# 컴퓨터그래픽스및실습

## 실습과제 10

---


| 과목명/분반 | 컴퓨터그래픽스및실습/01 | 담당교수 | 최영규 교수님 |
|---|---|---|---|
| 학 과 명 | 컴퓨터공학부 | 제 출 일 | 2023 / 11 / 24 |
| 학번 | 2019136098 | 이름 | 이우진 |

**실습과제10**

- Bresenham의 선분 그리기 알고리즘을 모든 입력 상황에 대해 처리할 수 있도록 확장하라.
- Midpoint Circle 알고리즘을 모든 입력 상황에 대해 처리할 수 있도록 확장하라. 특히 drawCirclePoint(x,y) 함수에서는 8 화소에 대해 그려야 할 것이다.

**각 문제에 대한 분석과 자신이 해결한 방법 설명**

Bresenham의 선분 그리기 알고리즘은 시작점과 끝점 사이의 수평 및 수직 거리를 계산해서 이를 픽셀의 위치를 오차 값에 기반하여 결정하였다.

Midpoint Circle 알고리즘은 주어진 원의 중심점과 반지름을 이용하여 초기 점을 설정하고, 결정 변수를 사용하여 다음 점의 위치를 결정한다. 원의 1/8만 계산하고, 나머지는 대칭성을 이용하여 그린다.

이를 이용하여 코드로 구현하였다. 또한 키보드 입력과 swith문을 이용하여 같은 선분으로 Bresenham의 선분 그리기 알고리즘과 Midpoint Circle 알고리즘 선택하여 출력할 수 있도록 하였다.

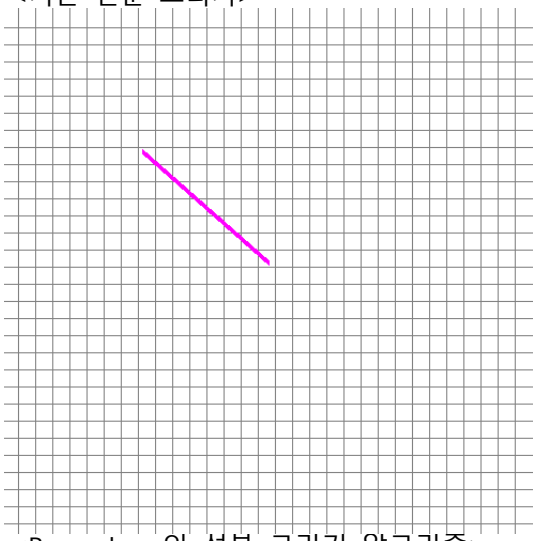
**자신이 구현한 주요 코드 및 코드 설명**

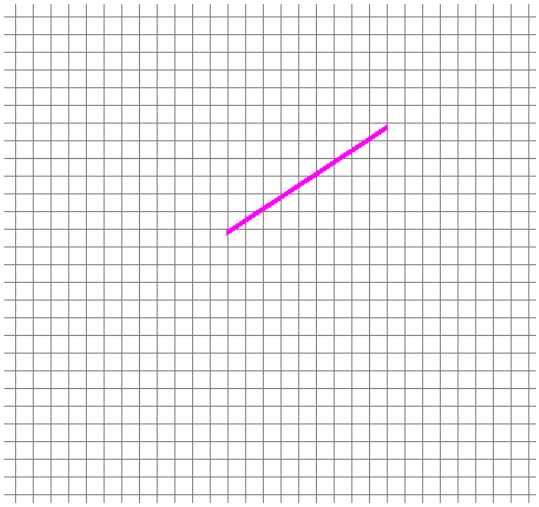
```
static bool bSwapXY = false; // displayPixel 함수에서 x와 y좌표를 바꿀지 여부를 제어하기 위해 선언
void displayPixel(int x, int y) {
	if (bSwapXY)
		swap(&x, &y);
	drawPixel(x, y);
} // 화면에 픽셀을 그리는 함수
void lineBresenham(GLint x0, GLint y0, GLint x1, GLint y1) {
	int dx = abs(x1 - x0), sx = x0 < x1 ? 1 : -1;
	int dy = -abs(y1 - y0), sy = y0 < y1 ? 1 : -1;
	// 시작점과 끝점 사이의 수평 및 수직 거리를 계산

	int err = dx + dy, e2;
	// 오차 값으로, 초기에는 dx와 dy의 합으로 설정

	for (;;) { // 무한 루프
		displayPixel(x0, y0); // 현재 위치의 픽셀 그리기
		if (x0 == x1 && y0 == y1) break; //  만약 현재 위치가 끝점에 도달했다면 루프를 종료
		e2 = 2 * err; // 다음 픽셀의 위치를 결정하는 데 사용
		if (e2 >= dy) { err += dy; x0 += sx; }
		// 오차 값이 dy보다 크거나 같다면, x좌표 갱신
		if (e2 <= dx) { err += dx; y0 += sy; }
		// 오차 값이 dx보다 작거나 같다면, y좌표 갱신
	}
} // bresenham의 선분 알고리즘 그리는 함수
void drawCirclePoint(int cx, int cy, int x, int y) {
	displayPixel(cx + x, cy + y);
```

```
		displayPixel(cx - x, cy + y);
		displayPixel(cx + x, cy - y);
		displayPixel(cx - x, cy - y);
		displayPixel(cx + y, cy + x);
		displayPixel(cx - y, cy + x);
		displayPixel(cx + y, cy - x);
		displayPixel(cx - y, cy - x);
} // 주어진 중심점을 기준으로 원의 8개의 점을 그리기
void circleMidPoint(GLint cx, GLint cy, GLint radius) {
		int x = 0;
		int y = radius;
		// x, y값 선언
		int d = 1 - radius;
		// 다음 점을 어디에 그릴지 결정하는 데 사용할 d 변수 선언

		drawCirclePoint(cx, cy, x, y); // 원의 첫 번째 점 그리기
		while (y > x) { // y가 x보다 클 때까지 반복
			if (d < 0) { // d가 0보다 작다면
				d = d + 2 * x + 3; //  y 좌표는 그대로 두고 x 좌표만 증가
			}
			else { // 그렇지 않으면,
				d = d + 2 * (x - y) + 5; // y 좌표를 감소시키고 x 좌표는 증가
				y--;
			}
			drawCirclePoint(cx, cy, x, y); // 새로운 x, y 좌표에 대해 원의 점들을 그리
기
			x++; // 원의 다음 점을 계산하기 위해 x좌표 1 증가
		}
} // circleMidPoint 그리는 함수
```

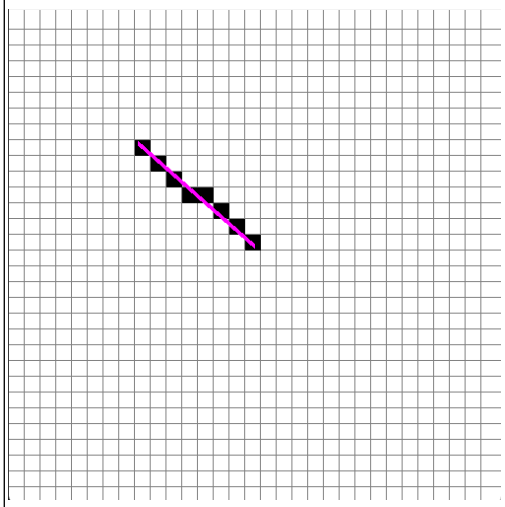
**다양한 입력에 대한 테스트 결과**

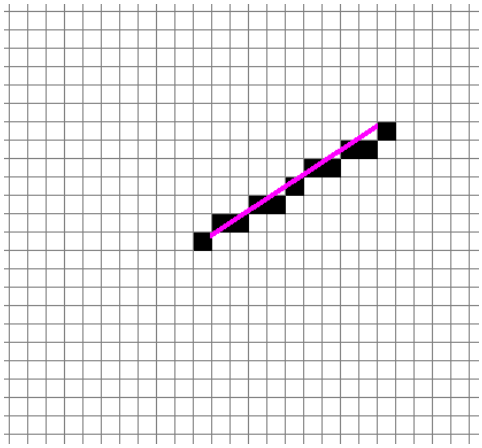
<기본 선분 그리기>



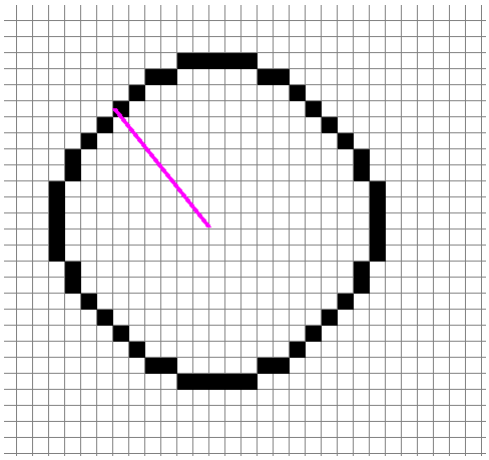


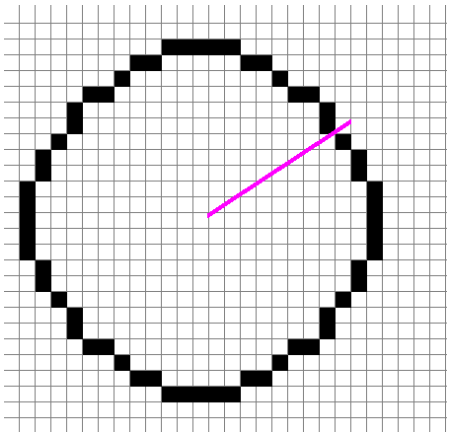
<Bresenham의 선분 그리기 알고리즘>





<Midpoint Circle 알고리즘>





**해당 문제에 대한 고찰**

교수님께서 올려주신 강의자료와 영상에서는 circleMidPoint를 그릴 때 radius와 x, y만을

이용하여 코드를 구현하셨는데 나는 그렇게는 되지 않아 cx와 cy를 추가 이용하여 코드를 구현하였다. 또한 다양한 입력 상황에 대해 처리할 수 있도록 확장하라고 하셔서 강의자료에 있는 코드를 수정하여 구현하였다.

radius와 x, y만으로 알고리즘을 어떻게 구현할 수 있는지 궁금했던 문제였던거 같다.

| **이번 과제에 대한 느낀점** |
| --- |
| 강의 자료 및 강의 영상에 코드가 나와있어서 코드를 구현하는데는 크게 어려움이 없었다. 이 과정을 통해 Bresenham의 선분 그리기 알고리즘과 Midpoint Circle 알고리즘을 이해하는 시간이 되었던 거 같다. 이번 과제에서는 2개 중 하나 고르는 과제여서 나머지 구현하지 않은 알고리즘도 구현해봐야겠다. |
| **궁금한 점이나 건의사항** |
| 궁금한 점이나 건의사항은 없습니다. |