A Simple and Efficient Collision Detection Algorithm and its Implementation[*]

Yanpeng Sun
Software Teaching Division of International Institute,
University of Qingdao
Qingdao, Shandong Province, China
meng_yu_sun@163.com

***Abstract*—Border collision is an important function in the game, but the existing methods, like bounding box, octree, etc. are complex and they require extensive scene statistics for support. However, XNA-based 3D racing game achieved a simple and effective collision detection algorithm, which could properly resolve problems emerged during the designing of racing game.**

Keywords*—*XNA; Racing game; Collision detection

I. Introduction

Border collision is an important function of the game as it's the hardware base for the fun of game. Now people have worked out a variety of collision detection methods, such as bounding box, octree, etc., but these methods are complex and require extensive scene statistics for support, thus they affect the development of the whole project due to heavy preparation work in earlier stages, although they boast satisfactory effectiveness.

During 3D racing game development, this paper uses XNA Game Studio, which is a set of managed class libraries used on Windows and Xbox 360. It is integrated in Visual C # 2005 Express Editions, and shows a group of new features in 2D and 3D game development. By using a new game component model and a new framework class library which is used on game development by Windows and Xbox360, we achieved collision detection of camera and provided a quick and easy way for game development.

II. Collision between racing car and road border

After researching and analysis, this paper tries to work out a sound and effective collision detection method suitable for 3D racing game. Key ideas are as follows:

First, collect road border data. Art designers could get the road border very easily from the scene model, and export the middle files of border data, and then supply these data to programs which would analyze and finish data collection automatically. Later, programs could figure out ubiety between cars and road borders. This approach lacks generality or commonality, but it's very convenient for this game and helpful for the overall development progress.

In the collision mode between racing car and road border, the coordinates for racing car will be located by the four wheels. First, set up equation of two lines in space through road border data. Then acquire the real time space coordinates of the racing car. Last, check out whether the coordinates of the racing car is between the two linear equations or not. Fig. 1 shows the collision mode between racing car and road border:

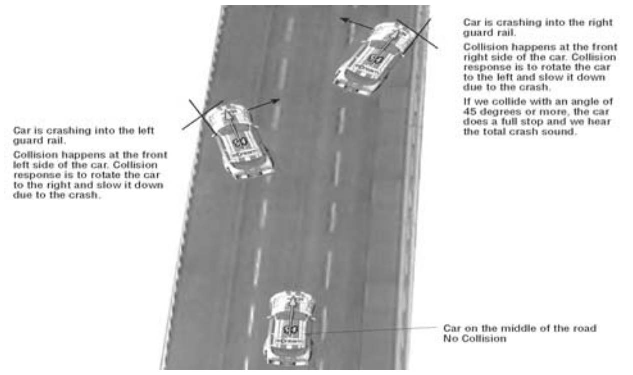
Car is crashing into the right guard rail.
Collision happens at the front right side of the car. Collision response is to rotate the car to the left and slow it down due to the crash.
If we collide with an angle of 45 degrees or more, the car does a full stop and we hear the total crash sound.
Car is crashing into the left guard rail.
Collision happens at the front left side of the car. Collision response is to rotate the car to the right and slow it down due to the crash.
Car on the middle of the road
No Collision

Fig. 1 Collision of racing car and road border

Based on the current coordinates of the racing car and road border data, finish collision detection through checking out whether the racing car is located within the boundary of the two road borders or not. When collision with the road boundary happened, calculate the intersection angle between the racing car and the road, and on the basis of the principle of mirror reflection, let the racing car drive toward the opposite direction with the same intersection angle, thus to complete the collision handling.

The direction of the racing car could be adjusted through changes in racing car's matrix information. Fig. 2 shows the collision for bifurcated surfaces:

[*] Fund Project: This paper received the fund of Institutional Research Project(JY0911)

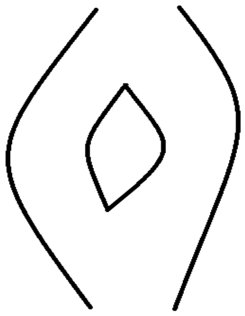
Fig. 2 the model of bifurcated surfaces

It’s the same as for bifurcated surfaces, but only need to detect one more track compared with single track. The following is the detection code of collisions between racing car and left-side border:

```
Public void OnCollisionDetectionLeft(object sender,
EventArgs e) { BCSound.PlayCue(BCAllSounds.Crash);
int WheelNum =
((BCCollisionDetection)sender).WheelNum;
//front wheels
if (WheelNum == 0 || WheelNum == 1)
{
if (Math.Abs(cosval) > 0.5f)
{
velocity *= 0.5f;
if (forward)
{rotationM.X = -0.1f;}
else
{rotationM.X = 0.1f;}}
else
{velocity = 0f;
blPosition -= cDirection * 10f;
}}
//back wheels
if (WheelNum == 2 || WheelNum == 3)
{if (Math.Abs(cosval) > 0.5f)
{velocity *= 0.5f;
if (forward)
{rotationM.X = -0.1f;}
else
{rotationM.X = 0.1f;}}
else
{velocity = 0f;
blPosition += cDirection * 10f;
}}}
```

III. Collision between Racing Car and Road Surface

In the collision mode between racing car and road surface, the coordinates for racing car will be located by the four wheels. We deem the coordinates for the middle of the two front wheels as front, and that of the two back wheels as back. The principle of collision detection between racing car and road surface as follows:

First, based on the X coordinate of front, calculate the point coordinates on both sides of the border with the same X value, and record them as P1 and P2.

Second, according to coordinates of P1 and P2, calculate and get a linear equation in space which has passed P1 and P2, deem it as Fun1

Lastly, input X coordinate of front into Fun1, we can get the road surface height h1 where the racing car was located. And then compare the road surface height and the Y value of the racing car’s front head, thus the collision detection can be completed now.

We could get the trail height h2 by the same way.

Fig. 3 shows the mathematical model of the collision between racing car and road surface

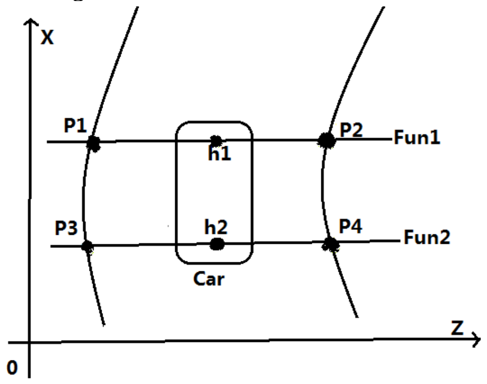

Fig. 3 Mathematical model of the collision between racing car and road surface

IV. Collision between Racing Cars

In the collision model between racing cars, the coordinates of the racing car will be acquired by a bounding sphere. Wrap the whole car body with a standard ball as large as possible, then transfer the collision between racing cars to collision between balls. The reason to choose a ball collision model is because it’s easy to detect and handle, and also easy to carry out, less data calculation as well. Fig. 4 shows the mathematical model of ball collision:

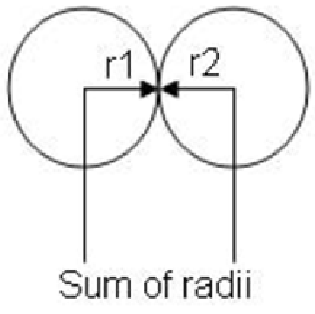

Fig. 4 Mathematical model of ball collision

V. Conclusion

Reference to a large number of existing scientific literatures, we refuse to adopt the matured algorithms in hand; but instead we map out a new algorithms according to the project requirement through bold design, innovation, especially for collision detection solution. The new algorithm can not only avoid the complexity in traditional collision detection, but also achieve more accurate results based on special requirements of project, and the algorithm has also been well applied to in practices.

REFERENCES

[1] Qiaomin Lin, Ping Lin, Ruchuan Wang, "Research on Collision Detection Algorithm in 3D Game Development", Computer Technology and Development 2010, 20(5)

[2] Sufeng Yin, Hanming Chen, Zhifeng Liang, Zhining Liu,"XNA-based Skinned Mesh Animation of 3D Characters [J]" Gansu Science and Technology 2009, 25(15)

[3] Wei Zhao, Wenhui Li, "A Fast Collision Detection Algorithm based on Sphere Hybrid Reconstruction [J]" Computer Science 2009,36(7)

[4] Xiaochun Yin, Ming Nie, " Collision Detection Research and Application in the Roam of 3D scene[J]" Computer Science and Technology 2010,6

[5] XNA 3.0 Game Programming Recipes[D] 2009