OOP Project 3

-“Four-ball” Billiard-

Team members: 이종아, 김규리, 전승민

Presentation speaker : 김규리

Contents

**1. Summary3**

**2. How to Compile and Execute4**

**3. Functionalities5**

3.1 Score Display**5**

3.2 Player’s turn change**5**

3.3 Change the movement of a ball**5**

3.4 Camera**5**

3.5 Guide about movement of the ball**5**

3.6 Error situation handling**5**

**4. Important Implementation Issues7**

4.1 Collision by Wall**7**

4.2 Collision with Balls**8**

**5. Result of SW system DESIGN9**

**6. Execution Results10**

**7. OOP Concepts & What we felt and learned16**

**8. Conclusion17**

# Summary

In this project, we developed a simple simulation of a “four-ball” billiard game. The billiard table as the user sees it consists of a window containing a rectangle table with 2 red balls and 1 yellow, 1 white balls, and 1 blue balls.

Now explain how to play billiard game. You can move the blue ball by right click. The blue ball is the position where you want to hit and shift the location of white or yellow ball. When the blue ball is more and more far from white or yellow ball the power of hitting the ball will be increase. After you decide the position of the blue ball you can push the space bar and the white or yellow ball will move. When white or yellow ball moving if white or yellow ball hit the 2 red ball without colliding with another player’s ball. The player will get one point. But when the white or yellow ball is moving, if the player’s ball hit the another player’s ball the score decrease 10 point. Before hit the ball, if player want to know how the ball will move, player can push the ‘a’ key then the guideline will show how the ball will move. If player want to rotate camera totally, player can push ‘tab’ key then the camera will rotate totally. If the blue ball is disappeared from the screen, player can push ’s’ key then the position of the blue ball will reset to the center of the billiard table. When you want to finish the game, push esc button and it will be finish.

# How to Compile and Execute

To compile and execute, use Visual Studio 2015. Open .dsw file. And

Open the project and Click the ‘Property’

Click the 'VC++Directories' and you should set up ‘Include Directories’ and ‘Library Directories’.

Edit include Directory, Choose the folder (where SDK is installed.)\Include.

Edit Library Directory, Choose the folder (where SDK is installed.)\Lib\x86

In property C/C++ -> general change debug information format into another thing.

Add MyVector.h and MyVector.cpp

# Functionalities

## Score Display

When player success to get the score, after every ball stop the score will add the 10 point. But if player’s ball hit another player’s ball or doesn’t hit any red ball, score will subtract the 10 point.

## Player’s turn change

After every ball stop to move, another player’s ball is set to move.

## Change the movement of a ball

When ball moving, if ball collide with wall or another ball every ball will change it’s movement naturally.

## Camera

In default mode camera only move in y coordinate, so it can make camera’s rotation convenient. If user want to move another coordinate, just push the tab key and it will be move camera in another coordinate.

## Guide about movement of the ball

If player want to know how the ball moving, player can predict the movement of the ball by guide line. If player want to see the guide line, just pushing ‘a’ (If player stop to push ‘a’, the guideline will disappear) and there will be guide line on the screen. The total chance to see the guide line is 10. Once the player push ‘a’ key the count of chance will decrease one.

## Error situation handling

While the player move blue ball, mistakenly player move the blue ball out of the billiard table, the blue ball is disappeared. And then player can’t move the blue ball again. In this situation if player push ‘space bar’ the balls move very fast and don’t stop. To prevent this situation, when the blue ball disappears from the table, player just push ’s’ and the blue ball’s position will reset to the center of table.

# Important Implementation Issues

## Collision By Wall

When ball is collided with wall we should concentrate on vertical vector, Speed of ball. After the collision, the ball will move similar before collision, the ball’s movement angle is same as incidence angle by law of reflection. Also, the horizon vector is also same as before. But the vertical vector is changed reversely so the movement of a ball is changed reversely maintaining the incidence angle.

## Collision with Balls

(x2 , z2)

(x1 , z1)

V1

V2

T1

D1

D2

T2

V1 = D1 + T1

V2 = D2 + T2

When two balls collide each other (both are not spinning), the direction of the balls are changed. As seen in the above image, in order to determine the new directions, we need a normal vector that crosses both balls through the middle. For this, it is necessary to move the balls to a place of collision. To find the position of the collision of two balls, we used the following method

When two balls collide, we call Vec(B) which is vector connect the center of two balls(the direction is ‘Other ball’s coordinate’ – ‘My Ball`s coordinate’ ) and call Vec(V) which is speed vector about My ball’s . After the collision Ball’s speed vector (V’) is Other ball’s Vec(D) + My ball`s Vec(T).

Vec(D) is (orthogonal project V into B) \* (B’s unit vector Vec(D))( D= ((V∙B)/(B∙B))\*B).

Vec(T) is V – D. Vec(V`) is Other Ball`s Vec(D) + My Ball`s Vec(T) (V`=(Other)D+(My)T)

# Result of SW system DESIGN

Inheritance

MyVector

Trace

CWall

CLight

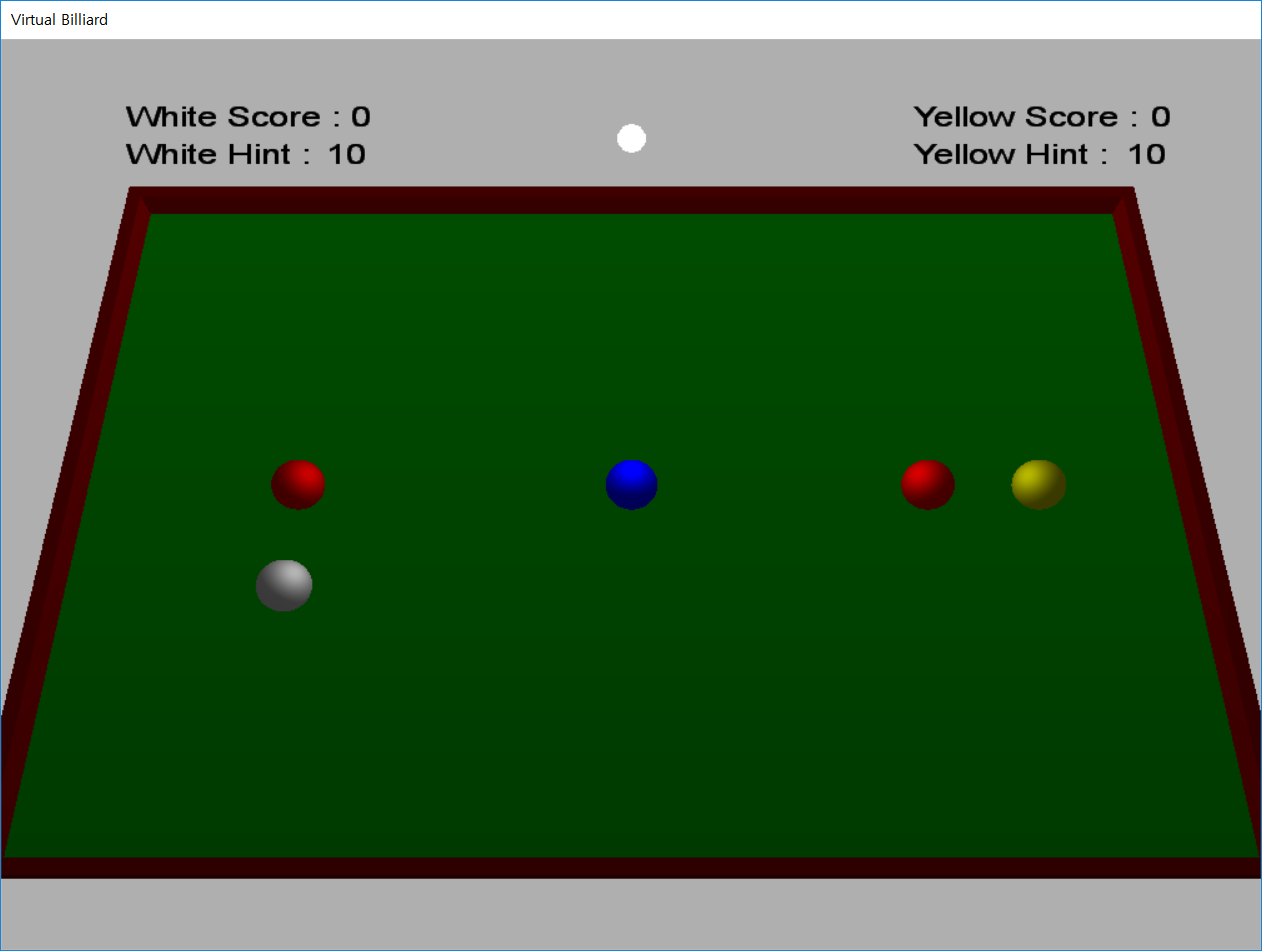
CSphere

virtualLego

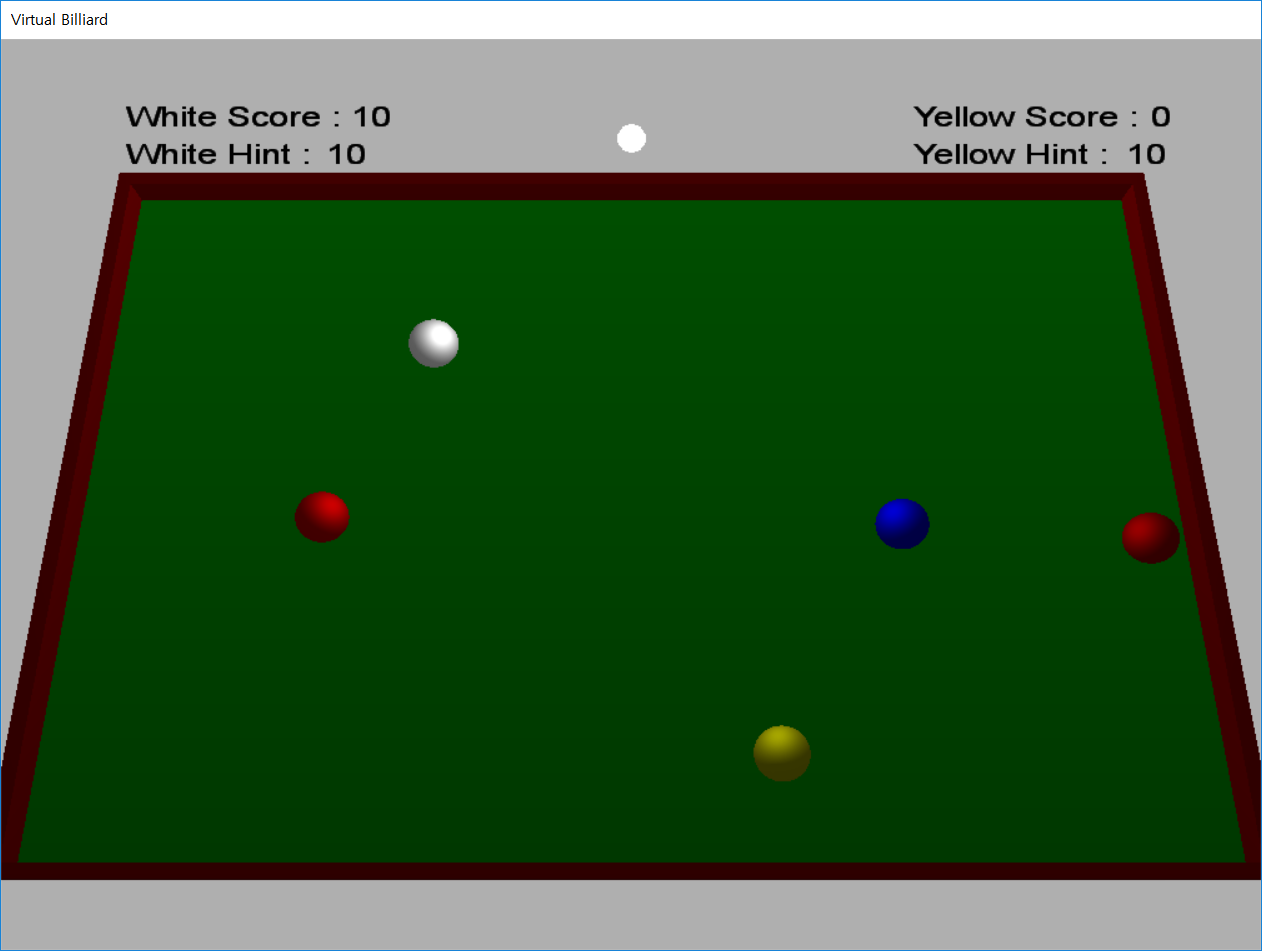
d3dUtillity

# Execution Results

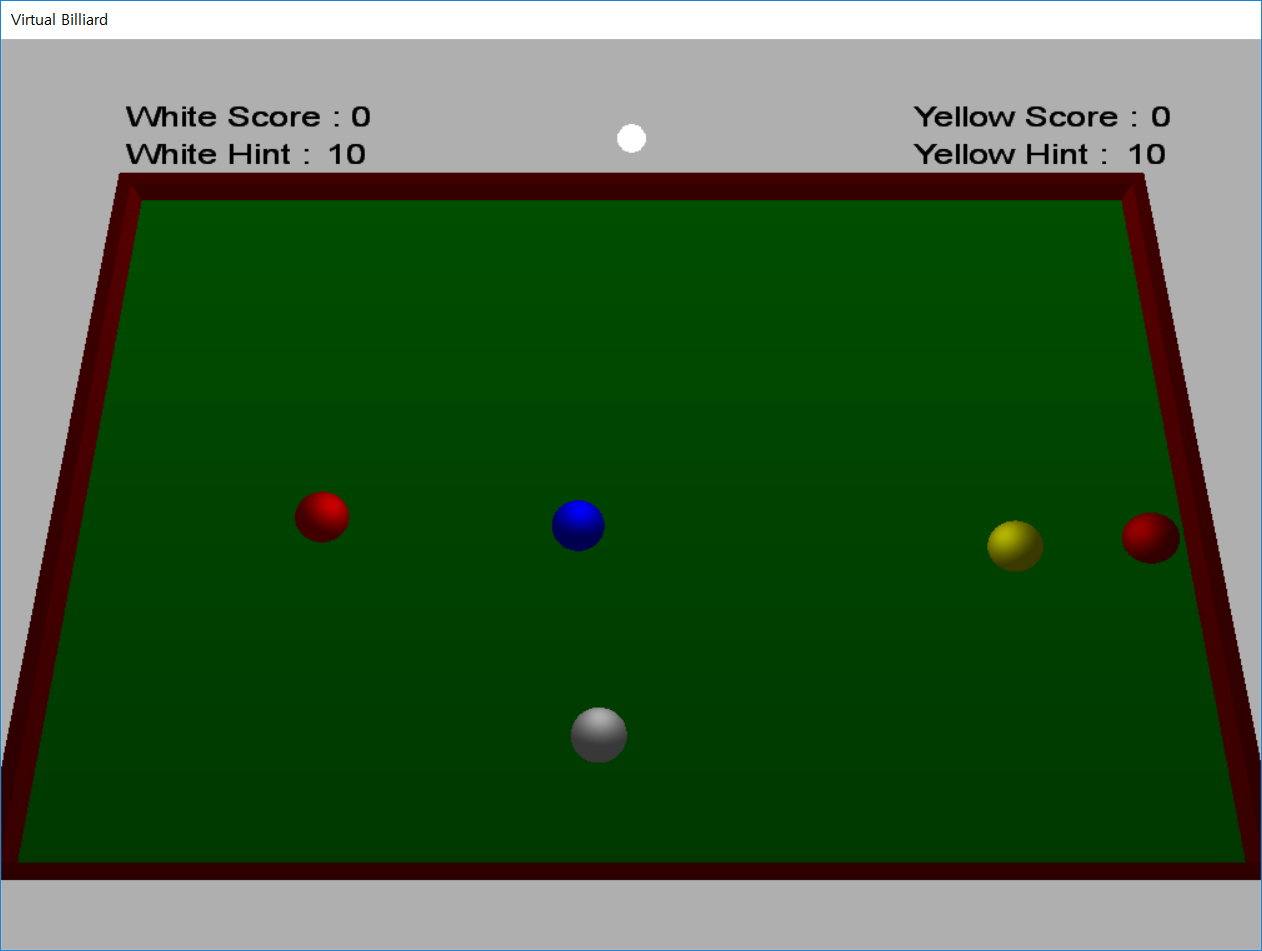
## Initial Screen Page



## When white ball player get points

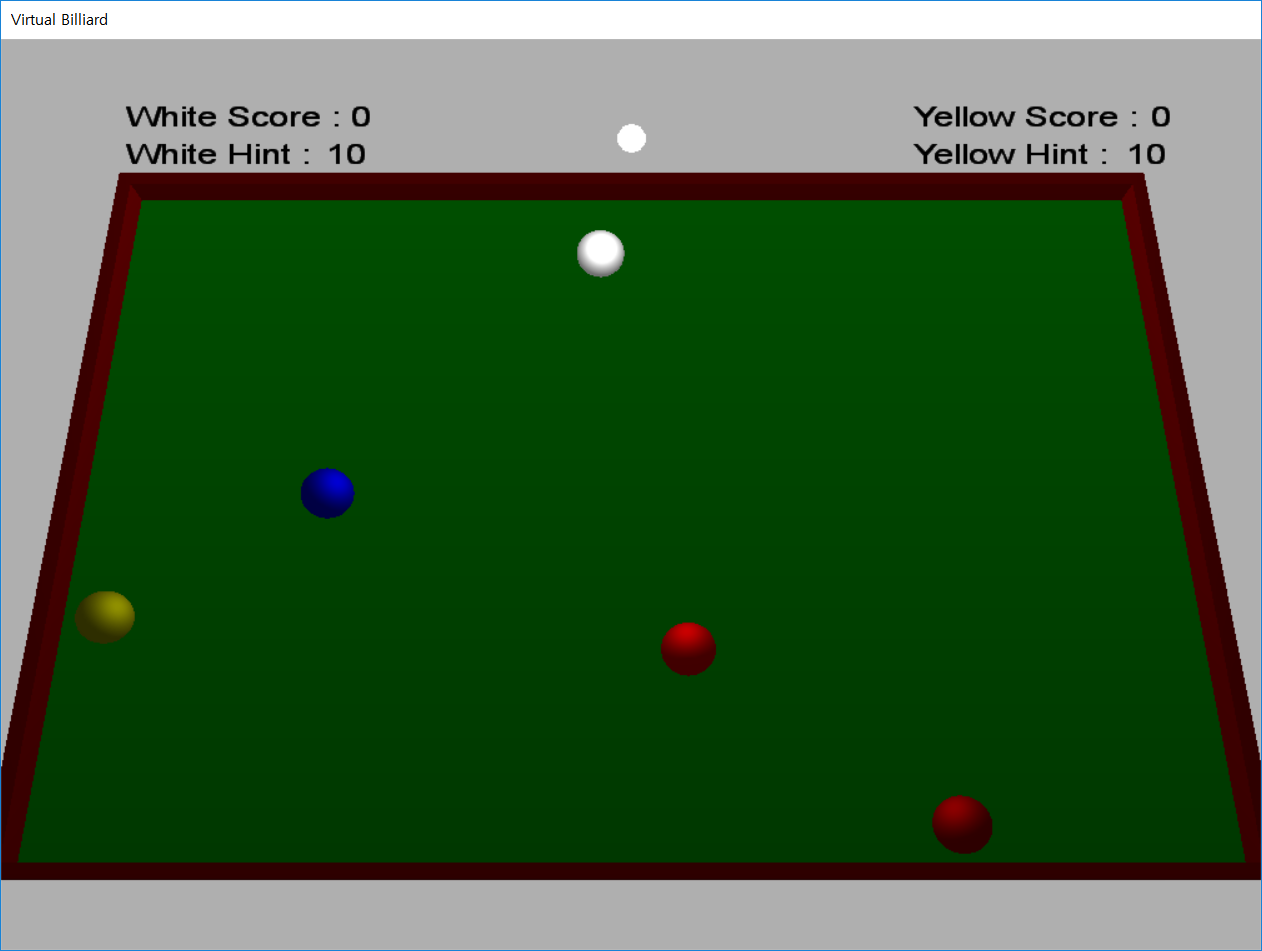


## After white ball player get point, in next turn result about player hit the yellow ball

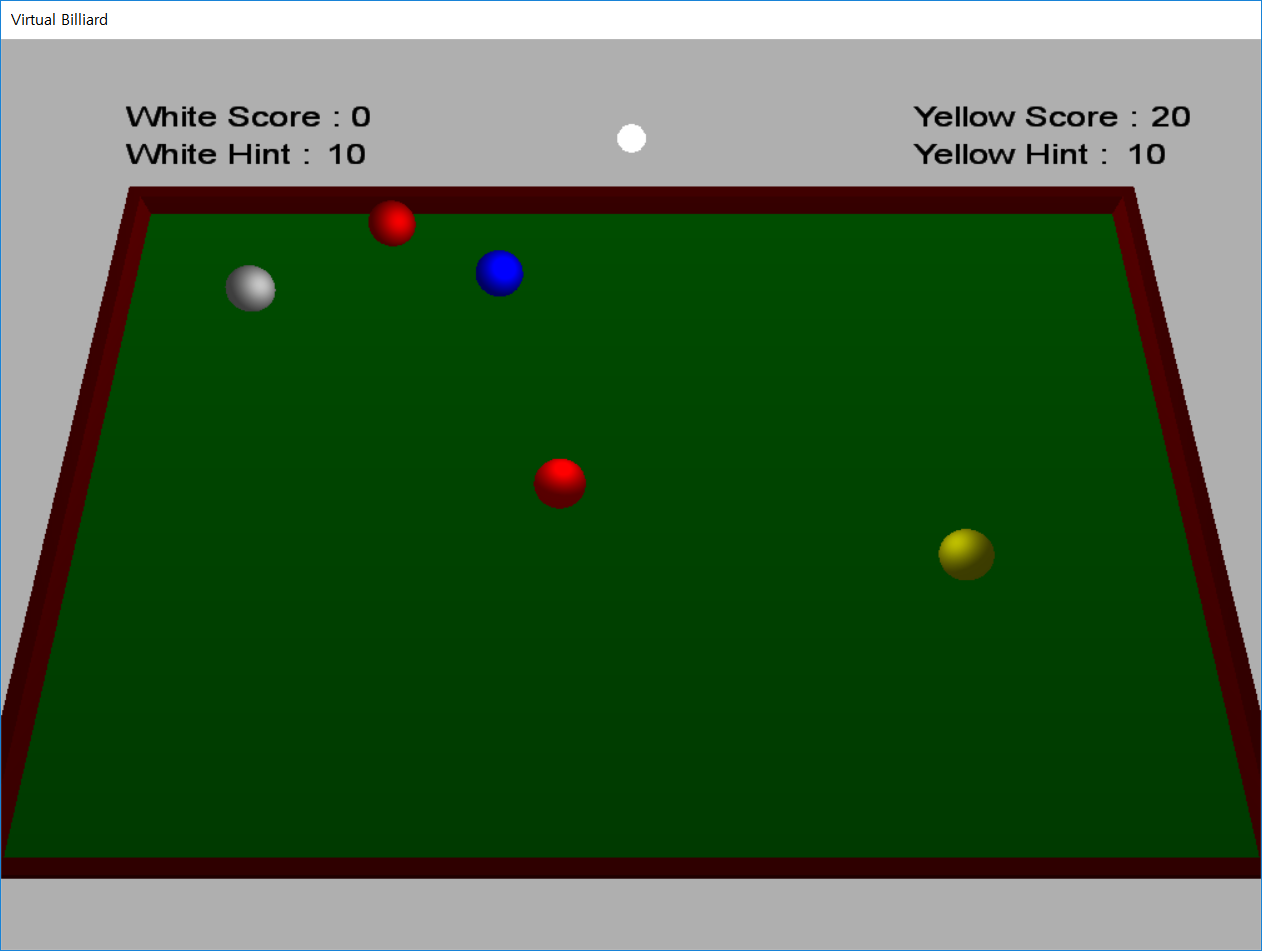


## After white ball player get point, in next turn result about player don’t hit any ball

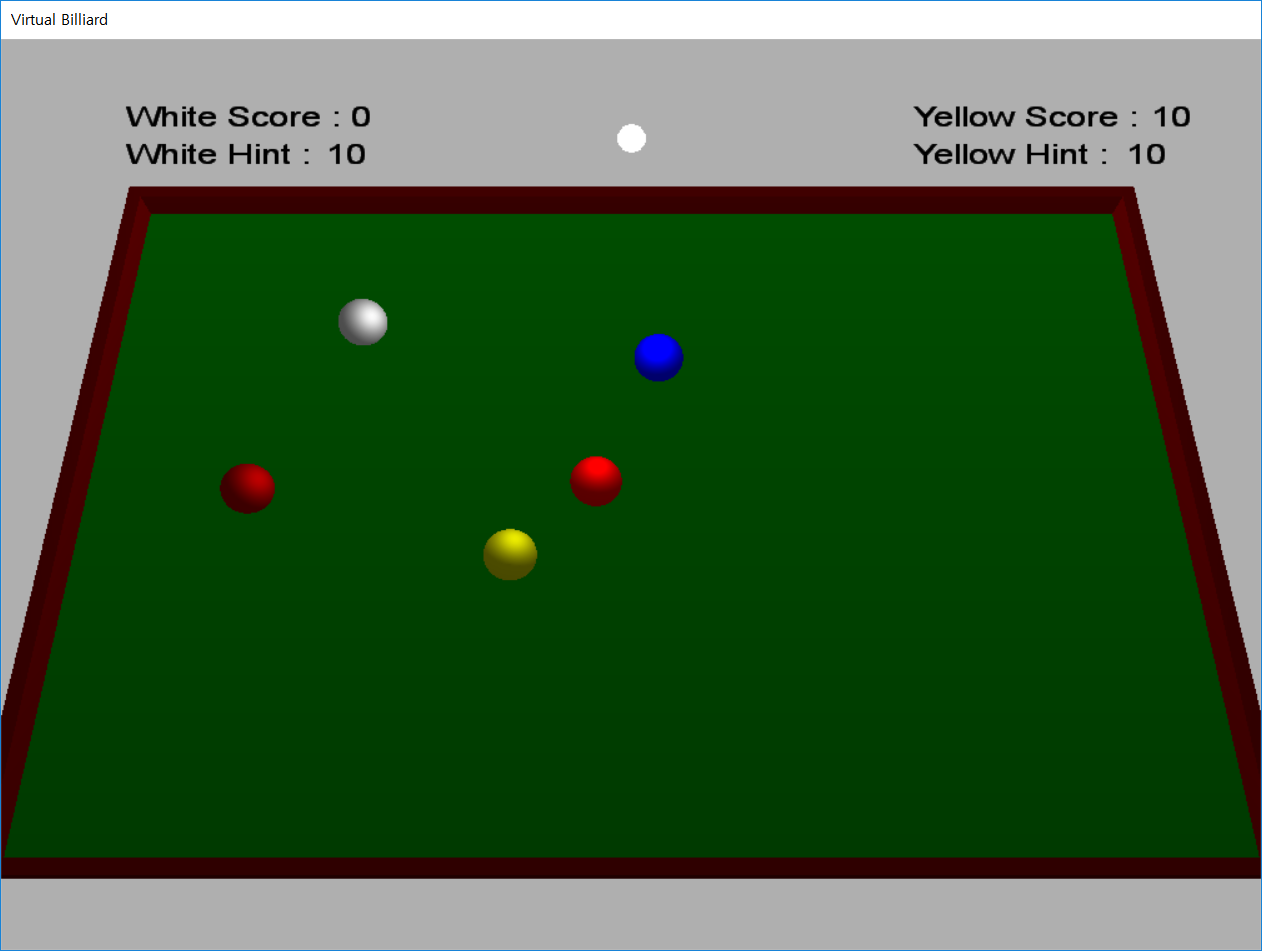




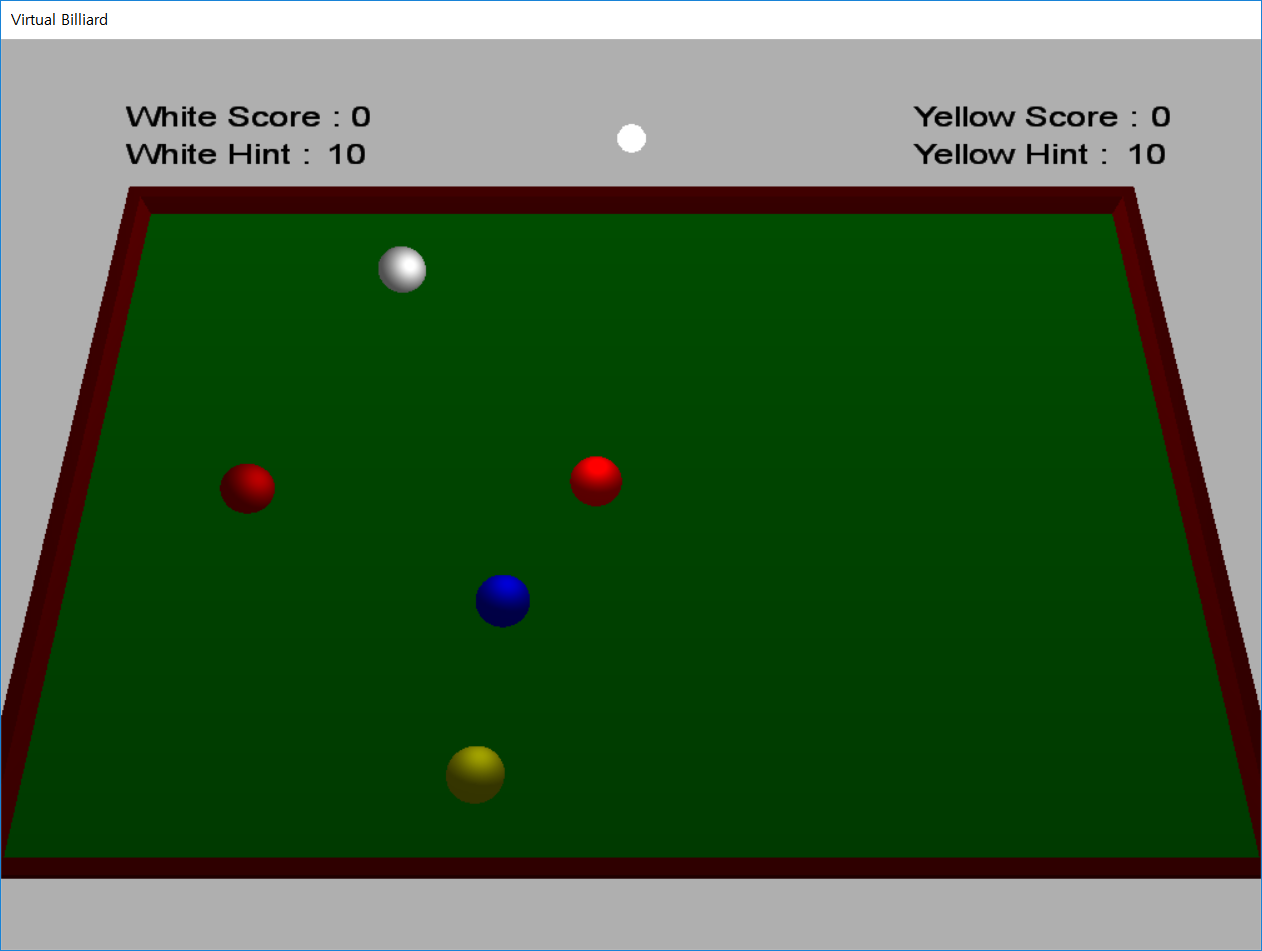
## When the yellow ball player get point.



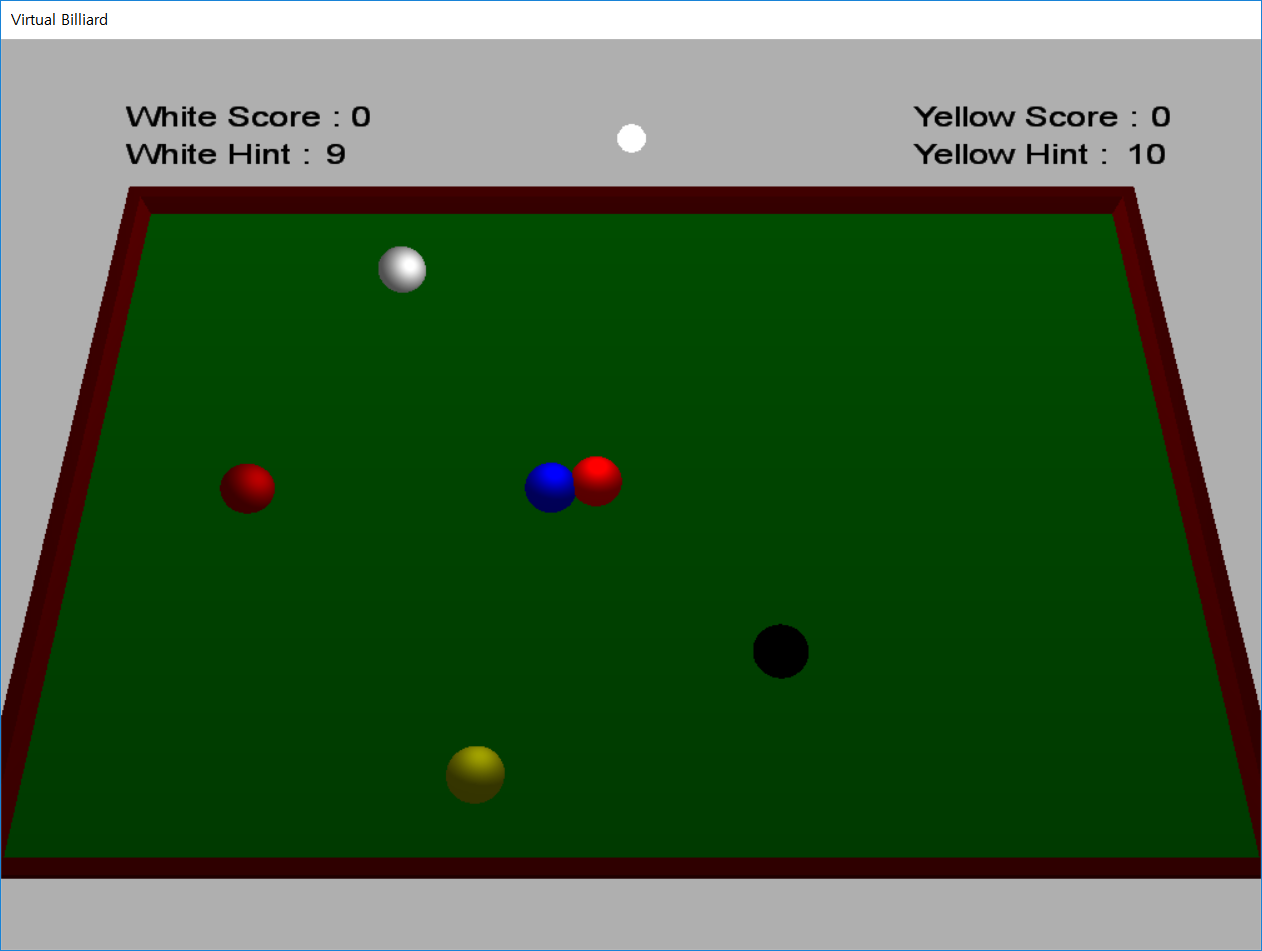
## After the yellow ball player get point, in next turn result about hit the white ball



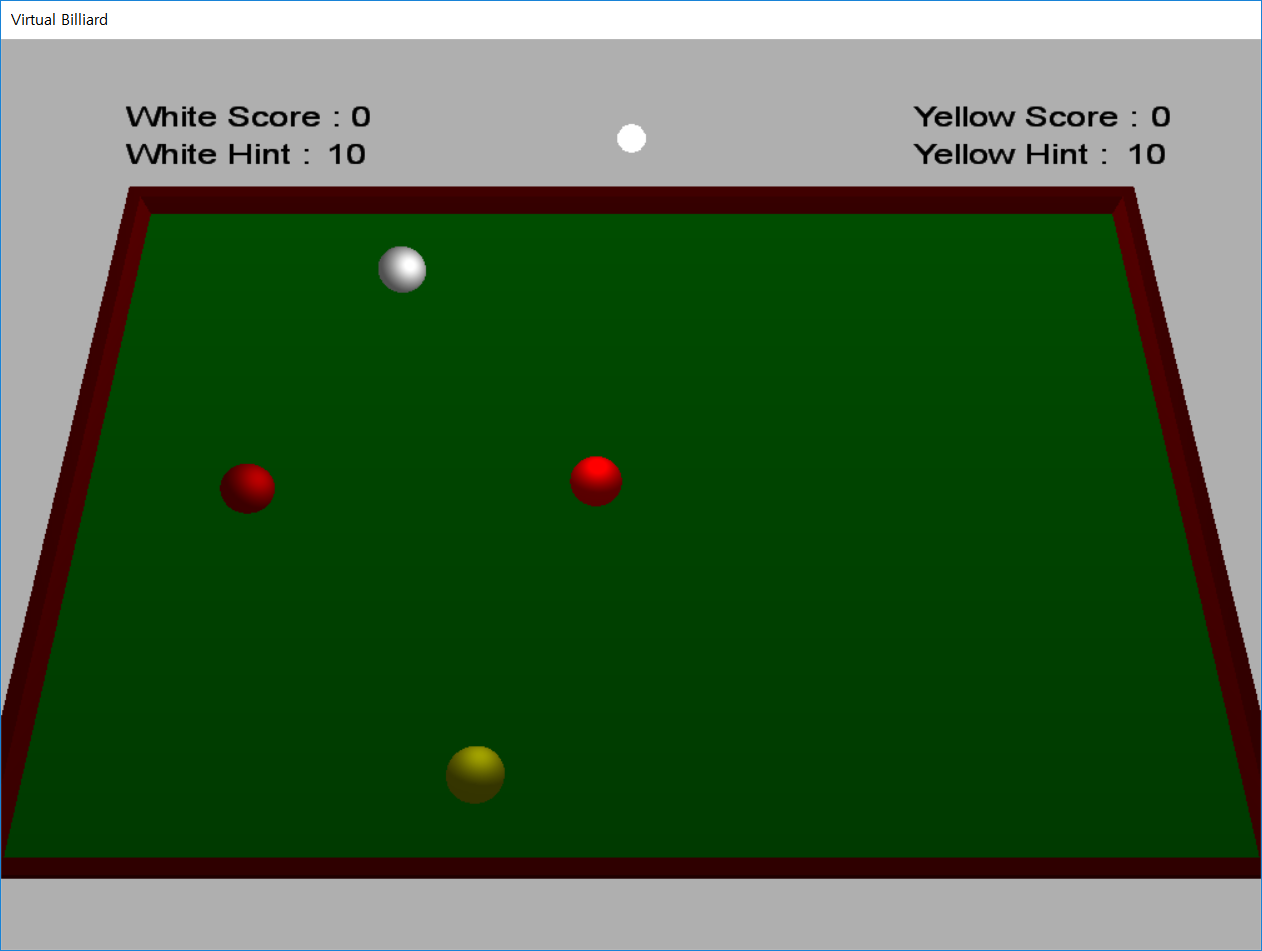
## After Yellow ball player get point, in next turn result about player don’t hit any ball

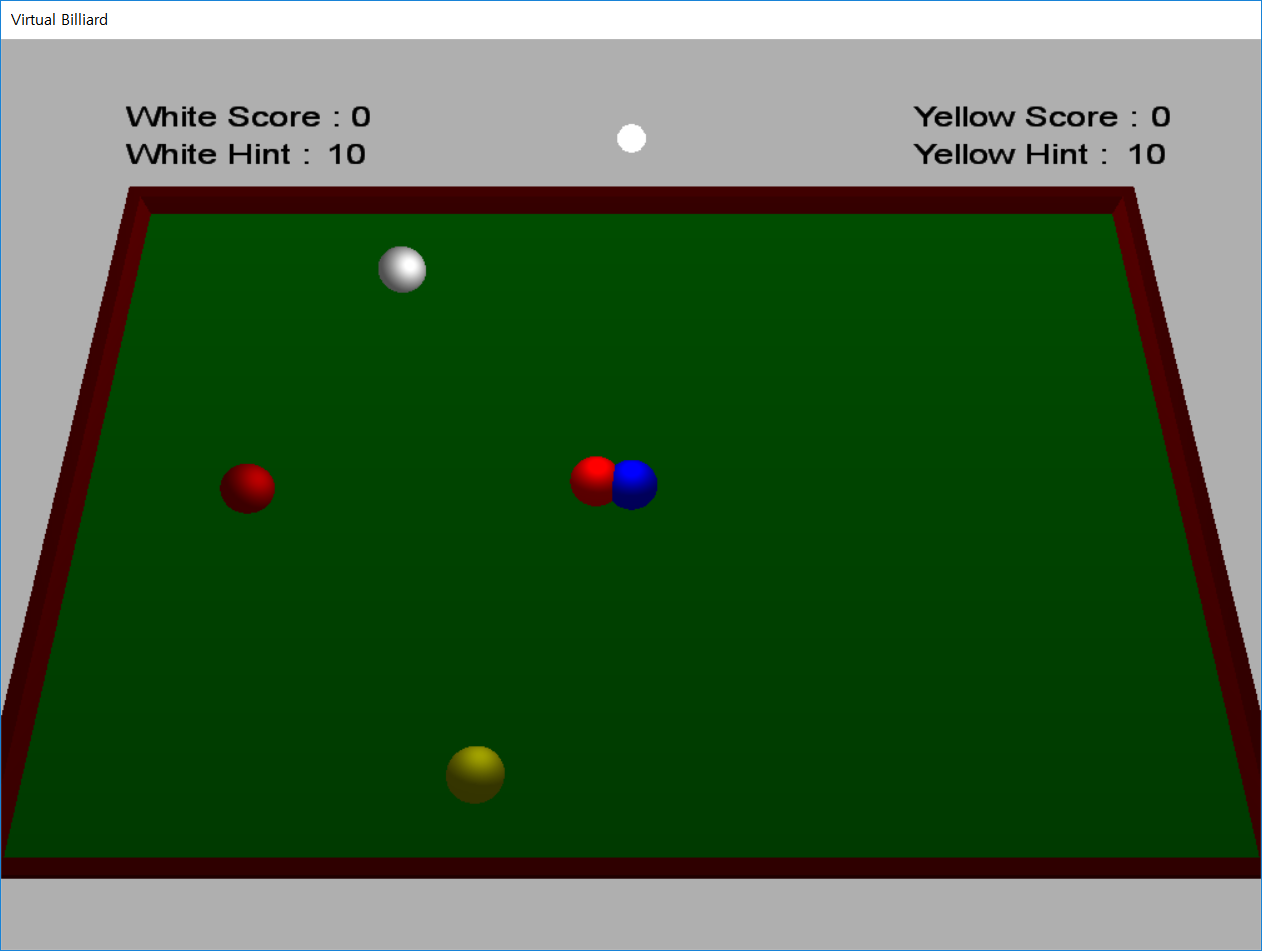


## Result of Pressing button ‘a’



## Result of Pressing button ‘s’





# OOP Concepts & What we felt and learned

Trace class is inherited from the CSphere. And hitBy function and SetDirection function make to the virtual function. So, we don’t need to edit another codes. And we make MyVector class to calculate easy about Ball’s velocity. And we do operator overloading matching with MyVector class. The other things are explained on the Picture of ‘Result of System Design’.

There are some hard things while we type the code.

First, the material on the screen has a shape but not realize on the table place so balls sometime overlapped. So, we need to compare the distance between two ball’s center, and ball’s diameter.

Second, when the ball collides with other ball or wall, the ball’s collide position is more forward than we expect. So the time of changing velocity also not match with our expectation, and then the ball just move forward not changing the direction and velocity. To solve this problem once we detect the ball’s collision, only that the ball is collide with other material, we set not to collide with same material again.

First we see the sample code, I think it will be very hard to understand the sample code. But thanks to the benefit of readability about data encapsulation, it’s easier to understand the code than what we expect. And then we think that the combine each other’s code will be very hard. Thanks to implementation independence about benefit of encapsulation, the code of each other doesn’t affect to another code. So it’s easier to combine the code than what we expect. We met everyday and doing homework everyday, so we could know each other more and more.

# Conclusion

We made project about "Four ball". "Four ball" is kind of billiard game with four balls and two players. We should convert real game to the 3D game by using DirectX. To convert we must study vector about control the movement of balls. Through a lot of trial and error we improve incommodity and add more functions what we need.

First, we add camera function. The sample code's camera movement is move through x, y axis so it's hard to move what we want. And in game, we know that we move y axis a lot of time more than x axis. So we change the setting. In default, the camera only move through y axis, when player push 'tab' key the camera will move through x, y axis.

Second, we add guideline system. The reason why we first add this system is what we didn't play well so we take very long time. So for us we decide to add guideline function. Thanks to guideline function we decrease the debugging time and players will easily play the game.

Through these process, we can complete the "Four ball" code.