“ OPEN FIRE ”

LAPORAN TUGAS BESAR DASAR PEMROGRAMAN

Disusun Oleh:

Muhammad Nezha Alfatah Chandrawisesa // 105221021

Ichiro Albert // 105221022

Muhammad Irsyad Zahrandi // 105221030

PROGRAM STUDI S-1 ILMU KOMPUTER

UNIVERSITAS PERTAMINA

JAKARTA SELATAN

2022

DAFTAR ISI

BAB I

Cara Kerja Program

1. Library yang kami pakai. Program ini memakai bahasa C++, untuk program kita dapat bekerja sesuai dan berjalan dengan baik, terdapat beberapa library tertentu yang harus kami tambahkan.

```
//================================={ include library }=====================================//

#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_timer.h>
#include <time.h>
#include <iostream>

using namespace std;
```

- **#include**: digunakan untuk memasukkan konten file header dalam program C++ kami
- **< SDL2/SDL.h >** : Library ini kami gunakan untuk me-
- **< SDL2/SDL_image.h >** : Agar dapat menampilkan segala media yang kita sudah masukkan ke dalam program tersebut
- **< SDL2/SDL_timer.h >** : Dengan library ini kita dapat
- **< time.h >**: Khusus library ini akan kami pakai untuk-
- **< iostream >** : Library yang sangat dibutuhkan untuk menjalankan sebuah program dengan memakai bahasa C++
- **using namespace std**: komputer perlu mengetahui kode untuk fungsi cout, cin dan perlu mengetahui namespace mana yang didefinisikan.

2. Mendefinisikan beberapa *Macros* (nilai konstan atau ekspresi) memakai **#define**

```
//=============================================================================================//

//=================================={ preparation }============================================//

#define NAMA_APLIKASI "Game by Kelompok 10"
#define IMAGE_BACKGROUND "image/background.jpg"
#define SCREEN_HEIGHT 750
#define SCREEN_WIDTH 1375

#define JUMLAH_HANTU 10
#define UKURAN_HANTU 250
#define KECEPETAN_HANTU 15
#define NYAWA_HANTU 1
#define IMAGE_HANTU "image/ghost.png"
```

- **#define**: dikenal juga sebagai macros yang digunakan untuk mendeklarasikan nilai atau ekspresi konstanta dengan CNAME yang dapat digunakan di seluruh program.

- **NAMA_APLIKASI & IMAGE_BACKGROUND**: Menunjukkan nama program dari kelompok kami dan memunculkan background dari output tersebut
- **SCREEN_HEIGHT & SCREEN_WIDTH**: Deklarasi char yang didalamnya mengandung value yang nantinya akan dipakai saat mulai membuat pengaturan ukuran layar
- **JUMLAH_HANTU & UKURAN_HANTU** : Ukuran yang ditentukan untuk bentuk hantu yang akan ditampilkan di output program teresebut, dan jumlah hantu yang akan disediakan dalam program tersebut.
- **JUMLAH KECEPETAN_HANTU** : Berfungsi untuk menentukan kecepatan hantu tersebut .
- **NYAWA_HANTU & IMAGE_HANTU**: berfungsi untuk memasukkan jumlah integer ke dalam hantu jika nilai tersebut =0, maka hantu tersebut akan menghilang, dan image hantu berfungsi untuk menampilkan bentuk hantu tersebut.

3. *Index*

```
//======================================{ block hantu }=============
//setting hantu
typedef struct setting_hantu
{
  int x;
  int y;
  int arahx;
  int arahy;
  int nyawa;
}
KOMPONEN_HANTU;
```

- **struct**: index untuk menyimpan data-data yang nantinya akan diakses ke berbagai fungsi, tetapi tidak untuk memuat data baru
- **typedef**: mempersingkat penulisan index
- Typedef struct setting_hantu(){}KOMPONEN_HANTU: membuat nama alias dari index struct
- **int x, y, arahx, arahy, nyawa**: isi dari index setting_hantu
- **KOMPONEN_HANTU:** nama singkat untuk memanggil index setting_hantu

4. Fungsi untuk mengatur pergerakan hantu

```
void gerak_hantu(KOMPONEN_HANTU *hnt) {
    int gerakan_x = rand() % (2*KECEPETAN_HANTU) - KECEPETAN_HANTU;
    int gerakan_y = rand() % (2*KECEPETAN_HANTU) - KECEPETAN_HANTU;


    hnt->x += gerakan_x;
    hnt->y += gerakan_y;

    if (hnt->y < 0)
    {
        hnt->y = 0;
    }
    if (hnt->x < 0)
    {
        hnt->x = 0;
    }
    if (hnt->x >SCREEN_WIDTH-1)
    {
        hnt->x = SCREEN_WIDTH-1;
    }
    if (hnt->y >SCREEN_HEIGHT-1)
    {
        hnt->y = SCREEN_HEIGHT;
    }
}
```

- **Void fungsi()** adalah fungsi tidak bertipe data (kalau di pascal lebih dikenal dengan procedure), void itu sama saja tidak dianggap. Void fungsi() digunakan apabila sebuah fungsi tidak memerlukan argument.
- Void g**erak_hantu(KOMPONEN_HANTU *hnt)**: merupakan sebuah fungsi void yang didalamnya terdapat parameter yang isinya index untuk mengatur geraknya hantu dan sejauh mana hantu tersebut bisa bergerak.
- **Gerakan_x & y** memakai fungsi rand() dan dimodulakan (2*KECEPATAN_HANTU) - KECEPATAN_HANTU lagi agar gerakan hantu stabil, dan masih bisa di ikuti.
- **hnt->x & y += gerakan_x:** hnt mengambil data dari index KOMPONEN_HANTU menyatakan bahwa nilai y isinya value gerakan_hantu yang didalamnya mengandung arah dan kecepatan hantu secara harizontal
- **hnt->y += gerakan_y:** hnt mengambil data dari index KOMPONEN_HANTU menyatakan bahwa nilai y isinya value gerakan_hantu yang didalamnya mengandung arah dan kecepatan hantu secara vertikal

5. Fungsi untuk gambar hantu

```
void gambar_hantu(SDL_Renderer* r, SDL_Texture* tex, KOMPONEN_HANTU *hnt)
{
    if (hnt->nyawa <= 0)
    {
        return;
    }
    SDL_Rect rect_hantu;
    rect_hantu.x = 0;
    rect_hantu.y = 0;
    rect_hantu.h = UKURAN_HANTU;
    rect_hantu.w = UKURAN_HANTU;

    rect_hantu.x = hnt->x;
    rect_hantu.y = hnt->y;
    SDL_RenderCopy(r, tex, NULL, &rect_hantu);

}
```

- g**erak_hantu(SDL_Renderer* r, SDL_Texture* tex, KOMPONEN_HANTU *hnt)**: merupakan sebuah fungsi void yang didalamnya terdapat program untuk mengatur *render*, tekstur, dan index KOMPONEN_HANTU
- **if (hnt->nyawa <= 0**: adalah percabangan yang difungsikan untuk pengechekan bila nyawa hantu sudah dibawah atau sama dengan 0 maka akan kembali (return), nyawa yang diambil tunjuk dari hnt yang merupakan alias dari KOMPONEN_HANTU.
- Jika data nyawa diatas tidak 0 yang dimana percabgangan diatas tidak akan diakses, maka fungsi ini akan membaca isi dari fungsi berikutnya
- **SDL_Rect: SDL_Rect** merupakan index dari SDL Library yang isi nya defines frame untuk menggambar hantu
- **SDL_Rect rect_hantu**: index SDL_Rect yang dialiaskan sebagai recr_hantu
- **rect_hantu.x = 0, recr_hantu.y = 0**: memasukan nilai ke x dan dari index rect_hantu yang dimana didalamnya pengaturan posisi horizontal dan vertikal frame hantu, kita isi 0 karena kita tidak perlu mengetahui posisi x dan y frame hantu dalam menggambar hantu
- **rect_hantu.w = UKURAN_HANTU, rect_hantu.h = UKURAN_HANTU**: memasukan nilai ke w dan h dari index rect_hantu yang dimana didalamnya mengandung pengaturan panjang dan lebar frame hantu, kita isi dengan value UKURAN_HANTU yang dideklarasikan pada #define UKURAN_HANTU diatas.
- **SDL_RenderCopy(SDL_Renderer * r, SDL_Texture * tex, const SDL_Rect * r, const SDL_Rect * rect_hantu)**: ini adalah contoh dari default fungsi SDL_renderCopyfungsi dari SDL Library untuk menempelkan gambar pada frame gambar dan didalamnya wajib ada value pengaturan frame gambar, texture gambar, dan frame gambarnya .

6. Fungsi untuk mendefenisikan apa itu “ketembak”

```
//====================================={ kena tembakan }=================================//

int definisi_ketembak(KOMPONEN_HANTU *hantuu, int x,int y)
{
    int x0 = hantuu->x;
    int x1 = x0+UKURAN_HANTU;
    int y0 = hantuu->y;
    int y1 = y0+UKURAN_HANTU;
    int resul = 0;
    cout << "x hantu " << x0 << ", y hantu " << y0 << ", x target " << x << ", y target " << y <<endl;


    if (x0 <= x && x1 >=x && y0 <= y && y1 >= y)
    {
        resul = 1;;
    }
    return resul;
}
```

- Disini adalah sebuah fungsi int subprogram untuk *definisi_ketembak* dengan berisikan deklarasi integer x, y, dan sebuah index KOMPONEN_HANTU yang dialiaskan sebagai *hantuu*.
- Terdapat deklarasi yang terdiri dari 5 yang dimana x0 sebagai hantuu yang dialiaskan sebagai x, x1 adalah algoritma dari x0 + dengan jumlah UKURAN_HANTU, y0 sebagai hantuu yang dialiaskan sebagai y, y1 adalah algoritma dari y0 + dengan jumlah UKURAN_HANTU, dan deklarasi integer *resul* yang memiliki nilai nol.
- Terdapat percabangan yang dimana ketika x0 lebih kecil sama dengan x dan x1 lebih besar sama dengan x, begitu juga dengan y maka *resul* sama dengan satu dan mengembalikan nilai resul tersebut.

7. Fungsi untuk cek jika hantu “ketembak”

```
void cek_ketembak(KOMPONEN_HANTU* daftar_sasaran,int x, int y)
{
    for(int i = 0; i < JUMLAH_HANTU; i++)
    {
        if (daftar_sasaran[i].nyawa == 0) continue;

        if (definisi_ketembak(&daftar_sasaran[i],x,y) == 1)
        {
            daftar_sasaran[i].nyawa = NYAWA_HANTU-DAMAGE_TEMBAKAN;
        }
    }
}
```

- **cek_ketembak(KOMPONEN_HANTU* daftar_sasaran, int x, int y)** : Sebuah fungsi void yang berisikan deklarasi integer x, y, dan sebuah index KOMPONEN_HANTU yang dialiaskan sebagai daftar_sasaran.
- **for (int i=0;i<JUMLAH_HANTU;i++)** : Sebuah perulangan for yang berisikan deklarasi integer i yang dimana i tersebut dimulai dari nol.
- **if(daftar_sasaran[i].nyawa == 0) continue;** : Sebuah percabangan yang dimana daftar sasaran dalam array[i] sama dengan nol

- **if(definisi_ketembak(&daftar_sasaran[i], x, y) ==1)** : Sebuah percabangan yang dimana ketika nilai tersebut sama dengan 1
- **daftar_sasaran[i].nyawa=NYAWA_HANTU-DAMAGE_TEMBAKAN** : jika nilai dari daftar sasaran tersebut sama dengan satu maka nilai dari nyawa hantu tersebut dikurangi dengan nilai damage tembakan.

8. Fungsi utama dimana didalamnya game loop dimulai dan layar hantu, serta target digambar

```
int main()
{
    //=========================={ setting screen }=========================================//
    // returns zero on success else non-zero
    if (SDL_Init(SDL_INIT_EVERYTHING) != 0)
    {
        cout << "error initializing SDL: " << SDL_GetError() <<endl;
    }
    SDL_Window* screen = SDL_CreateWindow(NAMA_APLIKASI,SDL_WINDOWPOS_CENTERED,SDL_WINDOWPOS_CENTERED,SCREEN_WIDTH, SCREEN_HEIGHT, 0);
    // your graphics hardware and sets flags
    Uint64 render_flags = SDL_RENDERER_ACCELERATED;
    // creates a renderer to render our images
    SDL_Renderer* render = SDL_CreateRenderer(screen, -1, render_flags);
    // SDL_Renderer = dictionary from SDL Library to create a surface to load an image from our computer memory into this game main memory
    SDL_Surface* img_screen_background;
    // IMG_Load = function from SDL Library to take image from our computer memory
    img_screen_background = IMG_Load(IMAGE_BACKGROUND);
    // SDL_Texture = dictionary from SDL Library that contain setting loads image from our game main meory to our graphics hardware memory
    SDL_Texture* screen_texture = SDL_CreateTextureFromSurface(render, img_screen_background);
    // SDL_FreeSurface = Function from SDL Library to clears image file from game main memory
    SDL_FreeSurface(img_screen_background);
    // SDL_Rect = dictionary from SDL Library that contain framewgork to create frame for our texture image
    SDL_Rect kanvas_background;
    //we fill the framework information here
    kanvas_background.x = 0;
    kanvas_background.y = 0;
    kanvas_background.h = SCREEN_HEIGHT;
    kanvas_background.w = SCREEN_WIDTH;
```

- **SDL_INIT**: Inisialisasi SDL. Fungsi ini harus dipanggil sebelum semua fungsi SDL lainnya. Parameter flags menentukan bagian SDL yang akan diinisialisasi. Jika inisialisasi tersebut nilainya tidak sama dengan nol maka SDL tidak akan dijalankan karena mengalami error.
- **SDL_Window* screen**: Buat *windows* dengan posisi, dimensi, dan *flags* yang ditentukan.
- Uint64 render_flags : Membuat render untuk merenderisasi file image tersebut.
- **SDL_Renderer* render** : Sebuah struktur yang berasal dari SDL Library untuk membuat permukaan untuk memunculkan sebuah gambar dari memori komputer kita menuju memori utama dari game ini yang dimana dialiaskan sebagai render.
- **SDL_Surface* img_screen_background** : Sebuah struktur yang berisi representasi data pixel yang efisien / dioptimalkan.
- **SDL_Texture* screen_texture** : Sebuah struktur yang berasal dari SDL Library yang mengandung representasi data piksel yang efisien dan spesifik driver.

- **SDL_FreeSurface** : Sebuah fungsi dari SDL Library untuk membersihkan file gambar dari memori utama game ini
- **SDL_Rect** : Sebuah index dari SDL Library yang mengandung framework untuk membuat frame untuk tekstur gambar kami

9. fungsi untuk membaca file png

```
//=========================={ setting gambar hantu dan target }============================//
IMG_Init( IMG_INIT_PNG ); //to make png image with transparant
```

- **IMG_Init(IMG_INIT_PNG)**: fungsi tersebut digunakan agar operasi tersebut dapat memuat format file .png yang merupakan format gambar yang kami gunakan untuk merepresentasikan hantu yang akan ditembak dan target (*cross-hair*) untuk menembak hantu-hantu tersebut..

10. Pengaturan gambar hantu

```
//-----------------------------[ setting gambar hantu ]---------------------------//
SDL_Surface* img_hantu;
img_hantu = IMG_Load(IMAGE_HANTU);
SDL_Texture* texture_hantu = SDL_CreateTextureFromSurface(render, img_hantu);
SDL_FreeSurface(img_hantu);
srand(time(NULL));   // Initialization, should only be called once.
KOMPONEN_HANTU hantus[JUMLAH_HANTU];
for (int i=0; i< JUMLAH_HANTU; i++)
{
    hantus[i].x = 500;
    hantus[i].y = 500;
    hantus[i].nyawa = NYAWA_HANTU;
}
```

- **SDL_Surface* img_hantu** merupakan index dari SDL Library yang didalamnya pengaturan membuat frame gambar, index ini dialiaskan sebagai img_hantu
- **IMG_Load(IMAGE_HANTU)**: fungsi dari SDL Library yang digunakan untuk mengambil gambar dari memory komputer kita dan ditaruh di memory program ini, fungsi IMG_Load membutuhkan path dari memory komputer kita untuk mengambil gambar tersebut yang dimana sudah kita deklarasikan di #define IMAGE_HANTU diatas
- **SDL_Texture* texture**: index dari SDL Library yang didalamnya memuat pengaturan texture gambar, dialiaskan sebagai texture.
- **texture** = SDL_CreateTextureFromSurface(render, image_hantu): mendeklarasikan index texture yang merupakan alias dari SDL_Texture didalamnya menjalankan fungsi SDL_CreateTextureFromSurface().

- **SDL_CreateTextureFromSurface(render, img_hantu**): fungsi dari SDL Library yang didalamnya mengandung pengaturan untuk membuat texture dari frame gambar yang ada. fungsi ini mewajibkan kita memasukan value SDL_Renderer dan SDL_Surface yang dimana kedua index ini sudah dialiaskan sebagai render dan image_hantu.
- SDL_FreeSurface(img_hantu): fungsi dari SDL Library untuk menghapus file gambar dari frame gambar, karena frame gambar telah diisi dengan texture yang diambil dari file gambar tersebut sehingga file tersebut tidak begitu banyak berguna lagi dan hanya memenuhi memory program. karena itu lebih baik membersihkannya dari frame gambar img_hantu
- **srand(time(NULL))**: Memanfaatkan jam internal komputer untuk mengontrol pilihan seed. Karena waktu terus berubah, seed itu selamanya berubah. Dengan kata lain, menghasilkan angka secara mengacak untuk menentukkan dimana hantu akan muncul ketika program dijalankan.
- **KOMPONEN_HANTU hantus[JUMLAH_HANTU]**: mendeklarasikan variable hantus sebagai variable yang mengandung elemen index KOMPONEN_HANTU dan menjadikannya array sebanyak JUMLAH_HANTU.
- Loop for untuk cek apakah jumlah hantu masih lebih besar dari 0 atau dengan kata lain, jumlah hantu belum habis.

11. Pengaturan untuk gambar target (*Cross-hair*)

```
//----------------------------[ setting gambar target ]-----------------------------//
SDL_Surface* img_target1;
SDL_Surface* img_target2;
SDL_Surface* img_target3;
img_target1 = IMG_Load(IMAGE_TARGET1);
img_target2 = IMG_Load(IMAGE_TARGET2);
img_target3 = IMG_Load(IMAGE_TARGET3);

SDL_Texture* targets[3];

SDL_Texture* texture_target;
texture_target = SDL_CreateTextureFromSurface(render, img_target1);
targets[0] = texture_target;
texture_target = SDL_CreateTextureFromSurface(render, img_target2);
targets[1] = texture_target;
texture_target = SDL_CreateTextureFromSurface(render, img_target3);
targets[2] = texture_target;

SDL_FreeSurface(img_target1);
SDL_FreeSurface(img_target2);
SDL_FreeSurface(img_target3);

SDL_Rect kanvas_target;
// SDL_Query = function from SDL Library SDL_Quis used to retrieve the basic settings of
// a texture, including the format, access, width, and height.
// and we need access to connects our texture with dest to control position
SDL_QueryTexture(texture_target, NULL, NULL, &kanvas_target.w, &kanvas_target.h);

kanvas_target.w = UKURAN_TARGET;
kanvas_target.h = UKURAN_TARGET;
kanvas_target.x = (SCREEN_WIDTH - kanvas_target.w) / 2;
kanvas_target.y = (SCREEN_HEIGHT - kanvas_target.h) / 2;
```

- **SDL_Surface* img_target1** merupakan index dari SDL Library yang didalamnya pengaturan membuat frame gambar, index ini dialiaskan sebagai img_target1. begitupun dengan SDL_Surface* img_target2, dan SDL_Surface* img_target3.
- **imgtarget1 = IMG_Load(IMAGE_TARGET1)**: menyatakan frame gambar imgtarget1 memuat fungsi IMG_Load(IMAGE_TARGET)
- **IMG_Load(IMAGE_TARGET)**: fungsi dari SDL Library yang digunakan untuk mengambil gambar dari memory komputer kita dan ditaruh di memory program ini, fungsi IMG_Load membutuhkan path dari memory komputer kita untuk mengambil gambar tersebut yang dimana sudah kita deklarasikan di #define IMAGE_TAGRET1 diatas. Begitupun dengan img_target2, dan img_target3.
- **SDL_Texture* targets[3**]: index dari SDL Library yang didalamnya memuat pengaturan texture gambar, dialiaskan sebagai texture dan dijadikan array.
- **SDL_Texture* texture_target** index dari SDL Library yang didalamnya memuat pengaturan texture gambar, dialiaskan sebagai texture_targe
- **texture_target = SDL_CreateTextureFromSurface(render, img_target1)**: fungsi dari SDL Library yang didalamnya mengandung pengaturan untuk membuat texture dari frame gambar yang ada. fungsi ini mewajibkan kita memasukan value SDL_Renderer dan SDL_Surface yang dimana kedua index ini sudah dialiaskan sebagai render dan img_target1. Begitupun dengan SDL_CreateTextureFromSurface(render, img_target2), dan SDL_CreateTextureFromSurface(render, img_target3)
- **texture_target = SDL_CreateTextureFromSurface(render, img_target1);**

  targets[0] = texture_target;

  texture_target = SDL_CreateTextureFromSurface(render, img_target2);

  targets[1] = texture_target;

  texture_target = SDL_CreateTextureFromSurface(render, img_target3);

  targets[2] = texture_target;

  ini merupakan sususan perintah dengan tujuan texture_target membaca setiap pengaturan texture dari img_target1, img_target2, img_target3 dan secara berurutan memasuki array targets.

- **SDL_FreeSurface(img_target1)**: fungsi dari SDL Library untuk menghapus file gambar dari frame gambar, karena frame gambar telah diisi dengan texture yang diambil dari file gambar tersebut sehingga file tersebut tidak begitu banyak berguna lagi dan hanya memenuhi memory program. karena itu lebih baik membersihkannya dari frame gambar img_target1. begitupun dengan frame img_target2, dan frame img_target3.

- SDL_Rect kanvas_target: index dari SDL Library yang didalamnya terdapat struktur pengaturan frame;
- kanvas_target.w = UKURAN_TARGET: pengaturan lebar frame yang value nya merupakan nilai dari #define UKURAN_TARGET
- kanvas_target.h = UKURAN_TARGET: pengaturan tinggi frame yang value nya merupakan nilai dari #define UKURAN_TARGET
- kanvas_target.x = (SCREEN_WIDTH - kanvas_target.w)/2: pengaturan posisi horizontal frame yang isinya nilai (SCREEN_WIDTH - kanvas_target.w)/2. kenapa begitu? untuk membuat posisi horizontal frame pas ditengah layar dengan komposisi lebar target
- kanvas_target.y = (SCREEN_WIDTH - kanvas_target.w)/2: pengaturan posisi horizontal frame yang isinya nilai (SCREEN_WIDTH - kanvas_target.h)/2. kenapa begitu? untuk membuat posisi horizontal frame pas ditengah layar dengan komposisi lebar target

12. Loop game utama dan kondisi-kondisi saat key dipencet

```
//==================================={ game loop }=========================================//
int close = 0;

while (!close)
{
    SDL_Event event;

    // Events management
    while (SDL_PollEvent(&event))
    {
        switch (event.type)
        {
        case SDL_QUIT:
            // handling of close button
            close = 1;
            break;

        case SDL_KEYDOWN:
            // keyboard API for key pressed
            switch (event.key.keysym.scancode)
            {
            case SDL_SCANCODE_X:
                {
                int tengahx = kanvas_target.x+(kanvas_target.w/2);
                int tengahy = kanvas_target.y+(kanvas_target.y/2);
                cek_ketembak(hantus,tengahx,tengahy);
                }
                break;
            case SDL_SCANCODE_W:
            case SDL_SCANCODE_UP:
                kanvas_target.y -= KECEPATAN_TARGET;
                 break;
             case SDL_SCANCODE_A:
             case SDL_SCANCODE_LEFT:
                 kanvas_target.x -= KECEPATAN_TARGET;
                 break;
             case SDL_SCANCODE_S:
             case SDL_SCANCODE_DOWN:
                 kanvas_target.y += KECEPATAN_TARGET;
                 break;
             case SDL_SCANCODE_D:
             case SDL_SCANCODE_RIGHT:
                 kanvas_target.x += KECEPATAN_TARGET;
                 break;
             case SDL_SCANCODE_M:
                 kursor_pilih_target++;
                 if (kursor_pilih_target > 2)
                 {
                     kursor_pilih_target = 0;
                 }
             default:
                 break;
             }
        }
    }
```

- **while (!close)**: ketika nilai *close* bernilai false, pada c++ nilai 0 akan disamadengankan dengan false sedangkan 1 disamadengankan dengan true, akan menjalankan loop.
- **SDL_Event event**: merupakan index dari SDL Library yang didalamnya data bermacam-macam event seperti SDL_PollEvent
- Dalam case **SDL_Keydown** terdapat switch case yang terdiri dari event, key, sym, dan scancode yang dimana semua dari itu berfungsi untuk mengatur keyboard API (Application Programming Interface) yang membuat jika kita

klik tombol-tombol yang disebutkan, maka akan jalan perintah-perintah sesuai tombol mana yang di klik

13. Kondisi agar target (hantu) tidak bergerak keluar batas layar

```
//------------pembatasan gerak target-------------------//
if (kanvas_target.x + kanvas_target.w > SCREEN_WIDTH){
    kanvas_target.x = SCREEN_WIDTH - kanvas_target.w;
}
if (kanvas_target.x < 0){
    kanvas_target.x = 0;
}
if (kanvas_target.y + kanvas_target.h > SCREEN_HEIGHT){
    kanvas_target.y = SCREEN_HEIGHT - kanvas_target.h;
}

if (kanvas_target.y < 0){
    kanvas_target.y = 0;
}
//-----------------------------------------------------//
```

- **if (kanvas_target.x + kanvas_target.w > SCREEN_WIDTH){**

  kanvas_target.x = SCREEN_WIDTH - kanvas_target.w

}

- **if (kanvas_target.x + kanvas_target.w < 0){**

  kanvas_target.x = 0

}

ini adalah kondisi dimana jika posisi vektor horinzontal frame target melebihi lebar layar maka posisi frame akan dipaksa untuk tetap didalam layar. Dan jika posisi horinzontal frame target kurang dari 0 maka posisi frame akan tetap di titik 0

- **if (kanvas_target.y + kanvas_target.h > SCREEN_WIDTH){**

  kanvas_target.y = SCREEN_WIDTH - kanvas_target.h

}

- **if (kanvas_target.y + kanvas_target.h < 0){**

kanvas_target.y = 0

}

ini adalah kondisi dimana jika posisi vektor vertikal frame target melebihi panjang layar maka posisi frame akan dipaksa untuk tetap didalam layar. Dan jika posisi vektor vertikal frame target kurang dari 0 maka posisi frame akan tetap di titik 0

14. Mulai menggambar layar, hantu dan target

```
// clears the screen
SDL_RenderClear(render);

//gambar layar//
SDL_RenderCopy(render, screen_texture, NULL, &kanvas_background);


//gambar hantu//
for (int i = 0; i < JUMLAH_HANTU; i++) {
    gerak_hantu(&hantus[i]);
    gambar_hantu(render,texture_hantu,&hantus[i]);

}

//gambar target//
SDL_RenderCopy(render, targets[kursor_pilih_target], NULL, &kanvas_target);


// SDL function for multiple rendering
SDL_RenderPresent(render);

// calculates to 60 fps
SDL_Delay(1000 / 60);
```

- **SDL_RenderClear(render)**: fungsi dari SDL Library untuk clear render yang sekarang ada
- **SDL_RenderCopy(render, screen_texture, NULL, &kanvas_background)**: fungsi dari SDL Library yang fungsinya menampilkan screen_texture pada frame gambar kanvas_background
- **for (int i =0; i < JUMLAH_HANTU; i++){**

  gerak_hantu(&hantus[i])

  gambar_hantu(render,texture_hantu.&hantus[i])

  }

ini adalah looping dimana disaat i = 0 dan i lebih kecil dari nilai JUMLAH_HANTU, maka fungsi gerak_hantu dengan isi data index array hantus akan dipanggil, lalu fungsi gambar_hantu dengan isi data render, frame gambar texture_hantu dan data index array hantus

- **SDL_RenderCopy(render, targets[kursor_pilih_target], NULL, &kanvas_target)**: fungsi dari SDL Library yang fungsinya menampilkan screen_texture pada frame gambar kanvas_target.
- **SDL_RenderPresent(render)**: memperbarui layar dengan index SDL_Render yang dialiaskan dengan render
- **SDL_Delay(1000/60)**: agar kecepatan frame per detik pada game setelah di jalankan optimal/lancar.

15. Ketika loop game telah berakhir

```
// destroy texture
SDL_DestroyTexture(texture_target);
SDL_DestroyTexture(texture_hantu);
SDL_DestroyTexture(screen_texture);

// destroy renderer
SDL_DestroyRenderer(render);

// destroy window
SDL_DestroyWindow(screen);

// close SDL
SDL_Quit();

return 0;
```

- SDL_DestroyTexture(texture_target): fungsi dari SDL Library untuk menghancurkan texture dari program kita setelah loop game selesai, agar tidak mengotori memory setelah game selesai. Begitupun dengan SDL_DestroyTexture(texture_hantu), dan SDL_DestroyTexture(screen_texture).
- SDL_DestroyWindow(screen): fungsi untuk menghancurkan layar setelah game loop selesai, agar windows ketutup saat game selesai.
- SDL_Quit(): fungsi dari SDL Library untuk menutup program ini

KETENTUAN - KETENTUAN YANG BERHASIL TERPENUHI:

1. Subprogram

(Fungsi untuk cek jika hantu “ketembak” [7])

2. Perulangan

(Loop game utama dan kondisi-kondisi saat key dipencet [12])

3. Percabangan

(kondisi agar target (hantu) tidak bergerak keluar batas layar [13])

4. Array

(Pengaturan gambar hantu [10])

5. Searching

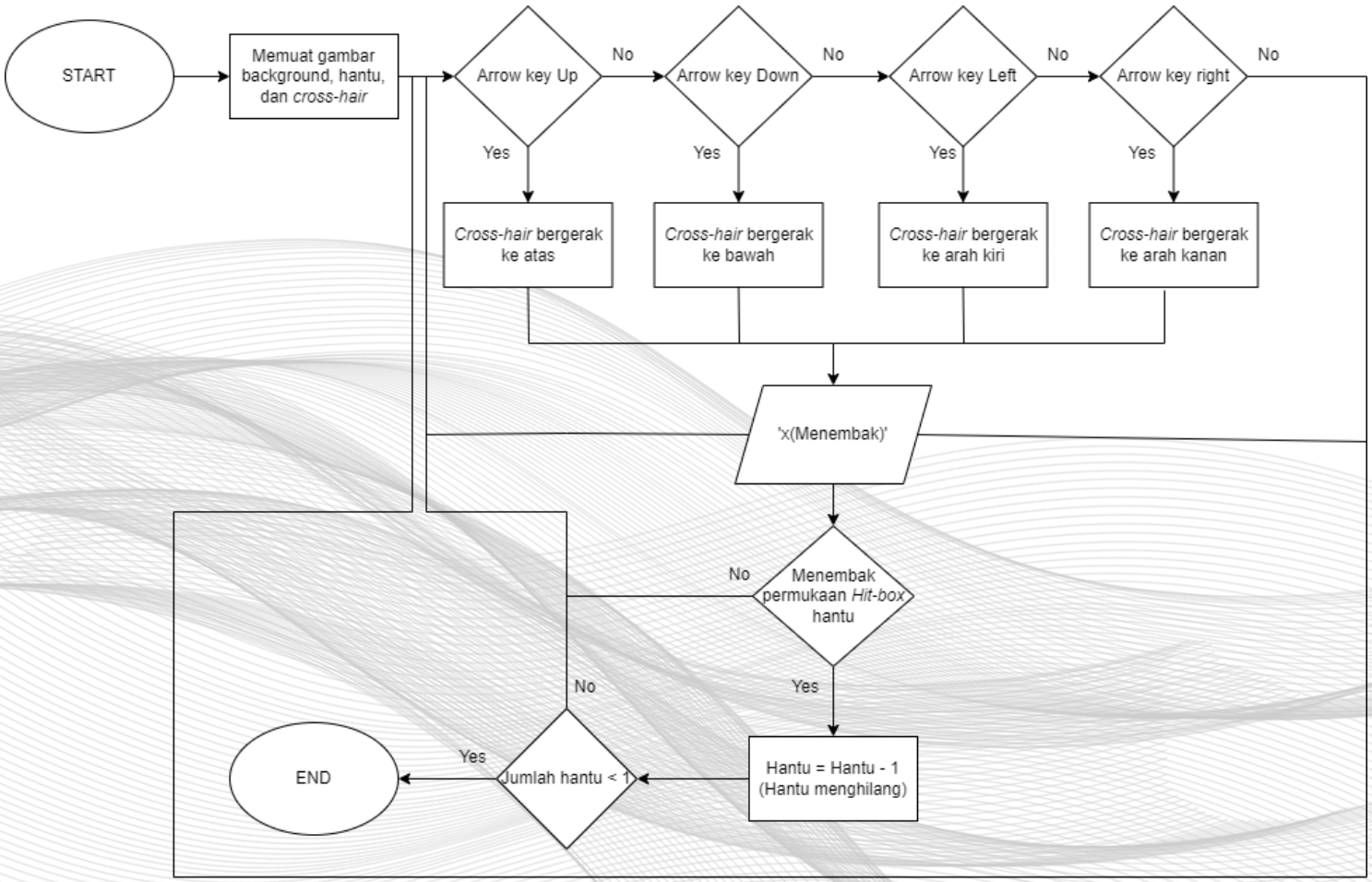
(Mulai menggambar layar, hantu dan target [14])

6. Operasi Data

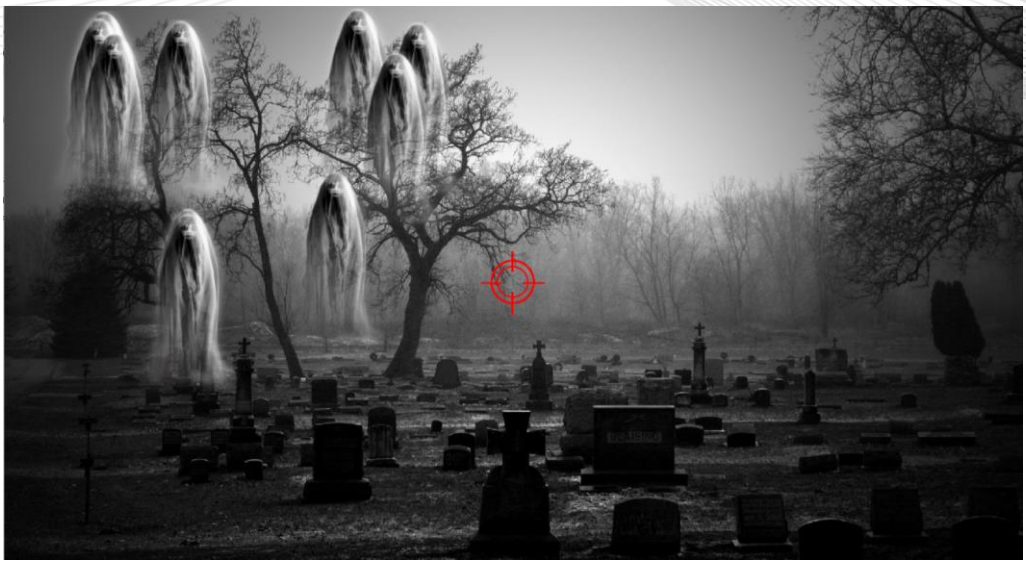
(Pengaturan untuk gambar target (*Cross-hair*) [11])

BAB II

HASIL DAN PEMBAHASAN



Preview:



Full Preview: https://youtu.be/7a5NDCpt9Do

Compiler

Project ini melibatkan banyak library SDL sehingga susah untuk di akses di komputer lain

yang tidak memiliki library SDL, karena itu kelompok kami mempersembahkan file executable kami yang sudah di compile menggunakan SDL static libs yang biasanya digunakan untuk mendistribusikan program game

```
compile static link file on linux = g++  -o hasil2 tubes2.cpp `sdl2-config --static-libs --cflags` -lGL -Wall /usr/lib/x86_64-linux-gnu/libSDL2.a /usr/lib/x86_64-linux-gnu/libSDL2_image.a -lpng -ltiff -ljpeg -lwebp
```

berikut adalah screenshot dari anggota kelompok kami Nezha dengan NIM 105221021 tentang cara compile di linux

```
compile static link on windows =  g++  tubes.cpp -IC:\Users\nezha\Downloads\SDL2\x86_64-w64-mingw32\include -LC:\Users\nezha\Downloads\SDL2\x86_64-w64-mingw32\lib -Wall -lmingw32 -lSDL2main -lSDL2 -lSDL2_image -o hasil.exe
```

berikut adalah screenshot dari anggota kelompok kami Nezha dengan NIM 105221021 cara compile di windows

Sehingga program game ini dapat dijalankan oleh pengguna windows ataupun linux dengan cara:

1. cd path/to/game/director

lalu

2. ./start_game_linux (untuk pengguna linux)

   atau

   ./start_game_windows.exe (untuk pengguna windows)

BAB III

KESIMPULAN DAN SARAN

- **KESIMPULAN**

  Dalam projek jangka panjang yang kami buat sebaik-baiknya ini, walaupun memang terlihat kompleks dan jauh dari pelajaran, pembuatan projek ini sungguh menjadi pengalaman yang sungguh luar biasa. Semoga dari laporan yang kami buat ini, bagi para pembaca, diharapkan dapat mencakup serta mencerna ilmu yang telah kami ambil. Dengan menggunakan SDL (*Simple DirectMedia Layer*) yang merupakan perpustakaan pengembangan perangkat lunak lintas platform yang dirancang untuk menyediakan lapisan abstraksi perangkat keras untuk komponen perangkat keras multimedia komputer, kami mengetahui akan potensi yang dimiliki *software* ini.

- **SARAN**

  Sangat wajar untuk kita masuk kuliah dan kurang paham akan apapun di bidang akademik. Ini tidak hanya untuk jurusan Ilmu Komputer tapi berlaku untuk semua jurusan. Jadi tolong katakan pada kita diri sendiri "Bagus, saya tidak tahu apa-apa jadi saya akan punya banyak hal untuk dipelajari selama beberapa tahun kedepan". Banyak juga kok lulusan Ilmu Komputer atau Teknik Informatikan yang bilang dia bisa pemrograman tetapi menurut saya kemampuannya masih masih jauh dibawah standar industri. Tapi banyak juga yang tidak lulusan kampus punya kemampuan yang bagus di bidang pemrograman.

  Jangan mudah menyerah! *May the force be with you.*.

DAFTAR PUSTAKA

https://id.quora.com/Apa-saran-Anda-untuk-saya-yang-masuk-fakultas-ilmu-komputer-tetapi-tidak-sepenuhnya-mengerti-pemrograman-sepenuhnya

https://wiki.libsdl.org/SDL_Event

https://elearning.universitaspertamina.ac.id/mod/assign/view.php?id=79979

https://www.educba.com/c-plus-plus-typedef/