**TITLE: DOODLE GAME**

**LAB EXAM**



**Spring 2021**

**CSE208L OOP Lab**

Submitted by: **WALIIYA RIZWAN**

Registration No. : **19PWCSE1766**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Sumayyea Salahuddin**

March 22,2021

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

# **DOODLE GAME AND SFML**

## INTRODUCTION:

In this lab exam we learned to create doodle game with help of Simple and Fast Library.

## SFML CLASSES:

### RENDER WINDOW:

#### DEFINITION:

RenderWindow is derived from sf::Window class of SFML, thus it inherits all its features: events, window management, OpenGL rendering, etc.

#### FUNCTIONS USED IN GAME:

1. **Constructor:** this constructor takes in the object of Video mode and the name of title bar.
2. **SetFramerateLimit:** limits the frame rate to a maximum fixed frequency.
3. **Draw:** this function is used to draw a drawable object to the render target.
4. **Display:** it is used to display on screen what has been rendered to the window so far
5. **Close:** It is used to close the window and destroy all the attached resources
6. **pollEvent:** This function is used to pop the event on top of the event queue, if any, and return it

### VIDEO MODE:

#### DEFINITION:

A video mode is defined by a width and a height (in pixels) and a depth (in bits per pixel). This allows to build windows with the same size or pixel depth as the current resolution.

#### FUNCTIONS USED IN GAME:

Here in the game it is used in the constructor of RenderWindow’s object app.

### SOUND BUFFER:

#### DEFINITION:

This class provides storage for audio samples defining a sound.

#### FUNCTIONS USED IN GAME:

1. **loadFromFile:** Loads the sound buffer from a file

### SOUND:

#### DEFINITION:

It is a lightweight object that plays loaded audio data from a sf::SoundBuffer. It should be used for small sounds that can fit in memory and should suffer no lag when they are played. Examples are gun shots, foot steps, etc.

#### FUNCTIONS USED IN GAME:

1. **setBuffer:** Set the source buffer containing the audio data to play
2. **play:** Start or resume playing the sound
3. **setLoop:** Set whether or not the sound should loop after reaching the end
4. **stop:** Set whether or not the sound should loop after reaching the end

### TEXTURE:

#### DEFINITION:

Texture class is used for Image Class living on the graphics card that can be used for drawing.

#### FUNCTIONS USED IN GAME:

**LoadFromFile**: loads the image form the file.

### SPRITE:

#### DEFINITION:

Sprite is class which can make drawable representation of a texture, with its own transformations, color,etc.

#### FUNCTIONS USED IN GAME:

**Sprite constructor:** takes in the object of texture class as an argument.

### FONT:

#### DEFINITION:

This class is used for loading and manipulating character fonts.

#### FUNCTIONS USED IN GAME:

**LoadFromFile:** this function is used her to load arial font from the the files.

### TEXT:

#### DEFINITION:

This class of text is used for graphical text that can be drawn to render target

#### FUNCTIONS USED IN GAME:

1. **setCharacterSize:** to set the character size.
2. **setPosition:** to set the position of the of the object.
3. **setFont:** to set the text’s font.
4. **setColor:** to set the global color of the text.
5. **setString:** to set the text’s string.

### EVENT:

#### DEFINITION:

Event is a union, which means that only one of its members is valid at a time The valid member is the one that matches the event type

#### FUNCTIONS USED IN GAME:

**Type**: to note about the type of event.

### OSTRINGSTREAM:

#### DEFINITION:

This is an output stream class to operate on strings.

#### FUNCTIONS USED IN GAME:

1. **Str():** takes a string object, whose content is copied .

## CODE:

#include<SFML/Graphics.hpp>

#include<SFML/Audio.hpp>

#include<iostream>

#include<sstream>

#include<time.h>

using namespace sf;

using namespace std;

class Doodle{

private:

struct Point{int x, y;};

int score;

public:

Doodle(){

score = 0;

}

void Game(){

srand(time(0));

RenderWindow app(VideoMode (1000,667), "Doodle Game");

app.setFramerateLimit(60);

SoundBuffer b;

if (!b.loadFromFile("canary.wav")){

cout<<"Sound Error"<<endl;

}

Sound s;

s.setBuffer(b);

s.play();

s.setLoop(true);

Texture t1, t2, t3;

t1.loadFromFile("images/background.jpg");

t2.loadFromFile("images/platform (3).png");

t3.loadFromFile("images/doodle.png");

Sprite sBackground(t1), sPlat(t2), sPers(t3);

Point Plat[10];

for(int i=0; i<10; i++){

Plat[i].x = rand()% 950;

Plat[i].y = rand()% 600;

}

int x=100, y=100, h=200;

float dx=0, dy=0;

while(app.isOpen()){

Event e;

while(app.pollEvent(e)){

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

app.close();

}

if(Keyboard::isKeyPressed(Keyboard::Right))

x+=4;

if(Keyboard::isKeyPressed(Keyboard::Left))

x-=4;

dy+=0.2;

y+=dy;

if(y>600){

dy=-10;

}

if(y<h){

for(int i=0; i<10; i++){

y=h;

Plat[i].y =Plat[i].y-dy;

if(Plat[i].y>600){

Plat[i].y =0;

Plat[i].x = rand()%950;

}

}

}

for (int i=0; i<10; i++){

if((x+50>Plat[i].x) && (x+20<Plat[i].x+68) && (y+70>Plat[i].y) && (y+70<Plat[i].y+14) && (dy>0))

{

dy=-10;

score++;

}

}

sPers.setPosition(x,y);

app.draw(sBackground);

app.draw(sPers);

for(int i=0; i<10; i++){

sPlat.setPosition(Plat[i].x, Plat[i].y);

app.draw(sPlat);

}

Font arial;

arial.loadFromFile("arial.ttf");

ostringstream ssScore;

ssScore<<"Score: "<<score;

Text lblScore;

lblScore.setCharacterSize(30);

lblScore.setPosition(10,10);

lblScore.setFont(arial);

lblScore.setColor(Color::Blue);

lblScore.setString(ssScore.str());

app.draw(lblScore);

app.display();

}

s.stop();

}

};

int main(){

Doodle d;

d.Game();

return 0;

}

## OUTPUT:

