KNIGHT'S QUEST

Autor: Łukasz Wróbel

TEMATYKA

- Knight’s Quest to gra umiejscowiona w średniowiecznym świecie fantasy. Jest to gra typu RPG, w której wcielamy się w rycerza, którego napotykają różne sytuacje.

- Gra opiera się na wyborach możliwych opcji. Istnieje kilka typów ekranów, z którymi się spotkamy:
  - Menu główne,
  - Informacja,
  - Dwa wybory,
  - Miasto,
  - Walka.
- Gra posiada także panel statystyk postaci i ekwipunek.

ZASADA DZIAŁANIA GRY

Uruchomienie gry otwiera okno menu głównego, z którego możemy wybrać poszczególne opcje. W zależności od wybranego przycisku otworzy się kolejne okno z odpowiednią zawartością. Wewnątrz funkcji while program obsługuje występowanie eventów takich jak wciśnięty klawisz lub przycisk myszy.

```
while (MENU.isOpen())
{
    Vector2i localPosition = sf::Mouse::getPosition(MENU);
    Event event;
    while (MENU.pollEvent(event))
    {
        if (event.type == Event::Closed)···
        if (event.type == Event::KeyPressed || event.type == Event::MouseButtonPressed)···
    }
    mainMenu.MouseSelect(localPosition.x, localPosition.y);
    MENU.clear();
    MENU.draw(background);
    mainMenu.draw(MENU);
    MENU.display();
}
return 1;
```

FUNKCJONALNOŚĆ

Kręgosłupem gry jest funkcja while wykonująca się póki okno gry jest otwarte. Wewnątrz pętli znajduje się konstrukcja switch, według której wykonuje się poszczególna część gry. Argumentem switcha jest wartość pola obiektu gracza zawierająca informacje o jego progresie w grze.

```
while (Gra.isOpen())
{
    Event aevent;
    Vector2i localGamePosition = sf::Mouse::getPosition(Gra);
    switch (player.progress)
    {
        case 0:
            if (doOnce)···
            while (Gra.pollEvent(aevent))···
            Gra.clear();

            info.MouseSelect(localGamePosition.x, localGamePosition.y);
            Gra.draw(infoBackground);
            info.draw(Gra);

            player.update();
            Gra.draw(playerBackground);
            player.draw(Gra);

            Opcje.close();
            Gra.display();
            break;
        case 1:
            if (doOnce)···
            while (Gra.pollEvent(aevent))···
            Gra.clear();

            city.MouseSelect(localGamePosition.x, localGamePosition.y);
            Gra.draw(cityBackground);
            city.draw(Gra);

            player.update();
            Gra.draw(playerBackground);
            player.draw(Gra);

            Opcje.close();
            Gra.display();
            break;
```

ZAPIS I WCZYTYWANIE POSTACI

- Po zamknięciu okna gry stan rozgrywki zapisuje się poprzez zapis wartości pól obiektu gracz w pliku tekstowym. Przy wybraniu opcji kontynuuj w menu głównym wartości pól obiektu gracz są wczytywane i ustawiane w grze.

```
void save(Player player, String path)
{
    fstream plik;
    plik.open(path, ios::trunc | ios::out);
    if (plik.good())
    {
        plik << player.progress << endl;
        plik << player.difficulty << endl;
        plik << player.attack << endl;
        plik << player.defense << endl;
        plik << player.maxHP << endl;
        plik << player.HP << endl;
        plik << player.gold << endl;
        plik << player.goldFound << endl;
        plik << player.level << endl;
        plik << player.expEarned << endl;
        plik << player.opt << endl;
        plik << player.potions << endl;
    }
    plik.close();
}
```

Funkcja „load()” przyjmuje w argumencie wskaźnik do zmiennej typu Player, ponieważ chcemy zmienić wartości globalnego obiektu.

```
void load(Player* player, String path)
{
    fstream plik;
    plik.open(path, ios::in);
    string buffer;
    if (plik.good())
    {
        std::getline(plik, buffer);
        if (buffer != "")
        {
            player->progress = stoi(buffer);
            std::getline(plik, buffer);
            player->difficulty = stoi(buffer);
            std::getline(plik, buffer);
            player->attack = stoi(buffer);
            std::getline(plik, buffer);
            player->defense = stoi(buffer);
            std::getline(plik, buffer);
            player->maxHP = stoi(buffer);
            std::getline(plik, buffer);
            player->HP = stoi(buffer);
            std::getline(plik, buffer);
            player->gold = stoi(buffer);
            std::getline(plik, buffer);
            player->goldFound = stoi(buffer);
            std::getline(plik, buffer);
            player->level = stoi(buffer);
            std::getline(plik, buffer);
            player->expEarned = stoi(buffer);
            std::getline(plik, buffer);
            player->opt = stoi(buffer);
            std::getline(plik, buffer);
            player->potions = stoi(buffer);
        }
        else…
    }
    else…
    plik.close();
    player->update();
}
```

```
22      //Tło postaci
23      Player player(300, 900);
24      RectangleShape playerBackground;
25      playerBackground.setSize(Vector2f(300, 900));
26      Texture playerBG;
27      playerBG.loadFromFile("content/Graphics/other/playerbg.jpg");
28      playerBackground.setTexture(&playerBG);
29
30      //Tło menu głównego
31      MainMenu mainMenu(1600, 900);
32      RectangleShape background;
33      background.setSize(Vector2f(1600, 900));
34      Texture MainMenuBG;
35      MainMenuBG.loadFromFile("content/Graphics/other/menubg.jpg");
36      background.setTexture(&MainMenuBG);
37
38      //Tło opcji
39      Options options(1600, 900);
40      RectangleShape backgroundOptions;
41      backgroundOptions.setSize(Vector2f(1600, 900));
42      Texture optionsBG;
43      optionsBG.loadFromFile("content/Graphics/other/optionsbg.jpg");
44      backgroundOptions.setTexture(&optionsBG);
45
46      //Tło ekranu końca gry
47      GameOver gameOver(1600, 900);
48      RectangleShape gameOverBackground;
49      gameOverBackground.setSize(Vector2f(1600, 900));
50      Texture gameOverBG;
51      gameOverBG.loadFromFile("content/Graphics/other/gameover.jpg");
52      gameOverBackground.setTexture(&gameOverBG);
53
54      //Tło ekranu dwóch opcji
55      TwoChoice twoChoice(1300, 900);
56      RectangleShape twoChoiceBackground;
57      twoChoiceBackground.setSize(Vector2f(1600, 900));
58      Texture twoChoiceBG;
59      twoChoiceBG.loadFromFile("content/Graphics/locations/travel.jpg");
60      twoChoiceBackground.setTexture(&twoChoiceBG);
```

OBIEKTOWE TWORZENIE POSZCZEGÓLNYCH EKRANÓW

Każdy z wymienionych wcześniej rodzajów ekranów zaimplementowany jest jako osobny obiekt. Każdy z ekranów posiada własne zmienne i funkcje.

EKRAN MENU GŁÓWNEGO

```h
#pragma once
#include <SFML/Graphics.hpp>
#include <iostream>

using namespace std;
using namespace sf;

#define Max_main_menu 4

class MainMenu
{
public:
    MainMenu(float width, float height);

    int MainMenuSelected;
    void draw(RenderWindow& window);
    void MoveUp();
    void MoveDown();
    int MouseSelect(int x, int y);

    int MainMenuPressed()
    {
        return MainMenuSelected;
    }
    ~MainMenu();

private:
    Font font;
    Text mainMenu[Max_main_menu];
};
```

```cpp
#include "MainMenu.h"

MainMenu::MainMenu(float width, float height)
{
    if (!font.loadFromFile("content/Fonts/isocpeui.ttf"))
    {
        cout << "No font is here!";
    }

    //Kontynuuj
    mainMenu[0].setFont(font);
    mainMenu[0].setFillColor(Color::Red);
    mainMenu[0].setOutlineColor(Color::Black);
    mainMenu[0].setOutlineThickness(1);
    mainMenu[0].setString("Kontynuuj");
    mainMenu[0].setCharacterSize(30);
    mainMenu[0].setPosition(width - 200, height - 250);
    //Nowa gra
    mainMenu[1].setFont(font);
    mainMenu[1].setFillColor(Color::White);
    mainMenu[1].setOutlineColor(Color::Black);
    mainMenu[1].setOutlineThickness(1);
    mainMenu[1].setString("Nowa gra");
    mainMenu[1].setCharacterSize(30);
    mainMenu[1].setPosition(width - 200, height - 200);
    //Opcje
    mainMenu[2].setFont(font);
    mainMenu[2].setFillColor(Color::White);
    mainMenu[2].setOutlineColor(Color::Black);
    mainMenu[2].setOutlineThickness(1);
    mainMenu[2].setString("Opcje");
    mainMenu[2].setCharacterSize(30);
    mainMenu[2].setPosition(width - 200, height - 150);
    //Wyjście
    mainMenu[3].setFont(font);
    mainMenu[3].setFillColor(Color::White);
    mainMenu[3].setOutlineColor(Color::Black);
    mainMenu[3].setOutlineThickness(1);
    mainMenu[3].setString(L"Wyjście");
```

FUNKCJE I ELEMENTY GRY

- Gra posiada następujące elementy:
  - Interfejs graficzny;
  - Elementy programowania obiektowego;
  - Zapis i wczytanie gry;
  - Punkty kontrolne;
  - Fabuła;
  - Symulacja walki opierająca się na statystykach;
  - Statystyki i poziom postaci;
  - Ekwipunek postaci;
  - Wewnętrzna ekonomia;
  - Proste efekty dźwiękowe;
  - Możliwość włączenia i wyłączenia dźwięku;
  - Możliwość zmiany poziomu trudności;
  - Możliwość sterowania za pomocą myszki lub klawiatury(WASD + E / strzałki + Enter)

DEMONSTRACJA GRY