PEMROGAMAN 4

MENCARI DAN MENGANALISA
SOURCE CODE GAME

Disusun oleh :

Galih Aulia Al Hakim 4210161028

PROGRAM STUDI TEKNOLOGI GAME
DEPARTEMEN TEKNOLOGI MULTIMEDIA KREATIF
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
SURABAYA

2018

1. Class – class pada game
   - Class Background

```
#ifndef _BACKGROUND_H_
#define _BACKGROUND_H_

#include "game.h"

class Background {
  const static int step = 1;
private:
  SDL_Surface* image;
  int x, y, x_, y_;
  int w, h;
public:
  Background(SDL_Surface*);

  void scroll();
  void unscroll();
  void reset();
  void draw(SDL_Surface*);
};

#endif /* BACKGROUND_H_ */
```

Class Background ini nantinya akan difungsikan untuk menscolling background saat player berjalan dan menghentikan background saat player berhenti berjalan atau mati. Dalam Class Background ini juga digunakan untuk menggambar background (meloop ulang) background yang ada.

```
#include "background.h"
Background::Background(SDL_Surface* s) {
  image = s;
  x = y= x_ = y_ = 0;
  w = s->w;
  h = s->h;
}
void Background::scroll() {
  x -= Background::step;
  if(x <= -w) {
    this->reset();
  }
}
void Background::unscroll() {
  x += Background::step;
  if(x <= -w) {
    this->reset();
  }
}
void Background::reset() {
  x = x_;
  y = y_;
}
void Background::draw(SDL_Surface* screen) {
  ApplySurface(x, y, image, screen);
  if(x <= -w + WIDTH) {
    ApplySurface(x + w, y, image, screen);
  }
}
```

- Class Bullet

```
#ifndef _BULLET_H_
#define _BULLET_H_
#include "game.h"
class Bullet {
private:
  int x, y;
  int xmove, ymove;
  SDL_Surface* image;
public:
  const static int MaxDistance = WIDTH;
  int dx, dy;

  Bullet(SDL_Surface *);
  Bullet(SDL_Surface*, int, int);
  Bullet(SDL_Surface*, int, int, int, int);

  void setXMove(int v) { xmove = v; }
  void setYMove(int v) { ymove = v; }

  int getX() { return x; }
  int getY() { return y; }

  void setX(int v) { x = v; }
  void setY(int v) {y = v; }
  void setXY(int a, int b) { x = a; y = b; }

  void move();
  void draw(SDL_Surface*);
};
#endif /* _BULLET_H_ */
```

Class Bullet ini nantinya akan difungsikan untuk mengatur gerak dari peluru atau tembakan dari senjata yang ditembakan oleh player(character). Dalam Class Bullet ini juga mengatur gambaran dari pelurunya sendiri ada pada fungsi draw.

```
#include "bullet.h"
Bullet::Bullet(SDL_Surface* i) {
  image = i;
  dx = dy = x = y= xmove = ymove = 0;
}
Bullet::Bullet(SDL_Surface* i, int x_, int y_) {
  image = i;
  x = x_;
  y = y_;
  dx = dy = xmove = ymove = 0;
}
Bullet::Bullet(SDL_Surface* i, int x_, int y_, int dx_, int dy_) {
  image = i;
  x = x_;
  y = y_;
  xmove = dx_;
  ymove = dy_;
  dx = dy = 0;
}
void Bullet::move() {
  x += xmove;
  y += ymove;
  dx += xmove;
  dy += ymove;
}
void Bullet::draw(SDL_Surface* s) {
  ApplySurface(x, y, image, s);
}
```

- Class Character

```
#ifndef _CHARACTER_H_
#define _CHARACTER_H_

#include "bullet.h"

#include <vector>
#include <fstream>

#include "SDL/SDL.h"
#include "SDL/SDL_image.h"

class Character {
private:
  string name;
  int x, y;
  double velocity_x, velocity_y;

  int life;

  double frame;
  int numframes[NUMSTATES]; // rest, walk, jump, attack

  MoveState state;

  bool jumping;

  SDL_Rect *clips[NUMSTATES]; // rest, walk, jump, attack
  SDL_Surface *image;

  SDL_Surface *bulletImage;
  std::vector<Bullet> bullets;
  Uint32 lastShot;

public:
  Character(std::string, int, int);
  Character() { };
  ~Character();
  bool load();
  void draw(SDL_Surface*);
  int nextFrame();
  void handleKeys(const bool[]);
  int getX() { return x; }
  int getY() { return y; }
  void setX(int v) { x = v; }
  void setY(int v) { y = v; }
  void fire();

  void hit(int damage=ZOMBIE_DAMAGE) {life -= damage; }
  bool alive() { return life > 0; }
  int health() { return life; }

};

#endif /*_CHARACTER_H_*/
```

Class Character ini nantinya akan difungsikan untuk mengatur nama yang nantinya akan diinputkan oleh user sebagai nama pada character yang user gunakan. Selanjutkan mengatur jalan gerak dari character itu sendiri, untuk menggerakkan character itu sendiri menggunakan velocity untuk menggerakkannya. Mengatur life, mengatur animasi yang perlu untuk character ini, mengatur *jump*, terdapat fungsi – fungsi yang dapat dipadukan pada class lain yakni fungsi fire.

Pada fungsi fire ini mengatur berapa kecepatan yang ditembakkan oleh player saat menembakkan tembakan.

```
167 void Character::fire() {
168   lastShot = SDL_GetTicks();
169   bullets.push_back(Bullet(bulletImage, x + CHARACTER_WIDTH / 2,
170                            y + CHARACTER_HEIGHT / 1.7, BULLET_SPEED, 0));
171 }
```

Pada Class Character ini juga mengatur untuk user dapat mengontrol character mereka dengan fungsi handleKey, yang dimana didalam fungsi ini terdapat key yang diinputkan untuk mengontrol character. Seperti *Jump, Attack,* dan berjalan

```
117 void Character::handleKeys(const bool keys[]) {
118   bool moved = false;
119   bool atk = false;
120
121   if(keys[SDLK_UP] && !jumping) {
122     state = JumpState;
123     frame = 0;
124     jumping = true;
125     velocity_y = Y_VELOCITY_STEP;
126     y += Y_VELOCITY_STEP;
127     moved = true;
128   }
129   if(jumping) {
130     moved = true;
131     if(y + velocity_y < HEIGHT - CHARACTER_HEIGHT) {
132       y = y + velocity_y;
133       velocity_y = velocity_y + GRAVITY;
134     } else {
135       jumping = false;
136       moved = false;
137       velocity_y = 0;
138       y = HEIGHT - CHARACTER_HEIGHT;
139     }
140   }
```

```
141   if(keys[SDLK_RIGHT]) {
142     x += CHARACTER_STEP;
143     moved = true;
144   }
145   if(keys[SDLK_LEFT]) {
146     x -= CHARACTER_STEP;
147     moved = true;
148   }
149
150   if(keys[SDLK_SPACE]) {
151     state = AttackState;
152     atk = true;
153     if(SDL_GetTicks() - lastShot > BULLET_DELAY) {
154       this->fire();
155     }
156   }
157
158   if(moved && !jumping && !atk) {
159     state = WalkState;
160   }
161
162   if(!moved && !atk) {
163     state = RestState;
164   }
165 }
```

- Class Fog

```
#ifndef _FOG_H_
#define _FOG_H_

#include "game.h"

class Fog {
  const static int step = 2;
private:
  SDL_Surface* fog;
  int x, y, x_, y_;
  int w, h;
public:
  Fog(SDL_Surface*);

  void setX(int v) { x = x_ = v; }
  void setY(int v) { y = y_ = v; }
  void setXY(int a, int b) { x = x_ = a; y = y_ = b; }

  void scroll();
  void reset();
  void draw(SDL_Surface*);
};

#endif /* _FOG_H_ */
```

Class Fog ini hampir sama kayak Class Background karena dalam Class Fog ini mengatur screen pada step ke-2 sedangkan dalam Class Background ini mengatur screen pada step ke-1.

- Class Music

```
#ifndef _MUSIC_H_
#define _MUSIC_H_

#include "SDL/SDL_mixer.h"

class Music {
  const static int audioRate = 22050;
  const static Uint16 audioFormat = AUDIO_S16;
  const static int audioChannels = 2;
  const static int audioBuffers = 4096;
 private:
  Mix_Music *music;
 public:
  static void init();
  Music() {};
  Music(std::string);
  ~Music();
  void load(std::string);
  void play();
  void stop();
  void loop();

};

#endif
```

Class Music ini nantinya akan mengatur music yang ada pada game ini. Seperti ngeload data music lalu memutarnya dan memberhentikannya apabila game dalam keadaan pause, dan mengatur loop pada music apabila music(BGM) sudah habis.

- Class Time

```
#ifndef _TIMER_H_
#define _TIMER_H_

class Timer {
private:
  int startTicks;
  int pausedTicks;
public:
  bool paused, started;

  Timer();
  void start();
  void pause();
  void unpause();
  int getTicks();
};

#endif /* _TIMER_H_ */
```

Class Time ini nantinya akan mengatur bagaimana saat game sedang start atau pause. Dalam Class Time ini terdapat fungsi start dan juga fungsi pause. Saat game posisi sedang pause, terdapat fungsi unpause untuk melanjutkan game tersebut.

- Class Zombie

```
#ifndef _ZOMBIE_H_
#define _ZOMBIE_H_

#include <iostream>
#include <fstream>

#include "game.h"
#include "character.h"

// TODO: inherit from Character? //
class Zombie {
private:
  string name;
  int x, y;

  int life;

  double frame;
  int numframes[ZNUMSTATES]; /* rest, walk, death, attack */

  ZombieState state;

  SDL_Rect *clips[ZNUMSTATES];
  SDL_Surface *image;

public:
  Zombie(std::string, int, int);
  Zombie() {};
  ~Zombie();
```

```
  bool load();
  void draw(SDL_Surface*);
  int nextFrame();
  void reactToPlayer(Character*);
  void attack() { state = ZAttackState; }

  int getX() { return x; }
  int getY() { return y; }
  void setX(int v) { x = v; }
  void setY(int v) { y = v; }
  void setXY(int a, int b) { x = a; y = b; }

  void hit(int damage=BULLET_DAMAGE) { life -= damage; }

  bool alive() { return life > 0; }
  int health() { return life; }

};

#endif /* _ZOMBIE_H_ */
```

Class Zombie ini berfungsi hampir sama seperti character seperti mengatur nama yang nantinya digunakan pada zombie, mengatur life, mengatur animasi yang digunakan, namun ada yang beda dari Class Zombie ini dibandingkan dengan Class Character yaitu, AI Zombie yang berjalan menuju player untuk melawan player sendiri. Dalam Class Zombie ini memadukan Class Character untuk mengatur arah bergerak dari Zombienya sendiri.

```
 98  void Zombie::reactToPlayer(Character* c) {
 99    int playerx = c->getX() + CHARACTER_WIDTH;
100    // unused
101    // int playery = c->getY() + CHARACTER_HEIGHT / 2;
102
103    if(state == ZDeathState || life <= 0) {
104      state = ZDeathState;
105      return;
106    }
107
108
109    if(abs(x - playerx) < ZOMBIE_WIDTH / 4) {
110      if(state != ZAttackState) frame = 0;
111      state = ZAttackState;
112      //TODO: Actual collision detection
113      c->hit();
114    } else {
115      state = ZWalkState;
116      if(playerx < x) {
117        x -= ZOMBIE_STEP;
118      } else {
119        x += ZOMBIE_STEP;
120      }
121    }
122  }
```

Selain arah gerak zombie yang menuju ke player, zombie ini juga melakukan attack pada player dan juga hancur(mati). Dalam Class Zombie ini juga mengatur wave kedatangan zombie yang berjalan menuju player.