of the initial velocity of the projection motion is transformed into the problem of finding the projection vector of a vector on a plane.

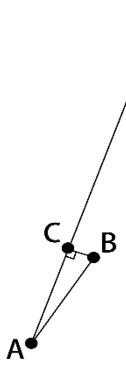

Fig. 4 Projection vector

This problem has been solved above. After the projection vector is obtained, the three-dimensional problem can be transformed into a two-dimensional problem [9]. It refers to the parabolic motion of the arrow on the XOY plane without air resistance. Its trajectory is shown in Figure 5.

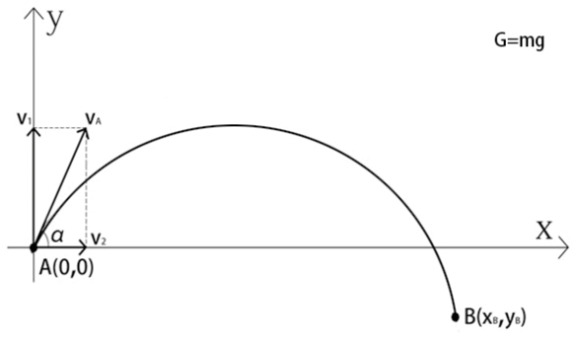

Fig.5 Parabolic motion

From the target point, it can be known that the horizontal distance and the vertical distance between the position when the arrow is shot and the target point are constant values. The angle α is the angle between the arrow and the horizontal plane when it is just projected. But the arrow can hit at different target points, so the change of this trajectory algorithm is its acceleration g.

According to Newton's second law:

$$v_a * \cos\alpha * t = x_b - 0 \tag{3}$$

$$v_a * \sin\alpha * t + {}^1\!/_2 * g * t^2 = y_b - 0 \tag{4}$$

The code for g is as follows, where ver is the magnitude of the vertical velocity, hor is the magnitude of the horizontal velocity, and calspeed is the magnitude of the initial velocity on this plane.

```
ver=Mathf.Cos(Vector3.Angle(this.transform.forward,V
ector3.up))*calspeed;
hor=Mathf.Cos(Vector3.Angle(this.transform.forward,V
ector3.up))*calspeed;
g=((ver*d/hor+h)*2*Mathf.Pow(hor,2))/(Mathf.Pow(d,2
));
```

After finding g, the variable t can be obtained by bringing in the above equations. For the rotation of the arrow, a linear interpolation algorithm can be used. Let the initial value of the rotation be the rotation value when the arrow is shot. The end of the rotation is the amount of rotation when the arrow points to a fixed target point. The arrow starts counting when it is just shot, and the time it takes to account for this time t is the degree of interpolation.

IV. Arrow Trajectory Algorithm and Simulation

This optimization algorithm has a higher degree of simulation for archery movement. Different bows and different arrows will correspond to different arrow movements. The motion of the arrow is affected by the accuracy of the bow, the initial velocity of the arrow, the distance, gravity, air resistance, and wind.

A. Arrow interpolation algorithm

In the game, in order to ensure the continuity of the arrow motion picture, the interpolation algorithm can be used to generate the motion tween. The interpolation algorithm of the arrow considers two factors: interpolation of position and interpolation of rotation. The specific implementation method is as follows:

In order to ensure the continuity of the arrow motion picture, an interpolation algorithm is needed. The interpolation algorithm considers both the position and rotation factors. When the arrow is shot, record the arrow shooting time, the arrow shooting position, the arrow shooting rotation angle, and the camera rotation angle. The core code of the interpolation algorithm is as follows:

```
startPos=this.transform.position;
startRot=this.transform.rotation;
FirePointPos=Cam.camera.ScreenToWorldPoint
        (new
Vector3(Screen.width/2.0f,Screen.height/2.0f, zdepth));
```

FirePointRot= Cam.transform.rotation;

Therefore, the direction of motion of the arrow on the horizontal plane is recalculated for each frame, that is, the direction of the projection vector is required for each frame; then, the position of each frame is no longer based on the arrow. The basis of the exit point is based on the position of the previous frame, that is, the vertical direction of the frame and the displacement in the direction of the horizontal plane are added to the previous frame.

B. *Projectile motion and air resistance*

Because of the difference in arrow shape, surface smoothness, and windward area, the air resistance of the arrow is different, which affects the motion track. In this paper, the mathematical model of air resistance is used in the process of arrow projection.

C. *Arrow initial speed and tension*

The form of tension is the degree of elastic deformation of the bow. The measure of the elastic deformation of the bow can in turn be converted to the displacement of the right-hand position, or the length of time that the current bow animation has been played. At the same time, it can also be converted to the percentage of the length of the current bow animation that has been played. In other words, the pull can be measured as a percentage of the length of the entire bow animation that has been played so far.

According to the formula that converts the elastic potential energy of the bow into the kinetic energy of the arrow:

$$1/_2 * k * x^2 = 1/_2 * m * v\text{^}2 \tag{5}$$

k is the elastic coefficient of the bowstring, and m is the mass of the arrow. It can be concluded that the initial velocity here is a variable that increases as the elastic deformation of the bow increases.

The minimum rate of fire and the maximum rate of fire for each bow are dependent on the ideal rate of fire of the bow. There is a functional relationship between them. In this paper, a positive proportional function is used to obtain the minimum initial velocity min_{sp} and the maximum initial velocity max_{sp} that each arrow can emit by dividing the coefficient k1 and multiplying the coefficient k2.

$$ideal_{sp}/k_1 = min_{sp} \tag{6}$$

$$ideal_{sp} * k_2 = max_{sp} \tag{7}$$

The function of the initial speed of the arrow and the playing time of the bow animation is:

$$\mathrm{y = a * x^2 + b * x + c(a > 0)} \tag{8}$$

$$-b/(2 * a) = t_1 \tag{9}$$

It can be seen from the above that the initial velocity of the arrow in the t1 state is minsp, and this value is also the maximum value of the function, that is:

$$(4 * a * c - b\text{^}2)/(4 * a) = min_{sp} \tag{10}$$

D. *Arrow initial speed and arrow weight*

Under the same pulling force of the same kind of bow, the initial velocity of the arrow is inversely proportional to the quality of the arrow. The lighter the arrow, the higher the initial velocity; the heavier the arrow, the smaller the initial velocity. In the paper, the quality of the arrow is set to 1, and the other arrows set the mass value according to the standard. The initial value of the arrow obtained above is divided by the mass value of the arrow to get the new arrow. speed.

E. *Collision detection in Unity3D*

Collision detection [10] is a commonly used feature in 3D games. The speed of objects in the game can be divided into normal speed motion and high-speed motion. When dealing with objects moving at normal speed, it is possible to use the collision detection method that comes with the Unity3D engine, such as collision detection of obstacles such as stones when NPC moves. However, objects moving at high speed may have the following conditions, as shown in Figure 6.

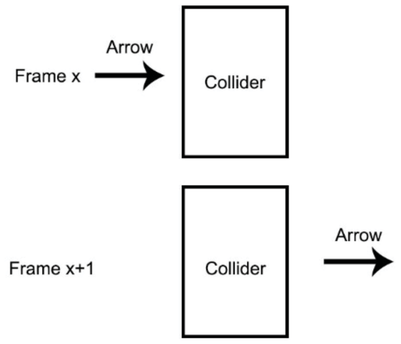

Fig.6 The high-speed motion of the arrow

The arrow is located to the left of the object at the xth frame, but is located to the right of the object at the x+1 frame when it is at the x+1 frame. When a computer renders an image, it renders one frame at a time. What happens between the frame and the frame is not known. If there is no frame where the arrow intersects the object, the engine's own collision detection cannot determine the target point of the arrow.

So, in order to solve this problem. Each frame of the arrow that emits a body will emit a ray. The starting point of this ray is the position of the arrow of this frame. Its direction is from the position of the arrow of this frame to the position of the arrow of the previous frame. Its length is the distance between the position of the arrow of this frame and the position of the arrow of the previous frame. If this ray collides with the NPC, it is considered that the arrow hits the NPC.

V. An Archery Game Demo

A. *Game introduction*

The game is an educational game, which is an example about arrow trajectory algorithm in this paper. This game mainly simulates the process of bow archery, including hunter movement module, pull bow module, and arrow motion process, as shown in Figure 7.

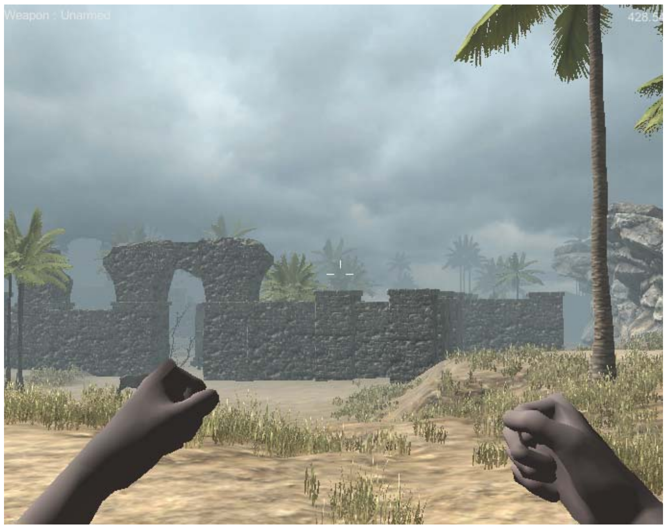

Fig.7 Game scene

B. Gameplay and system design

The game is divided into two modules, the UI and the bow archery mechanism. As shown in Figure 8, the bow arching mechanism includes the motion simulation of the arrow and the motion simulation of the bow.

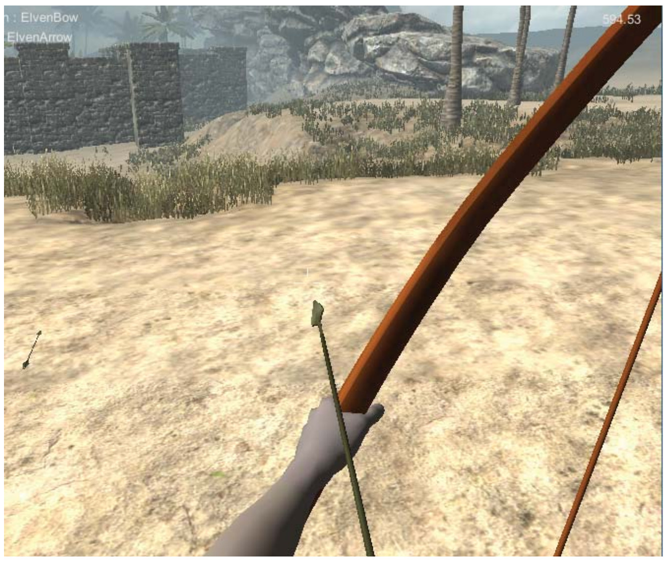

Fig.8 Simulation of bow and arrow

VI. CONCLUSION

This paper implements an arrow trajectory simulation algorithm, but there are still some shortcomings. Different bows, depending on the material, can cause the same archer to take different times when the different bows are full. This can also be simulated. In this paper, the arrow motion algorithm of the first-person archery game is obtained through research. However, this motion algorithm must do a lot of repetitive work in the next multiplexing. For the sake of convenience, this motion algorithm can be written as a plug-in for reuse.

The game considers the factors affecting archery movement more comprehensively, with higher degree of simulation and strong operability. Different bows and different arrows will correspond to different arrow movement trajectories. The final hit position of the arrow is no longer solely dependent on the position of the sighting of the target, but will depend on a variety of factors other than aiming, such as the player's state of motion, the accuracy of the bow, the initial speed of the arrow, the distance, gravity, Air resistance, players need to accumulate operational experience to improve archery skills.

REFERENCES

[1] X. H. Chen, A Study on the History of Archery in Ancient China, Soochow University, 2017

[2] W. Zhang, Design and implementation of campus security education VR game based on UE4, LiaoChen University.2018.6

[3] J. A. Chen, Q. P. Tan, Z. L. Li, A. M. Lu, Analysis and Comparison of Muscles Consistency in Different Bow – opening Techniques of Compound Bow.2018. Journal of Chengdu Sport University. 2018

[4] Z. J. Li, Sniper rifle ballistic trajectory calculation and the sniper training simulation research, Jilin University. 2016.4

[5] W. F. Hu, S. Zhao, Y. Ren, 3D model dynamic cutting technology based on game engine. 2017. 1.

[6] J. L. Hu, Research on the Integeration Method of 3-dimentional Terrain Model and Ground Feature Model based on Level of Detail, Lanzhou Jiaotong University. 2013

[7] C.Y. Wang, K. L. Zhu, Q. X. Wu, D, M. Lu, Adaptive display technology of high precision model based on local rendering, Journal of Zhejiang University, 2018.8

[8] M. Jiang, Teaching Research based on the Core Competencies of Physics Subjects---Taking the Projectile Motion Unit as an Examole. Sichuan Normal University. 2018.4

[9] Q. X. Jia, X. L. Li, J. Z. Song, X. Gao, G. Chen, H. B. Zhang, Projectile motion aerodynamic parameter identification and simulation, IEEE .2014

[10] Z. N. Chen, Power Optimization of Game Applications for Mobile Platform, University of Science and Technology of China. 2071.4