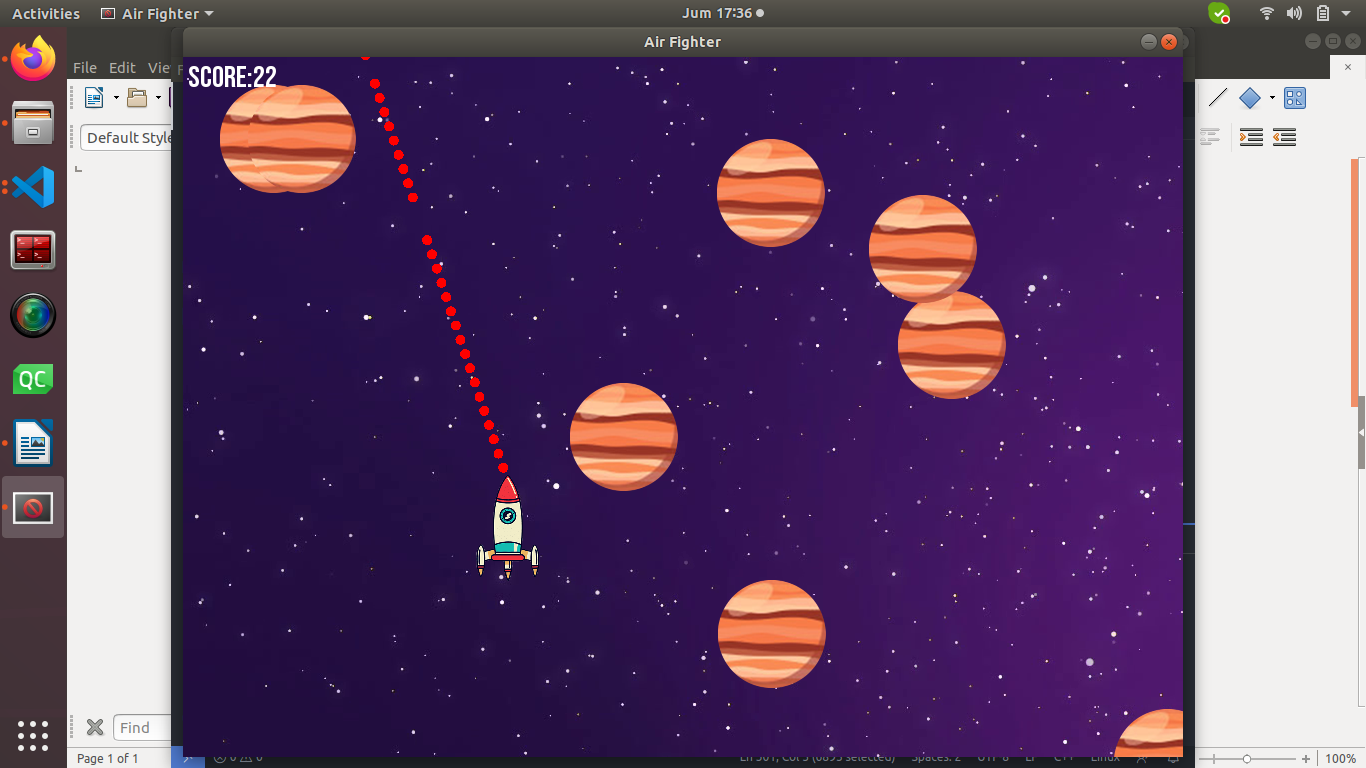
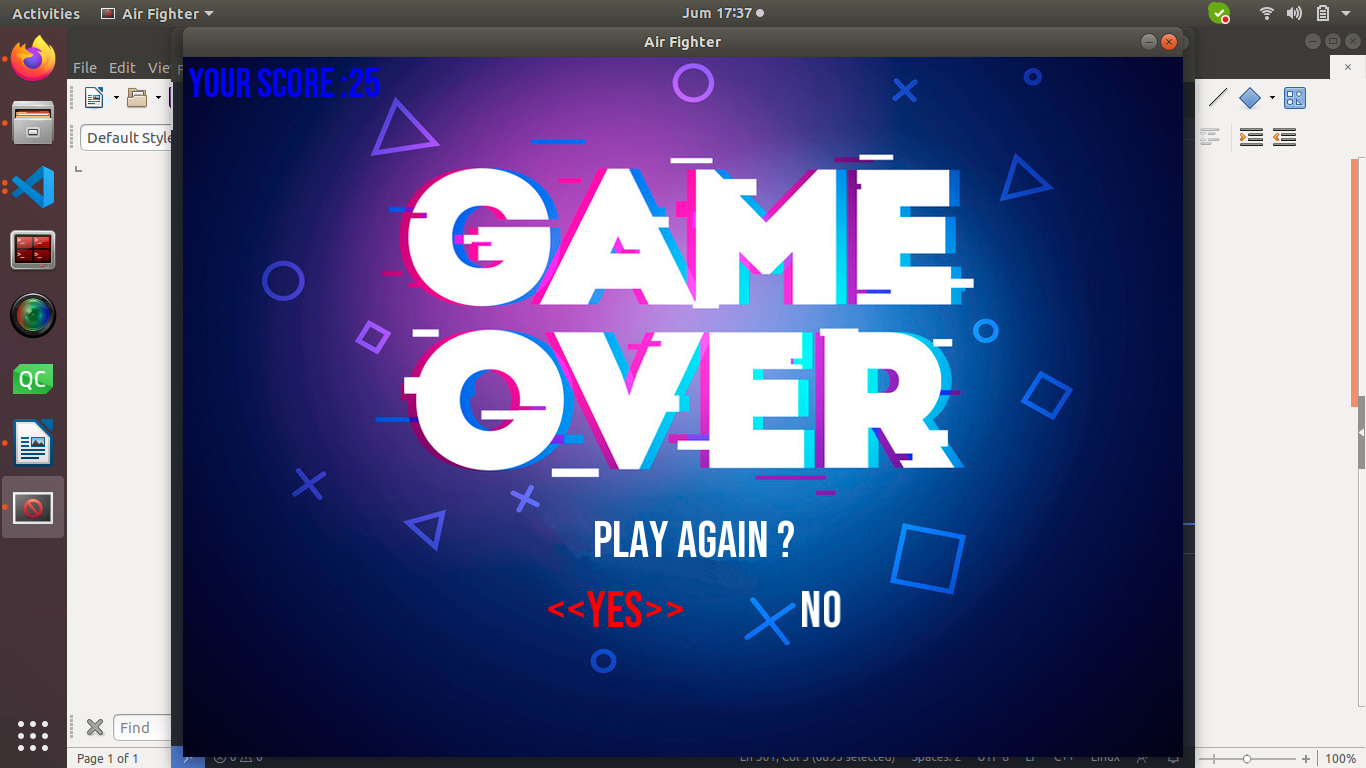
Nama : Nathanael Hutama Harsono

NRP : 07211940000044

Tanggal : 10/4/2020

**1. Hasil :**

****

****

**2. Program :**

#include<iostream>

#include<SFML/Graphics.hpp>

#include<SFML/Audio.hpp>

#include<math.h>

#include<vector>

#include<cstdlib>

#include<string>

#include<ctime>

#include"Menu.cpp"

using namespace sf;

class Bullet

{

public:

CircleShape shape;

Vector2f currVelocity;

float maxSpeed;

Bullet(float radius = 5.f)

: currVelocity(0.f, 0.f), maxSpeed(15.f)

{

shape.setRadius(radius);

shape.setFillColor(Color::Red);

}

};

std::string stringscore="", playagain="PLAY AGAIN ?";

int score,a;

int startGame();

int gameOver();

int awal;//untuk memulai game

RenderWindow window;

int main()

{

score = 0;

//sound untuk game

SoundBuffer gameSoundBuffer;

gameSoundBuffer.loadFromFile("game.wav");

//font

Font myfont;

myfont.loadFromFile("../font/BebasNeue-Regular.ttf");

srand(time(NULL));

Vector2i centerWindow((VideoMode::getDesktopMode().width / 2)-500, (VideoMode::getDesktopMode().height / 2)-390);

window.create(VideoMode(1000, 700), "Air Fighter", Style::Titlebar | Style::Close);

window.setPosition(centerWindow);

window.setFramerateLimit(60);

window.setMouseCursorVisible(false);

//texture

Texture background, enemy1, player;

//load images

if(!background.loadFromFile("../gambar/wallpaper.png")) return EXIT\_FAILURE;

if(!enemy1.loadFromFile("../gambar/jupiter.png")) return EXIT\_FAILURE;

if(!player.loadFromFile("../gambar/pesawat.png")) return EXIT\_FAILURE;

//sprite

Sprite Player(player), Enemy1(enemy1), Background(background);

//Player

Player.setScale(0.1f,0.1f);

Player.setOrigin(270.f,50.f);

//set posisi awal Player

Player.setPosition(Vector2f(window.getSize().x/2, window.getSize().y - 100.f) );

//Bullets

Bullet b1;

std::vector<Bullet> bullets;

//Enemy

Enemy1.setScale(0.2f,0.2f);

int spawnCounter1 = 0;

std::vector<Sprite> enemies1;

//Vectors

Vector2f playerCenter;

Vector2f mousePosWindow;

Vector2f aimDir;

Vector2f aimDirNorm;

startGame();

if (awal==1){

while (window.isOpen())

{

stringscore="SCORE:"+std::to\_string(score);

Text text(stringscore, myfont, 30);

text.setPosition(5,0);

Event event;

while (window.pollEvent(event))

{

if (event.type == Event::Closed)

window.close();

}

//Update

//Vectors

playerCenter = Vector2f(Player.getPosition());

mousePosWindow = Vector2f(Mouse::getPosition(window));

aimDir = mousePosWindow - playerCenter;

aimDirNorm = aimDir / (float)sqrt(pow(aimDir.x, 2) + pow(aimDir.y, 2));

//Player

//agar Player tidak keluar window

if (Player.getPosition().x < 35.f || Player.getPosition().x > window.getSize().x-40.f

|| Player.getPosition().y < 0 || Player.getPosition().y > window.getSize().y-100.f)

{Player.move(0.f,0.f);}

//menggerakkan Player

if (Player.getPosition().x > 35.f && Keyboard::isKeyPressed(Keyboard::A))

{Player.move(-6.f, 0.f);}

if (Player.getPosition().y > 0 && Keyboard::isKeyPressed(Keyboard::W))

{Player.move(0.f, -6.f);}

if (Player.getPosition().x < window.getSize().x-40.f && Keyboard::isKeyPressed(Keyboard::D))

{Player.move(6.f, 0.f);}

if (Player.getPosition().y < window.getSize().y-100.f && Keyboard::isKeyPressed(Keyboard::S))

{Player.move(0.f, 6.f);}

//Enemies

if (spawnCounter1 < 20)

spawnCounter1++;

if(spawnCounter1 >= 20 && enemies1.size() < 10)

{

Enemy1.setPosition(Vector2f(rand() % window.getSize().x, 0.f));

enemies1.push\_back(Enemy1);

spawnCounter1 = 0;

}

//Shooting

if (Mouse::isButtonPressed(Mouse::Left))

{

b1.shape.setPosition(playerCenter);

b1.currVelocity = aimDirNorm \* b1.maxSpeed;

bullets.push\_back(Bullet(b1));

}

for (int i = 0; i < bullets.size(); i++)

{

bullets[i].shape.move(bullets[i].currVelocity);

//Menghapus peluru yg keluar dr window

if (bullets[i].shape.getPosition().x < 0 || bullets[i].shape.getPosition().x > window.getSize().x

|| bullets[i].shape.getPosition().y < 0 || bullets[i].shape.getPosition().y > window.getSize().y)

{

bullets.erase(bullets.begin());

}

else

{

//Mendeteksi tabrakan Enemy dengan Bullet

for (int k = 0; k < enemies1.size(); k++)

{

if (bullets[i].shape.getGlobalBounds().intersects(enemies1[k].getGlobalBounds()))

{

bullets.erase(bullets.begin() + i);

enemies1.erase(enemies1.begin() + k);

score = score+1;

break;

}

}

}

}

//Draw

window.clear();

window.draw(Background);

//Draw Enemy

for (int i = 0; i < enemies1.size(); i++)

{

window.draw(enemies1[i]);

enemies1[i].move(0.f, rand()%5 + 1);

//mendeteksi tabrakan Enemy dengan Player

if (Player.getGlobalBounds().intersects(enemies1[i].getGlobalBounds()))

{

gameOver();

}

//me-reset Enemy yg sdh keluar window

if (enemies1[i].getPosition().y > window.getSize().y)

{

enemies1[i].setPosition(Vector2f(rand()%window.getSize().x, 0.f));

}

}

//Draw Player

window.draw(Player);

//Draw Bullet

for (int i = 0; i < bullets.size(); i++)

{

window.draw(bullets[i].shape);

}

window.draw(text);

window.display();

}

}

return 0;

}

int startGame(){

// Start start(window.getSize().x, window.getSize().y);

// while(window.isOpen())

// {

// window.clear();

// start.draw(window);

// }

return awal=1;

}

int gameOver(){

Menu menu(window.getSize().x, window.getSize().y);

Texture gameover;

if (!gameover.loadFromFile("../gambar/gameover.jpg")) return EXIT\_FAILURE;

Sprite Gameover(gameover);

SoundBuffer gameOver;

gameOver.loadFromFile("../sound/crash.wav");

Sound GameOver;

GameOver.setBuffer(gameOver);

GameOver.play();

while (window.isOpen())

{

Font myfont;

myfont.loadFromFile("../font/BebasNeue-Regular.ttf");

stringscore="YOUR SCORE :"+std::to\_string(score);

Text text(stringscore, myfont, 40);

text.setFillColor(Color::Blue);

text.setPosition(5,0);

Text playAgain(playagain, myfont, 50);

playAgain.setPosition(410,450);

Event event;

while (window.pollEvent(event))

{

switch (event.type)

{

case Event::KeyReleased:

switch (event.key.code)

{

case Keyboard::Left:

menu.MoveLeft();

break;

case Keyboard::Right:

menu.MoveRight();

break;

case Keyboard::Return:

switch (menu.GetPressedItem())

{

case 0:

std::cout << "Play again" << std::endl;

main();

break;

case 1:

std::cout << "Exit" << std::endl;

window.close();

break;

}

break;

}

break;

case Event::Closed:

window.close();

break;

}

}

window.clear();

window.draw(Gameover);

window.draw(text);

window.draw(playAgain);

menu.draw(window);

window.display();

}

}

**File Menu.hpp :**

#pragma once

#include<SFML/Graphics.hpp>

#define MAX\_NUMBER\_OF\_ITEMS 2

class Menu

{

public:

Menu(float width, float height);

~Menu();

void draw(sf::RenderWindow &window);

void MoveLeft();

void MoveRight();

int GetPressedItem() { return selectedItemIndex; }

private:

int selectedItemIndex;

sf::Font font;

sf::Text menu[MAX\_NUMBER\_OF\_ITEMS];

};

**File Menu.cpp :**

#include "Menu.hpp"

Menu::Menu(float width, float height)

{

font.loadFromFile("../font/BebasNeue-Regular.ttf");

menu[0].setFont(font);

menu[0].setCharacterSize(50);

menu[0].setFillColor(sf::Color::Red);

menu[0].setString("<<YES>>");

menu[0].setPosition(sf::Vector2f(width / (MAX\_NUMBER\_OF\_ITEMS + 1 ) \* 1 + 30, height / 2 + 170));

menu[1].setFont(font);

menu[1].setCharacterSize(50);

menu[1].setFillColor(sf::Color::White);

menu[1].setString("NO");

menu[1].setPosition(sf::Vector2f(width / (MAX\_NUMBER\_OF\_ITEMS + 1 ) \* 2 - 50, height / 2 + 170));

selectedItemIndex = 0;

}

Menu::~Menu()

{

}

void Menu::draw(sf::RenderWindow &window)

{

for (int i = 0; i < MAX\_NUMBER\_OF\_ITEMS; i++)

{

window.draw(menu[i]);

}

}

void Menu::MoveLeft()

{

if (selectedItemIndex > 0)

{

menu[selectedItemIndex].setString("NO");

menu[selectedItemIndex].setFillColor(sf::Color::White);

selectedItemIndex--;

menu[selectedItemIndex].setString("<<YES>>");

menu[selectedItemIndex].setFillColor(sf::Color::Red);

}

else

{

menu[selectedItemIndex].setString("YES");

menu[selectedItemIndex].setFillColor(sf::Color::White);

selectedItemIndex=1;

menu[selectedItemIndex].setString("<<NO>>");

menu[selectedItemIndex].setFillColor(sf::Color::Red);

}

}

void Menu::MoveRight()

{

if (selectedItemIndex + 1 < MAX\_NUMBER\_OF\_ITEMS)

{

menu[selectedItemIndex].setString("YES");

menu[selectedItemIndex].setFillColor(sf::Color::White);

selectedItemIndex++;

menu[selectedItemIndex].setString("<<NO>>");

menu[selectedItemIndex].setFillColor(sf::Color::Red);

}

else

{

menu[selectedItemIndex].setString("NO");

menu[selectedItemIndex].setFillColor(sf::Color::White);

selectedItemIndex=0;

menu[selectedItemIndex].setString("<<YES>>");

menu[selectedItemIndex].setFillColor(sf::Color::Red);

}

}