|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data oddania: 20.12.2017 | Imię i Nazwisko:  Michał Zięba | Wydział EiT  Kierunek: Teleinformatyka | Semestr III | Grupa I2 |
| Prowadzący: dr inż. Andrzej Urbański | | | | |

**Języki Programowania Obiektowego Projekt w C++.**

**1. Założenia:**

Program realizujący grę komputerową typu "dungeon crawler" napisany przy pomocy darmowej biblioteki SFML.

Gra składa się z 5 poziomów zapisanych w postaci cyfr w plikach z rozszerzeniem .txt.

Gra rozpoczyna się od wybrania klasy postaci (mag,wojownik lub złodziej).

Aby przejść grę należy zebrać wszystkie klucze znajdujące się na danym poziomie (w postaci skrzyń), na każdym poziomie na drodze gracza stoją przeciwnicy z którymi trzeba walczyć. Po pokonaniu przeciwników i zdobyciu wszystkich kluczy, należy przejść przez drzwi, gdzie gra wczytuje następny poziom.

**2. Klasy i ich hierarchia:**

Program składa się z **11 klas**, każda w oddzielnym pliku z rozszerzeniem .horaz z funkcji main, znajdującej się w pliku **main.cpp**.

W pliku main.cpp realizowane jest wyświetlanie gry w nieskończonej pętli oraz przełączanie między poszczególnymi jej oknami.

**klasy wchodzące w skład programu:**

**gra.h** (realizuje część związaną z grą, walkę, ruchy postaci, wczytywanie poziomów)

**postac.h** (klasa abstrakcyjna, interfejs dla klas postaci gracza)

**mag.h** (dziedziczy publicznie po klasie postac)

**zlodziej.h** (dziedziczy publicznie po klasie postac)

**wojownik.h** (dziedziczy publicznie po klasie postac)

**przeciwnik.h** (klasa abstrakcyjna, interfejs dla przeciwników)

**kosciotrup.h** (dziedziczy publicznie po klasie przeciwnik)

**menu.h** (klasa, która tworzy i rysuje główne menu programu)

**opcje.h** (klasa, która tworzy i rysuje podmenu odpowiadające za wybór rozdzielczości okna)

**wyborgracza.h** (klasa, która tworzy i rysuje podmenu odpowiadające za wybór klasy postaci przy rozpoczęciu gry)

**wyborpoziomow.h** (klasa, która tworzy i rysuje podmenu odpowiadające za wybór poziomu gry)

**3. Dodatkowe pliki wchodzące w skład projektu:**

Pliki z rozszerzeniem **.png**, zawierające tekstury postaci, ścian, przeciwników itp.

Plik z rozszerzeniem **.ttf** zawierający plik czcionki Arial.

oddzielny program **pomoc.cpp**, który pomaga w tworzeniu poziomów, zapisuje je w pliku z rozszerzeniem .txt

5 Plików z rozszerzeniem **.txt**,w których zapisane są poziomy do gry.

**4. Kod źródłowy i opis działania poszczególnych klas:**

**main.cpp**

#include <SFML/Graphics.hpp>

#include <iostream>

#include <cstdlib>

#include <ctime>

#include "gra.h"

#include "menu.h"

#include "opcje.h"

#include "wyborgracza.h"

#include "wyborpoziomow.h"

using namespace std;

using namespace sf;

enum class StatusGry

{

menus,

opcje,

poziom,

gra,

wyborpostaci,

};

int main()

{

StatusGry gdziejest = StatusGry::menus;

srand( time( NULL ) );

RenderWindow okno(VideoMode(1024,768,32), "GRA", Style::Close/\*, Style::Fullscreen\*/);

okno.setVerticalSyncEnabled(true);

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

opcje settings(okno.getSize().x, okno.getSize().y);

wyborgracza nowygracz(okno.getSize().x, okno.getSize().y);

wyborpoziomow nowypoziom(okno.getSize().x, okno.getSize().y);

gra nowa;

while(okno.isOpen())

{

switch(gdziejest)

{

case StatusGry::menus:

Event e;

while(okno.pollEvent(e))

{

switch (e.type)

{

case Event::KeyReleased:

switch (e.key.code)

{

case Keyboard::Up:

menu.UP();

break;

case Keyboard::Down:

menu.DOWN();

break;

case Keyboard::Return:

switch (menu.oddajwybor())

{

case 0:

gdziejest = StatusGry::wyborpostaci;

break;

case 1:

gdziejest = StatusGry::poziom;

break;

case 2:

gdziejest = StatusGry::opcje;

break;

case 3:

okno.close();

break;

}

default:

break;

}

break;

case Event::Closed:

okno.close();

default:

break;

}

}

okno.clear();

menu.rysuj(okno);

okno.display();

break;

case StatusGry::wyborpostaci:

while(okno.pollEvent(e))

{

switch (e.type)

{

case Event::KeyReleased:

switch (e.key.code)

{

case Keyboard::Up:

nowygracz.UP();

break;

case Keyboard::Down:

nowygracz.DOWN();

break;

case Keyboard::Escape:

gdziejest = StatusGry::menus;

break;

case Keyboard::Return:

switch (nowygracz.oddajgracza())

{

case 0:

nowa.wczytajgracza(1);

nowa.wczytajpoziom(nowa.P);

gdziejest = StatusGry::gra;

break;

case 1:

nowa.wczytajgracza(2);

nowa.wczytajpoziom(nowa.P);

gdziejest = StatusGry::gra;

break;

case 2:

nowa.wczytajgracza(3);

nowa.wczytajpoziom(nowa.P);

gdziejest = StatusGry::gra;

break;

}

default:

break;

}

break;

case Event::Closed:

okno.close();

default:

break;

}

}

okno.clear();

nowygracz.rysuj(okno);

okno.display();

break;

case StatusGry::gra:

while(okno.pollEvent(e))

{

switch (e.type)

{

case Event::KeyReleased:

switch (e.key.code)

{

case Keyboard::Up:

if(nowa.czywalka == false) nowa.UP();

else nowa.walkaUP();

break;

case Keyboard::Down:

if(nowa.czywalka == false) nowa.DOWN();

else nowa.walkaDOWN();

break;

case Keyboard::Left:

if(nowa.czywalka == false) nowa.LEFT();

break;

case Keyboard::Right:

if(nowa.czywalka == false) nowa.RIGHT();

break;

case Keyboard::Escape:

gdziejest = StatusGry::menus;

break;

case Keyboard::Return:

if(nowa.czywalka == true)

{

switch (nowa.oddajwybor())

{

case 0:

nowa.walka(1);

break;

case 1:

nowa.walka(2);

break;

case 2:

nowa.walka(3);

break;

}

}

default:

break;

}

break;

case Event::Closed:

okno.close();

default:

break;

}

}

okno.clear();

if(nowa.czywalka == false)

{

nowa.rysuj(okno);

}

else nowa.rysujwalka(okno);

okno.display();

break;

case StatusGry::opcje:

while(okno.pollEvent(e))

{

switch (e.type)

{

case Event::KeyReleased:

switch (e.key.code)

{

case Keyboard::Up:

settings.UP();

break;

case Keyboard::Down:

settings.DOWN();

break;

case Keyboard::Escape:

gdziejest = StatusGry::menus;

break;

case Keyboard::Return:

switch (settings.oddajopcje())

{

case 0:

okno.create(VideoMode(1024,768,32), "GRA", Style::Close);

okno.setVerticalSyncEnabled(true);

break;

case 1:

okno.create(VideoMode(1280,720,32), "GRA", Style::Close);

okno.setVerticalSyncEnabled(true);

break;

case 2:

okno.create(VideoMode(1366,768,32), "GRA", Style::Close);

okno.setVerticalSyncEnabled(true);

break;

case 3:

okno.create(VideoMode(1920,1080,32), "GRA", Style::Close);

okno.setVerticalSyncEnabled(true);

break;

case 4:

okno.create(VideoMode(okno.getSize().x,okno.getSize().y,32), "GRA", Style::Fullscreen);

okno.setVerticalSyncEnabled(true);

break;

}

default:

break;

}

break;

case Event::Closed:

okno.close();

default:

break;

}

}

okno.clear();

settings.rysuj(okno);

okno.display();

break;

case StatusGry::poziom:

while(okno.pollEvent(e))

{

switch (e.type)

{

case Event::KeyReleased:

switch (e.key.code)

{

case Keyboard::Up:

nowypoziom.UP();

break;

case Keyboard::Down:

nowypoziom.DOWN();

break;

case Keyboard::Escape:

gdziejest = StatusGry::menus;

break;

case Keyboard::Return:

switch (nowypoziom.oddajpoziom())

{

case 0:

nowa.wczytajpoziom(1);

gdziejest = StatusGry::menus;

break;

case 1:

nowa.wczytajpoziom(2);

gdziejest = StatusGry::menus;

break;

case 2:

nowa.wczytajpoziom(3);

gdziejest = StatusGry::menus;

break;

case 3:

nowa.wczytajpoziom(4);

gdziejest = StatusGry::menus;

break;

case 4:

nowa.wczytajpoziom(5);

gdziejest = StatusGry::menus;

break;

}

default:

break;

}

break;

case Event::Closed:

okno.close();

default:

break;

}

}

okno.clear();

nowypoziom.rysuj(okno);

okno.display();

break;

default:

break;

}

}

return 0;

}

W pliku main.cpp wykorzystywana jest klasa wyliczeniowa do przełączania się między aktywnym oknem gry, tutaj również realizowany jest odczyt naciśnięć klawiszy na klawiaturze i wywoływana jest odpowiednia funkcja.

**gra.h**

#pragma once

#include <SFML/Graphics.hpp>

#include <iostream>

#include <fstream>

#include <vector>

#include <sstream>

#include "mag.h"

#include "wojownik.h"

#include "zlodziej.h"

#include "postac.h"

#include "kosciotrup.h"

#include "przeciwnik.h"

#define ataki 3

using namespace sf;

using namespace std;

class gra

{

private:

int wybor;

Font czcionka;

Text menu[ataki];

Text napis[10];

int x,y,siz,hplimit,manalimit,L,Klucze;

Texture m1,t0,t1,t2,t3,t4,t5,t6;

Sprite tlo,mur,puste,potion,drzwi,post,kosci,postskal,kosciskal,skrzynia;

vector<vector<int> > pole;

fstream plik;

postac \*gracz;

vector<kosciotrup> szkielet;

public:

int P;

bool czywalka;

gra()

{

P = 1;

czywalka = false;

gracz = new mag;

wczytajpoziom(P);

t2.loadFromFile("mag32.png");

hplimit = gracz->hp;

manalimit = gracz->mana;

siz = 32;

L = 0;

Klucze = 0;

t0.loadFromFile("white32.png");

t1.loadFromFile("mur32.png");

t3.loadFromFile("kosciotrup32.png");

t4.loadFromFile("potion32.png");

t5.loadFromFile("skrzynia32.png");

t6.loadFromFile("drzwi32.png");

m1.loadFromFile("menu.png");

mur.setTexture(t1);

tlo.setTexture(m1);

post.setTexture(t2);

puste.setTexture(t0);

kosci.setTexture(t3);

potion.setTexture(t4);

skrzynia.setTexture(t5);

drzwi.setTexture(t6);

postskal.setScale(10,10);

postskal.setPosition(0,0);

postskal.setTexture(t2);

kosciskal.setScale(10,10);

kosciskal.setPosition(400,0);

kosciskal.setTexture(t3);

if (!czcionka.loadFromFile("arial.ttf"))

{

cout << "Nie wczytano czcionki!!";

}

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

{

napis[i].setFont(czcionka);

napis[i].setColor(Color::White);

}

napis[0].setPosition(Vector2f(0,320));

napis[1].setPosition(Vector2f(0,360));

napis[2].setPosition(Vector2f(0,400));

napis[3].setPosition(0,420);

napis[3].setString("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

napis[4].setPosition(340,140);

napis[4].setColor(Color::Red);

napis[4].setString("VS");

napis[5].setPosition(400,320);

napis[6].setPosition(400,460);

napis[7].setPosition(400,500);

napis[8].setCharacterSize(24);

napis[8].setStyle(Text::Bold);

napis[8].setColor(Color::Black);

napis[9].setCharacterSize(256);

napis[9].setStyle(Text::Bold);

napis[9].setColor(Color::Red);

menu[0].setFont(czcionka);

menu[0].setColor(Color::Red);

menu[0].setString("Atak");

menu[0].setPosition(Vector2f(0, 460));

menu[1].setFont(czcionka);

menu[1].setColor(Color::White);

menu[1].setString("Obrona");

menu[1].setPosition(Vector2f(0,510));

menu[2].setFont(czcionka);

menu[2].setColor(Color::White);

menu[2].setString("Atak specjalny");

menu[2].setPosition(Vector2f(0,560));

wybor = 0;

}

~gra() {}

void wczytajgracza(int a)

{

switch(a)

{

case 1:

gracz = new mag;

t2.loadFromFile("mag32.png");

break;

case 2:

gracz = new wojownik;

t2.loadFromFile("wojownik32.png");

break;

case 3:

gracz = new zlodziej;

t2.loadFromFile("zlodziej32.png");

break;

}

post.setTexture(t2);

hplimit = gracz->hp;

manalimit = gracz->mana;

}

void rysuj(RenderWindow &okno)

{

ostringstream napisy,koniec;

napisy << "HP: " << gracz->hp << ", MANA: " << gracz->mana << ", FURIA: " << gracz->furia <<", Pokonani przeciwnicy: " << L << ", Klucze do zebrania: " << Klucze ;

koniec << "KONIEC" << endl << "GRY";

napis[8].setString(napisy.str());

napis[9].setString(koniec.str());

okno.draw(tlo);

for(int i = 0; i<x ; i++)

{

for(int j = 0 ; j<y ; j++)

{

if(pole[i][j] == 2)

{

post.setPosition(j\*siz, i\*siz);

okno.draw(post);

}

if(pole[i][j] == 3)

{

kosci.setPosition(j\*siz,i\*siz);

okno.draw(kosci);

}

if(pole[i][j] == 4)

{

potion.setPosition(j\*siz,i\*siz);

okno.draw(potion);

}

if(pole[i][j] == 5)

{

skrzynia.setPosition(j\*siz,i\*siz);

okno.draw(skrzynia);

}

if(pole[i][j] == 6)

{

drzwi.setPosition(j\*siz,i\*siz);

okno.draw(drzwi);

}

if(pole[i][j] == 1)

{

mur.setPosition(j\*siz,i\*siz);

okno.draw(mur);

}

else if(pole[i][j] == 0)

{

puste.setPosition(j\*siz,i\*siz);

okno.draw(puste);

}

}

}

okno.draw(napis[8]);

if(P == 6)

{

okno.draw(napis[9]);

}

}

void wczytajpoziom(int a)

{

P = a;

Klucze = 0;

gracz->reset();

szkielet.clear();

czywalka = false;

L = 0;

switch(a)

{

case 1:

plik.open( "level1.txt", ios::in);

break;

case 2:

plik.open( "level2.txt", ios::in);

break;

case 3:

plik.open( "level3.txt", ios::in);

break;

case 4:

plik.open( "level4.txt", ios::in);

break;

case 5:

plik.open( "level5.txt", ios::in);

break;

default:

break;

}

plik >> x;

plik >> y;

pole.resize(x);

for (int i = 0; i<x; i++)

{

pole[i].resize(y);

}

for (int i = 0; i<x; i++)

{

for(int j = 0; j<y; j++)

{

plik >> pole[i][j];

if(pole[i][j] == 3)

{

szkielet.push\_back(kosciotrup());

}

if(pole[i][j] == 5)

{

Klucze++;

}

}

}

plik.close();

}

void miksturka()

{

if(gracz->hp + 50 >= hplimit) gracz->hp = hplimit;

else gracz->hp += 50;

if(gracz->mana + 25 >= manalimit) gracz->mana = manalimit;

else gracz->mana += 25;

}

void UP()

{

if(pole[gracz->x-1][gracz->y] == 1)

{

}

else if(pole[gracz->x-1][gracz->y] == 4)

{

miksturka();

pole[gracz->x--][gracz->y] = 0;

pole[gracz->x][gracz->y] = 2;

}

else if(pole[gracz->x-1][gracz->y] == 3)

{

czywalka = true;

pole[gracz->x--][gracz->y] = 0;

pole[gracz->x][gracz->y] = 2;

}

else if(pole[gracz->x-1][gracz->y] == 6)

{

if(Klucze == 0)

{

P++;

wczytajpoziom(P);

}

}

else if(pole[gracz->x-1][gracz->y] == 5)

{

Klucze--;

pole[gracz->x--][gracz->y] = 0;

pole[gracz->x][gracz->y] = 2;

}

else

{

pole[gracz->x--][gracz->y] = 0;

pole[gracz->x][gracz->y] = 2;

}

}

void DOWN()

{

if(pole[gracz->x+1][gracz->y] == 1)

{

}

else if(pole[gracz->x+1][gracz->y] == 4)

{

miksturka();

pole[gracz->x++][gracz->y] = 0;

pole[gracz->x][gracz->y] = 2;

}

else if(pole[gracz->x+1][gracz->y] == 3)

{

czywalka = true;

pole[gracz->x++][gracz->y] = 0;

pole[gracz->x][gracz->y] = 2;

}

else if(pole[gracz->x+1][gracz->y] == 6)

{

if(Klucze == 0)

{

P++;

wczytajpoziom(P);

}

}

else if(pole[gracz->x+1][gracz->y] == 5)

{

Klucze--;

pole[gracz->x++][gracz->y] = 0;

pole[gracz->x][gracz->y] = 2;

}

else

{

pole[gracz->x++][gracz->y] = 0;

pole[gracz->x][gracz->y] = 2;

}

}

void LEFT()

{

if(pole[gracz->x][gracz->y-1] == 1)

{

}

else if(pole[gracz->x][gracz->y-1] == 4)

{

miksturka();

pole[gracz->x][gracz->y--] = 0;

pole[gracz->x][gracz->y] = 2;

}

else if(pole[gracz->x][gracz->y-1] == 3)

{

czywalka = true;

pole[gracz->x][gracz->y--] = 0;

pole[gracz->x][gracz->y] = 2;

}

else if(pole[gracz->x][gracz->y-1] == 6)

{

if(Klucze == 0)

{

P++;

wczytajpoziom(P);

}

}

else if(pole[gracz->x][gracz->y-1] == 5)

{

Klucze--;

pole[gracz->x][gracz->y--] = 0;

pole[gracz->x][gracz->y] = 2;

}

else

{

pole[gracz->x][gracz->y--] = 0;

pole[gracz->x][gracz->y] = 2;

}

}

void RIGHT()

{

if(pole[gracz->x][gracz->y+1] == 1)

{

}

else if(pole[gracz->x][gracz->y+1] == 4)

{

miksturka();

pole[gracz->x][gracz->y++] = 0;

pole[gracz->x][gracz->y] = 2;

}

else if(pole[gracz->x][gracz->y+1] == 3)

{

czywalka = true;

pole[gracz->x][gracz->y++] = 0;

pole[gracz->x][gracz->y] = 2;

}

else if(pole[gracz->x][gracz->y+1] == 6)

{

if(Klucze == 0)

{

P++;

wczytajpoziom(P);

}

}

else if(pole[gracz->x][gracz->y+1] == 5)

{

Klucze--;

pole[gracz->x][gracz->y++] = 0;

pole[gracz->x][gracz->y] = 2;

}

else

{

pole[gracz->x][gracz->y++] = 0;

pole[gracz->x][gracz->y] = 2;

}

}

void walkaUP()

{

if (wybor - 1 >= 0)

{

menu[wybor].setColor(Color::White);

wybor--;

menu[wybor].setColor(Color::Red);

}

}

void walkaDOWN()

{

if (wybor + 1 < ataki)

{

menu[wybor].setColor(Color::White);

wybor++;

menu[wybor].setColor(Color::Red);

}

}

int oddajwybor()

{

return wybor;

}

void walka(int a)

{

int oddaj;

if(a == 1)

{

szkielet[L].hp -= gracz->atak();

napis[6].setString(gracz->czynnosc.str());

gracz->czynnosc.str("");

}

if(a == 2)

{

int spr = gracz -> obrona();

if(spr == 0)

{

oddaj = szkielet[L].atak()/2;

}

else if(spr == 1)

{

oddaj = szkielet[L].atak()\*(rand()&1);

int hpadd = 25\*(rand()&1);

if(gracz->hp + hpadd >= hplimit) gracz->hp = hplimit;

else gracz->hp += hpadd;

}

else oddaj = szkielet[L].atak();

napis[6].setString(gracz->czynnosc.str());

gracz->czynnosc.str("");

}

if(a == 3)

{

szkielet[L].hp -= gracz->atakspecjalny();

napis[6].setString(gracz->czynnosc.str());

gracz->czynnosc.str("");

}

if(a != 2)

{

oddaj = szkielet[L].atak();

}

if(szkielet[L].hp > 0)

{

gracz->hp -= oddaj;

ostringstream atakwroga;

atakwroga <<"Przeciwnik atakuje za " << oddaj << " pkt zycia";

napis[7].setString(atakwroga.str());

}

else if(szkielet[L].hp <= 0)

{

czywalka = false;

szkielet.erase(szkielet.begin()+L);

napis[6].setString("");

napis[7].setString("");

L++;

}

if(gracz->hp <= 0)

{

czywalka = false;

wczytajpoziom(P);

napis[6].setString("");

napis[7].setString("");

}

}

void rysujwalka(RenderWindow &okno)

{

okno.draw(tlo);

okno.draw(postskal);

okno.draw(kosciskal);

ostringstream hp,mana,furia,hpenemy;

hp << "Twoje HP: " << gracz->hp;

napis[0].setString(hp.str());

mana << "MANA(Mag): " << gracz->mana;

napis[1].setString(mana.str());

furia << "FURIA(Wojownik): " << gracz->furia;

napis[2].setString(furia.str());

hpenemy << "HP przeciwnika: " << szkielet[L].hp;

napis[5].setString(hpenemy.str());

for(int i = 0 ; i < 8 ; i ++)

{

okno.draw(napis[i]);

}

for(int i = 0 ; i < ataki ; i++)

{

okno.draw(menu[i]);

}

}

};

Plansza z grą realizowana jest w postaci tablicy dwuwymiarowej przechowywanej w pojemniku typu **vector,** wczytywana jest za pomocą funkcji **wczytajpoziom().**

W pojemniku typu **vector** przechowywani są również przeciwnicy z klasy **kosciotrup**.

**Konstruktor** klasy gra ładuje tekstury z plików .png, tworzy napisy, ustawia wszystkie zmienne na pola startowe, domyślnie tworzy również postać gracza jako **mag.**

funkcja **wczytajgracza()** odpowiada za wczytywanie postaci gracza realizuje to ustawiając wskaźnik typu **postac** na odpowiednią klasę mag,wojownik,zlodziej oraz wczytuje odpowiadającą im teksturę.

funkcja **rysuj()** odpowiada za wyświetlanie w oknie planszy z grą.

funkcja **wczytajpoziom()** realizuje kilka rzeczy, na początku resetuje pozycję gracza, czyści planszę z przeciwników, ustawia okno walki na **false**, następnie wczytuje określony poziom z pliku z rozszerzeniem **.txt** oraz ustawia nowych przeciwników.

funkcja **miksturka()** zarządza ustawionymi na planszy gry miksturkami, które może zbierać gracz.

funkcje **UP(), DOWN(), LEFT(), RIGHT()** odpowiadają za ruch gracza po planszy oraz wykrywanie **kolizji** ze ścianą, przeciwnikiem lub elementem do zebrania.

funkcja **rysujwalka()** odpowiada za rysowanie okna z walką, która następuje po dojściu gracza do przeciwnika, wtedy następuje automatyczne przełączenie z **rysuj()** na **rysujwalka()**.

funkcje **walkaUP() i walkaDOWN()** odpowiadają za poruszanie się po menu z walką.

funkcja **walka()** realizuje walkę z napotkanym przeciwnikiem.

**postac.h**

#pragma once

#include <sstream>

class postac

{

public:

int x,y,hp,mana,furia;

std::ostringstream czynnosc;

virtual int atak() = 0;

virtual int obrona() = 0;

virtual int atakspecjalny() = 0;

virtual void reset() = 0;

};

Jest to **klasa abstrakcyjna**, zawierająca **funkcje wirtualne**, służy jako **interfejs** dla klas postaci.

**mag.h**

#pragma once

#include "postac.h"

#include <iostream>

class mag : public postac

{

private:

int hplimit;

public:

int atak()

{

int liczba = (rand() % 10 ) + 10;

int manadd = (rand() % 10 ) + 10;

czynnosc << "Uderzasz przeciwnika kosturem za " << liczba << " pkt zycia";

if((mana + manadd) >= 100) mana = 100;

else mana += manadd;

return liczba;

}

int obrona()

{

int liczba = (rand() % 20 ) + 20;

if((mana - 25) < 0) czynnosc << "Nie masz dosc many, by sie uleczyc";

else

{

mana -= 25;

czynnosc << "Leczysz sie za " << liczba << " pkt zycia";

if(hp + liczba >= hplimit) hp = hplimit;

else hp += liczba;

}

return liczba;

}

int atakspecjalny()

{

if((mana - 25) < 0)

{

czynnosc << "Nie masz dosc many, by rzucic kule ognia";

}

else

{

mana -= 25;

int liczba = (rand() % 50 ) + 30;

czynnosc << "Rzucasz kule ognia za " << liczba << " pkt zycia";

return liczba;

}

return 0;

}

void reset()

{

x = 1;

y = 1;

hp = 100;

mana = 100;

furia = 0;

}

mag()

{

hplimit = 100;

x = 1;

y = 1;

hp = 100;

mana = 100;

furia = 0;

}

};

**Konstruktor** klasy mag ustawia jego podstawowe atrybuty zdrowie, manę, startowe położenie na planszy.

klasa mag **dziedziczy publicznie** po klasie **postac**, która jest **klasą abstrakcyjną**

funkcja **reset()** wykorzystywana jest do resetowania położenia i atrybutów postaci (w przypadku przegranej)

funkcje **atak(), obrona(), atakspecjalny()** realizują odpowiadające klasie postaci ataki.

**zlodziej.h**

#pragma once

#include "postac.h"

class zlodziej : public postac

{

private:

int hplimit;

public:

int atak()

{

int liczba = (rand() % 15 ) + 10;

czynnosc << "Atakujesz sztyletem za " << liczba << " pkt zycia";

return liczba;

}

int obrona()

{

czynnosc << "50% szans na unik i na uleczenie 25 HP";

return 1;

}

int atakspecjalny()

{

if(hp < 25) czynnosc << "Nie masz dosc hp, aby zaatakowac";

else

{

hp -= 25;

int liczba = (rand() % 50 ) + 50;

czynnosc << "Atak od tylu za " << liczba << " pkt zycia";

return liczba;

}

return 0;

}

void reset()

{

x = 1;

y = 1;

hp = 150;

mana = 0;

furia = 0;

}

zlodziej()

{

hplimit = 150;

x = 1;

y = 1;

hp = 150;

mana = 0;

furia = 0;

}

};

**Konstruktor** klasy zlodziej ustawia jego podstawowe atrybuty zdrowie i startowe położenie na planszy.

klasa zlodziej **dziedziczy publicznie** po klasie **postac**, która jest **klasą abstrakcyjną**

funkcja **reset()** wykorzystywana jest do resetowania położenia i atrybutów postaci (w przypadku przegranej)

funkcje **atak(), obrona(), atakspecjalny()** realizują odpowiadające klasie postaci ataki.

**wojownik.h**

#pragma once

#include "postac.h"

class wojownik : public postac

{

private:

int hplimit;

public:

int atak()

{

int liczba = (rand() % 25 ) + 10;

czynnosc << "Atakujesz mieczem za " << liczba << " pkt zycia";

return liczba;

}

int obrona()

{

int liczba = (rand() % 20 ) + 10;

furia += liczba;

czynnosc << "Blokujesz obrazenia o polowe, FURIA: +" << liczba;

return 0;

}

int atakspecjalny()

{

if(furia < 25) czynnosc << "Nie masz dosc furii, aby zaatakowac";

else

{

furia -= 25;

int liczba = (rand() % 30 ) + 30;

czynnosc << "Mocny Atak z wyskoku za " << liczba << " pkt zycia";

return liczba;

}

return 0;

}

void reset()

{

x = 1;

y = 1;

hp = 200;

mana = 0;

furia = 0;

}

wojownik()

{

hplimit = 200;

x = 1;

y = 1;

hp = 200;

mana = 0;

furia = 0;

}

};

**Konstruktor** klasy wojownik ustawia jego podstawowe atrybuty zdrowie, furię i startowe położenie na planszy.

klasa wojownik **dziedziczy publicznie** po klasie **postac**, która jest **klasą abstrakcyjną**

funkcja **reset()** wykorzystywana jest do resetowania położenia i atrybutów postaci (w przypadku przegranej)

funkcje **atak(), obrona(), atakspecjalny()** realizują odpowiadające klasie postaci ataki.

**przeciwnik.h**

#pragma once

class przeciwnik

{

public:

int hp,mana;

virtual int atak() = 0;

};

**klasa abstrakcyjna**, służy jako interfejs dla klas przeciwników.

**kosciotrup.h**

#pragma once

#include "przeciwnik.h"

using namespace std;

class kosciotrup : public przeciwnik

{

public:

int atak()

{

int liczba = (rand() % 10 ) + 10;

return liczba;

}

kosciotrup()

{

hp = 100;

mana = 0;

}

~kosciotrup()

{

}

};

**Konstruktor** klasy kosciotrup ustawia jego podstawowe atrybuty.

**Dziedziczy publicznie** po klasie przeciwnik, realizuje funkcję odpowiadającą za wyprowadzanie obrażeń w stronę gracza.

**menu.h**

#pragma once

#include "SFML/Graphics.hpp"

#include <iostream>

using namespace sf;

using namespace std;

#define ilosc 4

class Menu

{

private:

int wybor;

Font czcionka;

Text menu[ilosc];

Text napis;

Texture m1;

Sprite menu1;

public:

Menu(int x, int y)

{

m1.loadFromFile("menu.png");

menu1.setTexture(m1);

if (!czcionka.loadFromFile("arial.ttf"))

{

cout << "Nie wczytano czcionki!!";

}

napis.setFont(czcionka);

napis.setColor(Color::White);

napis.setString("Stworzyl: Michal Zieba");

napis.setPosition(Vector2f(x/4,0));

napis.setCharacterSize(64);

menu[0].setFont(czcionka);

menu[0].setColor(Color::Red);

menu[0].setString("Graj");

menu[0].setPosition(Vector2f(x/2, y/(ilosc + 1) \* 1));

menu[1].setFont(czcionka);

menu[1].setColor(Color::White);

menu[1].setString("Wybierz poziom");

menu[1].setPosition(Vector2f(x/2,y/(ilosc + 1) \* 2));

menu[2].setFont(czcionka);

menu[2].setColor(Color::White);

menu[2].setString("Opcje");

menu[2].setPosition(Vector2f(x/2,y/(ilosc + 1) \* 3));

menu[3].setFont(czcionka);

menu[3].setColor(Color::White);

menu[3].setString("Wyjscie");

menu[3].setPosition(Vector2f(x/2,y/(ilosc + 1) \* 4));

wybor = 0;

}

~Menu() {}

void rysuj(RenderWindow &okno)

{

okno.draw(menu1);

okno.draw(napis);

for(int i = 0 ; i < ilosc ; i++)

{

okno.draw(menu[i]);

}

}

void UP()

{

if (wybor - 1 >= 0)

{

menu[wybor].setColor(Color::White);

wybor--;

menu[wybor].setColor(Color::Red);

}

}

void DOWN()

{

if (wybor + 1 < ilosc)

{

menu[wybor].setColor(Color::White);

wybor++;

menu[wybor].setColor(Color::Red);

}

}

int oddajwybor()

{

return wybor;

}

};

klasa Menu realizuje wyświetlanie głównego menu gry.

W **konstruktorze** klasy wczytywane jest tło oraz napisy do wyświetlenia.

funkcja **rysuj()** odpowiada za wyświetlanie zawartości głównego menu w oknie, natomiast funkcje **UP()** i **DOWN()** odpowiadają za poruszanie się po menu.

**opcje.h**

#pragma once

#include <iostream>

using namespace std;

#define iloscopcji 5

class opcje

{

private:

int wybor;

Font czcionka;

Text menu[iloscopcji];

Text opcja;

Texture m1;

Sprite menu1;

public:

opcje(int x, int y)

{

m1.loadFromFile("menu.png");

menu1.setTexture(m1);

if (!czcionka.loadFromFile("arial.ttf"))

{

cout << "Nie wczytano czcionki!!";

}

opcja.setFont(czcionka);

opcja.setColor(Color::Red);

opcja.setString("Wybór Rozdzielczosci: ");

opcja.setPosition(Vector2f(x/4,0));

opcja.setCharacterSize(64);

menu[0].setFont(czcionka);

menu[0].setColor(Color::Red);

menu[0].setString("1024x768");

menu[0].setPosition(Vector2f(x/2, y/(iloscopcji + 1) \* 1));

menu[1].setFont(czcionka);

menu[1].setColor(Color::White);

menu[1].setString("1280x720");

menu[1].setPosition(Vector2f(x/2,y/(iloscopcji + 1) \* 2));

menu[2].setFont(czcionka);

menu[2].setColor(Color::White);

menu[2].setString("1366x768");

menu[2].setPosition(Vector2f(x/2,y/(iloscopcji + 1) \* 3));

menu[3].setFont(czcionka);

menu[3].setColor(Color::White);

menu[3].setString("1920x1080");

menu[3].setPosition(Vector2f(x/2,y/(iloscopcji + 1) \* 4));

menu[4].setFont(czcionka);

menu[4].setColor(Color::White);

menu[4].setString("Fullscreen");

menu[4].setPosition(Vector2f(x/2,y/(iloscopcji + 1) \* 5));

wybor = 0;

}

~opcje() {}

void rysuj(RenderWindow &okno)

{

okno.draw(menu1);

okno.draw(opcja);

for(int i = 0 ; i < iloscopcji ; i++)

{

okno.draw(menu[i]);

}

}

void UP()

{

if (wybor - 1 >= 0)

{

menu[wybor].setColor(Color::White);

wybor--;

menu[wybor].setColor(Color::Red);

}

}

void DOWN()

{

if (wybor + 1 < iloscopcji)

{

menu[wybor].setColor(Color::White);

wybor++;

menu[wybor].setColor(Color::Red);

}

}

int oddajopcje()

{

return wybor;

}

};

Jedno z **podmenu głównego menu**, konstrukcją przypomina główne menu, realizuje wyświetlanie zawartości opcji.

**wyborgracza.h**

#pragma once

#include <iostream>

using namespace std;

#define gracze 3

class wyborgracza

{

private:

int wybor;

Font czcionka;

Text menu[gracze];

Text napis[2];

Texture m1,s1;

Sprite menu1,sterowanie;

public:

wyborgracza(int x, int y)

{

m1.loadFromFile("menu.png");

menu1.setTexture(m1);

s1.loadFromFile("sterowanie.png");

sterowanie.setTexture(s1);

sterowanie.setPosition(300,100);

if (!czcionka.loadFromFile("arial.ttf"))

{

cout << "Nie wczytano czcionki!!";

}

napis[0].setFont(czcionka);

napis[0].setColor(Color::Red);

napis[0].setString("Wybór postaci:");

napis[0].setCharacterSize(64);

napis[0].setPosition(300,0);

menu[0].setFont(czcionka);

menu[0].setColor(Color::Red);

menu[0].setString("Mag");

menu[0].setPosition(Vector2f(0, y/(gracze + 1) \* 1));

menu[1].setFont(czcionka);

menu[1].setColor(Color::White);

menu[1].setString("Wojownik");

menu[1].setPosition(Vector2f(0,y/(gracze + 1) \* 2));

menu[2].setFont(czcionka);

menu[2].setColor(Color::White);

menu[2].setString("Zlodziej");

menu[2].setPosition(Vector2f(0,y/(gracze + 1) \* 3));

wybor = 0;

}

~wyborgracza() {}

void rysuj(RenderWindow &okno)

{

okno.draw(menu1);

okno.draw(napis[0]);

okno.draw(sterowanie);

for(int i = 0 ; i < gracze ; i++)

{

okno.draw(menu[i]);

}

}

void UP()

{

if (wybor - 1 >= 0)

{

menu[wybor].setColor(Color::White);

wybor--;

menu[wybor].setColor(Color::Red);

}

}

void DOWN()

{

if (wybor + 1 < gracze)

{

menu[wybor].setColor(Color::White);

wybor++;

menu[wybor].setColor(Color::Red);

}

}

int oddajgracza()

{

return wybor;

}

};

Jedno z **podmenu głównego menu**, konstrukcją przypomina główne menu, realizuje wyświetlanie menu z wyborem klasy postaci.

**wyborpoziomow.h**

#pragma once

#include <iostream>

using namespace std;

#define ilepoziomow 5

class wyborpoziomow

{

private:

int wybor;

Font czcionka;

Text menu[ilepoziomow];

Text opcja;

Texture m1;

Sprite menu1;

public:

wyborpoziomow(int x, int y)

{

m1.loadFromFile("menu.png");

menu1.setTexture(m1);

if (!czcionka.loadFromFile("arial.ttf"))

{

cout << "Nie wczytano czcionki!!";

}

opcja.setFont(czcionka);

opcja.setColor(Color::Red);

opcja.setString("Wybór Poziomu: ");

opcja.setPosition(Vector2f(x/4,0));

opcja.setCharacterSize(64);

menu[0].setFont(czcionka);

menu[0].setColor(Color::Red);

menu[0].setString("Poziom 1");

menu[0].setPosition(Vector2f(x/2, y/(ilepoziomow + 1) \* 1));

menu[1].setFont(czcionka);

menu[1].setColor(Color::White);

menu[1].setString("Poziom 2");

menu[1].setPosition(Vector2f(x/2,y/(ilepoziomow + 1) \* 2));

menu[2].setFont(czcionka);

menu[2].setColor(Color::White);

menu[2].setString("Poziom 3");

menu[2].setPosition(Vector2f(x/2,y/(ilepoziomow + 1) \* 3));

menu[3].setFont(czcionka);

menu[3].setColor(Color::White);

menu[3].setString("Poziom 4");

menu[3].setPosition(Vector2f(x/2,y/(ilepoziomow + 1) \* 4));

menu[4].setFont(czcionka);

menu[4].setColor(Color::White);

menu[4].setString("Poziom 5");

menu[4].setPosition(Vector2f(x/2,y/(ilepoziomow + 1) \* 5));

wybor = 0;

}

~wyborpoziomow() {}

void rysuj(RenderWindow &okno)

{

okno.draw(menu1);

okno.draw(opcja);

for(int i = 0 ; i < ilepoziomow ; i++)

{

okno.draw(menu[i]);

}

}

void UP()

{

if (wybor - 1 >= 0)

{

menu[wybor].setColor(Color::White);

wybor--;

menu[wybor].setColor(Color::Red);

}

}

void DOWN()

{

if (wybor + 1 < ilepoziomow)

{

menu[wybor].setColor(Color::White);

wybor++;

menu[wybor].setColor(Color::Red);

}

}

int oddajpoziom()

{

return wybor;

}

};

Jedno z **podmenu głównego menu**, konstrukcją przypomina główne menu, realizuje wyświetlanie menu z wyborem poziomu gry.