**Catch the Bird**

**Program:**

game.cpp

#include<GL/glut.h>

#include<iostream>

#include<cmath>

#include <stdlib.h>

#include <string>

#include <time.h>

#include <stdio.h>

#include"fish.h"

#include<list>

#include"bird.h"

using namespace std;

int flag=0;

Bird b(0.5,380);

int number=b.getpoints();

char s[]=": Score ";

char score[15];

int maxi=0;

//output score

// default font

void \*font = GLUT\_BITMAP\_TIMES\_ROMAN\_24;

void renderBitmapString(

float x,

float y,

float z,

void \*font,

char \*str) {

char \*c;

glRasterPos3f(x, y,z);

for (c=str; \*c != '\0'; c++) {

glutBitmapCharacter(font, \*c);

}

}

void drawcloud()

{

glScalef(20,20,0);

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

glBegin(GL\_POLYGON);

glVertex2f(0,2);

glVertex2f(1,3);

glVertex2f(2,2);

glVertex2f(1,1);

glEnd();

glBegin(GL\_POLYGON);

glVertex2f(1,2);

glVertex2f(2,3);

glVertex2f(3,2);

glVertex2f(2,1);

glEnd();

glBegin(GL\_POLYGON);

glVertex2f(2,2);

glVertex2f(3,3);

glVertex2f(4,2);

glVertex2f(3,1);

glEnd();

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

glBegin(GL\_LINE\_LOOP);

glVertex2f(0,2);

glVertex2f(1,3);

glVertex2f(1.5,2.5);

glVertex2f(2,3);

glVertex2f(2.5,2.5);

glVertex2f(3,3);

glVertex2f(4,2);

glVertex2f(3,1);

glVertex2f(2.5,1.5);

glVertex2f(2,1);

glVertex2f(1.5,1.5);

glVertex2f(1,1);

glEnd();

}

void loadEnvironment()

{

//drawing pond

glColor3f(0.6f, 0.8f, 1.0f);

glBegin(GL\_LINES);

glVertex2d(0,100);

glVertex2d(640,100);

glEnd();

glColor3f(0.0f, 0.0f, 0.6f);

glBegin(GL\_POLYGON);

glVertex2d(0,0);

glVertex2d(0,100);

glVertex2d(640,100);

glVertex2d(640,0);

glEnd();

// drawing cloud

for(int i=10;i<=640;i+=200)

{

glPushMatrix();

glTranslatef(i,420,0);

drawcloud();

glPopMatrix();

}

}

Fish f1(1,42,0,1,1);

Fish f2(300,52,1,0,1);

Fish f3(620,38,0.1,1,0.1);

Fish f4(147,22,0.3,0.5,0.3);

Fish f5(500,47,1,0,0.5);

Fish f6(550,39,1,1,0);

Fish f7;

Fish f8;

Fish f9;

Fish f10;

Fish \*f[] = {&f1,&f2,&f3,&f4,&f5,&f6,&f7,&f8,&f9,&f10};

list<Fish \*> fishes (f, f + sizeof(f) / sizeof(Fish) );

void renderScene(void)

{

if(b.getpoints()>=0)

{

glClearColor(0,0.7,0.7,1);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

loadEnvironment();

// render bird

b.drawBird();

b.roam();

//fishes

f1.drawFish();

f1.roam();

f2.roam();

f2.drawFish();

f3.drawFish();

f3.roam();

f4.drawFish();

f4.roam();

f5.drawFish();

f5.roam();

f6.drawFish();

f6.roam();

f7.drawFish();

f7.roam();

f8.drawFish();

f8.roam();

f9.drawFish();

f9.roam();

f10.drawFish();

f10.roam();

/\*

\* Display Score

\*/

glColor3f(1,1,0.5);

number=b.getpoints();

if(number>maxi){maxi=number;}

sprintf(score,"%s%d :",s,number);

renderBitmapString(500.0f, 400.5f, 0.0f, (void \*)font ,score);

}

else

{

glClearColor(0,0,0,1);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glColor3f(1,1,1);

sprintf(score,"MAX Score : %d ",maxi);

renderBitmapString(250.0f, 400.5f, 0.0f, (void \*)font ,score);

renderBitmapString(250.0f, 240.0f, 0.0f, (void \*)font ,"GAME OVER");

renderBitmapString(230.0f, 100.0f, 0.0f, (void \*)font ,"\t\t\t\t\tDeveloped By");

renderBitmapString(180.0f, 70.0f, 0.0f, (void \*)font ,"\t\tKiran Kilingar Nadumane");

glutIdleFunc(NULL);

}

glutSwapBuffers();

}

void changeSize(int w, int h) {

// Prevent a divide by zero, when window is too short

// (you cant make a window of zero width).

if(h == 0)

h = 1;

float ratio = 1.0\* w / h;

// Use the Projection Matrix

glMatrixMode(GL\_PROJECTION);

// Reset Matrix

glLoadIdentity();

// Set the viewport to be the entire window

glViewport(0, 0, w, h);

// Set the correct perspective.

gluPerspective(45,ratio,1,1000);

// Get Back to the Modelview

glMatrixMode(GL\_MODELVIEW);

}

void processNormalKeys(unsigned char key,int x,int y)

{

if(key == 27)

exit(0);

}

void processMouse(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN && (480-y)>120 && (480-y)<380)

{

b.fillFishBucket(fishes);

b.roam(x,480-y);

}

glutPostRedisplay();

}

int main(int argc,char \*\*argv)

{

//initialize glut

glutInit(&argc,argv);

glutInitWindowSize(640,480);

//glutInitWindowPosition(0,0);

glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH)-640)/2,(glutGet(GLUT\_SCREEN\_HEIGHT)-480)/2);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DEPTH | GLUT\_DOUBLE);

glutReshapeFunc(changeSize);

glutCreateWindow("Catch the Fish");

gluOrtho2D(0,640,0,480);

//write callback functions

glutDisplayFunc(renderScene);

glutKeyboardFunc(processNormalKeys);

glutIdleFunc(renderScene);

glutMouseFunc(processMouse);

//keep it in loop

glutMainLoop();

}

bird.h

static float remx,remy;

class Bird

{

float xpos;

float ypos;

float scalefactorx;

float scalefactory;

int flag;

int reached;

float theta;

int g;

int points;

std::list <Fish \*> fishes;

public:

Bird()

{

xpos = 0;

ypos = 0;

scalefactorx=8;

scalefactory=8;

flag=0;

reached =0;

g=0;

points=0;

}

Bird(float x,float y)

{

xpos=x;

ypos=y;

scalefactorx=8;

scalefactory=8;

flag=0;

reached=0;

g=0;

points=0;

}

void change\_sf(float fx,float fy)

{

scalefactorx=fx;

scalefactory=fy;

}

void reverse\_sf()

{

scalefactorx=-scalefactorx;

}

void cal\_theta(float t)

{

theta = t;

}

void fillFishBucket(std::list <Fish \*> f)

{

fishes = f;

}

void drawBird()

{

//head

glPushMatrix();

glTranslatef(xpos,ypos,0);

glScalef(scalefactorx,scalefactory,0);

glRotatef(theta,0,0,1);

glColor3f(1,1,0.5);

glBegin(GL\_POLYGON);

glVertex2d(1,3);

glVertex2d(2,4);

glVertex2d(3,4);

glVertex2d(4,3);

glVertex2d(3,2);

glVertex2d(2,2);

glEnd();

//eyes

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

glBegin(GL\_POLYGON);

glVertex2f(2.2,2.8);

glVertex2f(2.2,3.2);

glVertex2f(2.6,3.2);

glVertex2f(2.6,2.8);

glEnd();

//beak

glColor3f(1,0.4,0);

glBegin(GL\_TRIANGLES);

glVertex2d(4,3);

glVertex2d(5,2);

glVertex2d(3,2);

glEnd();

//body

glColor3f(1,1,0.5);

glBegin(GL\_TRIANGLES);

glVertex2d(0,2);

glVertex2d(3,2);

glVertex2d(1,0);

glEnd();

//feather

glColor3f(0.9f,0.9f,0.0f);

glBegin(GL\_POLYGON);

glVertex2f(0.8,1);

glVertex2f(0.8,1.5);

glVertex2f(1.2,1.5);

glVertex2f(1.2,1);

glEnd();

//leg

glColor3f(1,0,0);

glBegin(GL\_LINES);

glVertex2d(1,0);

glVertex2d(2,0);

glEnd();

glPopMatrix();

}

void roam(float xin=0,float yin=0)

{

if(xin!=0 && yin!=0) {g=1;remx=xin;remy=yin;}

if(g==1){xin=remx;yin=remy;}

if(g==0) // normal path

{

cal\_theta(-20);

if(xpos>=620)

{

flag=1;

change\_sf(-8,8);

}

if(xpos<=0)

{

flag=0;

change\_sf(8,8);

}

if(flag==0)

{

xpos = xpos+0.06;

ypos = ypos-sin(xpos\*10);

}

if(flag==1)

{

xpos = xpos-0.06;

ypos = ypos+sin(xpos\*10);

}

}

else //glide path

{

glide(xin,yin);

cal\_theta(-85);

if(scalefactorx<0 && xpos<xin || scalefactorx>0 && xpos>xin)reverse\_sf();

}

}

void glide(float xin,float yin)

{

float X1,Y1,X2,Y2,xInc=0,yInc=0;

double dx,dy,steps;

if(reached==0)

{

X1=xpos;Y1=ypos;X2=xin;Y2=yin;

dx=(X2-X1);

dy=(Y2-Y1);

steps=(abs(dx)>abs(dy))?abs(dx):abs(dy);/\* Find out whether to increment x or y\*/

xInc=dx/(float)steps;

yInc=dy/(float)steps;

xpos+=(xInc);

ypos+=(yInc);

if(xpos==xin && ypos==yin)

{

reached=1;

catch\_fish();

}

}

if(reached==1)

{

X1=xpos;Y1=ypos;

if(xpos<320){X2=0;Y2=380;}

if(xpos>=320){X2=640;Y2=380;}

dx=(X2-X1);

dy=(Y2-Y1);

steps=(abs(dx)>abs(dy))?abs(dx):abs(dy);/\* Find out whether to increment x or y\*/

xInc=dx/(float)steps;

yInc=dy/(float)steps;

xpos+=xInc;

ypos+=yInc;

if((xpos==0 && ypos==380) || (xpos==640 && ypos==380)){reached=0;g=0;};

}

}

int getpoints()

{

return points;

}

void catch\_fish()

{

float x,y;

for (std::list<Fish \*>::iterator it = fishes.begin(); it != fishes.end(); it++)

{

Fish \*f = \*it;

x = f->getxpos();

y = f->getypos();

//std::cout<<x<<","<<y<<std::endl;

if(abs(x-xpos)<50 && abs(y-ypos)<50)

{

points++;

f->resetfish();

}

else

{

points=points-2;

}

}

}

};

fish.h

static float t=0;

class Fish

{

float xpos;

float ypos;

float scalefactorx;

float scalefactory;

int flag;

float r,g,b;

float rem;

float range;

float resetx,resety;

public:

Fish()

{

xpos = rand() % 620;

ypos = rand() % 90;

resetx=1;

resety=0;

scalefactorx=4;

scalefactory=4;

flag=0;

r=((double) rand() / (RAND\_MAX));

g=((double) rand() / (RAND\_MAX));

b=((double) rand() / (RAND\_MAX));

srand (time(NULL));

range = 360;

}

Fish(float x,float y)

{

xpos=x;

ypos=y;

resetx=x;

resety=y;

scalefactorx=4;

scalefactory=4;

flag=0;

r=((double) rand() / (RAND\_MAX));

g=((double) rand() / (RAND\_MAX));

b=((double) rand() / (RAND\_MAX));

rem=y;

srand (time(NULL));

range=360;

}

Fish(float x,float y,float red,float green,float blue)

{

xpos=x;

ypos=y;

resetx=x;

resety=y;

scalefactorx=4;

scalefactory=4;

flag=0;

r=red;

g=green;

b=blue;

rem=y;

srand (time(NULL));

range=360;

}

void resetfish()

{

xpos=resetx;

ypos=resety;

r=((double) rand() / (RAND\_MAX));

g=((double) rand() / (RAND\_MAX));

b=((double) rand() / (RAND\_MAX));

}

float getxpos(){return xpos;}

float getypos(){return ypos;}

void change\_sf(float fx,float fy)

{

scalefactorx=fx;

scalefactory=fy;

}

void change\_color(float red,float green,float blue)

{

g=green;

r=red;

b=blue;

}

void drawFish()

{

//head

glPushMatrix();

glTranslatef(xpos,ypos,0);

glScalef(scalefactorx,scalefactory,0);

glColor3f(r,g,b);

glBegin(GL\_POLYGON);

glVertex2d(1,1);

glVertex2d(2,2);

glVertex2d(3,1);

glVertex2d(2,0);

glEnd();

//eyes

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

glBegin(GL\_POLYGON);

glVertex2f(2,1);

glVertex2f(2,1.5);

glVertex2f(2.5,1.5);

glVertex2f(2.5,1);

glEnd();

//body

glColor3f(r,g,b);

glBegin(GL\_POLYGON);

glVertex2f(1,1);

glVertex2f(0.7,1.5);

glVertex2f(0.5,1);

glVertex2f(0.7,0.5);

glEnd();

glPopMatrix();

}

void jump(float x, float theta, float r)

{

float xe = (x>r)? 0:x;

ypos = (rem+xe\*tan(theta)\*(1-(xe/r)));

}

void roam()

{

if(xpos>=620)

{

flag=1;

change\_sf(-scalefactorx,scalefactory);

}

if(xpos<=0)

{

flag=0;

change\_sf(-scalefactorx,scalefactory);

range= rand() % 620 + 1;

}

if(flag==0)

{

xpos = xpos+0.05;

jump(xpos,45,range);

}

if(flag==1)

{

xpos = xpos-0.05;

}

}

};

**Screenshots:**







