CSCI 6555 Computer Animation

**Lab 3 Report**

Lingfeng Zhou

GWid: G38336549

Professor：James Hahn

The George Washington University

**I. Brief description of the system**

The system uses the geometric data of a set of balls and the physical properties of these balls as input, and output the animated view of the balls colliding with each other and the floor. Each ball’s position and velocity are initialized as three dimensions vector, and each ball has its mass.

This lab is based on lab0 given by professor. Additional functions are the initial of the balls, floor display, distance calculation between two balls, dot product calculation, vector normalization, collision detection and the movement of the balls.

**II. Description of code**

* Firstly, assign some global variables

// Number of balls

int BallNum

// old velocity of the balls

GLfloat OldVel[8][3]

// new velocity of the balls

GLfloat NewVel[8][3]

// old position of the balls

GLfloat OldPos[8][3]

// new position of the balls

GLfloat NewPos[8][3]

// matrix of each ball

GLfloat BallMatrix[8][16]

// Geometric data for the balls

GLfloat BM[16]

// gravity

GLfloat gravity[3]

// dt

GLfloat dt

// coefficient of restitution

GLfloat res

// initial position of each ball

GLfloat Position[10][3]

// mass of balls

GLfloat Mass[10]

* Initialize the balls, assign the position of each ball to its geometric data matrix

void init( void ) {

for (int i=0;i<BallNum;i++){

BallMatrix[i][0]=1.0f;

BallMatrix[i][5]=1.0f;

BallMatrix[i][10]=1.0f;

for (int j=0;j<3;j++){

BallMatrix[i][12+j]=Position[i][j];

OldPos[i][j]=BallMatrix[i][12+j];

}

BallMatrix[i][15]=1.0f;

}

}

* Then display the floor

void DisplayFloor()

* Then calculate distance between two balls for collision detect

GLfloat CalDistance(GLfloat B1[3],GLfloat B2[3])

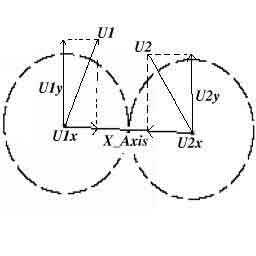
* Then calculate dot product of two vectors to find projection of one direction

GLfloat Dot(GLfloat V1[3],GLfloat V2[3])

* Before dot product, the vector should be normalized

void Normalise(GLfloat V[3])

* Then detect collisions between balls, handle the probable collision between ball n and other balls, the method is retrieved from <http://nehe.gamedev.net/tutorial/collision_detection/17005/>



U1 and U2 are the velocity vectors of the two spheres at the time of impact. There is an axis (X\_Axis) vector which joins the 2 centers of the spheres, and U1x, U2x are the projected vectors of the velocity vectors U1,U2 onto the axis (X\_Axis) vector.

U1y and U2y are the projected vectors of the velocity vectors U1,U2 onto the axis which is perpendicular to the X\_Axis. To find these vectors a few simple dot products are needed. M1, M2 is the mass of the two spheres respectively. V1,V2 are the new velocities after the impact, and V1x, V1y, V2x, V2y are the projections of the velocity vectors onto the X\_Axis.

In More Detail:

a) Find X\_Axis

X\_Axis = (center2 - center1);  
Unify X\_Axis, X\_Axis.unit();

b) Find Projections

U1x= X\_Axis \* (X\_Axis dot U1)  
U1y= U1 - U1x  
U2x =-X\_Axis \* (-X\_Axis dot U2)  
U2y =U2 - U2x

c) Find New Velocities

V1x= ((U1x \* M1)+(U2x\*M2)-(U1x-U2x)\*M2)/(M1+M2)  
V2x= ((U1x \* M1)+(U2x\*M2)-(U2x-U1x)\*M1)/( M1+M2)

d) Find The Final Velocities

V1y=U1y  
V2y=U2y  
V1=V1x+V1y  
V2=V2x+V2y

void DetectCollision(int n){

for (int i=n+1;i<BallNum;i++){

if (CalDistance(OldPos[n],OldPos[i])<1.01){

GLfloat X\_Axis[3];

GLfloat U1\_X[3], U1\_Y[3];

GLfloat U2\_X[3], U2\_Y[3];

for(int j=0;j<3;j++){

X\_Axis[j]=OldPos[i][j]-OldPos[n][j]; // find X\_Axis

}

Normalise(X\_Axis);

GLfloat a=Dot(X\_Axis,OldVel[n]);// find projection

for(int j=0;j<3;j++){

U1\_X[j]=a\*X\_Axis[j];// find projected vector for ball n

U1\_Y[j]=OldVel[n][j]-U1\_X[j];

}

// do the same as above

for(int j=0;j<3;j++){

X\_Axis[j]=OldPos[n][j]-OldPos[i][j];// find X\_Axis

}

Normalise(X\_Axis);

GLfloat b=Dot(X\_Axis,OldVel[i]);// find projection

for(int j=0;j<3;j++){

U2\_X[j]=b\*X\_Axis[j];// find projected vector for ball i

U2\_Y[j]=OldVel[i][j]-U2\_X[j];

}

// find new velocity

GLfloat V1\_X[3], V2\_X[3];

GLfloat V1\_Y[3], V2\_Y[3];

for(int j=0;j<3;j++){

V1\_X[j]=(U1\_X[j]\*Mass[n]+U2\_X[j]\*Mass[i]-(U1\_X[j]-U2\_X[j])\*Mass[i])/(Mass[n]+Mass[i]);

V2\_X[j]=(U1\_X[j]\*Mass[n]+U2\_X[j]\*Mass[i]-(U2\_X[j]-U1\_X[j])\*Mass[i])/(Mass[n]+Mass[i]);

V1\_Y[j]=U1\_Y[j];

V2\_Y[j]=U2\_Y[j];

OldVel[n][j]=V1\_X[j]+U1\_Y[j];// new velocity for ball n

OldVel[i][j]=V2\_X[j]+U2\_Y[j];// new velocity for ball i

}

}

}

}

void DetectCollision(int n)

* Then calculate the movement of one single ball n

void BallMovement(int n)

* Finally display the animation

void Display()