컴퓨터그래픽스및실습

실습과제 09

과목명/분반	컴퓨터그래픽스및실습/01	담당교수	최영규 교수님
학 과 명	컴퓨터공학부	제 출 일	2023 / 11 / 17
학번	2019136098	이름	이우진

실습과제09

선분 절단 알고리즘 구현하기
– 코헨-서덜랜드의 알고리즘 구현: clipLine_CS()
– 리앙-바스키 알고리즘 구현: clipLine_LB()
– 테스트 코드 작성
• 다음 코드 참조
• 콘솔 응용 프로그램으로 작성해도 됨
• 여러 영역의 점들을 이용해 테스트할 것

각 문제에 대한 분석과 자신이 해결한 방법 설명

실습과제09는 선분 절단 알고리즘인 코헨-서덜랜드의 알고리즘과 리앙-바스키 알고리즘을 구현하는 과제다. 이 과제를 해결하려면 우선 코헨-서덜랜드의 알고리즘과 리앙-바스키 알고리즘에 대해 알아야한다.

코헨-서덜랜드의 알고리즘은 2차원 공간을 9개 영역으로 분할한 다음 연산을 하는 알고리즘이다. Clipping Window를 0000으로 하고 위에는 1000(8), 아래는 0100(4), 왼쪽은 0001(1), 오른쪽은 0010(2)로 나타낸다. 그리고 accept와 reject 조건을 검사하며 교차점을 계산하여 0000에 들어가는 선분 좌표를 찾는 알고리즘이다. 이에 대해 코드로 구현은 8장 강의자료 26페이지의 pseudo 코드를 이용하여 c++로 구현하였다.

리앙-바스키 알고리즘은 선분을 매개변수 방정식으로 표현한 후, 그 매개변수의 범위를 윈도우 범위에 맞추어 Clipping하는 알고리즘이다. 네 개의 부등식 정형화를 통해 총 3개의 경우를 찾는데 평행선의 경우, 선분이 외부에서 내부로 들어가는 경우, 선분이 내부에서 외부로 나가는 경우 3가지가 있다. 이를 이용하여 cliptest를 진행하고 최종적으로 0000안에 남아있는 선분 좌표를 찾는 알고리즘이다. 이에 대해 코드로 구현은 8장 강의자료 33페이지의 pseudo 코드를 이용하여 c++로 구현하였다.

자신이 구현한 주요 코드 및 코드 설명

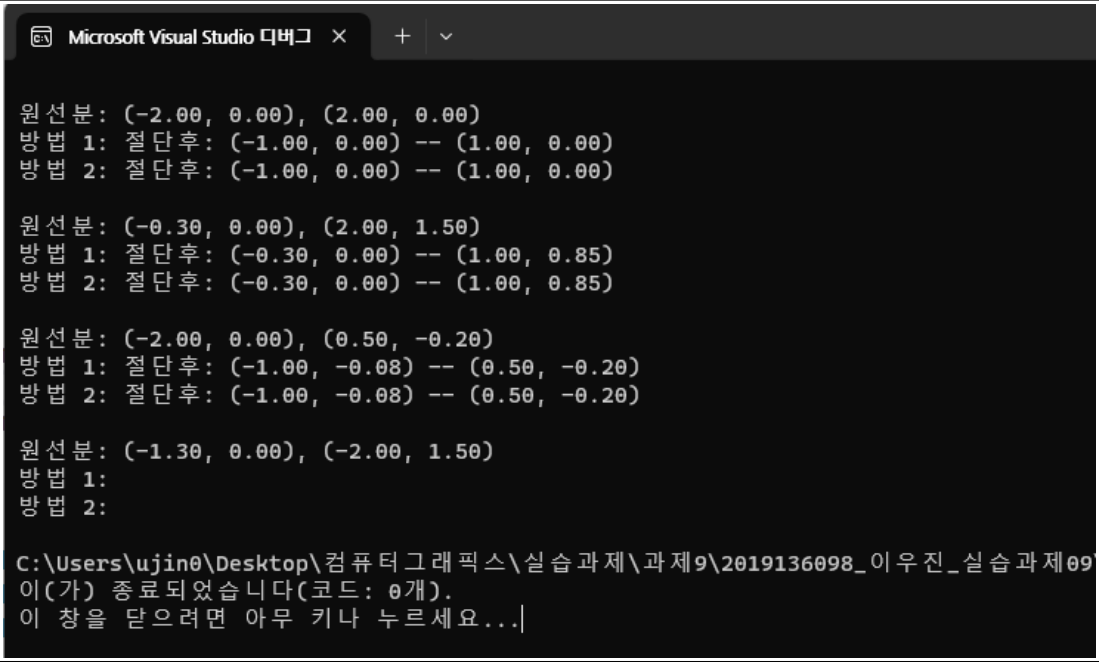
```
// 코헨-서더랜드 알고리즘
// Clipping Window 경계를 기준으로 점(x, y)의 위치를 결정하는 computeOutCode 함수
int computeOutCode(float x, float y, Point2D min, Point2D max) {
    int code = INSIDE;
    // 점이 Clipping Window의 왼쪽, 오른쪽, 아래쪽, 위쪽에 있는지 확인
    if (x < min.x) code |= LEFT;
    else if (x > max.x) code |= RIGHT;
    if (y < min.y) code |= BOTTOM;
    else if (y > max.y) code |= TOP;
    return code;
}
// Clipping Window 경계를 기준으로 점(x, y)의 위치를 결정하는 clipLine 함수
void clipLine(Point2D min, Point2D max, Point2D p1, Point2D p2) {
    int outcode0 = computeOutCode(p1.x, p1.y, min, max); // 첫 번째 점의 outcode 계산
    int outcode1 = computeOutCode(p2.x, p2.y, min, max); // 두 번째 점의 outcode 계산
    bool accept = false;
    while (true) {
        if (!(outcode0 | outcode1)) { // 두 점 모두 Clipping Window 내부에 있으면 accept
= true
```

```
                accept = true;
                break;
            }
            else if (outcode0 & outcode1) { // 두 점이 모두 동일한 경계 밖에 있으면 accept = false
                break;
            }
            else { // 두 개중 하나가 Clipping Window 내부에 있으면
                float x, y;
                int outcodeOut = outcode0 ? outcode0 : outcode1; // Clipping할 점의 outcode를 결정함
                // Clipping할 점과 Clipping Window 경계와의 교차점을 계산
                float slope = (p2.y - p1.y) / (p2.x - p1.x); // 선분의 기울기를 계산
                if (outcodeOut & TOP) { // 위쪽 경계와 교차하는 경우
                    x = p1.x + (max.y - p1.y) / slope;
                    y = max.y;
                }
                else if (outcodeOut & BOTTOM) { // 아래쪽 경계와 교차하는 경우
                    x = p1.x + (min.y - p1.y) / slope;
                    y = min.y;
                }
                else if (outcodeOut & RIGHT) { // 오른쪽 경계와 교차하는 경우
                    y = p1.y + (max.x - p1.x) * slope;
                    x = max.x;
                }
                else if (outcodeOut & LEFT) { // 왼쪽 경계와 교차하는 경우
                    y = p1.y + (min.x - p1.x) * slope;
                    x = min.x;
                }
                // 교차점으로 Clipping할 점의 위치를 업데이트하고 outcode 재계산
                if (outcodeOut == outcode0) {
                    p1.x = x;
                    p1.y = y;
                    outcode0 = computeOutCode(p1.x, p1.y, min, max);
                }
                else {
                    p2.x = x;
                    p2.y = y;
                    outcode1 = computeOutCode(p2.x, p2.y, min, max);
                }
            }
        }
        if (accept) { // 선분이 Clipping Window 내부에 있으면 결과를 출력
```

```
            printf("절단후: (%4.2f, %4.2f) -- (%4.2f, %4.2f)₩n", p1.x, p1.y, p2.x, p2.y);
        }
        else { // 선분이 Clipping Window 외부에 있으면 아무것도 출력하지 않음
            printf("₩n");
        }
    }
    // 리앙-바스키 알고리즘
    // Clipping Test를 수행하는 clipTest 함수
    bool clipTest(float p, float q, float& u1, float& u2) {
        float r;
        bool result = true;
        if (p < 0.0f) { // p < 0.0f이고
            r = q / p; // 교차하는 값을 계산
            if (r > u2) { // r > u2이면
                result = false; // Clipping 되지 않음
            }
            else if (r > u1) { // r > u1이면
                u1 = r; // u1 업데이트
            }
        }
        else if (p > 0.0f) { // p > 0.0f이고
            r = q / p; // 교차하는 값을 계산
            if (r < u1) { //r < u1이면
                result = false; // Clipping 되지 않음
            }
            else if (r < u2) { //r < u2이면
                u2 = r; // u2 업데이트
            }
        }
        else if (q < 0.0f) { // q < 0.0f이면
            result = false; // Clipping 되지 않음
        }
        return result;
    }
    // Clipping Window에 선분을 Clipping하는 함수
    void clipLine2(Point2D min, Point2D max, Point2D p1, Point2D p2) {
        float u1 = 0.0f, u2 = 1.0f; // 매개변수화된 선분의 초기 하한과 상한
        float dx = p2.x - p1.x, dy = p2.y - p1.y;
        // clipTest 함수를 이용하여 Clipping 테스트를 수행하고 Clipping Window에 선분의 위치를 결정
        if (clipTest(-dx, p1.x - min.x, u1, u2)) {
            if (clipTest(dx, max.x - p1.x, u1, u2)) {
                if (clipTest(-dy, p1.y - min.y, u1, u2)) {
```

```
                    if (clipTest(dy, max.y - p1.y, u1, u2)) {
                        if (u2 < 1.0f) {
                            p2.x = p1.x + u2 * dx;
                            p2.y = p1.y + u2 * dy; // 상한에 따라 끝점 업데이트
                        }
                        if (u1 > 0.0f) {
                            p1.x = p1.x + u1 * dx;
                            p1.y = p1.y + u1 * dy; // 하한에 따라 시작점 업데이트
                        }
                        // 선분이 Clipping Window 내부에 있으면 결과를 출력
                        printf("절단후: (%4.2f, %4.2f) -- (%4.2f, %4.2f)₩n", p1.x, p1.y, p2.x,
p2.y);
                    }
                }
            }
        }
        else { // 선분이 Clipping Window 외부에 있으면 아무것도 출력하지 않음
            printf("₩n");
        }
}
```

다양한 입력에 대한 테스트 결과

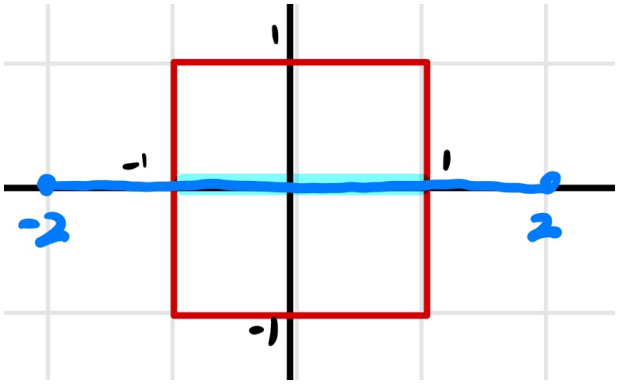


해당 문제에 대한 고찰

우선 이번 과제에서 clipping window는 왼쪽 아래 좌표 (-1, -1), 오른쪽 위 좌표 (1, 1)의 정사각형 모양을 가진다. 그래서 원선분에서 나온 답이 맞는지 확인하는 과정을 가진다. 먼저 교수님께서 올려주신 실습09 자료와 내가 구현한 코드의 답을 먼저 맞춰봤는데 원선분(-2, 0), (2, 0)과 원선분 (-0.3, 0.0), (2, 1.5)에서 방법 1의 절단 후 나타내는 것이 위치가 서로 바뀌어있는데 이는 문제가 없다고 생각이 된다. 그 이유는 절단 후 선분의 좌표를 표현하는 것이 이번 과제 문제이기에 순서가 바뀌어도 정삭적으로 답을 표현한

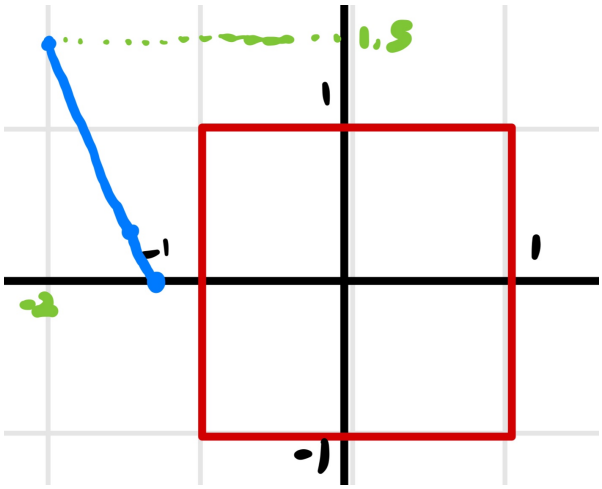
다고 생각을 했다.

이제 내가 구현한 코드에서 나온 답이 실제로도 맞는지를 파악해본다.
우선 첫 번째 원선분인 (-2.00, 0.00), (2.00, 0.00)일 때 경우이다. 아래 그래프를 보면 쉽게 파악할 수 있다.



파란색은 원선분, 빨간색 정사각형은 clipping window를 나타낸다. 코헨-서덜랜드 알고리즘이랑 리앙-바스키 알고리즘에서 구하는 값은 원선분에서 저 정사각형 안에 들어가는 부분을 찾기 위해 자르고 하는 거라서 그림으로 그려보면 원하는 답이 명확하게 보인다. 그림에서도 clipping window에 들어가는 원선분의 부위는 (-1,0), (1,0)인 것을 알 수 있다. 따라서 코드로 구현한 선분 절단 알고리즘은 정상적으로 구현이 되었다는 것을 확인할 수 있었다. 이 방법대로 두 번째와 세 번째 원선분의 값이 맞는 값인지 확인할 수 있을 것이다.

그럼 네 번째 원선분은 왜 둘다 값이 안나왔을까?
이 또한 그래프를 그려보면서 확인할 수 있다.



이 그래프를 보면 파란색이 원선분, 빨간색 정사각형이 clipping window인데 원선분이 정사각형에 들어가는 부분이 하나도 없는 것을 확인할 수 있다. 그래서 절단을 할 수 없고 그래서 2개의 선분 절단 알고리즘에서도 답을 구할 수 없었다.

이번 과제에 대한 느낀점

pseudo 코드가 있어서 코드를 구현하기가 수월했던 거 같다. 그리고 처음 실습자료의 예시를 봤을 때는 왜 저 값이 나왔는지 이해가 되지 않았는데 직접 손으로 그래프를 그려보면서 이 경우에는 이런 값이 나왔고 이 경우에는 값이 안 나왔는지 좀 더 쉽게 알게 되었다. 또한 강의를 들을 때는 모호했었는데 pseudo 코드와 강의자료를 읽어보면서 선분 절단 알고리즘을 이해하게 된 거 같다.

궁금한 점이나 건의사항
궁금한 점이나 건의사항은 없습니다.