

3조 프로젝트 결과보고서

제목 : CG Mini-Project

과	목	명	컴퓨터그래픽스
담	당	교	정 성 환
학		과	컴퓨터공학과
제	출	일	2017년 12월 10일

1. 프로젝트 개요

가. 추진목적 및 필요성

컴퓨터 그래픽스 시간에 배운 OpenGL를 직접 사용하여 그림판을 구현해보면서 폭 넓은 이해를 하는 것에 그 목적이 있다.

나. 추진 목표

윈도우에서 지원하는 그림판을 OpenGL을 사용하여 가능한 재현하는 것이 목표이다.

라. 역할 분담

김진우 - 기본구조 변경, 자유곡선, 별, 원그리기, 지우개, Undo, Redo

김태경 - 색상 반전

김태환 - UI, 축 반전, 색상 선택

마. 개발 환경

OS	window 10(64bit), Arch Linux(64bit)
개발 언어	C 언어
헤더파일	glut.h
	math.h
	stdlib.h

2. 수행 방법 및 주요 내용

가. 기본구조의 변경

화면에 곧 바로 그렸던 기존 방식과는 다르게 오브젝트 구조체와 정점의 구조체를 사용했다.

```
struct vertexNode
{
    GLint vx; // x 좌표
    GLint vy; // y 좌표
    struct vertexNode* next; //다음을 vertexNode를 가리킨다..
};

struct object
{
    GLfloat color[4]; //RGBA 값을 저장하는 곳
    struct vertexNode* vertex; // vertexNode의 Head
    int fill; // 짝찬 그림인지 여부
    int mode; // 그림 그리는 방식을 저장
    struct object* next; //다음 오브젝트를 가리킨다.
}*head, *tail, *lineObj, *draw, *redo;
// head, tail : Object의 처음과 끝을 나타낸다.
// lineObj : 자유 곡선을 만들기 위한 객체
// draw : 실시간으로 그려지는 것을 보여주기 위한 객체
// redo : redo를 사용했을 때 불러오기 위해 undo시 임시로 저장하기
        위한 객체의 시작지점.
```

구조체 안에 들어있는 정보는 drawList()를 통해 그려진다.

나. 자유곡선

마우스를 드래그 하는 동안의 경로를 리스트에 저장한다.

```
void motion(int x, int y) { // 마우스를 드래그 하는 동안 실행되는 함수
    if (draw_mode == DRAW && lineObj != NULL)
    {
        makeVertex(x, y, lineObj); //
        dragCheck = 1;
        drawObj(lineObj);

    }...
}
```

```

int main(int argc, char** argv)
{
    ...
    glutMotionFunc(motion); //마우스를 드래그 하는 동안 실행한다
    ...
}

```

다. 별그리기

별을 구성하는 10개의 점 좌표를 비율적으로 구성한다. 꼭 찬 별의 경우에는 GL_POLYGON이 오목 도형을 지원하지 않기 때문에 3각형을 3개를 그린다.

```

void CreateStar(int x1, int y1, int x2, int y2, struct object* p)
{
    int tmp = 0;
    if (x2 < x1) //
    {
        tmp = x2;
        x2 = x1;
        x1 = tmp;
    }
    if (y2 < y1)
    {
        tmp = y2;
        y2 = y1;
        y1 = tmp;
    }

    int xlength = x2 - x1; //별의 x 좌표 크기
    int ylength = y2 - y1; //별의 y 좌표 크기

    if (p->fill == 0) //별의 속이 비었을 경우
    {
        makeVertex(0.5*xlength + x1, y1, p);
        makeVertex(0.38*xlength + x1, 0.38*ylength + y1, p);
        makeVertex(x1, 0.4*ylength + y1, p);
        makeVertex(0.3*xlength + x1, 0.62*ylength + y1, p);
        makeVertex(0.19*xlength + x1, y2, p);
        makeVertex(0.5*xlength + x1, 0.77*ylength + y1, p);
    }
}

```

```

        makeVertex(0.81*xlength + x1, y2, p);
        makeVertex(0.7*xlength + x1, 0.62*ylength + y1, p);
        makeVertex(x2, 0.4*ylength + y1, p);
        makeVertex(0.62*xlength + x1, 0.38*ylength + y1, p);
    }//10개의 점에 좌표를 찍어서 잇는다.
    Else //별의 속이 짝 차 있는 경우
    {
        makeVertex(0.5*xlength + x1, y1, p);
        makeVertex(0.19*xlength + x1, y2, p);
        makeVertex(0.7*xlength + x1, 0.62*ylength + y1, p);
        //1번 삼각형

        makeVertex(0.5*xlength + x1, y1, p);
        makeVertex(0.3*xlength + x1, 0.62*ylength + y1, p);
        makeVertex(0.81*xlength + x1, y2, p);
        //2번 삼각형

        makeVertex(x1, 0.4*ylength + y1, p);
        makeVertex(0.5*xlength + x1, 0.77*ylength + y1, p);
        makeVertex(x2, 0.4*ylength + y1, p);
        //3번 삼각형
    } // 총 3개의 삼각형을 그려 별모양을 만든다.
}

```

라. 원그리기

해당 프로그램은 연산량을 신경 쓸 정도로 복잡하지 않아 브래스넘 알고리즘 보다는 간단하게 극좌표계를 사용하였다

```

void CreateCircle1(int centerx, int centery, int rad, int fill)
{
    struct object* p = makeObj(fill);
    for (int angle = 0; angle < 360; angle += 1)
    {
        int px = rad * cos(angle*0.017);
        int py = rad * sin(angle*0.017);

        makeVertex(centerx + px, wh - centery + py, p);
    }
    addObj(p, &head);
}

```

마. 지우개

일반적인 네모 그리기와 똑같지만 배경화면 색을 그림의 색으로 사용한다.

```
struct object* makeObj(int fill)
{
    ...
    Else // 지우개를 선택했을 경우
    {
        p->color[0] = bg_r;
        p->color[1] = bg_g;
        p->color[2] = bg_b;
        p->color[3] = 1.0;
    }
    return p;
}
```

바. Undo

오브젝트 리스트에서 마지막으로 생성된 오브젝트를 Redo 리스트로 옮긴다.(오브젝트 리스트에서는 제거);

```
void undo(struct object** head)
{
    struct object* p = *head; //검색의 시작지점 보통 head를 사용
    if ((p->next != NULL) && (p->next != tail))
    {
        while (p->next->next != tail)
            p = p->next; //마지막으로 전으로 생성된
                        // 오브젝트를 찾는다.
        push(p->next); //오브젝트의 다음을 Redo로 보낸다.
        p->next = tail; //p를 마지막이라고 표시한다.
        glutPostRedisplay(); //화면을 다시 그려준다.
    }
}
```

사. Redo

Redo 리스트에서 마지막으로 받은 오브젝트를 오브젝트 리스트로 옮긴다.

```
void Redo(struct object** head)
{
    struct object *p = *head;
    if ((redo != NULL) && (head != NULL))
    {
        while (p->next != tail)
        { // 마지막으로 생성된 오브젝트를 찾는다.
            p = p->next;
        }
        p->next = pop(); //Redo 리스트의 마지막값을 받는다.
        p->next->next = tail; // 방금 받은값을 마지막으로
        // 설정한다.
        glutPostRedisplay(); // 화면을 다시 그려준다.
    }
}
```

아. 색상 반전

RGB값에서 최대 색상은 1이기 때문에 Object에 저장되어 있는 RGBA 값에서 RGB 값을 1에서 원래 값을 빼준 값으로 바꿔준다.

```
void rever(struct object** head)
{
    bg_r = 1 - bg_r; //배경화면의 색도 반전시켜준다.
    bg_g = 1 - bg_g;
    bg_b = 1 - bg_b;

    struct object* p = *head;
    if (p->next != NULL)
    {
        while (p->next != tail) //마지막에 닿을 때 까지 반복
        {
            p = p->next;
            p->color[0] = 1 - p->color[0];
            p->color[1] = 1 - p->color[1];
            p->color[2] = 1 - p->color[2];
        }
    }
}
```

```

    }
    glutPostRedisplay(); //모두 바꾼 후 새 화면을 그린다.
}
}

```

자. UI

UI는 크게 메뉴 바와 기능 창으로 나뉘져 있으며 메뉴바에서는 끄기, 채우기, 배경색, 반전 등 화면을 전체적으로 바꾸는 기능이 존재하고, 기능 창엔 그리기와 색상 선택 같은 기능이 위치한다

```

void menubar() {
    glColor3f(0.9, 0.9, 0.9);
    glBegin(GL_POLYGON); //메뉴바를 그릴 공간을 만들어준다.
        glVertex2i(0, wh);
        glVertex2i(0, wh - 20);
        glVertex2i(ww, wh - 20);
        glVertex2i(ww, wh);
    glEnd();

    if (menubarcheck != 0) { //상단에서 메뉴가 선택됐을시
        int x1 = -5 * pow(menubarcheck - 1, 4)
            + 40 * pow(menubarcheck - 1, 3)
            - 85 * pow(menubarcheck - 1, 2)
            + 95 * (menubarcheck - 1);
        int x2 = -5 * pow(menubarcheck, 4)
            + 40 * pow(menubarcheck, 3)
            - 85 * pow(menubarcheck, 2)
            + 95 * menubarcheck;
        //선택된 x축의 계산한다.
        int y2 = wh - 20;
        int tmp;

        if (menubarcheck == 4) tmp = 80;
        else tmp = 50 * menubarcheck*menubarcheck
            - 130 * menubarcheck + 120;
        //메뉴아이템의 갯수를 계산한다.

        glColor3f(0.5, 0.5, 0.5);
        glBegin(GL_POLYGON); //선택된 메뉴를 색칠해준다.
            glVertex2i(x1, wh);
            glVertex2i(x1, wh - 20);

```



```

        glVertex2i(x2, wh - 20);
        glVertex2i(x2, wh);
    glEnd();

    glColor3f(0.9, 0.9, 0.9);
    y2 = wh - tmp;
    //메뉴아이템의 갯수의 크기만큼 네모를 그려준다.
    glBegin(GL_POLYGON);
        glVertex2i(x1, wh - 20);
        glVertex2i(x1, y2);
        glVertex2i(x2, y2);
        glVertex2i(x2, wh - 20);
    glEnd();

    //각 메뉴별 메뉴 아이템
    if (0 < menucheck && menucheck <= tmp / 20 - 1) {
        glColor3f(0.0, 0.0, 1.0);
        //마우스가 위치한 메뉴 아이템을 칠해준다.
        glBegin(GL_POLYGON);
            glVertex2i(x1, wh - menucheck * 20);
            glVertex2i(x1, wh - (menucheck + 1) * 20);
            glVertex2i(x2, wh - (menucheck + 1) * 20);
            glVertex2i(x2, wh - menucheck * 20);
        glEnd();
    }
    glColor3f(0.0, 0.0, 0.0);
    if (menubarcheck == 1)
    {
        StringText("Quit", 5, wh - 35);
    }
    else if (menubarcheck == 2)
    {
        StringText("On", 50, wh - 35);
        StringText("Off", 50, wh - 55);
    }
    else if (menubarcheck == 3)
    {
        StringText("Red", 95, wh - 35);
        StringText("Green", 95, wh - 55);
        StringText("Blue", 95, wh - 75);
        StringText("Cyan", 95, wh - 95);
        StringText("Magenta", 95, wh - 115);
        StringText("Yellow", 95, wh - 135);
    }

```

```

        StringText("White", 95, wh - 155);
        StringText("Black", 95, wh - 175);

    }
    else if (menubarcheck == 4)
    {
        StringText("Color", 205, wh - 35);
        StringText("X-axis", 205, wh - 55);
        StringText("Y-axis", 205, wh - 75);
    }

    glBegin(GL_LINE_LOOP);/
        glVertex2i(x1, wh - 20);
        glVertex2i(x1, y2);
        glVertex2i(x2, y2);
        glVertex2i(x2, wh - 20);
    glEnd();
    //창을 구분하는 선을 그어준다.
}

glColor3f(0.6, 0.6, 0.6);
glBegin(GL_LINES); //메뉴바를 구분해주는 선
    glVertex2i(0, wh - 20);
    glVertex2i(ww, wh - 20);
glEnd();

glBegin(GL_LINES); //메뉴바를 나눠주는 선을 그린다..
    glVertex2i(45, wh - 4);
    glVertex2i(45, wh - 16);

    glVertex2i(90, wh - 4);
    glVertex2i(90, wh - 16);

    glVertex2i(195, wh - 4);
    glVertex2i(195, wh - 15);

    glVertex2i(300, wh - 4);
    glVertex2i(300, wh - 15);
glEnd();

glColor3f(0.0, 0.0, 0.0); //메뉴바의 메뉴들
StringText("File", 5, wh - 15);

```

```

    StringText("Fill", 50, wh - 15);
    StringText("BackGround", 95, wh - 15);
    StringText("Inversion", 200, wh - 15);
}

void Functionbar()
{
    glColor3f(0.8, 0.8, 0.8);
    glBegin(GL_POLYGON); //기능창이 위치한 공간을 그려준다.
        glVertex2i(0, wh - 20);
        glVertex2i(0, 0);
        glVertex2i(80, 0);
        glVertex2i(80, wh - 20);
    glEnd();

    //기능 창에 있는 버튼들을 그려준다.
    Button(5, wh - 25);
    Button(45, wh - 25);
    Button(5, wh - 65);
    Button(45, wh - 65);
    Button(5, wh - 105);
    Button(45, wh - 105);
    Button(5, wh - 145);
    Button(45, wh - 145);

    if (buttoncheck != 0) //버튼이 눌러졌을 때 클릭 모션
    {
        int x;
        int y;

        //x,y좌표값을 계산해준다.
        if (buttoncheck % 2 == 0)
        {
            x = 45;
            y = 25 + 40 * (buttoncheck / 2 - 1);
        }
        else
        {
            x = 5;
            y = 25 + 40 * (buttoncheck / 2);
        }
    }
}

```

```

        glColor3f(0.5, 0.5, 0.5);
        glBegin(GL_POLYGON);
            glVertex2i(x, wh - y);
            glVertex2i(x, wh - (y + 30));
            glVertex2i(x + 30, wh - (y + 30));
            glVertex2i(x + 30, wh - y);
        glEnd();

        glColor3f(0.7, 0.7, 0.7);
        glBegin(GL_POLYGON);
            glVertex2i(x, wh - y);
            glVertex2i(x, wh - (y + 30));
            glVertex2i(x + 2, wh - (y + 30));
            glVertex2i(x + 2, wh - y);
        glEnd();

        glBegin(GL_POLYGON);
            glVertex2i(x + 2, wh - y);
            glVertex2i(x + 30, wh - y);
            glVertex2i(x + 30, wh - (y + 2));
            glVertex2i(x + 2, wh - (y + 2));
        glEnd();
        glColor3f(1.0, 1.0, 1.0);
        glBegin(GL_LINE_STRIP);
            glVertex2i(x, wh - (y + 30));
            glVertex2i(x + 30, wh - (y + 30));
            glVertex2i(x + 30, wh - y);
        glEnd();
    }

    //버튼의 Thumbnail을 그려준다.
    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_LINES);
        glVertex2i(11, wh - 49);
        glVertex2i(29, wh - 31);
    glEnd();

    glBegin(GL_TRIANGLES);
        glVertex2i(60, wh - 31);
        glVertex2i(51, wh - 49);
        glVertex2i(69, wh - 49);
    glEnd();

```

```

glPointSize(5.0);
glBegin(GL_POINTS);
glVertex2i(20, wh - 80);
glEnd();
glPointSize(1.0);

glBegin(GL_LINE_LOOP);
for (int angle = 0; angle < 360; angle += 1)
{
    int px = 10 * cos(angle*0.017);
    int py = 10 * sin(angle*0.017);

    glVertex2f(60 + px, wh - 80 + py);
}
glEnd();

int x1 = 10;
int x2 = 30;
int y1 = 110;
int y2 = 130;

glBegin(GL_LINE_LOOP);
    glVertex2i(0.5*20 + x1, wh - y1);
    glVertex2i(0.38*20 + x1, wh - 0.38*20 - y1);
    glVertex2i(x1, wh - 0.4*20 - y1);
    glVertex2i(0.3*20 + x1, wh - 0.62*20 - y1);
    glVertex2i(0.19*20 + x1, wh - y2);
    glVertex2i(0.5*20 + x1, wh - 0.77*20 - y1);
    glVertex2i(0.81*20 + x1, wh - y2);
    glVertex2i(0.7*20 + x1, wh - 0.62*20 - y1);
    glVertex2i(x2, wh - 0.4*20 - y1);
    glVertex2i(0.62*20 + x1, wh - 0.38*20 - y1);
glEnd();

glRasterPos2i(55, wh - 125);
glutBitmapCharacter(GLUT_BITMAP_9_BY_15, 'D');

glBegin(GL_LINE_LOOP);
    glVertex2i(10, wh - 150);
    glVertex2i(10, wh - 170);
    glVertex2i(30, wh - 170);
    glVertex2i(30, wh - 150);
glEnd();

```

```

eraser_thumbnail(); //지우개의 썸네일이다.

glBegin(GL_POLYGON); //색상창을 그려준다.
    glColor3f(1.0, 0.0, 0.0);
    glVertex2i(5, wh - 185);
    glColor3f(0.0, 1.0, 0.0);
    glVertex2i(5, wh - 255);
    glColor3f(0.0, 0.0, 1.0);
    glVertex2i(75, wh - 255);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2i(75, wh - 185);
glEnd();

glColor3f(0.0, 0.0, 0.0);
glBegin(GL_LINE_LOOP); //색상창의 테두리
    glVertex2i(5, wh - 185);
    glVertex2i(5, wh - 255);
    glVertex2i(75, wh - 255);
    glVertex2i(75, wh - 185);
glEnd();

glColor3f(r, g, b);
glBegin(GL_POLYGON); //선택한 색상을 띄워주는 창
    glVertex2i(20, wh - 265);
    glVertex2i(20, wh - 305);
    glVertex2i(60, wh - 305);
    glVertex2i(60, wh - 265);
glEnd();

glColor3f(0.0, 0.0, 0.0);
glBegin(GL_LINE_LOOP); //선택 색상창의 테두리
    glVertex2i(20, wh - 265);
    glVertex2i(20, wh - 305);
    glVertex2i(60, wh - 305);
    glVertex2i(60, wh - 265);
glEnd();

glColor3f(0.6, 0.6, 0.6);
glBegin(GL_LINES); //기능 창을 구별하는 선
    glVertex2i(80, wh - 20);
    glVertex2i(80, 0);
glEnd();

```

```
}
```

차. 축 반전

축 반전은 좌표를 완전히 뒤집으면 되기 때문에 최대 좌표에서 원래 있던 곳의 좌표를 빼준다. 또한 최대 값은 변하면 안되기 때문에 좌표를 비율로서 존재하게 만든다.

```
void makeVertex(int x, int y, struct object* p)
{
    struct vertexNode* v =
        (struct vertexNode*)malloc(sizeof(struct vertexNode));
    //malloc을 통해 새로 VertexNode를 만들어준다.
    initNode(v); // 초기화
    //Fuctionbar의 크기 만큼을 빼고
    //최대 크기인 drawsize로 나눈다.
    v->vx = 500 * (x - 80) / drawsize;
    //menubar의 크기 만큼을 빼고
    최대크기인 drawsize로 나눈다.
    v->vy = 500 * (y - 20) / drawsize;
    //x,y좌표는 int 값으로 되어있기 때문에
    //충분히 큰 값인 500을 곱해서 정수 값으로 만들어준다.
    addVertex(v, &p->vertex); // 처리 완료후 저장
}

void xInversion(struct object** head)
{
    struct object* p = *head;;
    if (p->next != NULL)
    {
        struct vertexNode* node; // 값을 사용을 위한 임시변수
        while (p->next != tail) // 모든 Object에 대해 실행
        {
            p = p->next;
            //Object 안의 모든 Vertex값을 반전시킨다.
            for (node = p->vertex; node != NULL;
                node = node->next)
            {
                //최대 값이 500이므로 500에서 뺀다.
            }
        }
    }
}
```

```

node->vx = 500 - node->vx;
    }
}
}

void yInversion(struct object** head)
{
    //y축에 대해 계산하는 것 빼고 위와 동일하다.
    struct object* p = *head;;
    if (p->next != NULL)
    {
        struct vertexNode* node;
        while (p->next != tail)
        {
            p = p->next;
            for (node = p->vertex; node != NULL;
                node = node->next)
            {
                node->vy = 500 - node->vy;
            }
        }
    }
}

```

카. 색상 선택

색상 선택은 마우스 좌표에서 값을 읽어오는 것이 아닌 RGB값이 존재한다고 가정하고 색상 보간을 통해 색을 구하였다.

```

void getRGB(int x, int y)
{
    //비율로 구하기 위해 최대값인 70으로 나눠준다.
    float RGBx = (float)x / 70;
    float RGBy = (float)y / 70;

    //양방향 선형보간을 사용하여 계산하면
    //r = (1 - RGBx)*(1 - RGBy);
    //g = (1 - RGBx)*RGBy;
    //b = RGBx*RGBy;
}

```



```

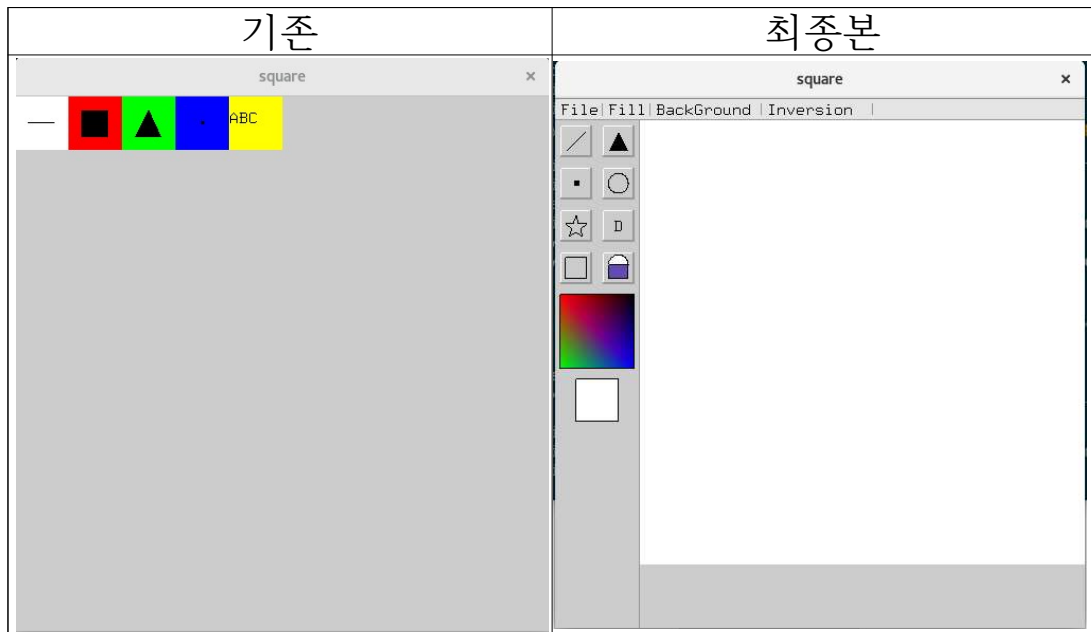
//구간별로 들어가는 색상이 다르다.
if(RGBx <= RGBy)
{
    if(RGBx < 1 - RGBy)
    {
        r = (1 - RGBx)*(1 - RGBy);
        g = (1 - RGBx)*RGBy;
        b = 0;
    }
    else
    {
        r = 0;
        g = (1 - RGBx)*RGBy;
        b = RGBx*RGBy;
    }
}
else
{
    r = (1 - RGBx)*(1 - RGBy);
    g = 0;
    b = RGBx*RGBy;
}
printf("R : %f G : %f B : %f\n", r, g, b);
glutPostRedisplay();
}

```

3. 수행 결과

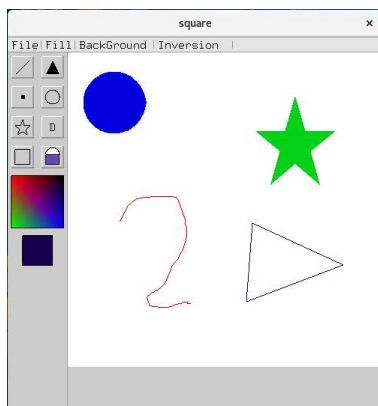
가. UI의 변화

기존의 조잡한 UI에서 나름 정리된 UI로 수정되었다.그리기 기능은 좌측의 functionbar로 나머지 기능들은 메뉴바로 옮겨졌다.



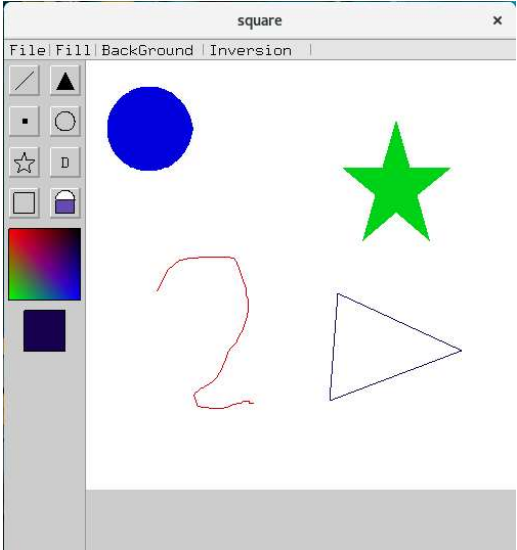
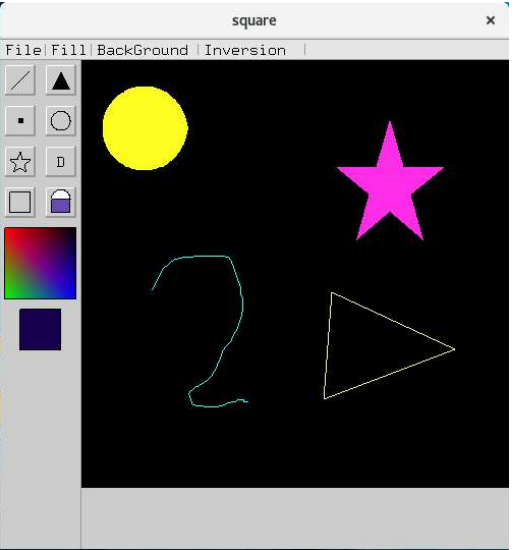
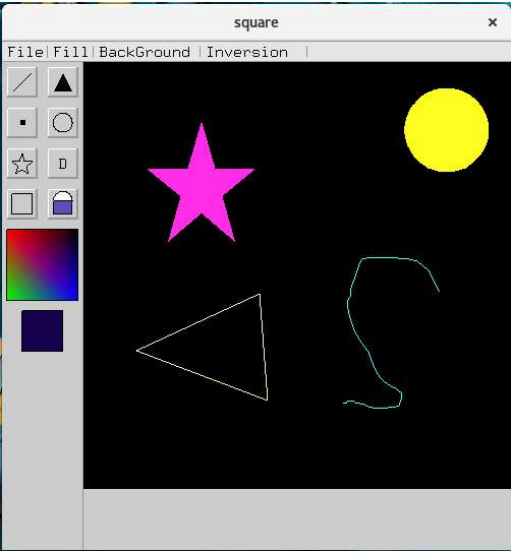
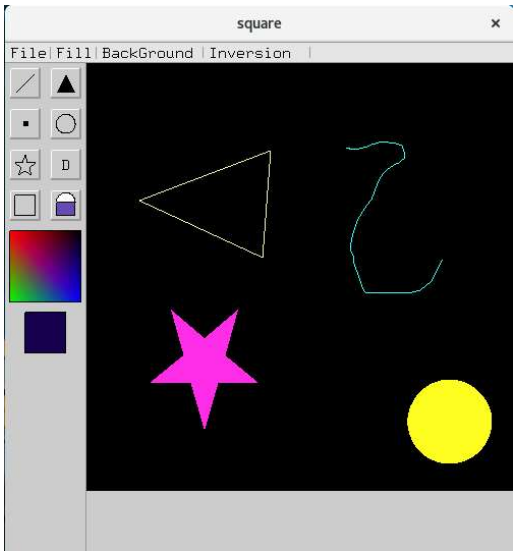
나. 그리기 기능 강화

원래 있던 그리기 기능에서 별그리기, 자유곡선, 원그리기, 지우개, 색상 선택등이 추가되었다.



다. 변형 효과

그림을 그리면 변형을 가할 수 없었던 원본과 달리 변형 효과들이 추가되었다.

원본	색반전
	
X축 반전	Y축 반전
	

4. 기여도 및 느낀점

- 김진우 : 55%(조장)

그동안 객체지향 방식이 어떤 점에서 편리한지 제대로 이해되지 않았지만 이번에 구조를 바꾸면서 C++이나 C#, JAVA였다면 클래스를 이용하여 편리하게 만들었을 것이라고 생각하며 객체지향 방식의 우수함을 깨달았습니다.

- 김태경 : 9%

이번 과제를 하면서 내가 여러가지 부족한 부분들이 많다는 것을 느꼈고, 이제부터라도 무엇을 보충해야 하는 지를 느꼈습니다. 역시 무엇인가를 해보지 않고 막연히 안다고 넘어가는 것보다는 한번 혹은 여러번 실행해보며, 부족한 부분을 채워야 한다는 것을 다시한번 느꼈습니다.

- 김태환 : 36%

역시 해봐야 모르고 지나쳤던 부분도 잡아낼 수 있다는 것을 느꼈다.