CG Practice 5

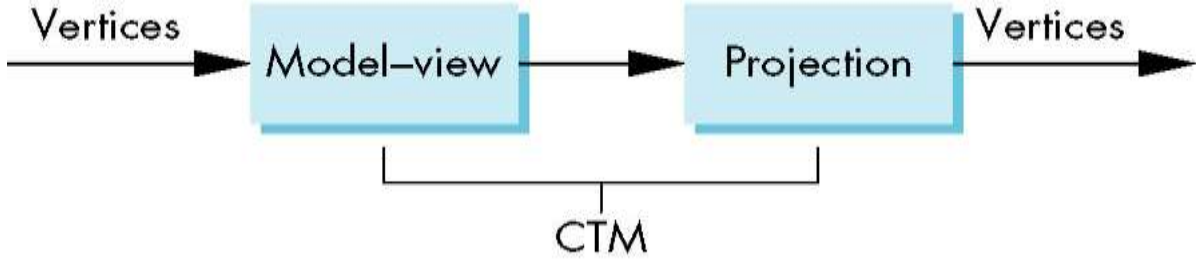
COLLEGE OF COMPUTING

HANYANG ERICA CAMPUS

Q YOUN HONG (홍규연)

Practice 4 Review

- Shader에서 viewing transformation 구현



CTM: Current Transform Matrix

- Camera transformation: Look-at() function
- Orthographic projection transformation: Ortho() function
- Perspective projection transformation: Frustum(), Perspective()

Objectives

- 2차원 곡선 Editor 만들기
  - 2차원 상의 3차 Hermite 곡선 그리기
  - Hermite 곡선 수정 기능 추가
  - Bezier 곡선 그리기

Program: Draw a 2D Hermite Spline Curve

(PROGRAM IN MAIN_HERMITE2D.CPP)

2D Hermite Curve Editor

- Idea: Draw an interactive Hermite Curve Editor in 2D
- (Cubic) Hermite Curve
  - A cubic curve satisfying the boundary constraints for positions p_0, p_1 and tangent vectors v_0, v_1
  - In terms of boundary constraints

$$f(t) = p_0(2t^3 - 3t^2 + 1) + p_1(-2t^3 + 3t^2) + v_0(t^3 - 2t^2 + t) + v_1(t^3 - t^2)$$

$$(0 \le t \le 1)$$

  - In terms of cubic polynomials

$$f(t) = (2p_0 - 2p_1 + v_0 + v_1)t^3 + (-3p_0 + 3p_1 - 2v_0 - v_1)t^2 + v_0 t + p_0$$

$$(0 \le t \le 1)$$

2D Hermite Curve Editor

- Make a data structure for 2D Hermite curve
  - Curve drawing – draw a curve with a set of line segments

```
struct HermiteCurve {
    vec2 BndPos[2];
    vec2 BndTan[2];

    vec2 points[NumCrvVertices];
    vec3 color;
    HermiteCurve(const vec2& p0, const vec2& p1, const vec2& v0, const vec2& v1)
    {
        BndPos[0] = p0;
        BndPos[1] = p1;
        BndTan[0] = v0;
        BndTan[1] = v1;
    }
…
```

2D Hermite Curve Editor

- Evaluation of a curve
  - Boundary constraints가 바뀔 때마다,

$$f(t) = p_0(2t^3 - 3t^2 + 1) + p_1(-2t^3 + 3t^2) + v_0(t^3 - 2t^2 + t) + v_1(t^3 - t^2)$$

다시 계산

⇒Observation: polynomial basis들은 t의 변화에 의해서만 변함

⇒미리 계산할 수 있음

2D Hermite Curve Editor

- Evaluation of a curve
  - Boundary constraints가 바뀔 때마다,

$$f(t) = p_0(2t^3 - 3t^2 + 1) + p_1(-2t^3 + 3t^2) + v_0(t^3 - 2t^2 + t) + v_1(t^3 - t^2)$$

다시 계산

⇒Observation: polynomial basis들은 t의 변화에 의해서만 변함

⇒미리 계산할 수 있음

```
GLfloat HermiteBasis[NumCrvVertices][4];
// precompute polynomial bases for Hermite spline
void precomputeHermiteBasis()
{
    GLfloat t, t2, t3;
    for (int i = 0; i < NumCrvVertices; i++) {
        t = i * 1.0 / (GLfloat)(NumCrvVertices - 1.0);
        t2 = t * t;
        t3 = t2 * t;
        HermiteBasis[i][0] = 2 * t3 - 3 * t2 + 1;
        HermiteBasis[i][1] = -2 * t3 + 3 * t2;
        HermiteBasis[i][2] = t3 - 2 * t2 + t;
        HermiteBasis[i][3] = t3 - t2;
    }
}
```

2D Hermite Curve Editor

- Evaluation of a curve
  - Boundary constraints가 바뀔 때마다,

$$f(t) = p_0(2t^3 - 3t^2 + 1) + p_1(-2t^3 + 3t^2) + v_0(t^3 - 2t^2 + t) + v_1(t^3 - t^2)$$

다시 계산

⇒Observation: polynomial basis들은 t의 변화에 의해서만 변함

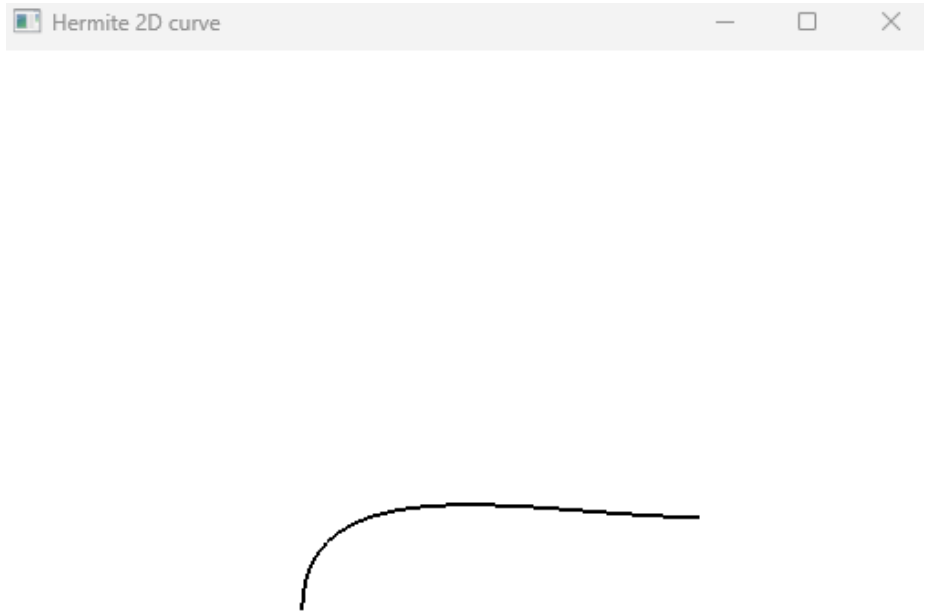
⇒미리 계산할 수 있음

⇒Curve의 evaluation에 필요한 연산 수가 줄어듬

```
void evalulate() {
       for (int i = 0; i < NumCrvVertices; i++) {
           for (int j = 0; j < 2; j++)
               points[i][j] = HermiteBasis[i][0] * BndPos[0][j] + HermiteBasis[i][1]
* BndPos[1][j]
               + HermiteBasis[i][2] * BndTan[0][j] + HermiteBasis[i][3] *
BndTan[1][j];
       }
   }
```

2D Hermite Curve Editor

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING



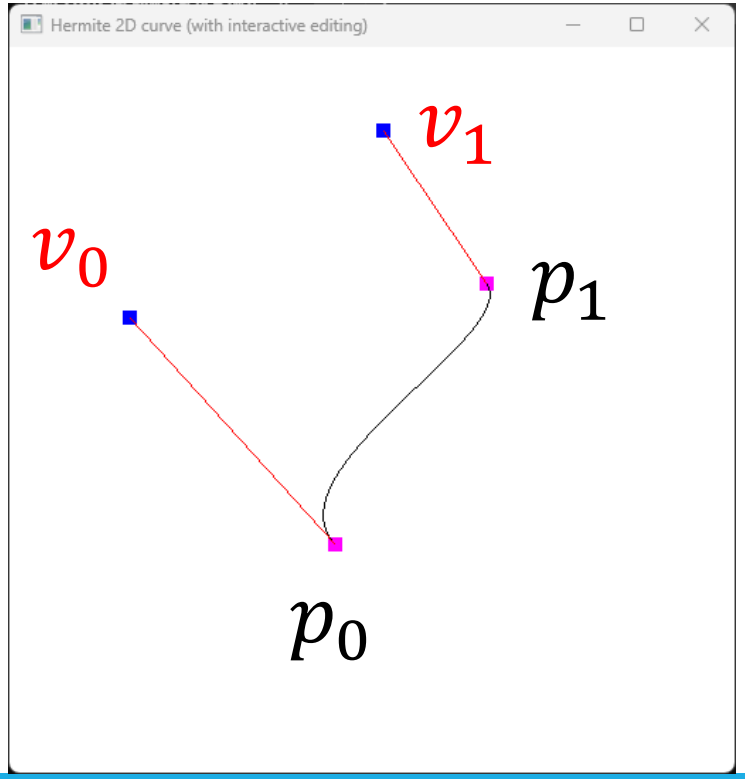
Hermite Curve 실습

- 추가 기능

① Boundary condition (양 끝점 p_0, p_1 + 두 점에서의 tangent vector v_0, v_1 를 화면에 같이 표시)

② 곡선 수정
- p_0, p_1을 끌어 당기면 곡선의 양 끝점 변환
- v_0, v_1를 끌어 당기면 곡선의 양 기울기가 변환

⇒ 결과적으로 곡선의 형태가 바뀜

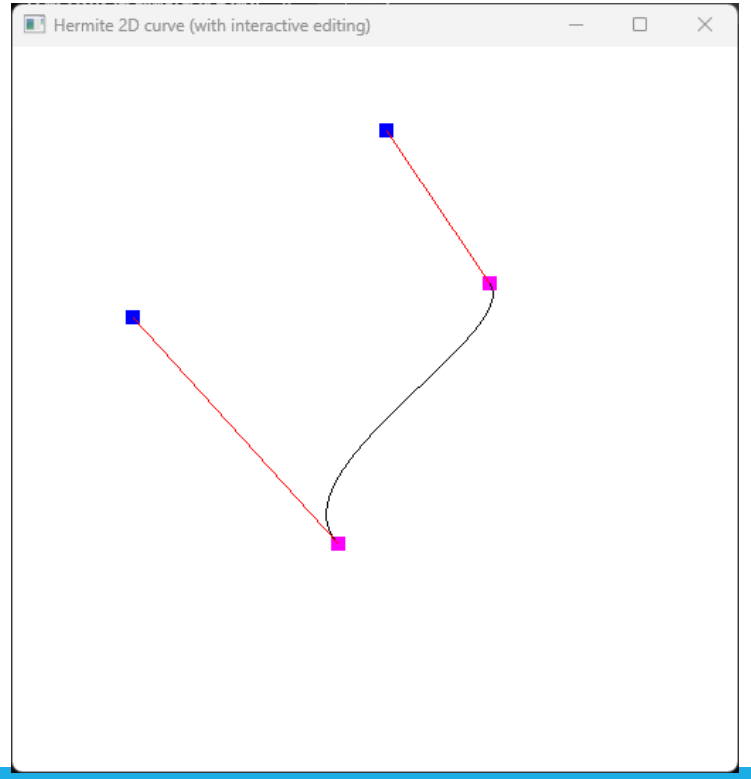


Hermite Curve 실습

① Boundary condition (양 끝점 p_0, p_1 + 두 점에서의 tangent vector v_0, v_1 를 화면에 같이 표시)

- Hint: 선을 먼저 그리고 점을 그 위에 그릴 수 있음
- Hint: 선과 점을 다른 배열에 저장할 수 있음

(각각 VAO, VBO필요하고 쓸 때마다 Binding을 다시 해야함)



② main_2dhermite2.cpp: 곡선 수정하기

A. mouse event, mouse move event 추가

```
int main(int argc, char **argv)
{
    Width = 512;
    Height = 512;
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(Width, Height);
    glutCreateWindow("Hermite 2D curve (with interactive editing) ");

    glewInit();
    init();


    glutDisplayFunc(display);
    glutKeyboardFunc( keyboard );
    glutMouseFunc(mouse);
    glutMotionFunc(mouseMove);

    glutMainLoop();
    return 0;
}
```

- glutMouseFunc: 마우스의 action이 일어나면 호출 (mouse 누르거나 뗄 때)
- glutMotionFunc: 화면 안에서 마우스를 움직일 때 호출

main_2dhermite2.cpp

B. mouse event

```
void mouse(GLint button, GLint action, GLint x, GLint y)
{
    if (GLUT_LEFT_BUTTON == button)
    {
        if (GLUT_LEFT_BUTTON == button)
        {
            switch (action)
            {
            case GLUT_DOWN:
                crv_edit_handle = HitIndex(&curve, x, Height - y);
                break;
            case GLUT_UP:
                crv_edit_handle = -1;
                break;
            default: break;
            }
        }
        glutPostRedisplay();
    }
}
```

마우스의 액션이 생길 때 호출

- button: 어떤 마우스 버튼이 눌렸는가
- action: 마우스를 누름/뗌?
- x,y: 마우스의 액션이 일어난 곳의 screen coordinate

→ 좌측 상단이 (0,0)

- HitIndex: 마우스의 액션이 곡선과 관련 있는지 판단

main_2dhermite2.cpp

C. HitIndex: 현재 마우스의 액션이 곡선의 handle(p_0, p_1, v_0, v_1)과 관련있는지 판단

In HitIndex(),

- Pixel 한칸의 가로, 세로 길이를 곡선이 정의된 좌표계에서 계산

```
vec2 pixelLen((right - left) / (GLfloat)(Width),
              (top - bottom) /  (GLfloat)Height);
```

- 화면 좌표로 표현된 마우스의 좌표를 곡선 좌표계에서 계산

```
mousePt.x = left + pixelLen[0] * (GLfloat)x;
mousePt.y = bottom + pixelLen[1] * (GLfloat)y;
```

→ mousePt는 곡선 좌표계 [left, right]x[bottom, top]에서의 마우스의 2차원 좌표임

main_2dhermite2.cpp

C. HitIndex: 현재 마우스의 액션이 곡선의 handle(p_0, p_1, v_0, v_1)과 관련있는지 판단

- mousePt가 p_0, p_1이나 $p_0 + v_0, p_1 + v_1$에 충분히 가까운지 판단

```
tmpVec = curve->BndPos[0] - mousePt;
dist[0] = dot(tmpVec, tmpVec);
tmpVec = curve->BndPos[1] - mousePt;
dist[1] = dot(tmpVec, tmpVec);

tmpVec = (curve->BndPos[0] + curve->BndTan[0]) - mousePt;
dist[2] = dot(tmpVec, tmpVec);
tmpVec = (curve->BndPos[1] + curve->BndTan[1]) - mousePt;
dist[3] = dot(tmpVec, tmpVec);

for (int i = 0; i < 4; i++) {
  if (mindist > dist[i]) {
     ret = i;
     mindist = dist[i];
  }
}
// if clicked within 10-pixel radius of one of the handles then return
if (mindist < 100.0 * dot(pixelLen, pixelLen)) return ret;
else return -1;
```

main_2dhermite2.cpp

D. mouse move event

- 이미 선택된 handle이 있고, 마우스를 누른 채 이동하면, 해당 handle을 업데이트

```
void mouseMove(GLint x, GLint y)
{
  if (crv_edit_handle != -1) {
    vec2 pixelLen((right - left) / (GLfloat)(Width), (top - bottom) / (GLfloat)Height);
    vec2 mousePt(left + (GLfloat)x * pixelLen[0],
                 bottom + (GLfloat)(Height - y) * pixelLen[1]);

    if (crv_edit_handle < 2)
      curve.BndPos[crv_edit_handle] = mousePt;
    else
      curve.BndTan[crv_edit_handle - 2] = mousePt - curve.BndPos[crv_edit_handle - 2];

    curve.updateForRendering(); Curve의 각 점의 위치가 update됨
    glutPostRedisplay();
}
```

main_2dhermite2.cpp

E. HermiteCurve structure revisited

- Curve와 시작/끝 vector를 line_strip으로 그리기:

points[N+4], colors[N+4]

- updateForRendering():

Curve 위의 점들 N개 뿐만 아니라, points[0], points[1], points[N+2], points[N+3]도 같이 계산

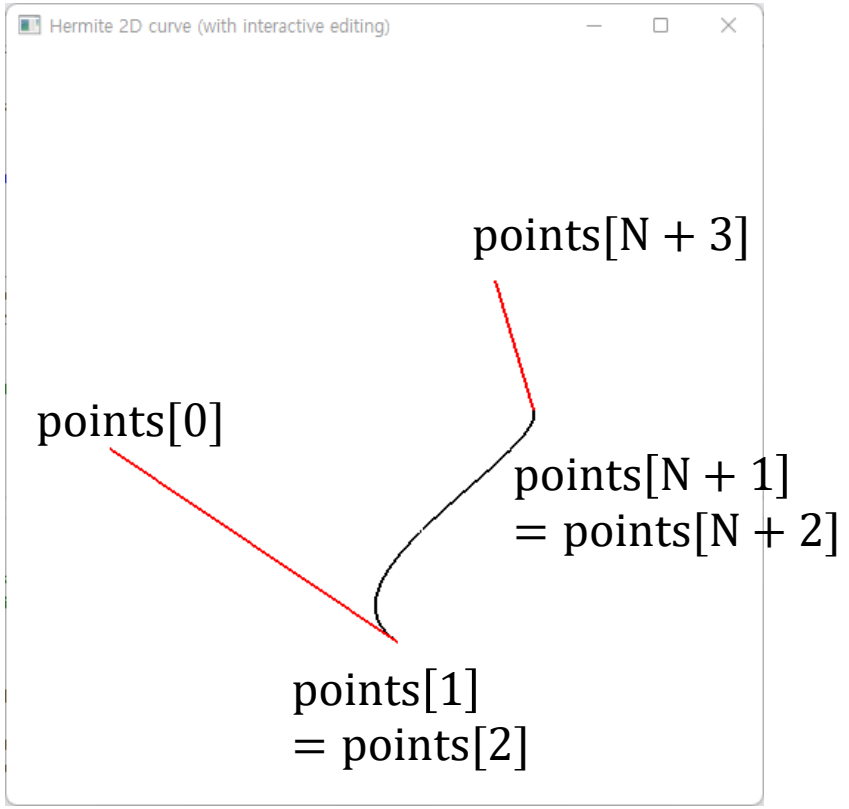
```
void updateForRendering() {
  evalulate();

  points[0] = BndPos[0] + BndTan[0];
  points[1] = BndPos[0];
  points[NumCrvVertices + 2] = BndPos[1];
  points[NumCrvVertices + 3] = BndPos[1] + BndTan[1];
}
```

- VAO,VBO를 멤버 변수로 포함:

buffer object안에 있는 points의 값이 계속 변하므로 display()에서

```
glBindVertexArray(curve.vao);
glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(curve.points), curve.points);
```

를 계속 호출함



main_2dhermite2.cpp

- shader의 in 변수들에 vertex array로 넘겨줄 때, c 프로그램에서 shader 변수의 주소를 찾는 대신, Vertex shader에서 직접적으로 변수의 주소를 설정해줄 수 있음

```
GLuint vPos = glGetAttribLocation(program, "vPosition");
glEnableVertexAttribArray(vPos);
glVertexAttribPointer(vPos, 2, GL_FLOAT, GL_FALSE, 0, BUFFER_OFFSET(0));
GLuint vCol = glGetAttribLocation(program, "vColor");
glEnableVertexAttribArray(vCol);
glVertexAttribPointer(vCol, 3, GL_FLOAT, GL_FALSE, 0, BUFFER_OFFSET(sizeof(curve.points));
```

```
#version 330

layout (location = 0) in vec2 vPosition;
layout (location = 1) in vec3 vColor;
out vec4 color;

uniform mat4 model_view;
uniform mat4 projection;

void main()
{
  gl_Position = projection * vec4(vPosition, 0.0, 1.0);
  color = vec4(vColor, 1.0);
}
```

```
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 2, GL_FLOAT,
            GL_FALSE, 0, BUFFER_OFFSET(0));

glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 3, GL_FLOAT,
      GL_FALSE, 0,
      BUFFER_OFFSET(sizeof(curve.points)));
```

3차 Bezier 곡선

- Bezier 곡선: n 개의 제어점(control point)들이 주어졌을 때, Bezier 곡선은 다음과 같이 정의된다.

$$f(t) = \sum_{i=0}^{n} P_i B_i^n(t) = \sum_{i=0}^{n} P_i \frac{n!}{i!\,(n-i)!}(1-t)^{n-i}t^i\,, (0 \le t \le 1)$$

  - 3차 Bezier 곡선의 경우

$$f(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3\ (0 \le t \le 1)$$

실습) 3차 Bezier 곡선 에디터

- 입력으로 점 4개 P_0, P_1, P_2, P_3가 주어질 때 3차 Bezier 곡선을 그리기
  - 제어점 P_0, P_1, P_2, P_3를 선분으로 이어 control polygon를 같이 그려보기
  - 제어점의 위치가 변할 때 곡선의 모양은 어떻게 변하는가?