CSCI 6555 Computer Animation

**Lab 4 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 as boids and a set of control points to control the leading teapot as input, and output the animated view of the movement of boids following the leading teapot in a 3D space. Use B spline to control the movement of the leading teapot. Each ball’s position and velocity are initialized as three dimensions vector, and the boids follow four rules which will affect behaviors: collision avoidance, velocity matching, flock centering and leading following rules.

This lab is based on lab1 and lab3. The control of leading teapot is just the same as lab1, and part of the display of balls is provided in lab3. Additional functions are used to achieve the movement of the boids and the four rules as introduced above.

**II. Description of code**

* Firstly, assign some global variables

// t parameter

GLfloat t=0;

// Number of balls

int BallNum=8;

// velocities of boids

GLfloat BoidVel[8][3]={0};

// positions of boids

GLfloat BoidPos[8][3]={0};

// matrix of each ball

GLfloat BallMatrix[8][16]={0};

// matrix of lead teapot

GLfloat TeapotMatrix[16]={0};

// Geometric data for the balls

GLfloat BM[16]={0};

// initial position of each ball

GLfloat Position[8][3]={{2.0f,5.5f,1.5f},{2.0f,8.5f,1.5f},{2.3f,7.0f,1.6f},{4.0f,5.0f,2.5f},{4.8f,7.0f,2.5f},{1.0f,7.0f,2.0f},{0.5f,3.2f,1.8f},{3.0f,4.6f,1.8f}};

// B Spline matrix

GLfloat BSplineMatrix[16]={-1.0f/6.0f,0.5f,-0.5f,1.0f/6.0f,0.5f,-1.0f,0,4.0f/6.0f,-0.5f,0.5f,0.5f,1.0f/6.0f,1.0f/6.0f,0,0,0};

// # of current point

int PointNumber = 0;

// point number

int PNumber = 5;

// The matrix to get Q=T·M·G

GLfloat Q[7] = {0};

// 4 parameters for Orientations and 3 for position given by quaternion

GLfloat Quaternion[5][7]={{1,0,0,0,-10,0,-25},{0,0,1,0,10,0,-25},{1,0,0,0,10,0,-5},{1,0,0,0,-10,0,-5},{1,0,0,0,-10,0,-15}};

// velocities after four apply four rules to affect behaviours

GLfloat R1Vel[3]={0};

GLfloat R2Vel[3]={0};

GLfloat R3Vel[3]={0};

GLfloat R4Vel[3]={0};

* Init function is same as lab3, 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;

}

}

* The following functions are same as lab1, to control the movement of leading teapot.

// comput Q=T·M·G

GLfloat ComputeQ(GLfloat T[4],GLfloat M[16],GLfloat G[4])

// nomalise the quanternion q, divide by |q|

void Normalize(GLfloat ParameterArray[7])

// Quaternion corresponding transform matrix, rotation of w around vector (x,y,z)

void getMatrix (GLfloat ParameterArray[7],GLfloat Matrix[16])

// Quaternion interpolation, MatrixType is BSplineMatrix in this lab

void Interpolation (GLfloat MatrixType[16],GLfloat Object[5][7],GLfloat Q[7])

* Then calculate distance between two balls for collision avoidance rule

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

* The collision avoidance rule, establish minimum required separation distance

void CARule(int N){

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

if (i!=N){

if (CalDistance(BoidPos[N],BoidPos[i])<4.5f){

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

R1Vel[j]=(R1Vel[j]-(BoidPos[i][j]-BoidPos[N][j]))/2000;

}

}

}

}

}

* The velocity matching rule, boids try to match velocity with nearby boids

void VMRule(int N){

GLfloat avg[3]={0};

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

avg[0]+=BoidVel[i][0];

avg[1]+=BoidVel[i][1];

avg[2]+=BoidVel[i][2];

}

// compute the average velocity of nearby boids

avg[0]=(avg[0]-BoidVel[N][0])/(BallNum-1);

avg[1]=(avg[1]-BoidVel[N][1])/(BallNum-1);

avg[2]=(avg[1]-BoidVel[N][2])/(BallNum-1);

// set the velocity toward the average velocity of nearby boids

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

R2Vel[i]=(avg[i]-BoidVel[N][i])/2000;

}

}

* The flock centering rule, boids try to move toward the center of mass of nearby boids

void FCRule(int N){

GLfloat center[3]={0};

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

center[0]+=BallMatrix[i][12];

center[1]+=BallMatrix[i][13];

center[2]+=BallMatrix[i][14];

}

// compute the centre of mass of nearby boids

center[0]=(center[0]-BallMatrix[N][12])/(BallNum-1);

center[1]=(center[1]-BallMatrix[N][13])/(BallNum-1);

center[2]=(center[1]-BallMatrix[N][14])/(BallNum-1);

// set the velocity toward the center

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

R3Vel[i]=(center[i]-BallMatrix[N][12+i])/5000;

}

}

* The leader following rule, boids try to follow the leading teapot

void LFRule(int N){

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

R4Vel[i]=(TeapotMatrix[12+i]-BallMatrix[N][12+i])/3000;

}

}

* After applying these four rules, calculate the movement of the boids

void BoidsMovement(){

GLfloat V1[3]={0};

GLfloat V2[3]={0};

GLfloat V3[3]={0};

GLfloat V4[3]={0};

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

CARule(i);// apply collision avoidance rule

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

V1[j]=R1Vel[j];

}

VMRule(i);// apply velocity matching rule

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

V2[j]=R2Vel[j];

}

FCRule(i);// apply flock centering rule

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

V3[j]=R3Vel[j];

}

LFRule(i);// apply leader following rule

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

V4[j]=R4Vel[j];

}

// compute the finally velocity and position of the boids

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

BoidVel[i][j]=BoidVel[i][j]+V1[j]+V2[j]+V3[j]+V4[j];

BoidPos[i][j]=BoidPos[i][j]+BoidVel[i][j]\*0.05;

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

}

}

}

* Display single ball and the teapot

void DisplayBall (int N){

glPushMatrix();

for (int i=0;i<16;i++){

BM[i]=BallMatrix[N][i];

}

glMultMatrixf(BM);

glutSolidSphere(0.5,15,15);

glPopMatrix();

}

void DisplayTeapot (){

glPushMatrix();

Interpolation(BSplineMatrix,Quaternion,Q);

glMultMatrixf(TeapotMatrix);

glColor3f(1.0f, 0.0f, 0.0f);

glutSolidTeapot(1.0);

glPopMatrix();

}

* Finally display the animation

void Display(){

DisplayTeapot ();

BoidsMovement();

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

DisplayBall (i);

}

}