CG homework : 배지어 곡선과 친해지기

1924073 최유현

코드설명

OpenGL을 사용하여 베지어 곡선을 그렸고, 이러한 곡선에 대한 충돌 박스인 AABB를 계산하여 화면상에 띄우는 프로그램을 작성해보았습니다.

프로그램을 실행시킨 사용자는 마우스를 통해 제어 점을 이동할 수 있습니다.

BezierCurve2d 로 네가지 제어점을 받아서 베지어 곡선을 생성합니다.

Display함수로 베지어 곡선과 AABB를 그립니다.

Mouse, mousemotion함수로 사용자 입력을 처리하고, 제어점을 이동시키는 이벤트 함수입니다.

하지만 프로그램을 실행시켰을 때, 배지어 곡선이 예상했던 결과와 다르게 단조 증가, 단조 감소 영역 구분이 잘 되지않았고, 그에 따라 마우스 매핑과 곡선의 매핑된 부분이 다른 오류가 발생했습니다. 원인을 찾는데에 추가 시간이 필요할 듯 해보입니다.

openGL화면

텍스트, 스크린샷, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명  
코드  
  
#include "gl/glut.h"

#include <cstdio>

#include <cmath>

#include <vector>

struct Vec2 {

float *x*, *y*;

};

class BezierCurve2D {

public:

BezierCurve2D(const std::vector<Vec2>& *controlPoints*) : *controlPoints*(*controlPoints*) {}

Vec2 Evaluate(float *t*) {

float *oneMinusT* = 1.0f - *t*;

float *t2* = *t* \* *t*;

float *oneMinusT2* = *oneMinusT* \* *oneMinusT*;

Vec2 *point*;

*point*.*x* = *oneMinusT2* \* *oneMinusT* \* *controlPoints*[0].*x* +

3 \* *oneMinusT2* \* *t* \* *controlPoints*[1].*x* +

3 \* *oneMinusT* \* *t2* \* *controlPoints*[2].*x* +

*t2* \* *t* \* *controlPoints*[3].*x*;

*point*.*y* = *oneMinusT2* \* *oneMinusT* \* *controlPoints*[0].*y* +

3 \* *oneMinusT2* \* *t* \* *controlPoints*[1].*y* +

3 \* *oneMinusT* \* *t2* \* *controlPoints*[2].*y* +

*t2* \* *t* \* *controlPoints*[3].*y*;

return *point*;

}

std::vector<Vec2> GetControlPoints() const {

return *controlPoints*;

}

private:

std::vector<Vec2> *controlPoints*;

};

Vec2 *mousePt* = { 1, 0 };

float *rotateAngle* = 0.0;

std::vector<Vec2> *ctrlPts*;

void display() {

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-5.0, 5.0, -5.0, 5.0);

*// Draw Bezier Curves and AABBs*

for (int *i* = 0; *i* < *ctrlPts*.size() - 3; *i* += 3) {

BezierCurve2D *curve*({ *ctrlPts*[*i*], *ctrlPts*[*i* + 1], *ctrlPts*[*i* + 2], *ctrlPts*[*i* + 3] });

glColor3f(1, 0, 0);

glBegin(GL\_LINE\_STRIP);

for (int *j* = 0; *j* <= 100; *j*++) {

auto *pt* = *curve*.Evaluate(*j* / 100.0);

glVertex2f(*pt*.*x*, *pt*.*y*);

}

glEnd();

auto *controlPoints* = *curve*.GetControlPoints();

*// Calculate AABB*

float *minX* = *controlPoints*[0].*x*, *maxX* = *controlPoints*[0].*x*;

float *minY* = *controlPoints*[0].*y*, *maxY* = *controlPoints*[0].*y*;

for (const auto& *pt* : *controlPoints*) {

if (*pt*.*x* < *minX*) *minX* = *pt*.*x*;

if (*pt*.*x* > *maxX*) *maxX* = *pt*.*x*;

if (*pt*.*y* < *minY*) *minY* = *pt*.*y*;

if (*pt*.*y* > *maxY*) *maxY* = *pt*.*y*;

}

*// Draw AABB*

glColor3f(0, 0, 1);

glBegin(GL\_LINE\_LOOP);

glVertex2f(*minX*, *minY*);

glVertex2f(*maxX*, *minY*);

glVertex2f(*maxX*, *maxY*);

glVertex2f(*minX*, *maxY*);

glEnd();

}

glutSwapBuffers();

}

void keyboard(unsigned char *key*, int *x*, int *y*) {

switch (*key*) {

case 27: *// ESC*

exit(0);

break;

}

glutPostRedisplay();

}

void mouseCoordinateTranslate(int *winX*, int *winY*) {

*mousePt*.*x* = *winX* / 50.0 - 5;

*mousePt*.*y* = (*winY* / 50.0 - 5) \* (-1.0);

}

void changeCtrlPts() {

for (auto& *pt* : *ctrlPts*) {

if (sqrt(pow(*pt*.*x* - *mousePt*.*x*, 2) + pow(*pt*.*y* - *mousePt*.*y*, 2)) < 0.5) {

*pt* = *mousePt*;

break;

}

}

}

void mouse(int *button*, int *state*, int *x*, int *y*) {

if (*state* == GLUT\_DOWN) {

mouseCoordinateTranslate(*x*, *y*);

changeCtrlPts();

}

glutPostRedisplay();

}

void mouseMotion(int *x*, int *y*) {

mouseCoordinateTranslate(*x*, *y*);

changeCtrlPts();

glutPostRedisplay();

}

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

glutInit(&*argc*, *argv*);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(1480, 100);

*ctrlPts*.push\_back({ -2, -1 });

*ctrlPts*.push\_back({ 0, 2 });

*ctrlPts*.push\_back({ 2, -1 });

*ctrlPts*.push\_back({ -1, 1 });

*ctrlPts*.push\_back({ 2, 2 });

*ctrlPts*.push\_back({ -2, 2 });

glutCreateWindow("OpenGL");

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

glutMouseFunc(mouse);

glutMotionFunc(mouseMotion);

glutMainLoop();

return 0;

}