컴퓨터 그래픽스 HW3　　제출 기한: 12월 6일 까지

1. 코드 실행 결과를 볼 수 있는 실행 화면 캡쳐, 어떻게 코드를 짰는지와 이에 대한 설명 및 토의 사항을 적은 리포트 (pdf 포맷)와 실행 코드 (.cpp)들을 하나의 파일로 압축하여 제출
2. 가시 공간, 물체, polyline의 색은 필요한 경우 적절히 선택하고 리포트에 간단히 설명하자.

1. 강의 자료에 있는 (Dropbox에 링크된) DDA 알고리즘 코드와 MidpointLine (Lecture 11, page44) 함수를 이용하여 브레스넘 알고리즘 OpenGL 코드를 완성해 보고자 한다. 원래 DDA 알고리즘 코드에서 불필요한 부분은 지워도 된다. 단, MidpointLine 함수에서 x2>x1이고 선분의 기울기는 1보다 작다고 가정하자.

1) 선분의 양 끝점의 (x, y) 좌표가 (0, 0), (9, 5)로 주어졌을 때 먼저 수업시간에 배운 방식으로 손으로 (Lecture 11, page 45-46) 처럼 픽셀들을 그리고 거기에 line rasterization 결과를 표시해 보자. 업데이트 되는 D값을 아래와 같이 보여주자.

```
D=?
D=?
..
```

2) 이번에는 브레스넘 알고리즘을 완성한 후에 1)에서의 예에 대해서 최초 결정 변수 D값부터 종료될 때까지의 D값을 아래와 같이 화면에 출력해보자. 1)의 결과와 동일한가?

```
D=?
D=?
..
```

3) 브레스넘 알고리즘은 다음 후보 픽셀들의 중점을 이용하여 선분의 rasterization을 수행한다. 2)번의 경우 총 몇 개의 중점을 이용하였는가? 이를 코드로 구현하여 화면에 출력하여 보자.

```
총 이용한 중점 수=?
```

2. 수업시간에 사용한 브레스넘 알고리즘의 코드 (Lecture 11, page44)에서 결정 변수 D값은 선분의 기울기가 1이하 (혹은 기울기의 절대값이 1이하)이고 x2>x1이라고 가정하였다. 이 함수가 x2>x1이면서 선분의 기울기가 1보다 큰 경우에 대해서도 동작하도록 코드를 바꾸고자 한다. 기울기가 1보다 크다면 처음 위치에서 북쪽 픽셀이나 동북쪽 픽셀 중의 하나를 선택하게 되므로 Lecture 11, page 32의 결정 변수 계산이 바뀔 것으로 예상 된다.

1) 기울기가 1보다 큰 경우에 결정 변수를 계산하는 방법을 수식으로 설명해보고 북쪽 픽셀과 동북쪽 픽셀을 선택 시에 각각 결정 변수를 update 하는 방법을 설명해 보자.

2) 1)에서 수정한 방법을 코드로 구현하여 선분의 기울기가 1보다 큰 경우에 대해서 (x1, y1, x2, y2)=(0, 0, 5, 9)에 대하여 수정한 코드를 실행해 보고 이 때 D값이 어떻게 변하는지 아래와 같이 출력해 보자.

```
D=?
D=?
..
```

3. 이러닝에 올라가있는 'block.off' 메쉬 파일은 off 파일 포맷을 사용하여 메쉬를 표현하였다. 파일을 편집기에서 열어보면 1번째 라인은 파일 포맷을 나타낸다 (이 부분은 메모리에 저장할 필요는 없다). 2번째 line은 vertex 수, face 수, edge 수를 의미 한다. 이 예에서는 edge 수를 사용하지 않으므로 무시하자. 그 다음 라인부터는 vertex 수만큼 (이 예에서는 2,132개) vertex 의 x, y, z 좌표 정보가 담겨 있다. 가장 위부터 vertex의 index를 나타낼 때 0번 vertex, 그 다음 line은 1번 vertex 이런 순서이다. 그 후에는 face 수만큼 line이 있는데 각 line 에는 각 face의 vertex 수, 각 face를 이루는 vertex

index (몇 번째 vertex) 정보가 있다. Off 파일 포맷에 대한 보다 많은 정보는 아래 링크를 활용하자. https://en.wikipedia.org/wiki/OFF_(file_format)

1) C나 C++의 파일 입출력을 이용하여 위의 'trim-star.off' 메쉬 파일을 읽어보자. 배열을 만들 때 1번째 line에서 읽은 vertex 수, face 수만큼 각각의 vertex, face 배열을 만들고 for 반복문을 이용하여 각 line 별로 읽으면서 Vertex의 x, y, z 좌표 정보 및 face 정보를 각 배열에 저장 하자.

2) 아래 Pseudo 코드를 이용하여 1)에서 각 배열에 저장한 메쉬를 polygon으로 OpenGL을 이용하여 출력해보고 그 결과를 스크린샷으로 레포트에 넣어보자. Meshlab으로 시각화한 결과와 비슷하게 나오는가? 각 polygon의 색은 임의로 정하고 적당한 가시 부피를 사용하자. 조명 처리는 따로 하지 않는다.

```
for (each face, f, in the mesh)
{
    glBegin(GL_POLYGON)
    for(each vertex, v, in face, f)
    {
        glVertex3f(vertex v의 좌표);
    }
glEnd();
 }
```

4. 아래는 3차 (cubic) Bezier curve를 2번 사용하고 끝에 polyline을 연결하여 만들었다. 적당한 가시공간을 설정하고 Bezier curve를 위한 control points들을 설정하고 아래와 같은 모양을 만들어 보자. 어떻게 control points들을 설정하였고 polyline을 사용하였는지 설명하고 실행 결과를 스크린샷으로 레포트에 넣어보자.