### ЗМІСТ

### 

СПЕЦИФІКАЦІЯ ПРОЕКТУ……………………………………………….……6

1. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ…………..….……7
   1. Загальні вимоги до програми…………………………………………….7
   2. Призначення та область застосування………………………….…….…9
   3. Функціональні вимоги……………………………………………………9
2. ОПИС ПРОГРАМИ…………………………………………………….……13
   1. Структура програми………………………………………………..……13
   2. Опис методів програми…………………………………………….……17
   3. Програмні засоби………………………………………………...………18
   4. Опис користувацького інтерфейсу………………………………..……18

ВИСНОВКИ……………………………………………………………...………22

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………....….……23

ДОДАТКИ………………………………………………………………..………24

Додаток А. Скролінг (текст) програми………………………………….….24

# 

# 

# СПЕЦИФІКАЦІЯ ПРОЕКТУ

# 1. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ

## 1.1 Загальні вимоги до програми

##### 1.1.1 Вимоги до графічного інтерфейсу користувача

До основних вимог інтерфейсу користувача можна віднести:

а) Робоча мова інтерфейсу – англійська.

б) Кнопок керування повинно бути сім:

1. вихід з гри;
2. повернення до головного меню(4);
3. початок гри;
4. правила гри;

в) Кольорова гама (палітра) за замовчуванням:

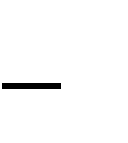
1. фон – картинка;
2. кольори фігур – див. Табл.1;
3. кнопки керування у вигляді тексту повинні бути білого кольору, якщо навести на них курсор мишки то їхній колір стає синім.

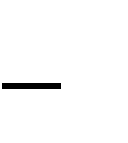
е) У програмі необхідно використати шрифт Arial Narrow та Arial,

напівжирного накреслень та курсив.

##### 1.1.2 Вимоги до архітектури програми

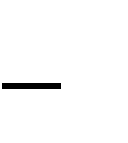
Використання стандартних операторів if, які реалізують:

 вибір режимів роботи в програмі;

 завантаження нової гри.

Використання стандартних класів, які реалізують:

завантаження шрифтів;



 завантаження текстур;

 завантаження аудіо файлів;

взаємодію між елементами програми.

##### 1.1.3 Вимоги до функціональності додатка

Основними вимогами до функціональності додатка є:

– Початок роботи програми з відображенням функціоналу,який відповідатиме вибраному користувачем режиму роботи.

– Генерація чергової фігури, яка з’являтиметься у потрібній частині ігрового поля.

– Виконання перевірки позиції фігури і досягнення границь поля.

– Очікування керуючих дій від користувача.

– При натисненні користувачем однієї з кнопок («finish», «start», «info», «back to menu» або «exit»), буде завершуватись гра, починатись,на екран виведуться правила гри, гравець повернеться в початкове меню, гра закриється.

– Після кліку на екран мишкою програма опрацює координати точки і побудує траєкторію польоту снаряду а також обчислює потрібний кут повороту пушки.

– Генерація наступної фігури, залежить від точки з координатами x,y на екрані та ходу гравця.

– Продовження гри до тих пір, допоки гравець не завершить гру або хтось не виграє.

– Відтворення мелодій , яка завантажується з файлу music формату оgg в потрібний момент.

– Проектування для розширення екрану full hd без можливості переходу до інших розширень.

– Вибір зовнішньго вигляду снаряду при натисненні на відповідні клавіші на клавіатурі (z,x,c,v);

– Показ стану життя для кожної з пушок в данний момент.

– Отримання та відображення підсумкової інформації по завершенню гри.

#### 1.2 Призначення та область застосування

***Мета роботи*** полягає у розробці розважaльної гри «Балістична гра» для комп'ютерів, які працюють під керуванням ОС Windows.

Реалізована версія гри «Балістична гра» призначена для розважання користувачів комп'ютерів, що працюють під управлінням ОС Windows, при проведенні дозвілля, тренуванні просторової уяви, та прогнозування своїх дій та їх наслідків.

***Область застосування*** – програмне забезпечення для недорогих комп'ютерів з обмеженими обчислювальними ресурсами.

## 1.3 Функціональні вимоги

До програмного забезпечення висуваються такі функціональні вимоги:

– Фігури, які повинна забезпечувати гра:

Таблиця 1 – Параметри фігур гри

|  |  |
| --- | --- |
| Вигляд фігури | Опис фігури |
|  | Гармата «1» |
|  | Гармата «2» |
|  | Снаряд «1» |
|  | Снаряд «2» |
|  | Снаряд «3» |
|  | Снаряд «4» |

– Поле гри – прямокутник висотою 1920 пікселів, шириною 1080 пікселів.

– Якщо довільний снаряд вилетить з ігрового поля за конкретні межі то він видалиться і хід завершиться.

– Мета гри – виграти в комп’ютера.

– Гра закінчується, коли у гравця смужка життя впаде до нуля або в комп’ютера.

– Гра повинна забезпечувати можливість продовження гри далі або її повне завершення з виходом до операційної системи.

– Функціональні вимоги до режиму гравця:

1) Гра повинна надавати можливість обертати пушку курсором мишки.

2) Напрям обертання та кут нормується.

3) Кут обертання – в межах від -40 до 30 градусів за один хід.

4) Пушка не повинна пересуватися.

5) Гра повинна забезпечувати можливість призупинення процесу гри без подальшим продовженням її з того самого місця (пауза).

# 2. ОПИС ПРОГРАМИ

## 2.1 Структура програми

##### 2.1.1 Модулі програми

Робота розробленого програмного забезпечення реалізується наступними модулями:

– RenderWindow – клас, в якому реалізовано управління вікном гри.

* Music – клас, в якому реалізовано роботу з звуками.
* Image – клас, в якому реалізовано роботу з картинками.
* Texture – клас, в якому реалізовано роботу з текстурами.
* Sprite – клас, в якому реалізовано створення об’єктів у вікні.

– RectangleShape клас, в якому реалізовано роботу з правильними геометричними фігурами.

* CircleShape клас, в якому реалізовано роботу з прямокутниками.
* Mouse – клас, в якому реалізовано роботу з мишкою.

– Keyboard – клас, в якому реалізовано роботу з клавішами на клавіатурі.

* Bullet – клас, в якому реалізовано характеристика снаряду.
* Button – клас, в якому реалізовано роботу з кнопками на екрані.
* Cannon – клас, в якому реалізовано робота з гарматами.

##### 2.1.2 Алгоритми роботи програми

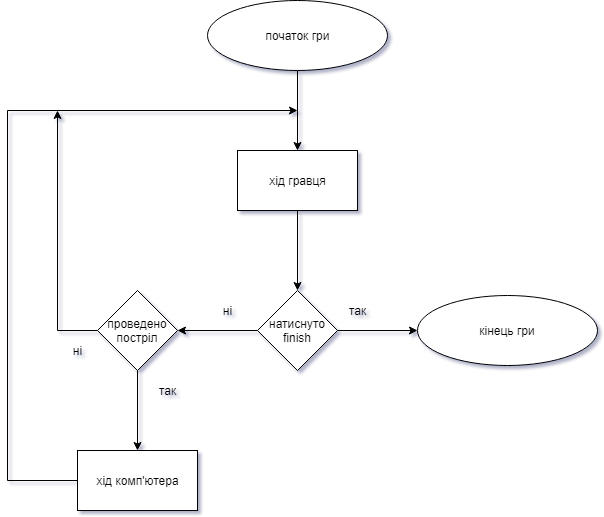


Рисунок 1 – Узагальнена схема роботи гри

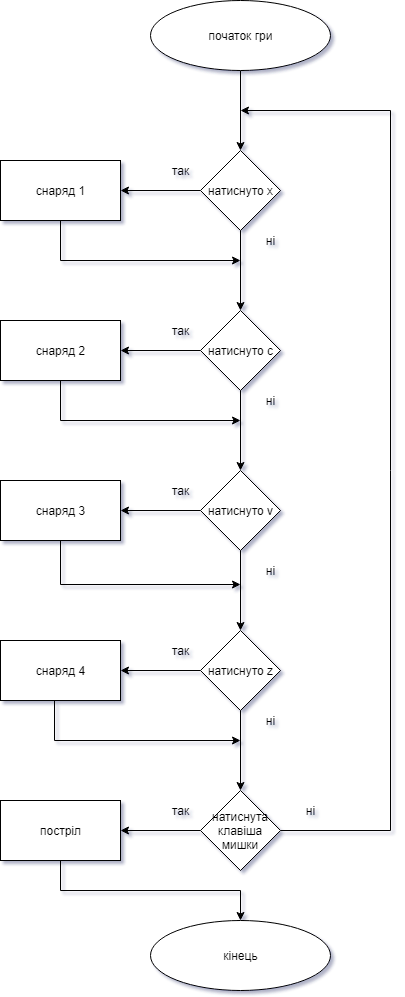


Рисунок 2 – Блок-схема режиму гравця

## 2.2 Опис методів програми

Список методів класу RenderWindow та їх опис наведено в табл.2.

Таблиця 2 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | clear() | Очищує вікно |
| 2. | display() | Виводить на екран об’єкти |
| 3. | draw() | Малює об’єкти у вікні |

Список методів класу Music та їх опис наведено в табл.3.

Таблиця 3 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | loadFromFile() | Загрузити звук з файлу |
| 2. | play() | Включає звук |
| 3. | setVolume() | Встановлює гучність |
| 4. | stop() | Зупиняє відтворення звуку |

Список методів класу Image та їх опис наведено в табл.4.

Таблиця 4 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | loadFromFile() | Загрузити картинку з файлу |
| 2. | setPosition() | Встановити координати картинки |
| 3. | getposition() | Отримання координат картинки |
| 4. | setScale() | Масштабування |

Список методів класу Texture та їх опис наведено в табл.5.

Таблиця 5 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | loadFromFile() | Загрузити картинку з файлу |
| 2. | loadFromImage() | Взяти загружену картинку |
| 3. | setTexture() | Встановити текстуру |

Список методів класу Sprite та їх опис наведено в табл.6.

Таблиця 6 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | setRoration() | Поворот на певний кут |
| 2. | setPosition() | Встановити координати спрайта |
| 3. | setScale() | Встановити масштабування |

Список методів класу RectangleShape та їх опис наведено в табл.7.

Таблиця 7 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | setRoration() | Встановити кут до горизонту |
| 2. | setPosition() | Встановити координати |
| 3. | setSize() | Встановлює розмір |
| 4. | getPosition() | Віддає координати |
| 5. | setFillColor() | Залити кольором |

Список методів класу CircleShape та їх опис наведено в табл.8.

Таблиця 8 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | setPosition() | Встановити координати |
| 2. | setFillColor() | Залити кольором |
| 3. | setRadius | Встановлює радіус |
| 4. | setRotate() | Обертання із заданною швидкістю |

Список методів класу Mouse та їх опис наведено в табл.9.

Таблиця 9 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | isButtonPressed() | Перевірка натискання кнопки мишки |
| 2. | getPosition() | Віддає координати |

Список методів класу Keyboard та їх опис наведено в табл.10.

Таблиця 10 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | isButtonPressed() | Перевірка натискання кнопки на клавіатурі |

Список методів класу Bullet та їх опис наведено в табл.11.

Таблиця 11 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | setPosition() | Встановити координати |
| 2. | getPosition() | Віддає координати |
| 3. | setTexture | Встановити текстуру |
| 4. | setSize() | Встановлює розмір ядра |

Список методів класу Button та їх опис наведено в табл.12.

Таблиця 12 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | setPosition() | Встановити координати кнопки |
| 2. | getPosition() | Віддає координати кнопки |
| 3. | setText | Встановити текст |
| 4. | setSize() | Встановлює розмір кнопки |
| 5. | isPressed() | Робить кнопку нажатою |

Список методів класу Cannon та їх опис наведено в табл.13.

Таблиця 13 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | setPosition() | Встановити координати пушки |
| 2. | getPosition() | Віддає координати пушки |
| 4. | setRotate() | Обертання із заданною швидкістю |
| 4. | setSize() | Встановлює розмір пушки |
| 5. | setOrange() | Задається вісь обертання пушки |
| 6. | setRoration() | Встановити кут до горизонту пушки |

## 2.3 Програмні засоби

Розробка реалізована засобами середовища Visual Studio на мові C++. Дане середовище є зручним у використанні для швидкого та якісного створення додатків на ОС Windows.

Windows забезпечує повний набір програмного забезпечення для комп’ютерних пристроїв: операційну систему, сполучні програми і ключові додатки.

Windows – це відкрита операційна система, яка була створена «з нуля», що дозволяє розробникам створювати програми, які можуть повною мірою скористатись функціями комп’ютерного пристрою.

## 2.4 Опис користувацького інтерфейсу

При запуску гри, гравець має можливість натиснути на такі кнопками: «Start», «Info», «Exit». Після натискання на «Start», починається гра.

Якщо натиснути «Info» то можна переглянути інформацію про гру і повернутися назад натиснувши кнопку «Back to menu».

Кнопка «Exit» відповідає за закриття програми.

Після виграшу або програшу гри появляється відповідне вікно з повідомленням на кнопка «Back to menu» клацнувши по якій можна повернутися до головного меню.

Розглянемо дії користувача при роботі з програмою. Користувач запускає гру. На екрані комп’ютера з’являється головне вікно програми.

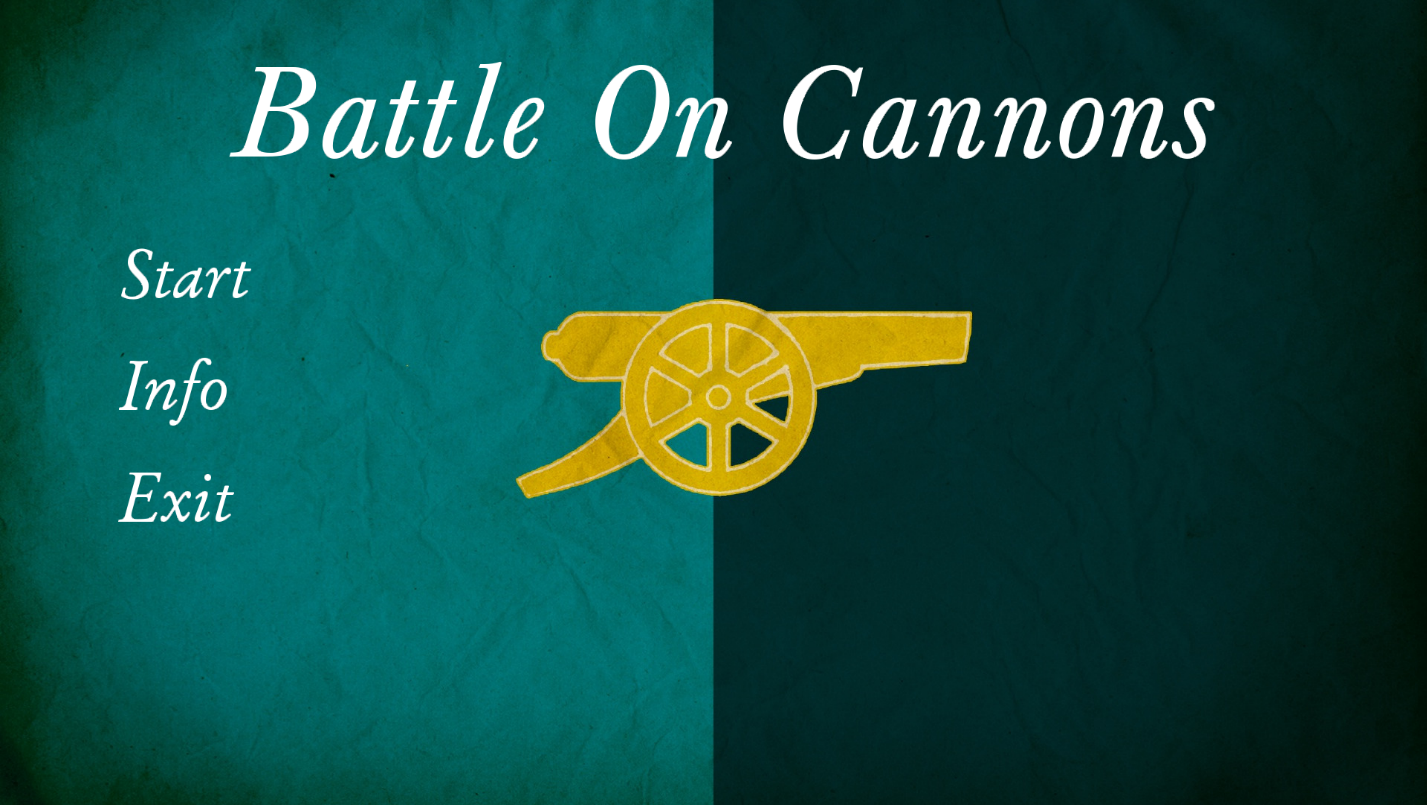


Рисунок 5 – Головне вікно програми

Тут ми бачимо три основні кнопки, три а саме:

* Start (старт) – кнопка для початку гри.
* Info (інформація) – кнопка дляперегляду режиму гри.
* Exit (вихід) – кнопка для тяжкого режиму гри.

При натисненні на кнопку «Info», нам відкривається таке вікно.



Рисунок 6 – Вікно з інформацією

В даному вікні користувач може ознайомитись з правилами гри.

При натисненні на кнопки «Start» нам відкривається вікно самої гри.



Рисунок 7 – Графічне відображення гри

В даному вікніклацаючи на екрані мишкою, користувач сміло може грати.

При закінченні гри вводимо результат (рис.8). При натисненні на кнопку «Back to menu», повертаємось в головне меню гри.



Рисунок 8 – Графічне відображення виграшу

Меню програшу даної гри представлена на рисунку 9.



Рисунок 9 – Графічне відображення програшу

При натисненні на кнопку «Back to menu», користувач повертається на головне вікно (див. рис.5).

# ВИСНОВКИ

В результаті виконання курсового проекту розроблена гра «Балістична гра» під OC Windows.

Розроблене програмне забезпечення має наступні можливості:

– Рухати пушку вверх-низ.

– Регулювати політ.

– Можливість вибрати зовнішній вигляд снаряду.

– Грати з комп’ютером.

– Переглядати інформацію по грі.

– Дізнатись результат.

В даній роботі використовуються Sfml – це проста та швидка мультимедійна бібліотека.Sfml є портативним і легким у використанні API для мультимедіа-програмування. Написано мовою C++, але доступні прив'язки і для C, Python, Ruby, OCaml, .Net. SFML забезпечує використання 2D графіки з апаратним прискорення OpenGL. SFML також може бути використаний для OpenGL-проектів. SFML надає різні модулі для

полегшення програмування ігор і мультимедійних додатків, що значно

спрощує розробку.

Дана розробка у майбутньому може бути розширена із добавленням нового функціоналу і видозміненою логікою обробки.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цехнер Марио. Програмирование игр под Windows. – СПб.: Москва, 2011. – 656с.

2. Дейтел П., Дейтел Х..Windows для раработчиков. – СПб.: Москва,2013. – 189 с.

3. Genymotion [Електронний ресурс] : [Веб-сайт]. – Електронні дані. –

Genymotion © 2016. – Режим доступу:

https://www.sfml-dev.org/tutorials/2.5/start-vc.php (дата звернення 15.04.2018)

https://www.visualstudio.com/ru/?rr=https%3A%2F%2Fwww.google.com.ua%2Fдата звернення (15.04.2018)

# ДОДАТКИ

## Додаток А. Скролінг (текст) програми

#include <SFML/Graphics.hpp>

#include <SFML/Audio.hpp>

#include <iostream>

#include <cstring>

#include <cstdlib>

#include <windows.h>

#include <time.h>

#include <math.h>

#include <conio.h>

using namespace sf;

using namespace std;

class Button

{

private:

Text text;

int size;

Font font;

bool fl;

string txt;

Vector2f position;

public:

Button(string txt,Font font, unsigned int size)

{

this->txt = txt;

this->size = size;

this->font = font;

text.setString(txt);

text.setFont(font);

text.setCharacterSize(size);

}

void ButtonOn()

{

this->fl = true;

}

void ButtonOff()

{

this->fl = false;

}

bool GetButton()

{

return fl;

}

void SetFont(Font font)

{

text.setFont(font);

}

void SetPosition(Vector2f pos)

{

this->position = pos;

text.setPosition(pos);

}

void SetPosition(float x,float y)

{

this->position.x = x;

this->position.y = y;

text.setPosition(x,y);

}

void SetSize(unsigned int size)

{

text.setCharacterSize(size);

}

int GetSize()

{

return text.getCharacterSize();

}

Vector2f GetPosition()

{

return text.getPosition();

}

void SetColor(int red, int green, int blue)

{

text.setFillColor(Color::Color(red,green,blue));

}

void SetText(string txt)

{

this->txt = txt;

}

string GetText()

{

return txt;

}

};

class Cannon

{

private:

Sprite cannon;

protected:

Texture texture;

float angle;

Vector2f position;

public:

Cannon()

{

angle = 0;

position.x = 0;

position.y = 0;

}

void LoadTextureFromFile(string road)

{

texture.loadFromFile(road);

}

virtual void SetTexture()

{

cannon.setTexture(texture);

}

virtual void SetAngle(float angle)

{

this->angle = angle;

cannon.setRotation(angle);

}

virtual void SetPosition(Vector2f position)

{

this->position = position;

cannon.setPosition(position);

}

virtual void SetPosition(float x,float y)

{

this->position.x = x;

this->position.y = y;

cannon.setPosition(x,y);

}

float GetAngle()

{

return angle;

}

Vector2f GetPosition()

{

return position;

}

};

class Bullet :public Cannon

{

private:

CircleShape bullet;

float radius;

public:

Bullet()

{

bullet.setPosition(0, 0);

radius = 1;

}

void SetRadius(float R)

{

radius = R;

bullet.setRadius(R);

}

void SetTexture(Texture &tex)

{

bullet.setTexture(&tex);

}

void SetAngle(float angle) override

{

this->angle = angle;

bullet.setRotation(angle);

}

void SetPosition(Vector2f position) override

{

this->position = position;

bullet.setPosition(position);

}

void SetPosition(float x, float y) override

{

this->position.x = x;

this->position.y = y;

bullet.setPosition(x, y);

}

};

float volume\_music\_menu = 0;

float bullet\_place\_stop = 0;

//--------------------------------------------------------------------------

int main()

{

srand(time(NULL));

RenderWindow window(VideoMode(1000, 800), "My game!", Style::Fullscreen);

Texture menu\_texture;

menu\_texture.loadFromFile("images/menu.jpg");

Sprite menu;

menu.setTexture(menu\_texture);

Music music\_menu,music\_menu\_game\_over, music\_menu\_game\_win,music\_shoot,music\_damage, music\_damage2;

music\_menu.openFromFile("music/menu.ogg");

music\_menu\_game\_win.openFromFile("music/game\_win.ogg");

music\_menu\_game\_over.openFromFile("music/game\_over.ogg");

music\_shoot.openFromFile("music/shoot.ogg");

music\_damage.openFromFile("music/damage.ogg");

music\_damage2.openFromFile("music/damage2.ogg");

music\_menu.play();

music\_menu.setVolume(volume\_music\_menu);

music\_menu.setLoop(true);

Font font\_game,font\_title;

font\_game.loadFromFile("font/game.ttf");

font\_title.loadFromFile("font/title.ttf");

Text Button\_exit("Exit", font\_game, 100);

Button\_exit.setPosition(160, 600);

Text Button\_info("Info", font\_game, 100);

Button\_info.setPosition(160, 450);

Text Button\_start("Start", font\_game, 100);

Button\_start.setPosition(160, 300);

Text Title("Battle On Cannons ", font\_title, 180);

Title.setPosition(310, 30);

Button exit("exit", font\_game, 10);

Button info("info", font\_game, 10);

Button start("start", font\_game, 10);

Button back("back to menu", font\_game, 10);

Button finish("finish", font\_game, 10);

float num\_button\_menu = 0;

while (window.isOpen())

{

music\_menu.setVolume(volume\_music\_menu);

Button\_start.setFillColor(Color::White);

Button\_info.setFillColor(Color::White);

Button\_exit.setFillColor(Color::White);

if (IntRect(160, 335, 180, 70).contains(Mouse::getPosition(window)))

{

Button\_start.setFillColor(Color::Blue);

num\_button\_menu = 1;

}

if (IntRect(160, 490, 150, 70).contains(Mouse::getPosition(window)))

{

num\_button\_menu = 2;

Button\_info.setFillColor(Color::Blue);

}

if (IntRect(160, 635, 160, 70).contains(Mouse::getPosition(window)))

{

num\_button\_menu = 3;

Button\_exit.setFillColor(Color::Blue);

}

if (Mouse::isButtonPressed(Mouse::Left) && num\_button\_menu == 1)

{

music\_menu.pause();

volume\_music\_menu = 0;

Image cannon\_image, cannon2\_image, heart\_image;

cannon2\_image.loadFromFile("images/cannon2.jpg");

cannon\_image.loadFromFile("images/cannon.jpg");

heart\_image.loadFromFile("images/heart.jpg");

cannon\_image.createMaskFromColor(Color(255, 255, 255));

cannon2\_image.createMaskFromColor(Color(255, 255, 255));

heart\_image.createMaskFromColor(Color(0, 0, 0));

Texture map\_tex, rust\_tex, old\_metal\_tex, stone\_tex, metal\_tex, cannon\_tex, cannon2\_tex, black\_tex, heart\_tex;

map\_tex.loadFromFile("images/map.jpg");

rust\_tex.loadFromFile("images/rust.jpg");

old\_metal\_tex.loadFromFile("images/old\_metal.jpg");

stone\_tex.loadFromFile("images/stone.jpg");

metal\_tex.loadFromFile("images/metal.jpg");

black\_tex.loadFromFile("images/black.jpg");

cannon\_tex.loadFromImage(cannon\_image);

cannon2\_tex.loadFromImage(cannon2\_image);

heart\_tex.loadFromImage(heart\_image);

Sprite map,cannon,cannon2,heart2,heart;

heart2.setTexture(heart\_tex);

heart2.scale(Vector2f(0.05f, 0.05f));

heart.setTexture(heart\_tex);

heart.scale(Vector2f(0.05f, 0.05f));

heart.setPosition(20, 20);

heart2.setPosition(1860, 20);

map.setTexture(map\_tex);

cannon.setTexture(cannon\_tex);

cannon.setPosition(1820, 785);

cannon.scale(Vector2f(0.1f, 0.1f));

cannon2.setTexture(cannon2\_tex);

cannon2.setPosition(100, 790);

cannon2.scale(Vector2f(0.1f, 0.1f));

Text Button\_finish("Finish", font\_title, 60);

Button\_finish.setPosition(880, 900);

Text z("z", font\_game, 40);

Text x("x", font\_game, 40);

Text c("c", font\_game, 40);

Text v("v", font\_game, 40);

z.setPosition(795, 955);

x.setPosition(887, 955);

c.setPosition(988, 955);

v.setPosition(1077, 955);

z.setFillColor(Color::Yellow);

x.setFillColor(Color::Yellow);

c.setFillColor(Color::Yellow);

v.setFillColor(Color::Yellow);

RectangleShape panel(Vector2f(400, 150));

RectangleShape hp(Vector2f(250, 20));

RectangleShape hp2(Vector2f(250, 20));

hp.setFillColor(Color::Color(255, 69, 0));

hp.setPosition(1603,25);

hp2.setFillColor(Color::Color(255, 69, 0));

hp2.setPosition(65, 25);

panel.setFillColor(Color::Color(105, 105, 105));

panel.setOutlineThickness(10);

panel.setOutlineColor(Color::Color(250, 150, 100));

panel.setPosition(750, 900);

CircleShape bullet\_rust(15, 500);

CircleShape bullet\_old\_metal(15, 500);

CircleShape bullet\_stone(15, 500);

CircleShape bullet\_metal(15, 500);

CircleShape bullet(10, 500);

bullet\_rust.setPosition(790, 1010);

bullet\_old\_metal.setPosition(880, 1010);

bullet\_stone.setPosition(980, 1010);

bullet\_metal.setPosition(1070, 1010);

bullet.setPosition(-50, 0);

bullet\_rust.setTexture(&rust\_tex);

bullet\_old\_metal.setTexture(&old\_metal\_tex);

bullet\_stone.setTexture(&stone\_tex);

bullet\_metal.setTexture(&metal\_tex);

Bullet Bullet;

Cannon cannon\_1, cannon\_2;

bullet.setTexture(&black\_tex);

bool finish = false;

bool step = false;

bool damage = false;

bool shoot = true;

bool bullet\_stop = false;

double MouseX, MouseY, angle, distance;

Sleep(100);

do

{

shoot = true;//щоб після натиснення фініш не було ще одного вистрілу

step = false;

music\_shoot.stop();

do

{

//хід гравця

cannon.setOrigin(790, 678);//вісь обертання 1 пушки

shoot = true;

Vector2f hp2\_size;

hp2\_size = hp2.getSize(); //полоса здоров'я

Vector2f hp\_size;

hp\_size = hp.getSize();

Vector2f hp\_position;

hp\_position = hp.getPosition();

bullet.setPosition(-100, 0);

if (hp2\_size.x <= 0) //програш

{

music\_damage2.stop();

music\_menu\_game\_over.play();

bool back\_menu = false;

finish = true;

step = true;

window.clear();

Texture menu\_game\_over\_tex;

menu\_game\_over\_tex.loadFromFile("images/menu\_game\_over.jpg");

Sprite menu\_game\_over;

menu\_game\_over.setTexture(menu\_game\_over\_tex);

Text Button\_back\_menu("Back to menu", font\_game, 100);

Text game\_over("GAME OVER !!!", font\_game, 150);

Button\_back\_menu.setPosition(700, 800);

game\_over.setPosition(495, 250);

game\_over.setFillColor(Color::Red);

while (back\_menu == false)

{

Button\_back\_menu.setFillColor(Color::Red);

num\_button\_menu = 0;

if (IntRect(700, 840, 510, 65).contains(Mouse::getPosition(window)))

{

num\_button\_menu = 6;

Button\_back\_menu.setFillColor(Color::Blue);

}

if (Mouse::isButtonPressed(Mouse::Left) && num\_button\_menu == 6)

{

back\_menu = true;

music\_menu\_game\_over.stop();

}

Sleep(20);

window.clear();

window.draw(menu\_game\_over);

window.draw(game\_over);

window.draw(Button\_back\_menu);

window.display();

}

}

if (hp\_size.x <= 0) //виграш

{

music\_damage.stop();

music\_menu\_game\_win.play();

bool back\_menu = false;

finish = true;

step = true;

window.clear();

Texture menu\_game\_over\_tex;

menu\_game\_over\_tex.loadFromFile("images/menu\_game\_over.jpg");

Sprite menu\_game\_over;

menu\_game\_over.setTexture(menu\_game\_over\_tex);

Text Button\_back\_menu("Back to menu", font\_game, 100);

Text game\_over("YOU WIN !!!", font\_game, 150);

Button\_back\_menu.setPosition(700, 800);

game\_over.setPosition(610, 250);

game\_over.setFillColor(Color::Yellow);

while (back\_menu == false)

{

Button\_back\_menu.setFillColor(Color::Yellow);

num\_button\_menu = 0;

if (IntRect(700, 840, 510, 65).contains(Mouse::getPosition(window)))

{

num\_button\_menu = 7;

Button\_back\_menu.setFillColor(Color::Blue);

}

if (Mouse::isButtonPressed(Mouse::Left) && num\_button\_menu == 7)

{

back\_menu = true;

music\_menu\_game\_win.stop();

}

Sleep(20);

window.clear();

window.draw(menu\_game\_over);

window.draw(game\_over);

window.draw(Button\_back\_menu);

window.display();

}

}

if (hp\_size.x > 0 && hp2\_size.x > 0)

{

Vector2f pos\_cannon2 = cannon2.getPosition();

Vector2i pos\_mouse = Mouse::getPosition(window);//забираем коорд курсора

Vector2f pos\_mouse\_in\_game = window.mapPixelToCoords(pos\_mouse);//переводим их в игровые (уходим от коорд окна)

MouseX = pos\_mouse\_in\_game.x - pos\_cannon2.x;

MouseY = pos\_mouse\_in\_game.y - pos\_cannon2.y;

cannon2.setOrigin(550, 680); //центр обертання пушки

angle = (atan2(MouseY, MouseX) \* 180 / 3.14159265) + 35;

if (angle > 35)

angle = 35;

if (angle < -40)

angle = -40;

cannon2.setRotation(angle);

distance = sqrt(MouseX\*MouseX + MouseY \* MouseY);

Button\_finish.setFillColor(Color::Yellow);

num\_button\_menu = 0;

if (IntRect(885, 920, 145, 42).contains(Mouse::getPosition(window)))

{

Button\_finish.setFillColor(Color::Color(30, 144, 255));

num\_button\_menu = 5;

}

if (Mouse::isButtonPressed(Mouse::Left) && num\_button\_menu == 5) { finish = true; shoot = false; step = true; }

if (Keyboard::isKeyPressed(Keyboard::Z)) { bullet.setTexture(&rust\_tex); }

if (Keyboard::isKeyPressed(Keyboard::X)) { bullet.setTexture(&old\_metal\_tex); }

if (Keyboard::isKeyPressed(Keyboard::C)) { bullet.setTexture(&stone\_tex); }

if (Keyboard::isKeyPressed(Keyboard::V)) { bullet.setTexture(&metal\_tex); }

if (Mouse::isButtonPressed(Mouse::Left) && shoot == true) //вистріл гравця

{

damage = false;

step = true;

music\_shoot.play();

float bulletX = 80 + 60 \* cos(angle\*3.14159265 / 180);

float bulletY = 740 + 60 \* sin(angle\*3.14159265 / 180);

if (angle <= -35) { bulletX -= 25;bulletY += 10; } //корекція снаряду

if (angle < -30 && angle>-35) { bulletX -= 20; bulletY += 5; }

if (angle < -25 && angle>-30) { bulletX -= 20; bulletY += 5; }

if (angle < -20 && angle>-25) { bulletX -= 17.5;bulletY += 5; }

if (angle < -15 && angle>-20) { bulletX -= 15; }

if (angle < -10 && angle>-15) { bulletX -= 10; }

if (angle < -5 && angle>-10) { bulletX -= 10; }

if (angle < 0 && angle>-5) { bulletX -= 4;bulletY -= 5; }

if (angle < 5 && angle>0) { bulletX -= 4;bulletY -= 2; }

if (angle < 10 && angle>5) { bulletX += 2;bulletY -= 5; }

if (angle < 15 && angle>10) { bulletX += 5;bulletY -= 2; }

if (angle < 20 && angle>15) { bulletX += 5;bulletY -= 3; }

if (angle < 25 && angle>20) { bulletX += 10;bulletY -= 4; }

if (angle < 30 && angle>25) { bulletX += 13;bulletY -= 2; }

if (angle <= 35 && angle>30) { bulletX += 15; }

float t0 = (15 \* pos\_mouse\_in\_game.y - 13430) / -790;

float t = t0;

while (bulletY<790) //політ снаряду

{

if (bulletX > 1950) { bulletY = 800; }

bullet.setPosition(bulletX, bulletY);

bulletX += distance / 100;

bulletY -= t;

t -= 0.1;

if (bulletX > 1780 && bulletX < 1870 && bulletY > 740 && bulletY < 803 && damage == false)//перевірка попадання

{

music\_damage.play();

damage = true;

hp\_size.x -= 0.1\*sqrt(pow(distance / 100, 2) + pow(pos\_mouse\_in\_game.y, 2)); //система урона

hp.setSize(hp\_size);

hp\_position.x += 0.1\*sqrt(pow(distance / 100, 2) + pow(pos\_mouse\_in\_game.y, 2));

hp.setPosition(hp\_position);

}

Sleep(8);

window.clear();

window.draw(map);

window.draw(cannon);

window.draw(cannon2);

window.draw(panel);

window.draw(bullet\_rust);

window.draw(bullet\_old\_metal);

window.draw(bullet\_stone);

window.draw(bullet\_metal);

window.draw(Button\_finish);

window.draw(bullet);

window.draw(heart);

window.draw(heart2);

window.draw(hp2);

window.draw(hp);

window.draw(z);

window.draw(x);

window.draw(c);

window.draw(v);

window.display();

}

}

Sleep(15);

window.clear();

window.draw(map);

window.draw(cannon);

window.draw(cannon2);

window.draw(panel);

window.draw(bullet\_rust);

window.draw(bullet\_old\_metal);

window.draw(bullet\_stone);

window.draw(bullet\_metal);

window.draw(Button\_finish);

window.draw(bullet);

window.draw(heart);

window.draw(heart2);

window.draw(hp2);

window.draw(hp);

window.draw(z);

window.draw(x);

window.draw(c);

window.draw(v);

window.display();

}

} while (step == false);//хід гравця

Vector2f pos\_bullet = bullet.getPosition();

step = false;

music\_shoot.stop();

if (finish == false)

{

do //хід бота

{

shoot = true;

Vector2f hp2\_size;

hp2\_size = hp2.getSize(); //полоса здоров'я

Vector2f hp\_size;

hp\_size = hp.getSize();

Vector2f hp\_position;

hp\_position = hp.getPosition();

if (hp2\_size.x <= 0) //програш

{

music\_damage2.stop();

music\_menu\_game\_over.play();

bool back\_menu = false;

finish = true;

step = true;

window.clear();

Texture menu\_game\_over\_tex;

menu\_game\_over\_tex.loadFromFile("images/menu\_game\_over.jpg");

Sprite menu\_game\_over;

menu\_game\_over.setTexture(menu\_game\_over\_tex);

Text Button\_back\_menu("Back to menu", font\_game, 100);

Text game\_over("GAME OVER !!!", font\_game, 150);

Button\_back\_menu.setPosition(700, 800);

game\_over.setPosition(495, 250);

game\_over.setFillColor(Color::Red);

while (back\_menu == false)

{

Button\_back\_menu.setFillColor(Color::Red);

num\_button\_menu = 0;

if (IntRect(700, 840, 510, 65).contains(Mouse::getPosition(window)))

{

num\_button\_menu = 6;

Button\_back\_menu.setFillColor(Color::Blue);

}

if (Mouse::isButtonPressed(Mouse::Left) && num\_button\_menu == 6)

{

back\_menu = true;

music\_menu\_game\_over.stop();

}

Sleep(20);

window.clear();

window.draw(menu\_game\_over);

window.draw(game\_over);

window.draw(Button\_back\_menu);

window.display();

}

}

if (hp\_size.x <= 0) //виграш

{

music\_damage.stop();

music\_menu\_game\_win.play();

bool back\_menu = false;

finish = true;

step = true;

window.clear();

Texture menu\_game\_over\_tex;

menu\_game\_over\_tex.loadFromFile("images/menu\_game\_over.jpg");

Sprite menu\_game\_over;

menu\_game\_over.setTexture(menu\_game\_over\_tex);

Text Button\_back\_menu("Back to menu", font\_game, 100);

Text game\_over("YOU WIN !!!", font\_game, 150);

Button\_back\_menu.setPosition(700, 800);

game\_over.setPosition(610, 250);

game\_over.setFillColor(Color::Yellow);

while (back\_menu == false)

{

Button\_back\_menu.setFillColor(Color::Yellow);

num\_button\_menu = 0;

if (IntRect(700, 840, 510, 65).contains(Mouse::getPosition(window)))

{

num\_button\_menu = 7;

Button\_back\_menu.setFillColor(Color::Blue);

}

if (Mouse::isButtonPressed(Mouse::Left) && num\_button\_menu == 7)

{

back\_menu = true;

music\_menu\_game\_win.stop();

}

Sleep(20);

window.clear();

window.draw(menu\_game\_over);

window.draw(game\_over);

window.draw(Button\_back\_menu);

window.display();

}

}

if (hp\_size.x > 0 && hp2\_size.x > 0)

{

step = true;

Vector2f pos\_cannon = cannon.getPosition();

Vector2f pos\_mouse\_in\_game;

bullet.setPosition(-100, 0);

int random;//ймовірність

random = rand() % 5;

if (bullet\_stop == false)//початкова точність вистрілу

{

bullet\_stop = true;

bullet\_place\_stop = pos\_cannon.x;

}

if (random < 3) //стріляє точніше

{

float bullet\_place\_stop\_temp = 0;

do

{

pos\_mouse\_in\_game.x = rand() % 1920;

angle = rand() % 70 - 35;//(-35...35)

MouseX = MouseY = 0;

MouseX = pos\_cannon.x - pos\_mouse\_in\_game.x;

MouseY = MouseX \* tan(3.14159265 / 180 \* (angle));

pos\_mouse\_in\_game.y = pos\_cannon.y - MouseY;

float t0 = (13430 - 14 \* pos\_mouse\_in\_game.y) / 790;

float t = t0;

float bulletX = 1820 - 53.5 \* cos(angle\*3.14159265 / 180);

float bulletY = 732 + 53.5 \* sin(angle\*3.14159265 / 180);

distance = sqrt(MouseX\*MouseX + MouseY \* MouseY);

if (angle <= -30 && angle>-35) { bulletX -= 40; bulletY += 60; } //корекція снаряду

if (angle <= -25 && angle>-30) { bulletX -= 30; bulletY += 60; }

if (angle <= -20 && angle>-25) { bulletX -= 20; bulletY += 50; }

if (angle <= -15 && angle>-20) { bulletX -= 20; bulletY += 40; }

if (angle <= -10 && angle>-15) { bulletY += 23; }

if (angle <= -5 && angle>-10) { bulletY += 10; }

if (angle <= 0 && angle>-5) { bulletX -= 5; bulletY += 10; }

if (angle <= 5 && angle>0) { bulletX -= 4; bulletY -= 2; }

if (angle <= 10 && angle>5) { bulletX += 2; bulletY -= 5; }

if (angle <= 15 && angle>10) { bulletX += 5; bulletY -= 14; }

if (angle <= 20 && angle>15) { bulletX += 5; bulletY -= 24; }

if (angle <= 25 && angle>20) { bulletX += 5; bulletY -= 35; }

if (angle <= 30 && angle>25) { bulletX += 8; bulletY -= 40; }

if (angle <= 35 && angle > 30) { bulletX += 10; bulletY -= 60; }

while (bulletY<800) //політ снаряду теоретичний

{

bullet.setPosition(bulletX, bulletY);

bulletX -= distance / 100;

bulletY -= t;

t -= 0.1;

}

bullet\_place\_stop\_temp = bulletX;

} while (bullet\_place\_stop\_temp < -1\*bullet\_place\_stop || bullet\_place\_stop\_temp > bullet\_place\_stop);

if (fabs(bullet\_place\_stop\_temp) < 50)

{

bullet\_place\_stop\_temp = 200;

}

bullet.setPosition(-100, 0);

bullet\_place\_stop = fabs(bullet\_place\_stop\_temp);

pos\_mouse\_in\_game.y = pos\_cannon.y - MouseY;

distance = sqrt(MouseX\*MouseX + MouseY \* MouseY);

float t0 = (13430 - 14 \* pos\_mouse\_in\_game.y) / 790;

float t = t0;

float temp\_angle = 35;

float bulletX = 1820 - 53.5 \* cos(angle\*3.14159265 / 180);

float bulletY = 732 + 53.5 \* sin(angle\*3.14159265 / 180);

if (1)

{

if (angle <= -30 && angle>-35) { bulletX -= 40; bulletY += 60; } //корекція снаряду

if (angle <= -25 && angle>-30) { bulletX -= 30; bulletY += 60; }

if (angle <= -20 && angle>-25) { bulletX -= 20; bulletY += 50; }

if (angle <= -15 && angle>-20) { bulletX -= 20; bulletY += 40; }

if (angle <= -10 && angle>-15) { bulletY += 23; }

if (angle <= -5 && angle>-10) { bulletY += 10; }

if (angle <= 0 && angle>-5) { bulletX -= 5; bulletY += 10; }

if (angle <= 5 && angle>0) { bulletX -= 4; bulletY -= 2; }

if (angle <= 10 && angle>5) { bulletX += 2; bulletY -= 5; }

if (angle <= 15 && angle>10) { bulletX += 5; bulletY -= 14; }

if (angle <= 20 && angle>15) { bulletX += 5; bulletY -= 24; }

if (angle <= 25 && angle>20) { bulletX += 5; bulletY -= 35; }

if (angle <= 30 && angle>25) { bulletX += 8; bulletY -= 40; }

if (angle <= 35 && angle > 30) { bulletX += 10; bulletY -= 60; }

}

while (temp\_angle > angle) //повертаєм пушку бота на потрібний кут

{

temp\_angle -= 0.4;

cannon.setRotation(temp\_angle);

if (1)

{

Sleep(15);

window.clear();

window.draw(map);

window.draw(cannon);

window.draw(cannon2);

window.draw(panel);

window.draw(bullet\_rust);

window.draw(bullet\_old\_metal);

window.draw(bullet\_stone);

window.draw(bullet\_metal);

window.draw(Button\_finish);

window.draw(bullet);

window.draw(heart);

window.draw(heart2);

window.draw(hp2);

window.draw(hp);

window.draw(z);

window.draw(x);

window.draw(c);

window.draw(v);

window.display();

} //малюєм

}

music\_shoot.play();

while (bulletY<800) //політ снаряду

{

if (bulletX < 0) { bulletY = 800;bulletX = -100; }

bullet.setPosition(bulletX, bulletY);

bulletX -= distance / 100;

bulletY -= t;

t -= 0.1;

if (bulletX > 50 && bulletX < 150 && bulletY > 750 && bulletY < 810 && damage == false)//перевірка попадання

{

music\_damage2.play();

damage = true;

hp2\_size.x -= 0.1\*sqrt(pow(distance / 100, 2) + pow(pos\_mouse\_in\_game.y, 2)); //система урона

hp2.setSize(hp2\_size);

}

Sleep(8);

window.clear();

window.draw(map);

window.draw(cannon);

window.draw(cannon2);

window.draw(panel);

window.draw(bullet\_rust);

window.draw(bullet\_old\_metal);

window.draw(bullet\_stone);

window.draw(bullet\_metal);

window.draw(Button\_finish);

window.draw(bullet);

window.draw(heart);

window.draw(heart2);

window.draw(hp2);

window.draw(hp);

window.draw(z);

window.draw(x);

window.draw(c);

window.draw(v);

window.display();

}

}

else

{

bullet.setPosition(-100, 0);

pos\_mouse\_in\_game.x = rand() % 1920;

angle = rand() % 70 - 35;//(-35...35)

MouseX = MouseY = 0;

MouseX = pos\_cannon.x - pos\_mouse\_in\_game.x;

MouseY = MouseX \* tan(3.14159265 / 180 \* (angle));

pos\_mouse\_in\_game.y = pos\_cannon.y - MouseY;

float t0 = (13430 - 14 \* pos\_mouse\_in\_game.y) / 790;

float t = t0;

float temp\_angle = 35;

float bulletX = 1820 - 53.5 \* cos(angle\*3.14159265 / 180);

float bulletY = 732 + 53.5 \* sin(angle\*3.14159265 / 180);

distance = sqrt(MouseX\*MouseX + MouseY \* MouseY);

while (temp\_angle > angle) //повертаєм пушку бота на потрібний кут

{

temp\_angle -= 0.4;

cannon.setRotation(temp\_angle);

if (1)

{

Sleep(15);

window.clear();

window.draw(map);

window.draw(cannon);

window.draw(cannon2);

window.draw(panel);

window.draw(bullet\_rust);

window.draw(bullet\_old\_metal);

window.draw(bullet\_stone);

window.draw(bullet\_metal);

window.draw(Button\_finish);

window.draw(bullet);

window.draw(heart);

window.draw(heart2);

window.draw(hp2);

window.draw(hp);

window.draw(z);

window.draw(x);

window.draw(c);

window.draw(v);

window.display();

} //малюєм

}

music\_shoot.play();

if (angle <= -30 && angle>-35) { bulletX -= 40; bulletY += 60; } //корекція снаряду

if (angle <= -25 && angle>-30) { bulletX -= 30; bulletY += 60; }

if (angle <= -20 && angle>-25) { bulletX -= 20; bulletY += 50; }

if (angle <= -15 && angle>-20) { bulletX -= 20; bulletY += 40; }

if (angle <= -10 && angle>-15) { bulletY += 23; }

if (angle <= -5 && angle>-10) { bulletY += 10; }

if (angle <= 0 && angle>-5) { bulletX -= 5; bulletY += 10; }

if (angle <= 5 && angle>0) { bulletX -= 4; bulletY -= 2; }

if (angle <= 10 && angle>5) { bulletX += 2; bulletY -= 5; }

if (angle <= 15 && angle>10) { bulletX += 5; bulletY -= 14; }

if (angle <= 20 && angle>15) { bulletX += 5; bulletY -= 24; }

if (angle <= 25 && angle>20) { bulletX += 5; bulletY -= 35; }

if (angle <= 30 && angle>25) { bulletX += 8; bulletY -= 40; }

if (angle <= 35 && angle > 30) { bulletX += 10; bulletY -= 60; }

while (bulletY<800) //політ снаряду

{

if (bulletX < 0) { bulletY = 800; }

bullet.setPosition(bulletX, bulletY);

bulletX -= distance / 100;

bulletY -= t;

t -= 0.1;

if (bulletX > 50 && bulletX < 150 && bulletY > 750 && bulletY < 810 && damage == false)//перевірка попадання

{

music\_damage2.play();

damage = true;

hp2\_size.x -= 0.1\*sqrt(pow(distance / 100, 2) + pow(pos\_mouse\_in\_game.y, 2)); //система урона

hp2.setSize(hp2\_size);

}

Sleep(8);

window.clear();

window.draw(map);

window.draw(cannon);

window.draw(cannon2);

window.draw(panel);

window.draw(bullet\_rust);

window.draw(bullet\_old\_metal);

window.draw(bullet\_stone);

window.draw(bullet\_metal);

window.draw(Button\_finish);

window.draw(bullet);

window.draw(heart);

window.draw(heart2);

window.draw(hp2);

window.draw(hp);

window.draw(z);

window.draw(x);

window.draw(c);

window.draw(v);

window.display();

}

}

}

} while (step == false);//хід бота

}

} while (finish == false);

music\_menu.play();

}

if (Mouse::isButtonPressed(Mouse::Left) && num\_button\_menu == 2)

{

Sleep(100);

bool button = false;

Text Button\_info\_exit("Back to menu", font\_game, 100);

Button\_info\_exit.setPosition(700, 800);

Text Devel("Developer : Artem Morarash", font\_game, 35);

Devel.setPosition(70, 950);

Text Text\_info\_0("Welcome!", font\_game, 100);

Text\_info\_0.setPosition(780, 40);

Text Text\_info\_1("In this game you have to kill the enemy for the smallest amount of shots.", font\_game, 60);

Text\_info\_1.setPosition(70, 200);

Text Text\_info\_2("With each next shot the enemy will shoot more precisely.", font\_game, 60);

Text\_info\_2.setPosition(65, 270);

Text Text\_info\_3("Time and number of shells is unlimited.", font\_game, 60);

Text\_info\_3.setPosition(70, 340);

Text Text\_info\_4("The damage to the projectile depends on its trajectory.", font\_game, 60);

Text\_info\_4.setPosition(70, 410);

Text Text\_info\_5("The more curve the trajectory is, the less damage.", font\_game, 60);

Text\_info\_5.setPosition(70, 480);

Text Text\_info\_6("Damage does not depend on the core.", font\_game, 60);

Text\_info\_6.setPosition(70, 550);

do

{

if (volume\_music\_menu > 10000) { volume\_music\_menu = 10000; }

volume\_music\_menu += 0.13;

Button\_info\_exit.setFillColor(Color::White);

num\_button\_menu = 0;

if (IntRect(700, 840, 510, 65).contains(Mouse::getPosition(window)))

{

num\_button\_menu = 4;

Button\_info\_exit.setFillColor(Color::Blue);

}

if (Mouse::isButtonPressed(Mouse::Left) && num\_button\_menu == 4)

{

button = true;

}

Sleep(20);

window.clear();

window.draw(menu);

window.draw(Button\_info\_exit);

window.draw(Devel);

window.draw(Text\_info\_0);

window.draw(Text\_info\_1);

window.draw(Text\_info\_2);

window.draw(Text\_info\_3);

window.draw(Text\_info\_4);

window.draw(Text\_info\_5);

window.draw(Text\_info\_6);

window.display();

} while (button == false);

}

if (Mouse::isButtonPressed(Mouse::Left) && num\_button\_menu == 3) { window.close(); }

if (volume\_music\_menu > 10000) { volume\_music\_menu = 10000; }

num\_button\_menu = 0;

volume\_music\_menu += 0.13;

Sleep(10);

window.clear();

window.draw(menu);

window.draw(Title);

window.draw(Button\_start);

window.draw(Button\_info);

window.draw(Button\_exit);

window.display();

}

return 0;

}