IF3260 – Grafika Komputer

Laporan Tugas Ujian Tengah Semester

Oleh:

Achmad Fahrurrozi Maskur	13515026
M. Umar Fariz Tumbuan	13515050
Akmal Fadlurohman	13515074
Diki Ardian Wirasandi	13515092
David Theosaksomo	13515131
Fadhil Imam Kurnia	13515146

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung

2018

I. Deskripsi Aplikasi

Program UTS-REMIX merupakan gabungan/kompilasi dari beberapa subprogram. Subprogram-subprogram yang dikompilasi yaitu Vector Letter, Plane with Laser, Plane in Viewport (auto-move), Plane in Viewport (user-controllable) dan ITB Map. Program ini dapat menerima input berupa mouse yang dapat menerima event *pointer movement*, klik kanan dan klik kiri. Saat dijalankan, program akan menampilkan menu utama beserta pointer yang dapat digerakkan user. Untuk memilih menu, user dapat mengarahkan pointer ke menu yang akan dipilih dan klik kiri untuk memilih menu yang saat ini ditunjuk pointer.

Gambar 1.1 Tampilan Menu Utama

II. Daftar Program

A. Vector Letter

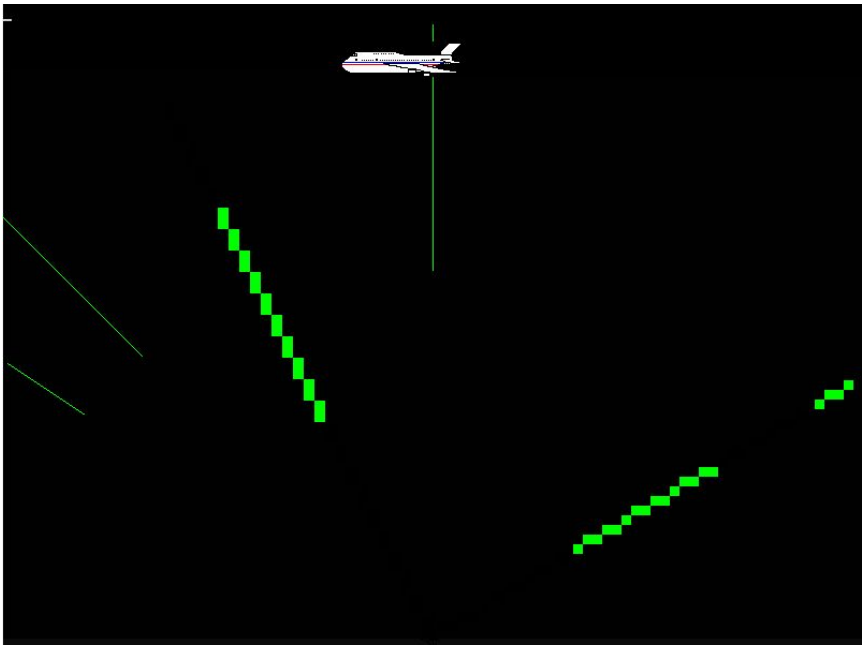
Program Vector Letter terdapat pada menu “font”. Program ini akan mencetak ke layar huruf-huruf yang di *input* oleh user. Huruf-huruf ini dibentuk menggunakan vector dan diwarnai dengan metode *raster*. Metode *raster* bekerja mewarnai sebuah objek yang disusun oleh beberapa vektor dengan mengisikan warna pada bidang yang dibatasi oleh dua vektor dalam setiap iterasi untuk setiap baris yang menyusun objek tersebut. Dalam setiap iterasi, program akan memeriksa *critical point* dari objek tersebut agar seluruh

bidang dapat terwarnai. Program ini digunakan juga untuk menampilkan tulisan pada menu.

Gambar 1.2 Vector Letter

B. Plane With Laser

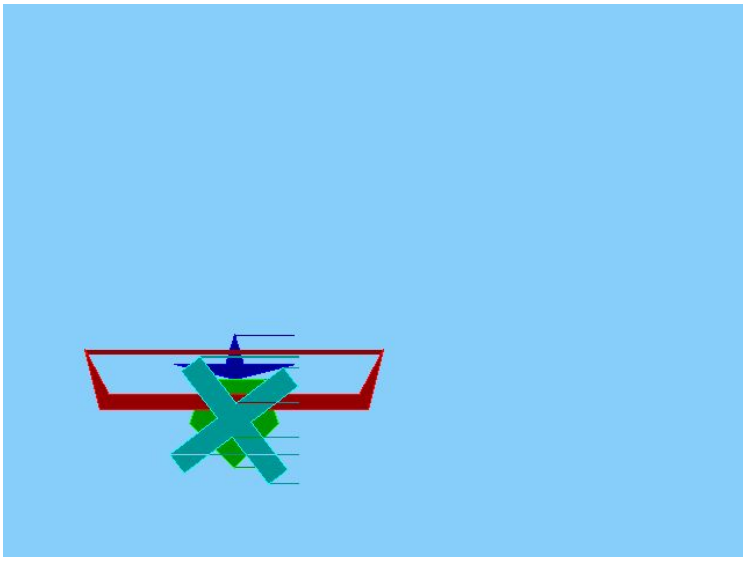
Program Plane With Laser terdapat pada menu “Animasi Pesawat 1”. Program ini akan menampilkan animasi pesawat yang bergerak secara horizontal ke kiri. Program juga akan menampilkan animasi laser berbentuk garis dengan beragam ketebalan yang bergerak dari satu titik pusat ke segala arah sehingga seolah-olah laser tersebut menyerang pesawat yang sedang melintas.



Gambar 1.3 Plane with Laser

C. Plane in Viewport (auto-move)

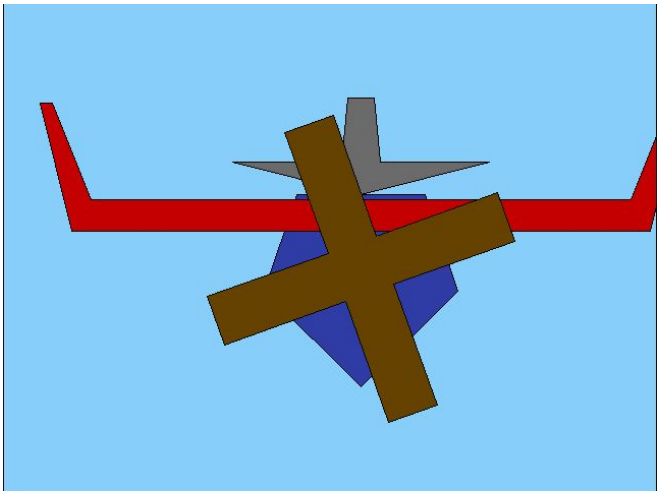
Program Plane in Viewport (auto-move) terdapat pada menu "Animasi Pesawat 2". Program ini akan menampilkan pesawat yang dapat bergerak ke kanan, kiri, atas, bawah, mendekat, dan menjauhi layar dalam viewport. Pesawat yang ditampilkan juga memiliki baling-baling yang dapat berotasi bersamaan dengan pergerakan pesawat.



Gambar 1.4 Plane in Viewport (auto-move)

D. Plane in Viewport (user-controllable)

Program Plane in Viewport (user-controllable) terdapat pada menu "Animasi Pesawat 3". Program ini akan menampilkan pesawat dalam *viewport*. Pesawat dapat digerakkan berdasarkan masukan dari pengguna. Saat pesawat bergerak, baling-baling pada pesawat juga akan ikut berputar.



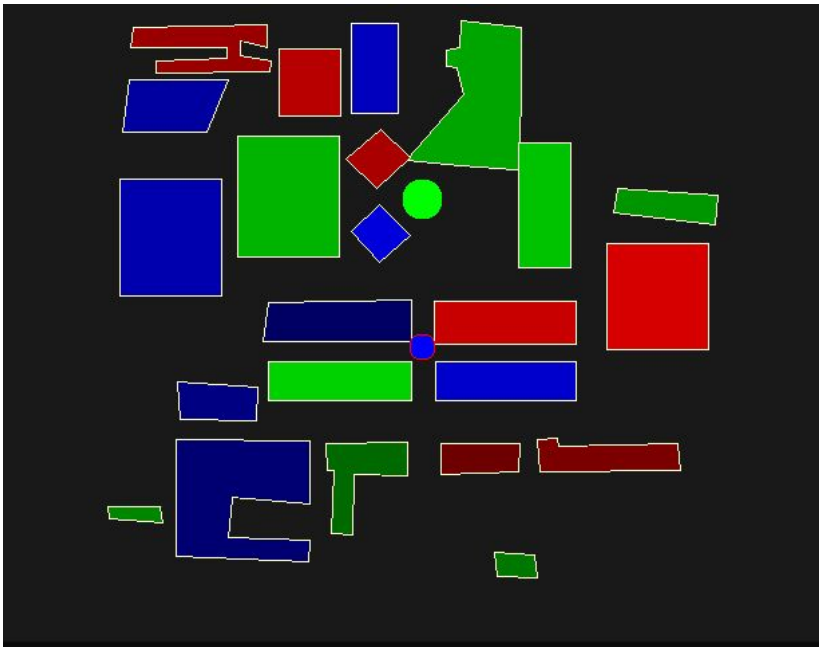
Gambar 1.5 Plane in Viewport (user-controllable)

E. ITB Map

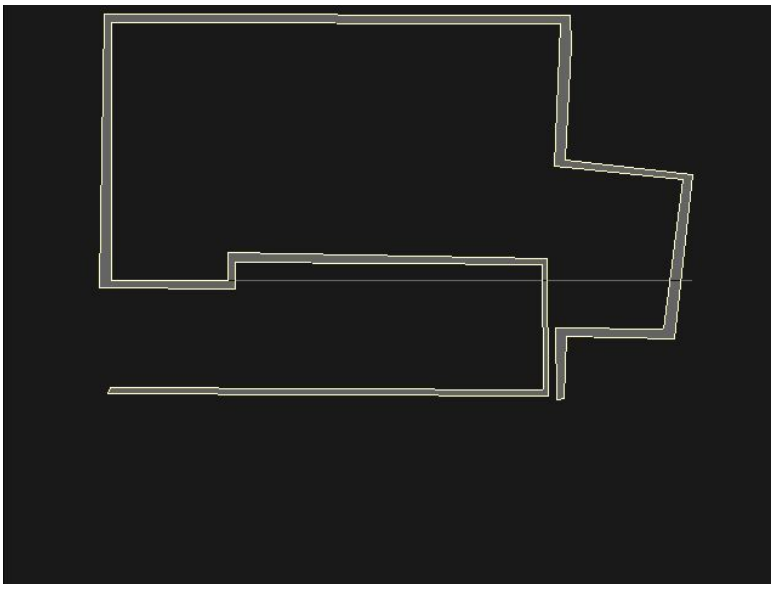
Program ITB Map terdapat pada menu “Peta ITB”. Program ini akan menampilkan sebagian wilayah ITB dari kawasan 4 labtek hingga gedung oktagon. Pada peta, jalan digambarkan dengan sebuah garis, gedung digambarkan dengan sebuah *polygon*, dan terdapat bentuk lingkaran yang merepresentasikan kolam Indonesia Tenggelam (Intel) dan taman didepan gedung oktagon.

Program memiliki 3 mode dalam menampilkan peta ITB. Mode pertama, Peta menampilkan bangunan dan jalan. Mode kedua, Peta menampilkan bangunan saja. Mode ketiga, Peta menampilkan jalan saja. Ketiga mode ini dapat di *toggle* oleh pengguna.

Gambar 1.5 ITB Map (Buildings + Roads)



Gambar 1.6 ITB Map (Buildings only)



Gambar 1.7 ITB Map (Roads only)

III. Snippet Kode

```
#include "headers/fbp.h"

#define MOUSE_SPEED 12

const char *template_bitmap_font =
"data/template_bitmap_font.io";

void drawMainMenu(BitmapFont* bf, int x, int y);
void closeProgram(BitmapFont* bf, int mouse_y);
void showVecLetters(BitmapFont* bf, int mouse_y);
void show_plane1(BitmapFont* bf, int mouse_y);
void show_plane2(BitmapFont* bf, int mouse_y);
void show_plane3(BitmapFont* bf, int mouse_y);
void openMap(BitmapFont* bf, int mouse_y);

int window_x;
int window_y;
int max_window_x;
int max_window_y;

Mouse *mouse;

int main(int argc, char **argv) {
```

```
     initializeFBP();
    viewport_x = 200;
    viewport_y = 500;

    window_x = (vinfo.xres - viewport_width) / 2;
    window_y = (vinfo.yres - viewport_height) / 2;
    max_window_x = window_x + viewport_width;
    max_window_y = window_y + viewport_height;

    mouse = initMouse(window_x, window_y, max_window_x,
max_window_y, MOUSE_SPEED);
    if (mouse == 0) {
        return 0;
    }

    BitmapFont* bitmapFont =
initBitmapFont(template_bitmap_font);

    critColor = rgbaToInt(250,250,250,0);
    frameColor = rgbaToInt(247,247,247,0);


    while(RUNNING) {
        scanMouse(mouse);
        if(mouse->isEvent) {
            clearScreen();
            clearViewPort(rgbaToInt(25,25,25,25));
            drawMainMenu(bitmapFont, window_x, window_y);
            drawPointer(mouse);
            if (mouse->isRightClick) {
                mouse->positionY = window_y;
            }
            if (mouse->isLeftClick) {
                showVecLetters(bitmapFont, mouse->positionY);
                show_plane1(bitmapFont, mouse->positionY);
                show_plane2(bitmapFont, mouse->positionY);
                show_plane3(bitmapFont, mouse->positionY);
                openMap(bitmapFont, mouse->positionY);
                closeProgram(bitmapFont, mouse->positionY);
            }
        }
    }
    printf("bye!\n");

    munmap(fbp, screensize);
    close(fbfd);
}
```

```
void closeProgram(BitmapFont* bf, int mouse_y) {
    int line_height_5 = bf->char_height*5;
    int line_height_3 = bf->char_height*3;

    int upperBound = window_y + 2*line_height_5 +
6*line_height_3;
    int lowerBound = window_y + 2*line_height_5 +
7*line_height_3;

    if (mouse_y < lowerBound && mouse_y > upperBound) {
        clearViewPort(rgbaToInt(0,0,0,0));
        clearScreen();
        render();
        exit(0);
    }
}

void drawMainMenu(BitmapFont* bf, int x, int y) {
    int line_height_5 = bf->char_height*5;
    int line_height_3 = bf->char_height*3;
    drawBitmapString(bf, x, y, "MENU UTAMA", 5);
    drawBitmapString(bf, x, y + 2*line_height_5, " > FONT",
3);
    drawBitmapString(bf, x, y + 2*line_height_5 +
line_height_3, " > ANIMASI PESAWAT 1", 3);
    drawBitmapString(bf, x, y + 2*line_height_5 +
2*line_height_3, " > ANIMASI PESAWAT 2", 3);
    drawBitmapString(bf, x, y + 2*line_height_5 +
3*line_height_3, " > ANIMASI PESAWAT 3", 3);
    drawBitmapString(bf, x, y + 2*line_height_5 +
4*line_height_3, " > PETA ITB", 3);
    drawBitmapString(bf, x, y + 2*line_height_5 +
6*line_height_3, "KELUAR", 2);
}

void showVecLetters(BitmapFont* bf, int mouse_y) {
    viewport_x = 0;
    viewport_y = 0;
    int line_height_5 = bf->char_height*5;
    int line_height_3 = bf->char_height*3;

    int upperBound = window_y + 2*line_height_5;
    int lowerBound = window_y + 2*line_height_5 +
line_height_3;

    if (mouse_y < lowerBound && mouse_y > upperBound) {
```

```
        loadLetters("assets/VecLetterSpec.txt");
        clearViewPort(rgbaToInt(0,0,0,0));
        system("/bin/stty raw");
        int offsetX = MARGIN_HORIZONTAL;
        int offsetY = MARGIN_VERTICAL;
        char input = 0;
        clearScreen();
        render();
        while (RUNNING && (input=getchar()) != 27) {
            clearScreen();
            render();
            // char input[100];
            // printf("%s: ", "Masukkan input");
            // scanf("%99[0-9a-zA-Z .]", input);
            // if (input[strlen(input)-1] == '.') {
            //     return;
            // }
            // for(int i = 0; input[i]; i++) {
            //     input[i] = toupper(input[i]);
            // }

            // for (int i=0; i<vinfo.yres/17; i++) {
              // printf("\n");
            // }
            // printf("\n");

            input = toupper(input);

            // system("clear");

            COLOR = rgbaToInt(255, 225, 0, 0);
            BORDER_COLOR = rgbaToInt(255, 0, 0, 0);

            if (input == ' ') {
                offsetX += MARGIN_HORIZONTAL*4;
                render();
                continue;
            }
            if (input == 13) {
                offsetX = MARGIN_HORIZONTAL;
                offsetY += (vinfo.yres/90)*MARGIN_VERTICAL;
                render();
                continue;
            }
            drawLetters(input, &offsetX, &offsetY);
            render();
        }
        system("/bin/stty cooked");
```

```
    }

    return;
}

struct termios orig_termios;

void reset_terminal_mode()
{
    tcsetattr(0, TCSANOW, &orig_termios);
}

void set_conio_terminal_mode()
{
    struct termios new_termios;

    /* take two copies - one for now, one for later */
    tcgetattr(0, &orig_termios);
    memcpy(&new_termios, &orig_termios, sizeof(new_termios));

    /* register cleanup handler, and set the new terminal
mode */
    atexit(reset_terminal_mode);
    cfmakeraw(&new_termios);
    tcsetattr(0, TCSANOW, &new_termios);
}

int kbhit()
{
    struct timeval tv = { 0L, 0L };
    fd_set fds;
    FD_ZERO(&fds);
    FD_SET(0, &fds);
    return select(1, &fds, NULL, NULL, &tv);
}

int getch()
{
    int r;
    unsigned char c;
    if ((r = read(0, &c, sizeof(c))) < 0) {
        return r;
    } else {
        return c;
    }
}

// int main(int argc, char *argv[])
```

```
// {

//     while (!kbhit()) {
//          do some work
//     }
//     (void)getch(); /* consume the character */
// }

void show_plane1(BitmapFont* bf, int mouse_y) {
    int line_height_5 = bf->char_height*5;
    int line_height_3 = bf->char_height*3;

    int upperBound = window_y + 2*line_height_5 +
line_height_3;
    int lowerBound = window_y + 2*line_height_5 +
2*line_height_3;

    if (mouse_y < lowerBound && mouse_y > upperBound) {
     viewport_x = 200;
        viewport_y = 500;
     char* fileName = "assets/plane_bitmap.txt";

     struct f_Image* plane = f_loadImage(fileName);
        plane->posX = vinfo.xres;
        int t1 = 0, t2 = 0, t3 = 0, t4 = 0, t5 = 0, t6 = 0;
        centerX = vinfo.xres/2/SCALE, fullY =
vinfo.yres/SCALE - 1;
        int delay = 0;

     system("clear");

        set_conio_terminal_mode();
        int ditekan = 0;
        while (RUNNING && !ditekan) {
            plane->posX--;
            if (plane->posX < -plane->width) {
                plane->posX = vinfo.xres;
                plane->posY += 7;
            }
            drawObject(plane, 1);

            drawLaser(centerX, fullY, 10, -10, SCALE, &t1);
            drawLaser(vinfo.xres/18, vinfo.yres/9-1, 5, -3,
9, &t2);
            drawLaser(centerX, fullY, 0, -10, SCALE, &t3);
            drawLaser(centerX, fullY, -10, -10, SCALE, &t4);
            drawLaser(vinfo.xres/20, vinfo.yres/10-1, -5,
-10, 10, &t5);
```

```
            drawLaser(centerX, fullY, -3, -2, SCALE, &t6);
            if (delay %10 == 0) {
                t1++; t2++; t3++; t4++; t5++; t6++;
            }
            delay++;

            usleep(3000);
            if (kbhit()) {
                int a;
                a = getch();
                if (a == 27) {
                    ditekan = 1;
                }
            }
        }
        reset_terminal_mode();
        f_freeImage(plane);
    }
}

void show_plane2(BitmapFont* bf, int mouse_y) {

    int line_height_5 = bf->char_height*5;
    int line_height_3 = bf->char_height*3;

    int upperBound = window_y + 2*line_height_5 +
2*line_height_3;
    int lowerBound = window_y + 2*line_height_5 +
3*line_height_3;

    if (mouse_y < lowerBound && mouse_y > upperBound) {

     viewport_x = 0;
        viewport_y = 0;
     loadLetters("assets/plane2.txt");

        for (int i=0; i<vinfo.yres/17; i++) {
          printf("\n");
        }
        printf("\n");
        int f;
        int marginX = 150*SCALE;
        int posX = marginX;
        int posY = vinfo.yres/2;
        system("clear");
        float degree = 10;
        float wingDeg = 0;
        float zoom = 1;
```

```
        set_conio_terminal_mode();
        int ditekan = 0;
        while (RUNNING && !ditekan) {
            clearScreen();
            clearViewPort(rgbaToInt(135,206,250,0));
            wingDeg+=10;
            if(posX < vinfo.xres/(1.5*SCALE) && zoom == 1){
                if(degree < 10){
                    degree+=1;

                } else if(degree == 10){
                    posX+=4;

                }else{
                    degree = 10;
                }
            }
            if(posX >= vinfo.xres/(1.5*SCALE) && zoom < 3){
                if(degree > 0){
                    degree-=1;
                    posX+=5;

                } else if(degree == 0){
                    zoom+= 0.025;

                }else{
                    degree = 0;
                }
            }
            if(zoom >= 3 && posX >marginX){
                if(degree > -10){
                    degree-=1;
                    posX-=6;

                } else if(degree == -10){
                    posX-=8;

                }else{
                    degree = -10;
                }


            }
            if(posX < marginX){
                posX = marginX;
            }
            if(posX == marginX && zoom !=1){
                if(degree < 0){
```

```
                    degree+=1;

                } else if(degree == 0){
                    zoom-= 0.025;

                }else{
                    degree = 0;
                }
            }
            if(zoom < 1){
                zoom =1;
            }
            drawVector('C', posX, posY, rgbaToInt(0,0,255,0),
rgbaToInt(0,0,150,0), degree, 50, 50, zoom);
            drawVector('B', posX, posY, rgbaToInt(0,255,0,0),
rgbaToInt(0,150,0,0), degree, 50, 50, zoom);
            drawVector('A', posX, posY, rgbaToInt(255,0,0,0),
rgbaToInt(150,0,0,0), degree, 50, 50, zoom);
            drawVector('D', posX, posY,
rgbaToInt(0,255,255,0), rgbaToInt(0,150,150,0),wingDeg, 50,
60, zoom);
            render();
            usleep(30000);
            if (kbhit()) {
                int a;
                a = getch();
                if (a == 27) {
                    ditekan = 1;
                }
            }
        }
        reset_terminal_mode();
    }
}

void show_plane3(BitmapFont* bf, int mouse_y) {

    int line_height_5 = bf->char_height*5;
    int line_height_3 = bf->char_height*3;

    int upperBound = window_y + 2*line_height_5 +
3*line_height_3;
    int lowerBound = window_y + 2*line_height_5 +
4*line_height_3;

    if (mouse_y < lowerBound && mouse_y > upperBound) {
        viewport_x = 200;
        viewport_y = 500;
```

```
        // Initialize vector objects
        VectorPath* badan_bawah =
createVectorPathFromFile("assets/plane3/badan_bawah.txt");
        if (badan_bawah == NULL) {
            printf("Failed to load badan bawah\n");
            return;
        }
        VectorPath* sayap_utama =
createVectorPathFromFile("assets/plane3/sayap.txt");
        if (sayap_utama == NULL) {
            printf("Failed to load sayap utama\n");
            return;
        }
        VectorPath* sayap_belakang =
createVectorPathFromFile("assets/plane3/sayap_belakang.txt");
        if (sayap_belakang == NULL) {
            printf("Failed to load sayap belakang\n");
            return;
        }
        VectorPath* baling_baling =
createVectorPathFromFile("assets/plane3/baling2.txt");
        if (baling_baling == NULL) {
            printf("Failed to load baling-baling\n");
            return;
        }

        int count = 0;
        int dx = 10;

        void translatePlane() {
            translatePath(badan_bawah, 200, 500);
            translatePath(sayap_belakang, 200, 500);
            translatePath(sayap_utama, 200, 500);
            translatePath(baling_baling, 200, 500);
        }


        dilatatePath(badan_bawah, 50, 50, 2);
        dilatatePath(sayap_belakang, 50, 50, 2);
        dilatatePath(sayap_utama, 50, 55, 2);
        dilatatePath(baling_baling, 50, 60, 2);

        clearScreen();

        char c = 0;
        // Start animation and render
        system("/bin/stty raw");
```

```
        clearViewPort(rgbaToInt(135,206,250,0));
        rotatePath(baling_baling, 10,  50, 60);

        // drawVectorPathClipping(sayap_belakang,
rgbaToInt(0,0,0,0),rgbaToInt(107,107,107,0), 500, 500);
        // drawVectorPathClipping(badan_bawah,
rgbaToInt(2,2,2,0),rgbaToInt(48,60,165,0), 500, 500);
        // drawVectorPathClipping(sayap_utama,
rgbaToInt(1,1,1,0),rgbaToInt(196,0,0,0), 500, 500);
        // drawVectorPathClipping(baling_baling,
rgbaToInt(3,3,3,0),rgbaToInt(102,66,0,0), 500, 500);

        drawVectorPath(sayap_belakang,
rgbaToInt(0,0,0,0),rgbaToInt(107,107,107,0), 500, 500);
        drawVectorPath(badan_bawah,
rgbaToInt(2,2,2,0),rgbaToInt(48,60,165,0), 500, 500);
        drawVectorPath(sayap_utama,
rgbaToInt(1,1,1,0),rgbaToInt(196,0,0,0), 500, 500);
        drawVectorPath(baling_baling,
rgbaToInt(3,3,3,0),rgbaToInt(102,66,0,0), 500, 500);

        render();

        while (RUNNING && (c=getchar()) != 27) {
            clearViewPort(rgbaToInt(135,206,250,0));
            rotatePath(baling_baling, 10,  50, 60);

            // drawVectorPathClipping(sayap_belakang,
rgbaToInt(0,0,0,0),rgbaToInt(107,107,107,0), 500, 500);
            // drawVectorPathClipping(badan_bawah,
rgbaToInt(2,2,2,0),rgbaToInt(48,60,165,0), 500, 500);
            // drawVectorPathClipping(sayap_utama,
rgbaToInt(1,1,1,0),rgbaToInt(196,0,0,0), 500, 500);
            // drawVectorPathClipping(baling_baling,
rgbaToInt(3,3,3,0),rgbaToInt(102,66,0,0), 500, 500);

            drawVectorPath(sayap_belakang,
rgbaToInt(0,0,0,0),rgbaToInt(107,107,107,0), 500, 500);
            drawVectorPath(badan_bawah,
rgbaToInt(2,2,2,0),rgbaToInt(48,60,165,0), 500, 500);
            drawVectorPath(sayap_utama,
rgbaToInt(1,1,1,0),rgbaToInt(196,0,0,0), 500, 500);
            drawVectorPath(baling_baling,
rgbaToInt(3,3,3,0),rgbaToInt(102,66,0,0), 500, 500);

            render();

            if(c == 'w' || c == 'W'){
```

```
                viewport_y -= VIEWPORT_SPEED;
            } else if(c == 'a' || c == 'A'){
                viewport_x -= VIEWPORT_SPEED;
            } else if(c == 's' || c == 'S'){
                viewport_y += VIEWPORT_SPEED;
            } else if(c == 'd' || c == 'D'){
                viewport_x += VIEWPORT_SPEED;
            } else if(c == 'z' || c == 'Z'){
                dilatatePath(badan_bawah, 50, 50, 1.1);
                dilatatePath(sayap_belakang, 50, 50, 1.1);
                dilatatePath(sayap_utama, 50, 55, 1.1);
                dilatatePath(baling_baling, 50, 60, 1.1);
            } else if(c == 'x' || c == 'X'){
                dilatatePath(badan_bawah, 50, 50, 0.9);
                dilatatePath(sayap_belakang, 50, 50, 0.9);
                dilatatePath(sayap_utama, 50, 55, 0.9);
                dilatatePath(baling_baling, 50, 60, 0.9);
            }

            if(viewport_x < 0)
                viewport_x = 0;
            if(viewport_y < 0)
                viewport_y = 0;
            if(viewport_x > WORLD_WIDTH - viewport_width)
                viewport_x = WORLD_WIDTH - viewport_width;
            if(viewport_y > WORLD_HEIGHT - viewport_height)
                viewport_y = WORLD_HEIGHT - viewport_height;
            usleep(3000);
        }
        freeVectorPath(sayap_belakang);
        freeVectorPath(badan_bawah);
        freeVectorPath(sayap_utama);
        freeVectorPath(baling_baling);
        system("/bin/stty cooked");
    }
}

void openMap(BitmapFont* bf, int mouse_y) {
    int line_height_5 = bf->char_height*5;
    int line_height_3 = bf->char_height*3;

    int upperBound = window_y + 2*line_height_5 +
4*line_height_3;
    int lowerBound = window_y + 2*line_height_5 +
5*line_height_3;

    if (mouse_y < lowerBound && mouse_y > upperBound) {
     viewport_x = 200;
```

```
        viewport_y = 500;
     int renderRoad = 1;
        int renderBuilding = 1;

        int numOfGedung = 25;
        VectorPath** gedung =
createVectorPathFromSVG("assets/map_buildings.txt",
numOfGedung);
        VectorPath** jalan =
createVectorPathFromSVG("assets/map_roads.txt", 1);

        clearScreen();
        clearViewPort(rgbaToInt(25,25,25,25));
        if (renderBuilding == 1) {
            for (int i = 0; i < numOfGedung; i++) {
                if (i % 3 == 0) {
                    drawVectorPath(gedung[i],
rgbaToInt(255,255,200 + i,0), rgbaToInt(0,0,100 + i * 5,0),
0, 0);
                } else if (i % 3 == 1) {
                    drawVectorPath(gedung[i],
rgbaToInt(255,255,200 + i,0), rgbaToInt(0,100 + i * 5,0,0),
0, 0);
                } else {
                    drawVectorPath(gedung[i],
rgbaToInt(255,255,200 + i,0), rgbaToInt(100 + i * 5,0,0,0),
0, 0);
                }
            }

            drawCircle(340, 580, 10, rgbaToInt(255,0,0,0),
rgbaToInt(0,0,255,0));
            drawCircle(340, 470, 15, rgbaToInt(9,255,0,0),
rgbaToInt(0,255,0,0));
        }

        if (renderRoad == 1) {
            drawVectorPath(jalan[0],
rgbaToInt(255,255,199,0), rgbaToInt(100,100,100,0), 0, 0);
        }

        render();

        char c = 0;
        // Start animation and render
        system("/bin/stty raw");
     while (RUNNING && (c=getchar()) != 27) {
            if (c != 'w' && c != 'W' && c != 'A' && c != 'a'
```

```
&& c != 'S' && c != 's' && c != 'D' && c != 'd' && c != '1'
&& c != '2')
            {
                continue;
            }
          clearScreen();
            clearViewPort(rgbaToInt(25,25,25,25));
            if (renderBuilding == 1) {
                for (int i = 0; i < numOfGedung; i++) {
                    if (i % 3 == 0) {
                        drawVectorPath(gedung[i],
rgbaToInt(255,255,200 + i,0), rgbaToInt(0,0,100 + i * 5,0),
0, 0);
                    } else if (i % 3 == 1) {
                        drawVectorPath(gedung[i],
rgbaToInt(255,255,200 + i,0), rgbaToInt(0,100 + i * 5,0,0),
0, 0);
                    } else {
                        drawVectorPath(gedung[i],
rgbaToInt(255,255,200 + i,0), rgbaToInt(100 + i * 5,0,0,0),
0, 0);
                    }
                }

                drawCircle(340, 580, 10,
rgbaToInt(255,0,0,0), rgbaToInt(0,0,255,0));
                drawCircle(340, 470, 15,
rgbaToInt(9,255,0,0), rgbaToInt(0,255,0,0));
            }

            if (renderRoad == 1) {
                drawVectorPath(jalan[0],
rgbaToInt(255,255,199,0), rgbaToInt(100,100,100,0), 0, 0);
            }

            render();

            if(c == 'w' || c == 'W'){
                viewport_y -= VIEWPORT_SPEED;
            } else if(c == 'a' || c == 'A'){
                viewport_x -= VIEWPORT_SPEED;
            } else if(c == 's' || c == 'S'){
                viewport_y += VIEWPORT_SPEED;
            } else if(c == 'd' || c == 'D'){
                viewport_x += VIEWPORT_SPEED;
            } else if (c == '1') {
                if (renderBuilding == 1) {
                    renderBuilding = 0;
```

```
                } else {
                    renderBuilding = 1;
                }
            } else if (c == '2') {
                if (renderRoad == 1)
                {
                    renderRoad = 0;
                }
                else
                {
                    renderRoad = 1;
                }
            }
     }
        for (int i = 0; i < numOfGedung; i++) {
            freeVectorPath(gedung[i]);
        }
        freeVectorPath(jalan[0]);

        free(gedung);
        free(jalan);
        system("/bin/stty cooked");
    }
}
```

IV. Pembagian Tugas

13515026:
- Membuat critical point vector letter
- Membuat controller untuk menggerakkan pesawat pada tugas 4
- Membersihkan bug di setiap program

13515050:
- Membuat ADT vector letter
- Membuat objek pesawat tugas 1
- Membuat ADT vector path
- Membuat viewport
- Membuat susunan peta ITB

13515074:
- Membuat ADT vector letter
- Membuat parser file eksternal vector letter

13515092:
- Membuat fungsi pewarnaan objek
- Membuat mouse click listener

13515131:
- Membuat vector letter
- Membuat objek pesawat tugas 2 dan animasinya
- Membuat viewport
- Membuat parser SVG

13515146:
- Membuat animasi laser pada tugas 2
- Membuat mouse click listener
- Membuat tampilan menu tugas uts