3D 모바일 게임 내의 가상카메라를 위한 자이로 센서의 민감도 설정 방법

백인식, 김종국
고려대학교 전기전자공학부
E-mail: bispro@korea.ac.kr, jongkook@korea.ac.kr

Method of Sensitivity Configuration of Gyro Sensor for Virtual Camera inside 3D Mobile Game

Baek Insik and Jong-Kook Kim
School of Electrical Engineering, Korea University

Abstract

In this paper, we present a method for smartphone users to have a better user experience when playing 3D mobile games using the gyro sensor. We designed the rotation of the virtual camera in the game world to be proportional to the real-world's rotation. We have also made the sensitivity configuration possible for users to manipulate.

1. Introduction

As more and more users are playing games on their smartphones, companies are developing more sophisticated games. One category is 3D games and the virtual camera ([1]) in the game engine works as the eyes that allow players to see the game world. Changing the users' view point of the virtual camera in smartphone games is restricted due to limited number of input buttons, while in PC games it could be simply done by pressing a certain keyboard key or by moving mouse.

But in the general case of mobile games, change the option to the view point is not implemented or feasible with the cumbersome touch joystick. Even the most famous mobile racing game Asphalt™ does not have such feature. Even for certain mobile games which have view point changing feature, it is done with restricted settings process and is not as 'smart' as it is expected to be. We implemented this feature that enables the synchronization of the real-world and game world using the Unity3D game engine ([3]). That is, the rotation angle of the smartphone in the real world corresponds to the rotation angle of the virtual camera in the game world. Thus, Users can rotate the smartphone and observe objects in the game world intuitively. Figure 1 shows a simple demonstration of the test application.

After the implementation, we discovered that users need to excessively rotate their smartphone to rotate their view point. If the rotation that occurs inside of the game world is larger than the rotation in the real-world, this problem is solved. Therefore, we used the Slerp (Spherical linear interpolation) technique ([2]) and the extrapolation method ([4]) to adjust the sensitivity of the rotation. As far as the authors know, this is the first implementation for 3D games on mobile smartphones. Then, a survey was conducted to find the range of tolerable sensitivity and the most preferred sensitivity.

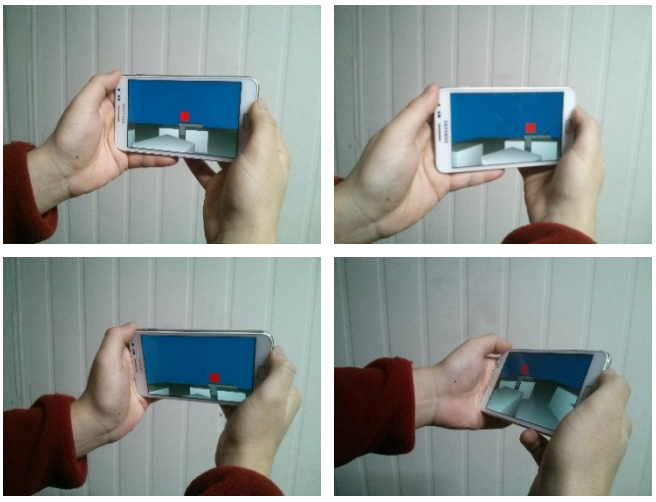

Figure 1. Demonstration of virtual camera rotation in game world

2. Description of implementation

2.1. Obtaining real-world rotation angle in Unity3D

As far as the motion sensor and the Android platform is concerned, Unity3D game engine ([3]) only provides APIs to obtain the acceleration value of the device. Thus, we had to obtain the rotation value using gyro sensor by accessing Android API from the inside of Unity3D game engine. For this purpose, we've packed Java files and cross-compiled .dex binaries into .jar package and called Java functions from a C# script by means of JNI (Java native interface), which links Java and other languages like C++ or C#, etc. We had to transform the obtained rotation value which is in Euler angle form into Quaternion ([2]). Quaternion is a general form that represents a rotation in 3D computer graphics. Figure 2 depicts the overall process of obtaining the rotation value.

We interlocked the real-world rotation angle with rotation angle of the virtual camera ([1]) which exists inside the game world.

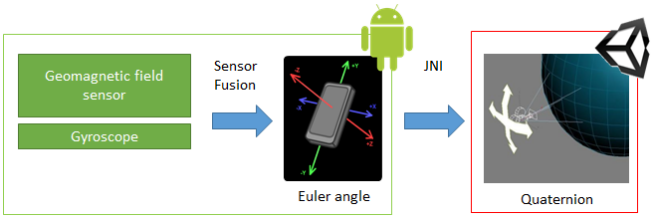

Figure 2. The process of obtaining rotation from Unity3D

2.2. Changing Sensitivity

When we interlock the rotation of the smartphone and the rotation of the virtual camera without compensation, users have to rotate their smartphone 45 degrees in order to rotate the view 45 degrees within the game. This excessive movement will cause awkwardness due to the frequent arm twisting, head nodding, and panning while playing 3D mobile games. To prevent such incidents, we had to reduce the range of users' physical rotation by using Slerp (Spherical linear interpolation, [2]). The Slerp is a geometric formula that is used to animate natural 3D rotation in 3D graphics, which enables constant-speed interpolation along the spherical surface. In our case, Slerp is used to scale the sensitivity of rotation.

The Slerp calculation is done using the first Quaternion which points to the forward direction in the game-world and the second Quaternion which represents the real-world rotation of the smartphone. The expression can be found in *Equation 1*. This statement holds true for the interpolation parameter **t** ranging from 0 to 1.

$$\boldsymbol{Q_{adjusted}(t) = Slerp(Q_{forward}, Q_{original}; t \mid_{0 \le t \le 1})}$$

Equation 1. Adjusted rotation for $0 \le \mathbf{t} \le 1$

As Unity3D API only supports the interpolation parameter **t** equal to or less than 1, we needed to implement the Slerp operation for sensitivity value greater than 1. This operation can also be referred to Slerp extrapolation. We implemented this feature using existing Unity3D API and the approach could be found in [4].

For extrapolation parameter **t** greater than 1, **t** can be separated with integer part **n** and fractional part **f**. From the definition of Slerp, the operation result when **t** is 1 is equal to $Q_{original}$. As the addition of parameters is equivalent to multiplication of Quaternions, operation result for **t** greater than 1 is shown in Equation 2. By applying Equation 2, we could calculate Slerp operation for every **t** greater than or equal to 0.

$$\begin{aligned}&\boldsymbol{Q_{adjusted}(t \mid_{t=n+f})}\\ &\boldsymbol{= Slerp(Q_{forward}, Q_{original}; 1)^n * Slerp(Q_{forward}, Q_{original}; f \mid_{0 \le f \le 1})}\\ &\boldsymbol{= Q_{original}{}^n * Slerp(Q_{forward}, Q_{original}; f \mid_{0 \le f \le 1})}\end{aligned}$$

Equation 2. Adjusted rotation for **t** > 1

By changing the interpolation and extrapolation parameter **t**, a real number equal to or greater than 0, we can reduce or increase the range of the rotation. This parameter **t** can also be called sensitivity because the compensated rotation angle is directly proportional to the given rotation. Sensitivity value 1 means the rotation in the game world exactly corresponds to the real-world rotation. A value smaller than 1 means rotation in the game world is less sensitive than the actual rotation, and a value larger than 1 means the opposite.

3. Application example - Spider Decimation VR

This application is a remake of the famous mobile game Angry Birds™, a 2D platform game that throws bombs to targets with a slingshot and wreck objects in numerous stages. We reinterpreted the original concept and implemented the game in a 3D world. This game is so named because the goal is to kill all the spiders in the game and letting users to experience VR (virtual reality). The main feature of this game is that the rotation angle of the slingshot corresponds to the rotation angle of the virtual camera in the game world. The direction of the bomb thrown from a slingshot is also coupled with the rotation angle of the slingshot, making it possible for users to shoot bombs more intuitively, so that users can experience the virtual reality. Figure 3 shows what the game looks like. The rotation sensitivity can range between 0 and 10.

Figure 3. <Spider Decimation VR> gameplay

4. User sensitivity test

The survey was conducted to find the best sensitivity value that is most preferable by users. Five participants were chosen randomly. The sensitivity value ranged from 0 to 10. First, to find the tolerable sensitivity range, the test subjects

were asked to find the minimum and the maximum sensitivity value that they could tolerate. Second, the subjects were asked to find the best sensitivity. The tolerable range of sensitivity and the best sensitivity they found are shown in Table 1. The Figure 4 shows Table 1 using a diagram. The vertical lines represent each participant's range of the tolerable sensitivity, while the boxed value represents the best value for the subject.

Test Subject	Min Tolerance	Best Sensitivity	Max Tolerance
HJH	1.32	2.53	3.73
JSW	0.46	2.45	4.87
KJW	0.53	1.33	3.25
YSH	1.31	2.98	5.05
HHW	0.55	2.67	3.27

Table 1. Preference for sensitivity values.

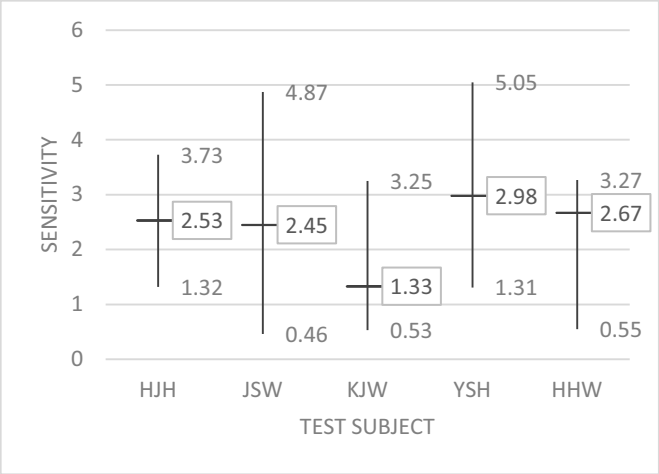

Figure 4. Diagram of preference for sensitivity values.

From Table 1, we found that the tolerable ranges and the most preferred sensitivity values vary for different participants. From this observation, we can make a conjecture that sensitivity configuration for each user is necessary. Second, the sensitivity values that the test subjects found most preferred were all greater than 1, which infers that our attempt of extrapolation is necessary for 3D gaming experience.

5. Conclusion

We implemented a feature that links 3D game world and real-world by using gyro sensor and enhanced the rotation experience by implementing the extrapolated Slerp technique. We conveyed a survey and found that the sensitivity configuration is necessary because users felt more comfortable when sensitivity is increased.

References

[1] Rollings, Andrew; Ernest Adams, "Fundamentals of Game Design", Prentice Hall, 2006

[2] Shoemake, K. "Animating rotation with quaternion curves." SIGGRAPH Computer Graphics 19, pp. 245-254, July 1985

[3] Unity 3D game engine, http://unity3d.com/ , accessed on March 2014

[4] Extrapolating Quaternion Rotation, http://answers.unity3d.com/questions/168779/extrapolating-quaternion-rotation.html , updated on September 2011, accessed on March 2014