Seoul National University
College of Engineering
Department of Naval Architecture and Ocean Engineering
1, Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea

Spring 2023

컴퓨터 이용 선박 설계

(Computer-Aided Ship Design)

PA #3

Instructor name	김 태 완
Student name	장 민 준
Department	조선해양공학과
Student ID	2019-10738
Submission date	23/05/09
Grade	

1. 구현 과정

A. PA2에서 구현한 메서드의 연장

2. 결론

3. Reference

A. PA2에서 구현한 메서드의 연장

기존에 제출했던 Programming Assignment #2에서 Bezier Curve 객체를 parameter로 받아 새로운 Bezier Curve를 만드는 메서드를 구현하는 것을 시도해보았다. 이번 보고서에는 그에 대해 상세히 설명할 것이다.

이번 PA에서 이용한 BezierCurve Class의 선언은 헤더 파일에 담아두었다.

"BezierCurve.hpp"

```
#ifndef BEZIERCURVE_H
#define BEZIERCURVE_H

#include <stdexcept>
#include <Eigen/Dense>

using namespace Eigen;

class BezierCurve {
public:
    BezierCurve();
    BezierCurve(const Vector2d& C0, const Vector2d& P1, const Vector2d&
P2, const Vector2d& C3);
    BezierCurve(BezierCurve& frontBezier, const Vector2d& final_point);
    BezierCurve(BezierCurve& frontBezier, const Vector2d& third_point,
const Vector2d& final_point);
    BezierCurve(BezierCurve& frontBezier, const Vector2d& P1, const
Vector2d& P2, const Vector2d& C3);
    BezierCurve(const Matrix<double, 4,2>& point);

    Vector2d getControlPoint(int index) const;
    Vector2d getBezierPoint(double t);
    void draw_Bezier(float z);
    void draw_PoC_triangle_case1(BezierCurve& BezierCurve, float z);
    void draw_PoC_triangle_case2(BezierCurve& BezierCurve, float z);
    void draw_PoC_triangle_case3(BezierCurve& BezierCurve, float z);
    void draw_Bezier_side(float z);
```

```
private:
    Vector2d C0_;
    Vector2d C1_;
    Vector2d C2_;
    Vector2d C3_;
    float ratio = 1.0/20.0; // 축척
    double origin_x = -50.0; // 원점 설정
    double origin_y = 50.0; //원점 설정
    double chord_length;
    Vector2d Bezier(double t, const Vector2d& C0, const Vector2d& C1,
const Vector2d& C2, const Vector2d& C3) ;
    void findControlPoints(const Vector2d& P1, const Vector2d& P2);
    void Continuity2_BezierCurve(BezierCurve& frontBezier, const
Vector2d& final_point);
    void Continuity1_BezierCurve(BezierCurve& frontBezier, const
Vector2d& third_point, const Vector2d& final_point);

};


#endif // BEZIERCURVE_H
```

각 구성요소에 대해서 설명하겠다.

```
BezierCurve();
BezierCurve(const Vector2d& C0, const Vector2d& P1, const Vector2d&
P2, const Vector2d& C3);
BezierCurve(BezierCurve& frontBezier, const Vector2d& final_point);
BezierCurve(BezierCurve& frontBezier, const Vector2d& third_point,
const Vector2d& final_point);
BezierCurve(BezierCurve& frontBezier, const Vector2d& P1, const
Vector2d& P2, const Vector2d& C3);
BezierCurve(const Matrix<double, 4,2>& point);
```

BezierCurve의 생성자들이다.

첫 번째는 4개의 POC를 통해 BezierCurve를 생성, 마지막 생성자는 Matrix를 통해 생성이며 기능은 첫 번째와 똑같다.

위 Constructor들 중 중요한 것은 BezierCurve 객체를 받아서 새 BezierCurve를 생성하는 Constructor들인데, 이는 각각 C2 continuity, C1 continuity, C0 continuity를 통해 생성하는 Constructor이다.

C2 continuity를 이용하는 constructor부터 살펴보도록 하자

```
BezierCurve::BezierCurve(BezierCurve& frontBezier, const Vector2d&
final_point){ // C2 continuity
    Continuity2_BezierCurve(frontBezier, final_point);
}


void BezierCurve::Continuity2_BezierCurve(BezierCurve& frontBezier,
const Vector2d& final_point){// C2 continuity로 BezierCurve 연결
    C0_ = frontBezier.C3_; // C0 condition
    C3_ = final_point; // 끝 점 설정
    chord_length = (final_point-C0_).norm(); // l2 설정
    double scale_factor = chord_length / frontBezier.chord_length; //
chord_length 이용
    C1_ = (scale_factor+1)*frontBezier.C3_ -
scale_factor*frontBezier.C2_; // C1 condition,(C3-C2)/l1 =(C5-C4)/l2
    C2_ = 2*C1_-C0_+scale_factor*scale_factor*(frontBezier.C3_-
2*frontBezier.C2_+frontBezier.C1_);
}
```

Continuity2_BezierCurve을 이용하여 새 객체의 C0_, C1_, C2_, C3_을 할당한다.

C2 continuity에 의해 C0_, C1_, C2_가 정해지기 때문에, final point를 받아 C3_를 설정해준다. 이를 통해 온전한 BezierCurve가 생성이 된다.

두 번째로는 C1 continuity를 이용하는 constructor이다.

```
BezierCurve::BezierCurve(BezierCurve& frontBezier, const Vector2d&
third_point, const Vector2d& final_point){ // C1 continuity
    Continuity1_BezierCurve(frontBezier, third_point, final_point);
}


void BezierCurve::Continuity1_BezierCurve(BezierCurve& frontBezier,
const Vector2d& third_point, const Vector2d& final_point){
    C0_ = frontBezier.C3_; // C0 condition
    C3_ = final_point; // 끝 점 설정
    chord_length = (final_point - C0_).norm();
    double scale_factor = chord_length / frontBezier.chord_length; //
chord_length 이용
    C1_ = (scale_factor+1)*frontBezier.C3_ -
scale_factor*frontBezier.C2_; // C1 condition,(C3-C2)/l1 =(C5-C4)/l2
```

```
    C2_ = (27*third_point - C0_ - 6*C1_-8*C3_)/12; // third point
t=2/3일때 가정 삽입
}
```

C1 continuity를 통해 C0_, C1_까지는 결정이 되지만, BezeirCurve를 완성하려면 DOF에 따라 점 두개가 필요하며, 따라서 위와 같이 두 점을 받아 BezierCurve를 완성을 한다.

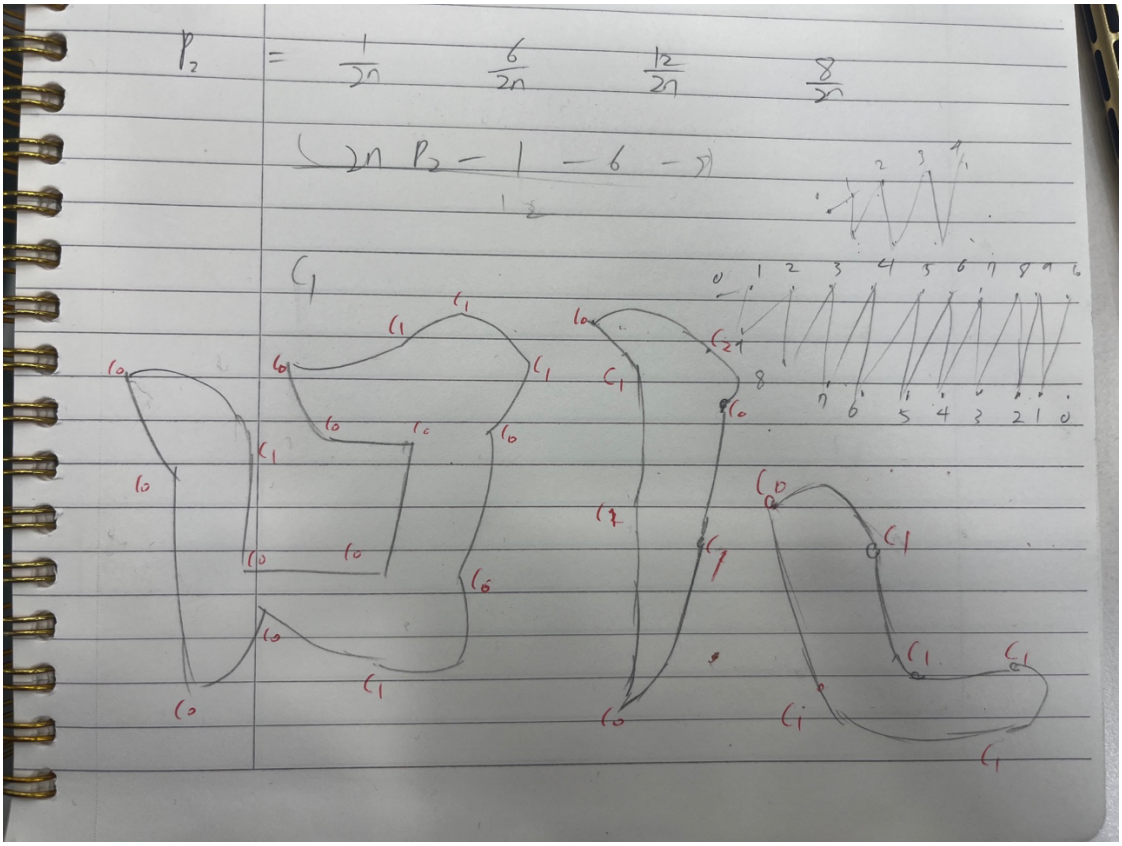
C0 continuity는 기존 생성자와 동일하며, frontBezier로부터 C0_을 받는 것만 차이가 있다.

```
BezierCurve::BezierCurve(BezierCurve& frontBezier, const Vector2d& P1,
const Vector2d& P2, const Vector2d& C3){ // C0 continuity
    C0_ = frontBezier.C3_;
    C3_ = C3;
    findControlPoints(P1, P2);
}
```

위 생성자를 바탕으로, POC를 받아 글자를 구현하도록 하자

PA2에서는 각 커브마다 POC 4개가 필요해, 초성 커브 16개, 중성 커브 7개, 종성 커브 6개 총 (16+7+6)*4 =. 116개의 POC가 필요했다.

PA3에서는, 글자에 따라 다음과 같이 continuity를 설정하여, 필요한 POC의 개수를 축소하였다.



이로 인해 총 필요한 POC 개수는 75개로 줄어든다.

텍스트 파일에 POC에 대한 정보를 담고, 점들을 받아 BezierCurve로 변환을 할 수 있다.

Visualization 내부에 다음과 같이 정보를 받아서 변환을 하고

```
ifstream poc("/Users/minjun/Downloads/PA#3 2/PoC.txt");

    // 파일 열기 실패 시 예외 처리
    if (!poc.is_open()) {
        cerr << "Error: Unable to open PoC.txt file." << endl;
        return;
    }

    Matrix<double, 44, 2> first;
    Matrix<double, 17, 2> middle;
    Matrix<double, 14, 2> last;

    for (int i = 0; i < 44; i++){
        for(int j = 0; j < 2; j++){
            poc >> first(i, j);
        }
    }
    for (int i = 0; i < 17; i++){
        for(int j = 0; j < 2; j++){
            poc >> middle(i, j);
        }
    }
    for (int i = 0; i < 14; i++){
        for(int j = 0; j < 2; j++){
            poc >> last(i, j);
        }
    }
    // 파일 닫기
    poc.close();

    std::vector<BezierCurve> first_hanguel(16);
    std::vector<BezierCurve> middle_hanguel(7);
    std::vector<BezierCurve> last_hanguel(6);

    first_hanguel[0] = BezierCurve(first.row(0), first.row(1), first.row(2), first.row(3)); //최초
    first_hanguel[1] = BezierCurve(first_hanguel[0], first.row(4), first.row(5)); //C1
    first_hanguel[2] = BezierCurve(first_hanguel[1], first.row(6), first.row(7)); //C1
    first_hanguel[3] = BezierCurve(first_hanguel[2], first.row(8), first.row(9)); //C1
    first_hanguel[4] = BezierCurve(first_hanguel[3], first.row(10), first.row(11), first.row(12)); //C0
    first_hanguel[5] = BezierCurve(first_hanguel[4], first.row(13), first.row(14), first.row(15)); //C0
    first_hanguel[6] = BezierCurve(first_hanguel[5], first.row(16), first.row(17)); //C1
    first_hanguel[7] = BezierCurve(first_hanguel[6], first.row(18), first.row(19), first.row(20)); //C0
    first_hanguel[8] = BezierCurve(first_hanguel[7], first.row(21), first.row(22), first.row(23)); //C0
    first_hanguel[9] = BezierCurve(first_hanguel[8], first.row(24), first.row(25), first.row(26)); //C0
    first_hanguel[10] = BezierCurve(first_hanguel[9], first.row(27), first.row(28), first.row(29)); //C0
    first_hanguel[11] = BezierCurve(first_hanguel[10], first.row(30), first.row(31)); //C1
    first_hanguel[12] = BezierCurve(first_hanguel[11], first.row(32), first.row(33), first.row(34)); //C0
    first_hanguel[13] = BezierCurve(first_hanguel[12], first.row(35), first.row(36), first.row(37)); //C0
    first_hanguel[14] = BezierCurve(first_hanguel[13], first.row(38), first.row(39), first.row(40)); //C0
    first_hanguel[15] = BezierCurve(first_hanguel[14], first.row(41), first.row(42), first.row(43)); //C0

    middle_hanguel[0] = BezierCurve(middle.row(0), middle.row(1), middle.row(2), middle.row(3)); //  최초
    middle_hanguel[1] = BezierCurve(middle_hanguel[0], middle.row(4)); // C2
    middle_hanguel[2] = BezierCurve(middle_hanguel[1], middle.row(5), middle.row(6), middle.row(7)); // C0
    middle_hanguel[3] = BezierCurve(middle_hanguel[2], middle.row(8), middle.row(9)); // C1
    middle_hanguel[4] = BezierCurve(middle_hanguel[3], middle.row(10), middle.row(11), middle.row(12)); // C0
    middle_hanguel[5] = BezierCurve(middle_hanguel[4], middle.row(13), middle.row(14)); // C1
    middle_hanguel[6] = BezierCurve(middle_hanguel[5], middle.row(15), middle.row(16)); // C1

    last_hanguel[0] = BezierCurve(last.row(0), last.row(1), last.row(2), last.row(3)); //  최초
    last_hanguel[1] = BezierCurve(last_hanguel[0], last.row(4), last.row(5)); // C1
    last_hanguel[2] = BezierCurve(last_hanguel[1], last.row(6), last.row(7)); // C1
    last_hanguel[3] = BezierCurve(last_hanguel[2], last.row(8), last.row(9)); // C1
    last_hanguel[4] = BezierCurve(last_hanguel[3], last.row(10), last.row(11)); // C1
    last_hanguel[5] = BezierCurve(last_hanguel[4], last.row(12), last.row(13)); // C1
```

Bezier curve를 openGL을 통해 구현하는 메서드를 만든다.

```
void draw_Bezier(float z);
void draw_PoC_triangle_case1(BezierCurve& BezierCurve, float z);
void draw_PoC_triangle_case2(BezierCurve& BezierCurve, float z);
void draw_PoC_triangle_case3(BezierCurve& BezierCurve, float z);
void draw_Bezier_side(float z);
```

draw_Bezier는 BezierCurve를 그리는 메서드로, 구현은 다음과 같다.

```
void BezierCurve::draw_Bezier(float z){
    glPointSize(5.0f); // 정점 크기 설정
    glBegin(GL_LINE_STRIP); // 직선 그리기
    for(int i = 0; i < 11; i++){
        glVertex3f(origin_x+ratio*this->getBezierPoint(i/10.0).x(),
origin_y-ratio*this->getBezierPoint(i/10.0).y(), z);
    }
    glEnd();
}
```

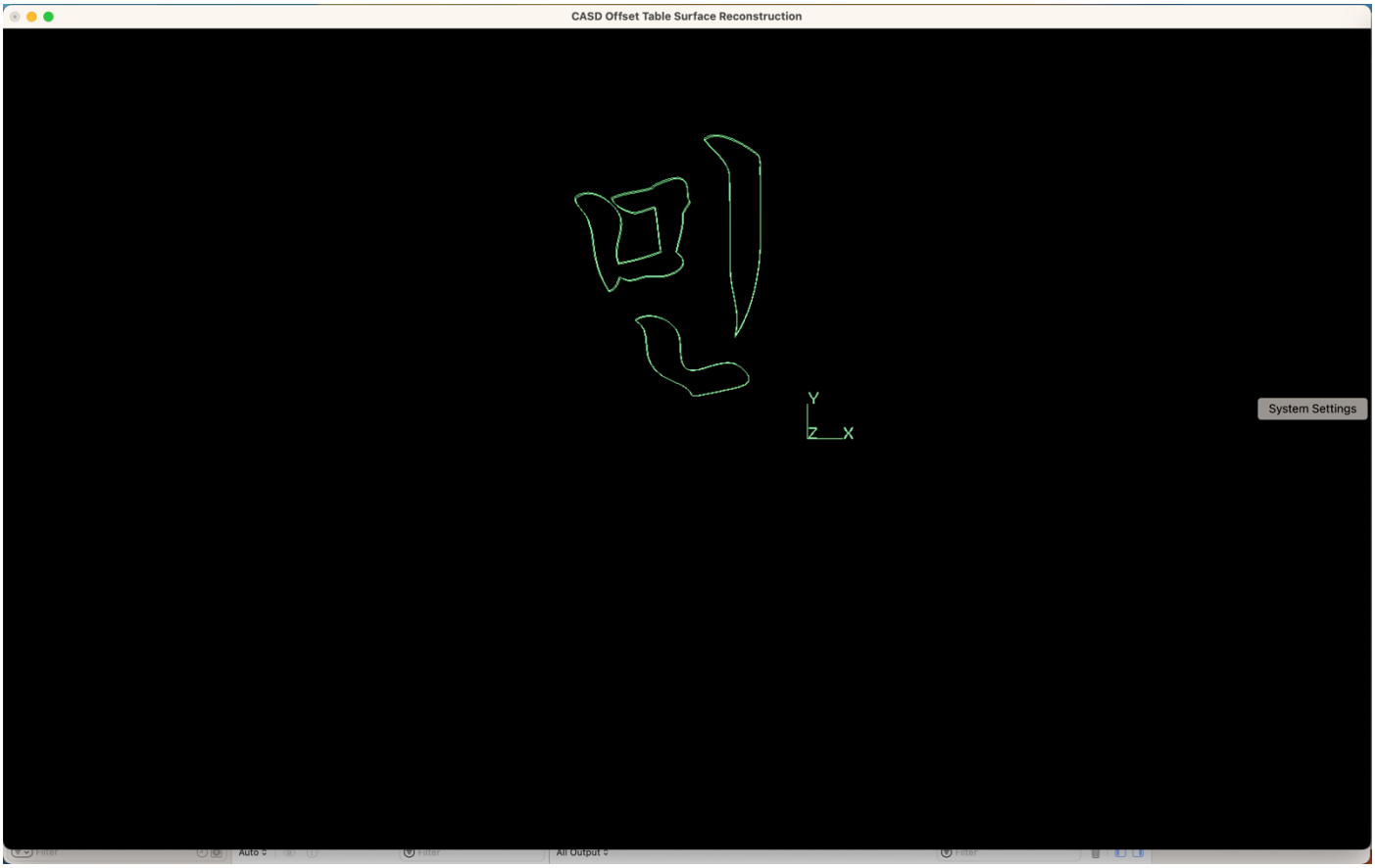
PA2의 구현 과정과 매우 유사하다.

이 메서드를 이용하여, drawName 안에 다음과 같이 작성할 수 있다.

```
for(int i = 0; i < 16; i++){
        first_hanguel[i].draw_Bezier(0);
    }
    for(int i = 0; i < 7; i++){
        middle_hanguel[i].draw_Bezier(0);
    }
    for(int i = 0; i < 6; i++){
        last_hanguel[i].draw_Bezier(0);
    }
    for(int i = 0; i < 16; i++){
        first_hanguel[i].draw_Bezier(1);
    }
    for(int i = 0; i < 7; i++){
        middle_hanguel[i].draw_Bezier(1);
    }
    for(int i = 0; i < 6; i++){
        last_hanguel[i].draw_Bezier(1);
    }
```

간단한 반복문을 통해, 각 커브들을 그려나갈 수 있다

이를 visualization한 결과는 다음 그림과 같다.



색칠하기 위해 polygon을 구성하여 색을 칠해야 하는데, 사용되는 메서드는 draw_PoC_triangle이다.

```
void BezierCurve::draw_PoC_triangle_case1(BezierCurve& BezierCurve,
float z){
    glPointSize(5.0f); // 정점 크기 설정
    for(int i = 0; i < 10; i++){
        glBegin(GL_TRIANGLES);
        glVertex3f(origin_x+ratio*this->getBezierPoint(i/10.0).x(),
origin_y-ratio*this->getBezierPoint(i/10.0).y(), z);
        glVertex3f(origin_x+ratio*BezierCurve.getBezierPoint(1-
i/10.0).x(), origin_y-ratio*BezierCurve.getBezierPoint(1-i/10.0).y(),
z);
        glVertex3f(origin_x+ratio*this->getBezierPoint((i+1)/10.0).x(),
origin_y-ratio*this->getBezierPoint((i+1)/10.0).y(), z);
        glEnd();
    }
    for(int i = 0; i< 10; i++){
        glBegin(GL_TRIANGLES);
```

```
glVertex3f(origin_x+ratio*BezierCurve.getBezierPoint(i/10.0).x(),
origin_y-ratio*BezierCurve.getBezierPoint(i/10.0).y(), z);
        glVertex3f(origin_x+ratio*this->getBezierPoint(1-i/10.0).x(),
origin_y-ratio*this->getBezierPoint(1-i/10.0).y(), z);

glVertex3f(origin_x+ratio*BezierCurve.getBezierPoint((i+1)/10.0).x(),
origin_y-ratio*BezierCurve.getBezierPoint((i+1)/10.0).y(), z);
        glEnd();
    }
}
void BezierCurve::draw_PoC_triangle_case2(BezierCurve& BezierCurve,
float z){
    glPointSize(5.0f); // 정점 크기 설정
    for(int i = 0; i < 10; i++){
        glBegin(GL_TRIANGLES);
        glVertex3f(origin_x+ratio*this->getBezierPoint(i/10.0).x(),
origin_y-ratio*this->getBezierPoint(i/10.0).y(), z);
        glVertex3f(origin_x+ratio*BezierCurve.getBezierPoint(0).x(),
origin_y-ratio*BezierCurve.getBezierPoint(0).y(), z);
        glVertex3f(origin_x+ratio*this->getBezierPoint((i+1)/10.0).x(),
origin_y-ratio*this->getBezierPoint((i+1)/10.0).y(), z);
        glEnd();
    }
}
void BezierCurve::draw_PoC_triangle_case3(BezierCurve& BezierCurve,
float z){
    glPointSize(5.0f); // 정점 크기 설정
    for(int i = 0; i < 10; i++){
        glBegin(GL_TRIANGLES);
        glVertex3f(origin_x+ratio*this->getBezierPoint(i/10.0).x(),
origin_y-ratio*this->getBezierPoint(i/10.0).y(), z);
        glVertex3f(origin_x+ratio*this->getBezierPoint((i+1)/10.0).x(),
origin_y-ratio*this->getBezierPoint((i+1)/10.0).y(), z);
        glVertex3f(origin_x+ratio*BezierCurve.getBezierPoint(1-
i/10.0).x(), origin_y-ratio*BezierCurve.getBezierPoint(1-i/10.0).y(),
z);
        glEnd();
    }
    for(int i = 0; i< 10; i++){
        glBegin(GL_TRIANGLES);
```

```
glVertex3f(origin_x+ratio*BezierCurve.getBezierPoint(i/10.0).x(),
origin_y-ratio*BezierCurve.getBezierPoint(i/10.0).y(), z);

glVertex3f(origin_x+ratio*BezierCurve.getBezierPoint((i+1)/10.0).x(),
origin_y-ratio*BezierCurve.getBezierPoint((i+1)/10.0).y(), z);
        glVertex3f(origin_x+ratio*this->getBezierPoint(1-i/10.0).x(),
origin_y-ratio*this->getBezierPoint(1-i/10.0).y(), z);
        glEnd();
    }
}
void BezierCurve::draw_PoC_triangle_case4(BezierCurve& BezierCurve,
float z){
    glPointSize(5.0f); // 정점 크기 설정
    for(int i = 0; i < 10; i++){
        glBegin(GL_TRIANGLES);
        glVertex3f(origin_x+ratio*this->getBezierPoint(i/10.0).x(),
origin_y-ratio*this->getBezierPoint(i/10.0).y(), z);
        glVertex3f(origin_x+ratio*this->getBezierPoint((i+1)/10.0).x(),
origin_y-ratio*this->getBezierPoint((i+1)/10.0).y(), z);
        glVertex3f(origin_x+ratio*BezierCurve.getBezierPoint(0).x(),
origin_y-ratio*BezierCurve.getBezierPoint(0).y(), z);
        glEnd();
    }
}
```

origin_X, origin_y를 통해 그려나갈 포인트들의 원점을 조정하고, ratio로 축척을 조절한다. 이들은 헤더 파일에 선언 및 초기화되어있기 때문에 크기는 이들을 조정함으로서 이뤄질 수 있다.

여기서, openGL의 glBegin(GL_TRIANGLES)은 점들의 반시계방향으로 평면을 형성하기 때문에, 위와 같이 반복문을 작성해야 평면을 채울 수 있다. POC 3,4 는 뒷면을 채울 때 쓰는 메서드로, 시계방향으로 점들을 출력하면 된다

이 메서드를 통해 DrawName에 다음과 같이 작성한다.

```
//앞면
    first_hanguel[0].draw_PoC_triangle_case1(first_hanguel[15], 1);
    first_hanguel[1].draw_PoC_triangle_case1(first_hanguel[14], 1);
    first_hanguel[4].draw_PoC_triangle_case1(first_hanguel[13], 1);
    first_hanguel[6].draw_PoC_triangle_case1(first_hanguel[12], 1);
```

```
    first_hanguel[8].draw_PoC_triangle_case1(first_hanguel[11], 1);
    first_hanguel[9].draw_PoC_triangle_case1(first_hanguel[10], 1);
    first_hanguel[2].draw_PoC_triangle_case2(first_hanguel[14], 1);
    first_hanguel[3].draw_PoC_triangle_case2(first_hanguel[14], 1);
    first_hanguel[5].draw_PoC_triangle_case2(first_hanguel[13], 1);
    first_hanguel[7].draw_PoC_triangle_case2(first_hanguel[12], 1);

    middle_hanguel[0].draw_PoC_triangle_case1(middle_hanguel[6], 1);
    middle_hanguel[2].draw_PoC_triangle_case1(middle_hanguel[5], 1);
    middle_hanguel[3].draw_PoC_triangle_case1(middle_hanguel[4], 1);
    middle_hanguel[1].draw_PoC_triangle_case2(middle_hanguel[6], 1);

    last_hanguel[0].draw_PoC_triangle_case1(last_hanguel[5], 1);
    last_hanguel[1].draw_PoC_triangle_case1(last_hanguel[4], 1);
    last_hanguel[2].draw_PoC_triangle_case1(last_hanguel[3], 1);

    //뒷면
    first_hanguel[15].draw_PoC_triangle_case3(first_hanguel[0], 0);
    first_hanguel[14].draw_PoC_triangle_case3(first_hanguel[1], 0);
    first_hanguel[13].draw_PoC_triangle_case3(first_hanguel[4], 0);
    first_hanguel[12].draw_PoC_triangle_case3(first_hanguel[6], 0);
    first_hanguel[11].draw_PoC_triangle_case3(first_hanguel[8], 0);
    first_hanguel[10].draw_PoC_triangle_case3(first_hanguel[9], 0);
    first_hanguel[2].draw_PoC_triangle_case4(first_hanguel[14], 0);
    first_hanguel[3].draw_PoC_triangle_case4(first_hanguel[14], 0);
    first_hanguel[5].draw_PoC_triangle_case4(first_hanguel[13], 0);
    first_hanguel[7].draw_PoC_triangle_case4(first_hanguel[12], 0);

    middle_hanguel[6].draw_PoC_triangle_case3(middle_hanguel[0], 0);
    middle_hanguel[5].draw_PoC_triangle_case3(middle_hanguel[2], 0);
    middle_hanguel[4].draw_PoC_triangle_case3(middle_hanguel[3], 0);
    middle_hanguel[1].draw_PoC_triangle_case4(middle_hanguel[6], 0);

    last_hanguel[5].draw_PoC_triangle_case3(last_hanguel[0], 0);
    last_hanguel[4].draw_PoC_triangle_case3(last_hanguel[1], 0);
    last_hanguel[3].draw_PoC_triangle_case3(last_hanguel[2], 0);
```

앞면과 뒷면은 다음과 같이 채울 수 있고, 옆면의 경우는 draw_Bezier_side를 이용한다

```
void BezierCurve::draw_Bezier_side(float z){
    glPointSize(5.0f); // 정점 크기 설정
    for(int i = 0; i < 10; i++){
```

```
        glBegin(GL_QUADS);
        glVertex3f(origin_x+ratio*this->getBezierPoint(i/10.0).x(),
origin_y-ratio*this->getBezierPoint(i/10.0).y(), 0);
        glVertex3f(origin_x+ratio*this->getBezierPoint(i/10.0).x(),
origin_y-ratio*this->getBezierPoint(i/10.0).y(), z);
        glVertex3f(origin_x+ratio*this->getBezierPoint((i+1)/10.0).x(),
origin_y-ratio*this->getBezierPoint((i+1)/10.0).y(), z);
        glVertex3f(origin_x+ratio*this->getBezierPoint((i+1)/10.0).x(),
origin_y-ratio*this->getBezierPoint((i+1)/10.0).y(), 0);
        glEnd();
    }
}
```

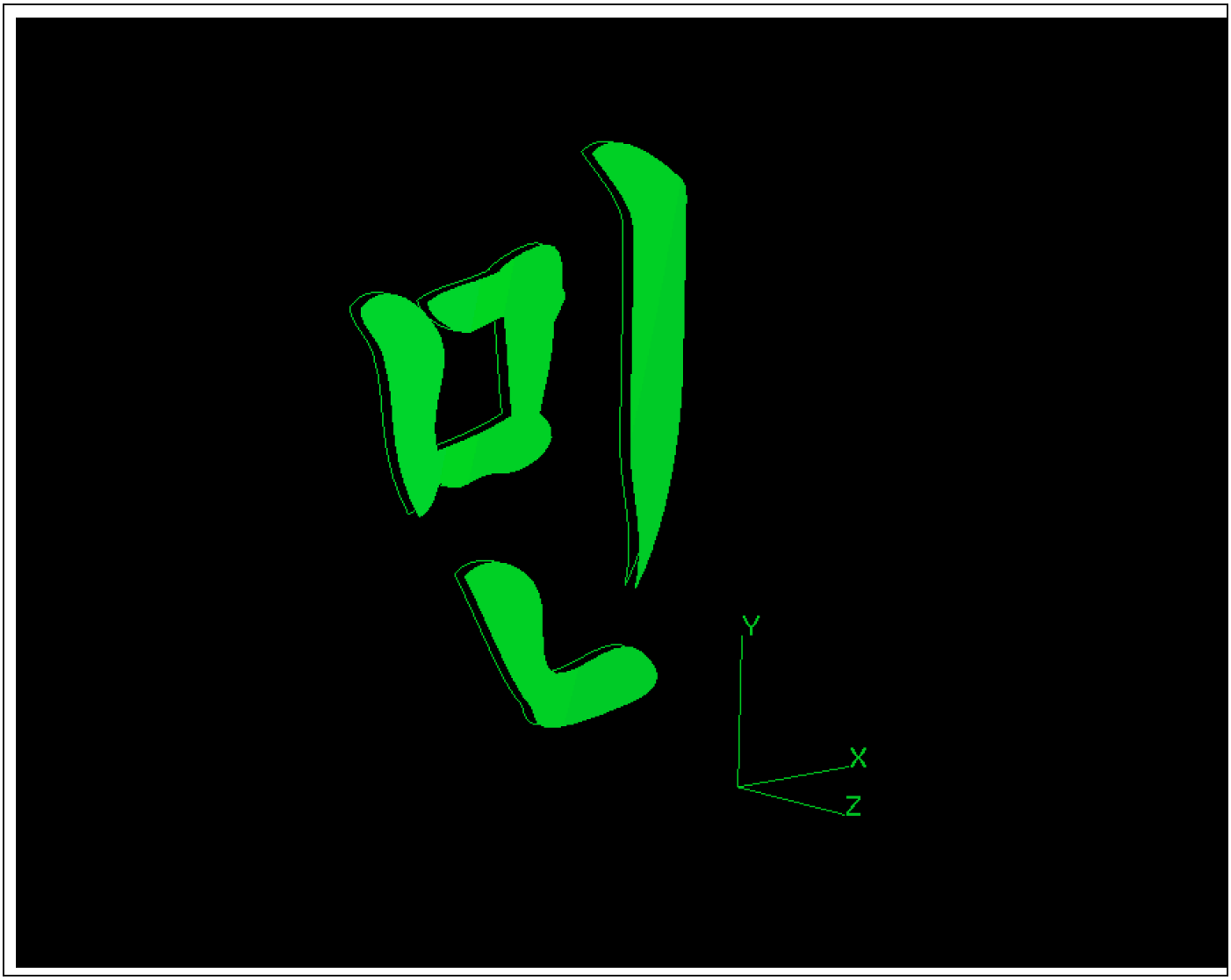
이 또한 반시계방향에 맞춰서 GL_QUADS를 이용하여 구현한다.

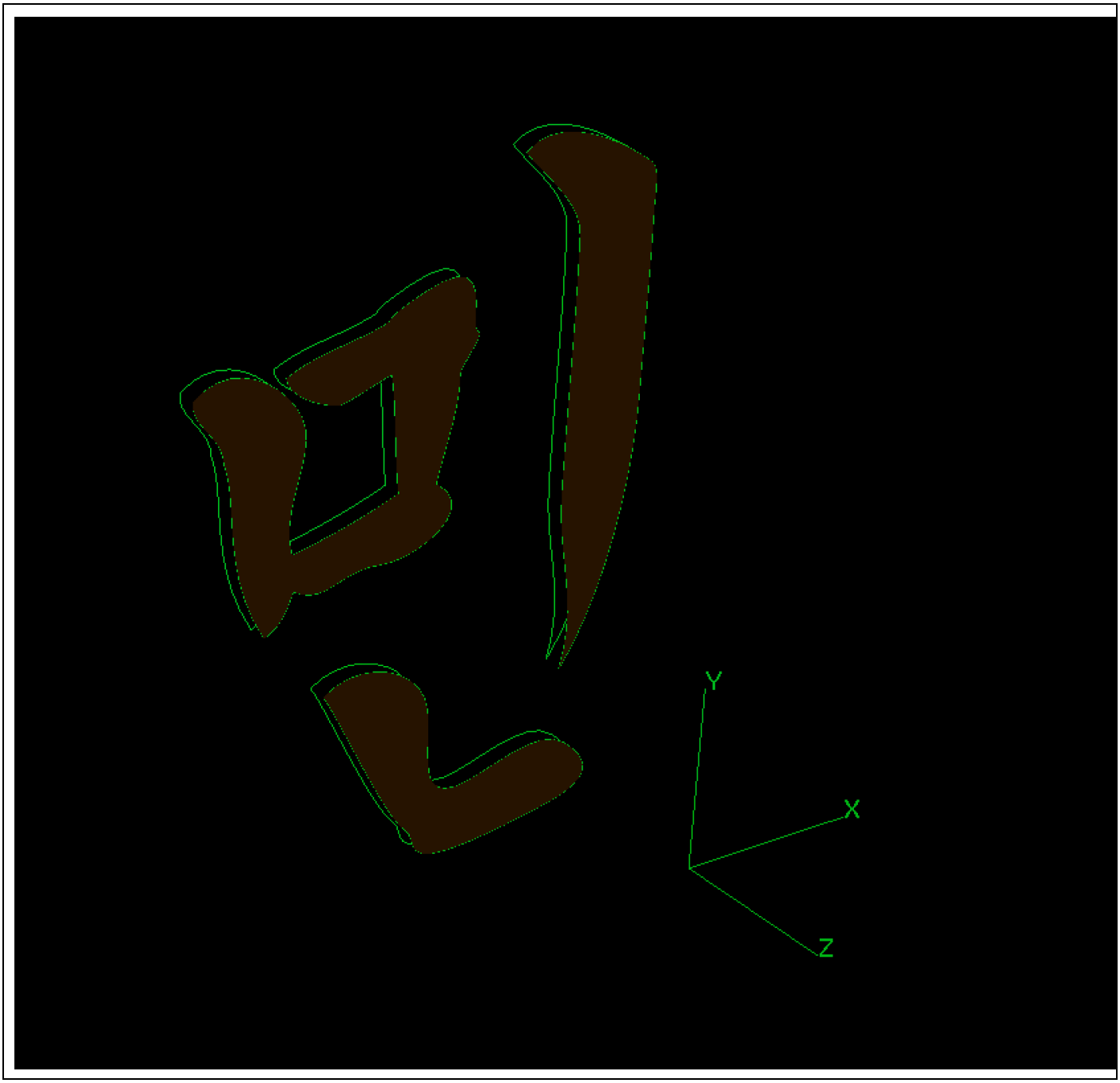
```
//옆면
    for(int i = 0; i < 16; i++){
        first_hanguel[i].draw_Bezier_side(1);
    }
    for(int i = 0; i < 7; i++){
        middle_hanguel[i].draw_Bezier_side(1);
    }
    for(int i = 0; i < 6; i++){
        last_hanguel[i].draw_Bezier_side(1);
    }
```

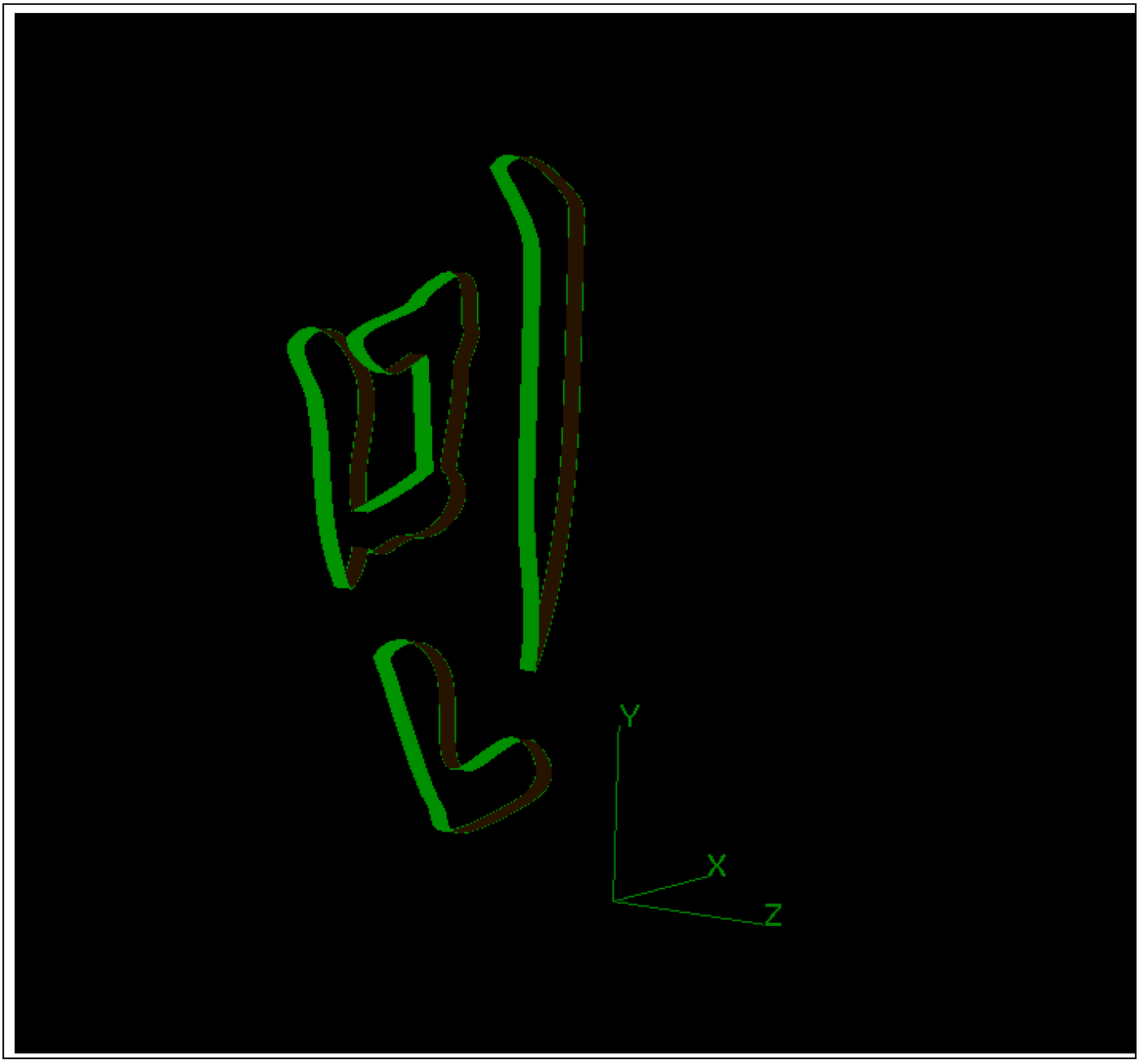
DrawName에 위와 같이 구현해주면,

다음과 같은 결과를 얻을 수 있다.

구현 중에 면이 겹치면, 고동색으로 출력되는 문제를 겪어, 각 면마다 구현한 결과를 첨부한다.


믿
Y
X
Z


Y
X
Z


Y
X
Z

결론

PA#3를 통해서, openGL을 이용해 Bezier Curve의 3차원 구현을 해 보았다. 기존 PA#2에서 continuity condition을 사용한 점과 3차원 구현이라는 점을 제외하고는 매우 유사했으나 과제를 통해서 기초적인 openGL의 사용법을 익혔고, 향후 구현할 BezierCurve의 기초가 될 것이다.

Reference

1. 2019-10738 조선해양공학과 장민준 PA#2 보고서