R E P O R T

컴퓨터그래픽스및실습

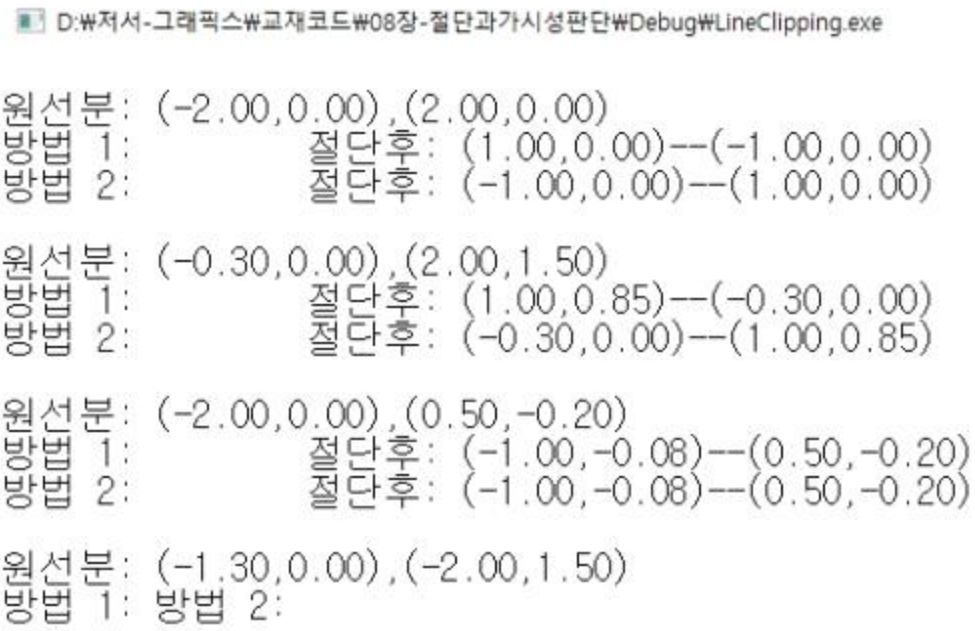
실습과제 09

과목명	컴퓨터그래픽스및실습
분반	01
교수	최 영 규
학번	2020136129
이름	최 수 연
제출일	2022년 11월 13일 일요일

[문제 분석 및 해결 방법]

[문제 분석]

- 선분 절단 알고리즘 구현하기
  - 코헨-서덜랜드의 알고리즘 구현: clipLine_CS()
  - 리앙-바스키 알고리즘 구현: clipLine_LB()
  - 테스트 코드 작성



[해결 방법]

- 먼저 코헨-서덜랜드 알고리즘의 경우에는 Encode, SwapIfNeeded, Accept, Reject 조건에 대한 함수를 정의하고, 본 함수들을 사용하여 가시부피인 윈도우 크기에 맞게 선분을 자르는 코드를 구현한다.
- 리앙-바스키 알고리즘의 경우에는 네 개의 부등식을 통해, Δx, $-\Delta x$, Δy, $-\Delta y$를 정의하고, 가시부피인 윈도우 크기의 상하좌우에 대하여 선분이 외부에서 내부로 들어가는지, 선분이 내부에서 외부로 나가는지, 아니면 선분이 수직선 또는 수평선인지 처리를 하여 윈도우 크기에 맞게 선분을 갱신한다.

[주요 설명 코드]

Cohen-Sutherland

```
enum Boundary { LEFT = 1, RIGHT = 2, BOTTOM = 6, TOP = 8 };
// 열거형을 통해 상하좌우에 대한 비트 위치를 나타냄
char Encode(Point2D p, Point2D min, Point2D max) {
    unsigned char code = 0x00;
    if (p.x < min.x) code = code | LEFT;
    if (p.x > max.x) code = code | RIGHT;
    if (p.y < min.y) code = code | BOTTOM;
    if (p.y > max.y) code = code | TOP;
    return (code);
} // 선분에 대한 한 점과 가시부피의 min, max를 받아 해당 점을 min, max에 따라 상하좌우에 대해 코드화하여 반환한다.
```

```
void SwapIfNeeded(char& c1, char& c2, Point2D& p, Point2D& q) {
    if (c1 == 0) {
        Point2D r = p; p = q; q = r;
        char tmp = c1; c1 = c2; c2 = tmp;
    }
} // code1이 0이면 두 점을 Swap 한다.
inline bool Accept(char a, char b) { return (a | b) == 0; }
inline bool Reject(char a, char b) { return (a & b) != 0; }
// 인라인 함수를 사용하여 Accept와 Reject 함수의 반환 값을 bool 형으로 반환한다. 반환할 때 Accept는 인코딩된 코드 값 a 또는 b가 0이면 true를 반환하고, 아니면 false를 반환한다. Reject의 경우에는 인코딩된 코드 값 a와 b가 0이 아닐 때 true를 반환하고, 아니면 false를 반환한다.
```

```
void clipLine_CS(Point2D min, Point2D max, Point2D p, Point2D q) {
    char code1, code2; // 인코딩한 값을 받는 code1, code2을 선언한다.
    bool done = false, draw = false;
   // 코드가 끝났음을 알리는 done과 코드를 출력하기 위한 draw를 선언한다.
    float m = 0;
    while (!done) {
        code1 = Encode(p, min, max);
        code2 = Encode(q, min, max);
        if (Accept(code1, code2)) {
            done = draw = true;
        }
        else if (Reject(code1, code2)) done = true;
        else {
            SwapIfNeeded(code1, code2, p, q);
            if (p.x != q.x) m = (q.y - p.y) / (q.x - p.x);
            if (code1 & LEFT) {
                p.y += (min.x - p.x) * m;
```

```
                p.x = min.x;
            }
            else if (code1 & RIGHT) {
                p.y += (max.x - p.x) * m;
                p.x = max.x;
            }
            else if (code1 & BOTTOM) {
                if (q.x != p.x) q.x += (min.y - p.y) / m;
                q.y = min.y;
            }
            else if (code1 & TOP) {
                if (q.x != p.x) q.x += (max.y - p.y) / m;
                q.y = max.y;
            }
        }
    }
    if (draw) drawLine(p, q);
```

} // 만약 done이 false일 경우 while문을 계속 반복하는데, while문 내부에서는 code1과 code2를 Encode 함수에 각 점과 min, max 값을 인수로 넣어 반환한 값을 넣는다. 그리고 if문을 통해 Accept 함수에 code1, code2를 넣었을 때 true가 될 경우, 두 점 모두 가시부피 안에 있으므로 바로 결과를 출력한다. 그렇지 않고 만약 Reject 함수에 code1, code2를 넣었을 때 true가 될 경우, 두 점 모두 가시부피 안에 있지 않으므로, 화면에 출력하지 않고 바로 while문을 종료한다. 그렇지 않을 경우, 두 점 중 하나는 가시부피 외부, 하나는 가시부피 내부에 있는 경우로, 가시부피인 윈도우의 상하좌우 경계선에 맞게 선분을 잘라서 출력한다.

Liang-Barsky

```
bool clipTest(float p, float q, float u1, float u2) {
    float r;
    bool result = true;
    if (p < 0) {            // 외부에서 내부로 들어감
        r = q / p;
        if (r > u2) result = false;
        else if (r > u1) {
            u1 = r;
            g_u1 = u1;
        }
    }
    else if (p > 0) {       // 내부에서 외부로 나감
        r = q / p;
        if (r < u1) result = false;
        else if (r < u2) {
            u2 = r;
            g_u2 = u2;
```

```
        }
    }
    else {                  // 수직 또는 수평선 (평행선)
        if (q < 0) result = false;
    }
    return result;
}
```

// 만약 점 p가 0보다 작으면, 점 p는 외부에 있는 경우로, u1을 갱신해준다. 이때 r을 점q/점p 값을 저장하고, r이 만약 u2보다 크면 reject 한다. 그렇지 않고, r이 u1보다 클 경우, u1을 r 값으로 갱신해준다.

// 만약 점 p가 0보다 크면, 점 q가 외부에 있는 경우로, u2을 갱신해준다. 이때 r을 점q/점p 값을 저장하고, r이 만약 u1보다 작으면 reject 한다. 그렇지 않고, r이 u2보다 작을 경우, u2를 r 값으로 갱신해준다.

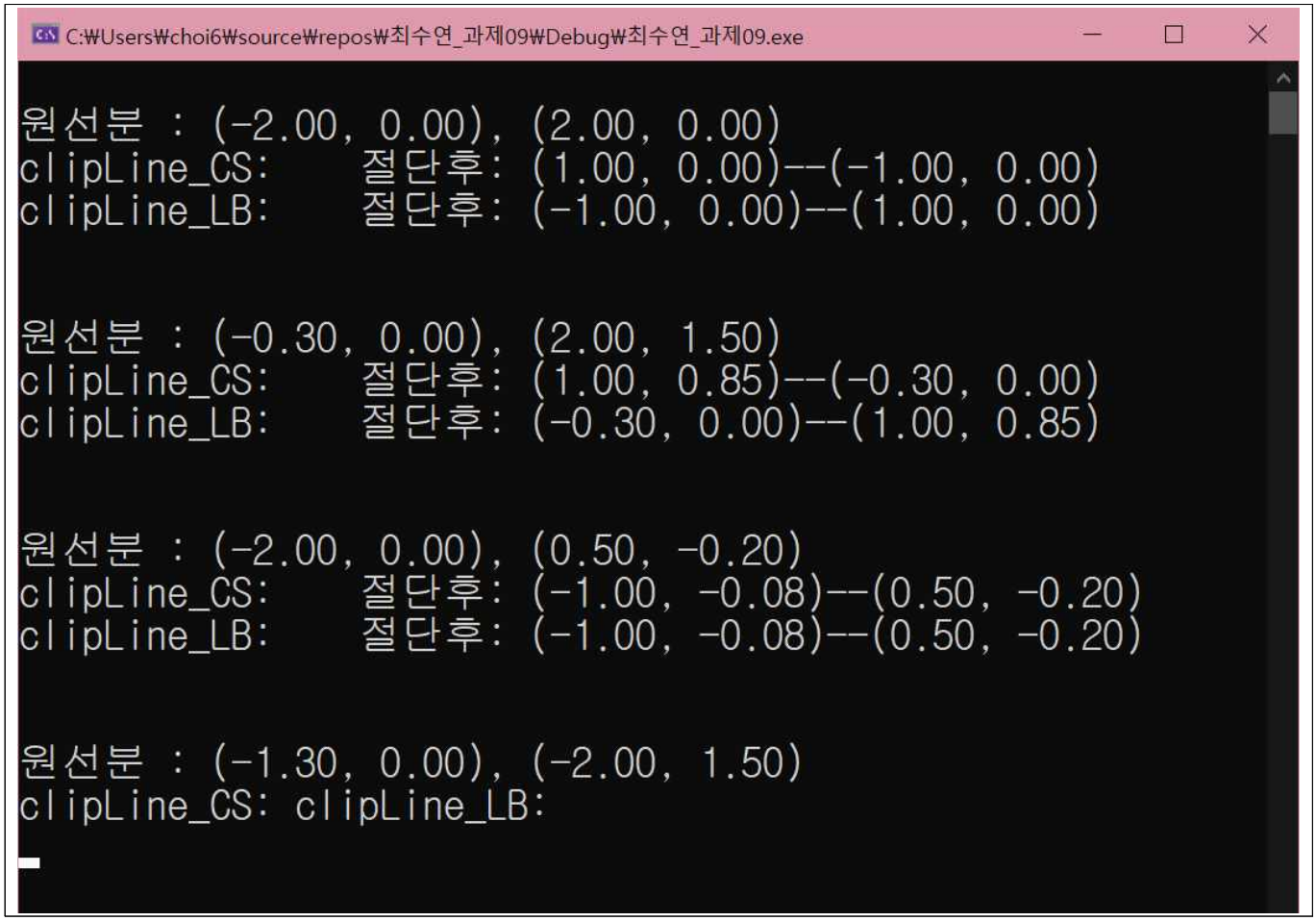
// 만약 위 두 경우 모두 아니면 점 p가 0으로, 수직 또는 수평선과 같은 평행선일 경우에 해당한다. 이때 만약 점 q가 0보다 작으면 해당 직선이 경계 밖에 있는 것이므로 reject 한다.

```
void clipLine_LB(Point2D min, Point2D max, Point2D p, Point2D q) {
    float dx = q.x - p.x, dy = q.y - p.y;
    if (clipTest(-dx, p.x - min.x, g_u1, g_u2)) {                  // left
        if (clipTest(dx, max.x - p.x, g_u1, g_u2)) {               // right
            if (clipTest(-dy, p.y - min.y, g_u1, g_u2)) {          // lower
                if (clipTest(dy, max.y - p.y, g_u1, g_u2)) {       // upper
                    if (g_u2 < 1) {
                        q.x = p.x + g_u2 * dx;
                        q.y = p.y + g_u2 * dy;
                    }
                    if (g_u1 > 0) {
                        p.x = p.x + g_u1 * dx;
                        p.y = p.y + g_u1 * dy;
                    }
                }
            }
        }
    }
    if((g_u1 != 0) || (g_u2 != 1)) drawLine(p, q);
}
```

// 두 점에 대한 Δx, Δy를 선언하고, clipTest() 함수를 호출하여 상하좌우에 대한 clipTest를 진행한다. clipTest() 함수를 통해 반환된 값이 만약 true이면, if문 내부로 들어오고, 상하좌우에 대한 선분의 검사가 끝나면 clipTest() 함수에서 갱신된 u1, u2에 대해, u2가 만약 1보다 작으면 점 q를 갱신하고, u1이 0보다 크면 점 p를 갱신한다.

// u2가 만약 1보다 작을 경우, 점 p에 u2와 x, y, 각각 Δx, Δy를 곱한 값을 더하여 점 q를 갱신하고, u1이 만약 0보다 클 경우, 점 p에 u1과 x, y, 각각 Δx, Δy를 곱한 값을 더하여 점 p점을 갱신한다.

[실행 결과]



[고찰 및 느낀점]

이번 8장 절단 및 가시성 판단에 대한 이론 수업을 들었을 때는 처음에 이해가 잘되지 않아서 걱정이 많았다. 이번 주차는 심지어 강의 영상도 따로 없어서, 과제를 하는 데에도 이해가 안 되는 부분에 대한 이론 내용을 인터넷으로 따로 찾아서 다시 보느라 시간이 오래 걸렸다. 그래도 두 알고리즘을 열심히 공부한 덕분에 기말고사 준비할 때는 좀 더 수월할 것 같다.