



Dokumentacja programu

Wielosobowa gra wzorowana na planszowej grze Monopoly w uproszczonej demonstracyjnej wersji.

Program jest projektem utworzonym na zaliczenie kursu Programowania Obiektowego prowadzonym przez Pana Mgr inż. Mariusza Ostrowskiego na Politechnice Wrocławskiej. Aplikacja powstała z użyciem kompilatora Code::Blocks w wersji 17.12, języka C++ oraz biblioteki zewnętrznej SFML dostępnej pod adresem <https://www.sfml-dev.org/index.php> .

Autor:

Wojciech Jakięła

■ Najważniejsze założenia

Przedstawiony program oraz dokumentacja powstała na jego bazie jest wersją demonstracyjną, niezawierającą pełnych funkcjonalności gry. Aplikacja nie jest tworzona w celach komercyjnych, a jej dalszy rozwój uzależniony będzie od dobrowolnego zainteresowania i chęci rozwoju autorów.

■ Cele do osiągnięcia

Aplikacja została przygotowana na potrzeby kursu Programowania Obiektowego wspomnianego wcześniej. Naszym celem jest zdobycie oraz rozwinięcie umiejętności wchodzących w zakres owego kursu oraz stworzenie sensownego programu, z którego utworzenia będziemy mieli satysfakcję.

■ Dalszy rozwój

Aplikację planujemy rozwinąć o 2 dodatkowych graczy (do tej pory dostępna jest wersja dla dwóch osób), program ma mieć funkcję losowania kolejności graczy. Chcemy także dodać funkcje pozwalające zakupić domki na posiadanych przez nas polach po zdobyciu wszystkich pól danego koloru. Funkcjonalność ta nie była potrzebna do demonstracji, więc zostawiliśmy ją na koniec. Najważniejszym elementem który chcemy wykonać jest zoptymalizowanie i poprawienie kodu na bardziej czytelny. Bardzo chcielibyśmy poprawić też warstwę graficzną, gdyż do tej pory skupiliśmy się bardziej na samym silniku gry. Z większych funkcjonalności silnika chcielibyśmy dodać także opcję licytacji oraz wyprzedania posiadanego pola.

■ Używane skróty

ECTS	Waluta w świecie gry.
------	-----------------------

*Sprawdź, który z twoich znajomych
ma „smykałkę do interesów”, a który
zbankrutuje tracąc swoje ostatnie oszczędności.*

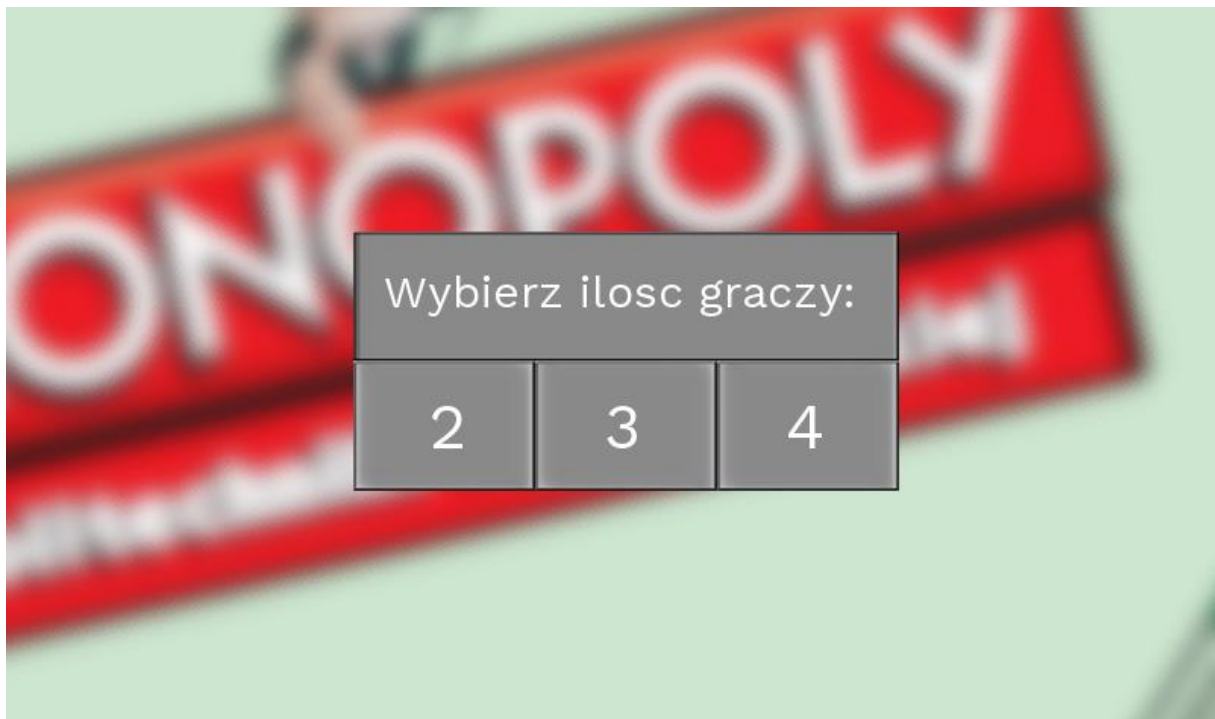
Ogólny opis programu

Program w pierwszej fazie uruchamia okienko z ustawieniami takimi jak wybór ilości graczy oraz koloru pionka dla każdego gracza. Następnie przechodzimy do właściwej części – czyli samej gry. Aplikacja działa w systemie turowym. Gracz rzuca kośćmi, przesuwa swój pionek i decyduje o zakupie pola na którym stoi. Każdy gracz jest ograniczony pewnym budżetem, który początkowo wynosi 1500 jednostek.

Etap 1 (USTAWIENIA)

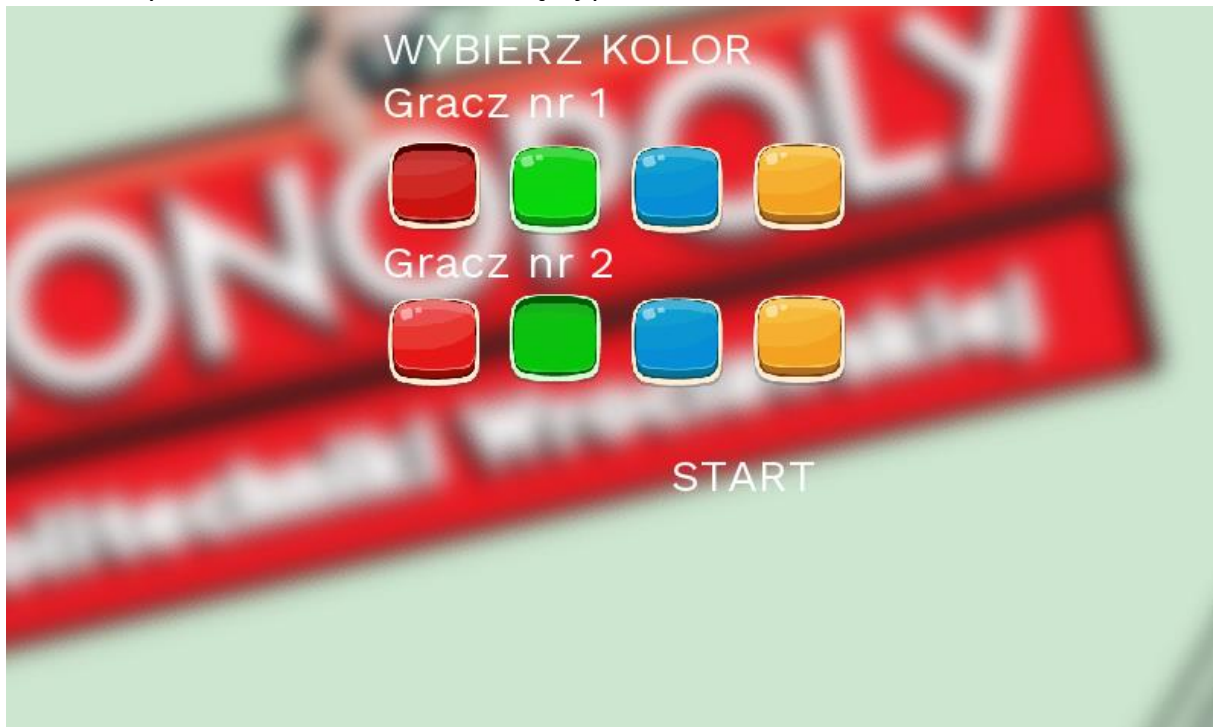
W pierwszej części wyświetlane jest nam okienko przedstawiające autorów programu oraz ogólnie informacje o używanych czcionkach i projekcie.

Dalej przechodzimy klawiszem ENTER do wyboru ilości graczy. Gra będzie przeznaczona dla 2-4 osób. W tle jest wyblurowana animacja planszy gry.




Po wybraniu odpowiadającej nam opcji przechodzimy do ustawień koloru pionków.

Dostępne mamy 4 kolory: czerwony, zielony, niebieski oraz żółty. Niemożliwe jest wybranie takich samych kolorów dla dwóch lub więcej pionków.



Po tych wszystkich ustawieniach możemy przejść do właściwej części zabawy.



TURA:

Ilość ECTS wszystkich graczy:

GRACZ1: 1500

GRACZ2: 1500

GRACZ3:

GRACZ4:

Karty aktualnego gracza:

Gracz	Gracz 1	Gracz 2	Gracz 3	Gracz 4	Gracz 5	Gracz 6	Gracz 7	Gracz 8	Gracz 9	Gracz 10	Gracz 11	Gracz 12	Gracz 13	Gracz 14	Gracz 15
D-1	D-2	B-1	B-2	B-3	A-1	A-2	A-3	A-4	A-5	A-6	H-1	H-2	H-3	H-4	H-5
C-4	C-5	C-6	T-3	E-1	Pol	W	Przystanek Autobusowy	Kolejka	Kolejka	Kolejka	Kolejka	Kolejka	Kolejka	Kolejka	Kolejka

Pole **NEUTRALNE** – gracz po prostu kończy turę.

Pole **SPECJALNE** – gracz, który stanął na polu specjalnym bierze kartę z odpowiedniego stosu kart „Kolokwium” lub „Odpowiedź” i wykonuje się akcja opisana na karcie. Następnie karta zwracana jest na spód stosu, lub w przypadku karty „Wyjście z więzienia” zatrzymywana przez gracza, który ją trafił. Kartę tę, można także zlicytować.

Pole **START** – pole opisane „Wejście na PWR”

Ekran gry prezentuje następujące elementy:

- Planszę przerobioną przez nas na potrzeby aplikacji, zawierającą znane budynki Politechniki Wrocławskiej.
- Ilość ECTS (waluty) każdego gracza
- Turę obecnego gracza
- Karty aktualnego gracza
- Kości do gry
- Przyciski odpowiednio: do zakupu domku/hotelu, do zakupu pola, wyciszenia muzyki, zrobienia przelewu na konto innego gracza oraz przycisk końca tury.

Po wciśnięciu przycisku przelewu prezentuje się nam następujące okno:



W oknie wybieramy sumę punktów jaką chcemy przelać na konto innego gracza. Liczba ta dynamicznie wyświetla nam się w odpowiednim miejscu. By zakończyć operację klikamy przycisk *Zakończ* widoczny w prawym dolnym rogu okna przelewu.

Najeżdżając na interesujące nas pole na planszy wyświetli nam się karta danego pola wraz z opisem.

The screenshot displays a board game interface. On the left is a portion of the game board with various colored squares and icons. In the center, a property card titled 'KARTA NIERUCHOMOŚCI D-1' is shown, detailing a plot of land with four buildings and their respective ECTS values.

KARTA NIERUCHOMOŚCI D-1

KOMORNE 2ECTS

Z 1 Budynkiem: 10ECTS
 Z 2 Budynkami: 30ECTS
 Z 3 Budynkami: 90ECTS
 Z 4 Budynkami: 160ECTS
 Z Laboratorium: 250ECTS

Wartość hipoteki 30ECTS
 (Budynki kosztują 50 hazardy)

On the right side of the interface, there is a panel with the following information:

- Ilość ECTS wszystkich graczy:**
 - GRACZ1: 1500
 - GRACZ2: 1500
 - GRACZ3:
 - GRACZ4:
- TURA:** (indicated by a green dot)
- Karty aktualnego gracza:** A grid of 16 cards, each with a color, a letter/number code (e.g., B-1, B-2, B-3, A-1, A-2, A-3, A-4, A-5, A-6, A-7, A-8, A-9, A-10, A-11, A-12, A-13, A-14, A-15, A-16), and a small icon.
- At the bottom of the panel are four buttons: 'Kup domki', 'Kup pole', 'Zrób przelew', and 'Koniec tury'.

KONIEC GRY

Warunki końca rozgrywki mogą zostać ustalone przez graczy. Proponujemy, aby gra kończyła się gdy w grze zostanie jedyny gracz, który zachował swoje punkty waluty lub w dowolnym momencie na który zgodzą się gracze – w takim wypadku kolejność miejsc może zależeć od ilości waluty i posiadanych pól.

Klasy główne:

- Gracz
- Pionek
- Kostka
- Interfejs

Klasy poboczne:

- Przycisk
- Pole planszy

Szczegółowy opis klas:

Gracz

Klasa gracza przechowuje informacje o posiadanej gotówce oraz wektorach przesunięcia każdego z pionków. Obsługuje ruch pionków i rzut kośćmi.

Wymagane nagłówki:

```
#include "Pionek.h"
```

```
#include <string>
```

Klasa zaprzyjaźniona: Interfejs.

Konstruktor: `Gracz(float g=1500.0f, Pionek* p=nullptr)`

g – początkowa wartość gotówki

p – wskaźnik na pionek gracza

Atrybuty publiczne:

float gotowka; - gotówka gracza

Pionek pionek;* - wskaźnik na pionek gracza

Metody publiczne:

```
int rzucKoscmi()
```

Funkcja zwraca losową wartość rzutu kością (od 1 do 6)

```
void ruszPionkiem(int tura)
```

Funkcja przesuwania pionkiem gracza o jedno pole planszy w zależności od tury gracza.

Przyjmuje wartości od 0 do 1

1 – przesuwaj gracza nr 1

0 – przesuwaj gracza nr 2

Kostka

Kostka jest klasą obsługującą wygląd kostki widocznej w interfejsie. Klasa dziedziczy po `sf::Drawable` pochodzącej z zewnętrznej biblioteki SFML

Wymagane nagłówki:

```
#include <SFML/Graphics.hpp>
```

Konstruktor: `Kostka(sf::Texture &pT)`

`pT` – tekstura, którą zostaje ustawiony sprite kostki

Atrybuty publiczne:

`sf::Sprite sprite` – sprite naszej kostki widoczny w interfejsie

Metody publiczne:

```
void draw(sf::RenderTarget &target, sf::RenderStates states) const
```

funkcja wirtualna zastępująca tę z klasy `Drawable`, rysująca nasz sprite na ekranie.

Dokumentacja funkcji `draw` znajduje się w dokumentacji biblioteki SFML

```
void ustawKostke(int stan)
```

Funkcja ustawiająca liczbę oczek na spritcie naszej kostki w zależności od argumentu `stan`. Powinna przyjmować za argumenty liczby całkowite od 1 do 6. Odpowiednim argumentem jest metoda z klasy *Gracz* – `rzucKoscmi()`;

Pionek

Funkcja `pionek` obsługuje sprite naszego pionka. Także dziedziczy po klasie `Drawable`.

Wymagane nagłówki:

```
#include <SFML/Graphics.hpp>
```

Konstruktor: `Pionek(sf::Texture &pT)`

Konstruktor działa indentyfikacyjnie jak w klasie `Kostka` oraz posiada analogiczne zmienne i metody pozwalające rysować nasz pionek na ekranie (sprite, metoda `draw`)

Przycisk

Klasa ta obsługuje przyciski wyboru koloru dla każdego z graczy. Ustawia kolory CZERWONY, ZIELONY, NIEBIESKI, ŻÓŁTY. Klasa dziedziczy po wspomnianych wcześniej klasach Drawable i Transformable.

Wymagane nagłówki:

```
#include <SFML/Graphics.hpp>
```

Konstruktor: Przycisk(sf::Texture &pT, float x, float y);

pT – tekstura przycisku pobierana z funkcji głównej

x oraz y – współrzędne położenia przycisku

Atrybuty publiczne:

bool wcisniety – zmienna opisująca stan przycisku, 1 oznacza wciśnięty, 0 niewciśnięty. Domyślnie konstruktor ustawia tę zmienną na 0

Metody publiczne:

void wcisnij() – funkcja zmieniająca stan przycisku z 0 na 1 i odwrotnie oraz ustawiająca odpowiedni sprite w zależności od stanu.

Interfejs

Klasa obsługująca cały interfejs gry. Odpowiada za przelewy między graczami, obsługę poruszania pionków i kości, wyciszanie muzyki, obsługująca karty, budynki i cały silnik gry.

Wymagane nagłówki:

```
#include "Gracz.h"
```

```
#include "PolePlanszy.h"
```

```
#include <SFML/Graphics.hpp>
```

```
#include <SFML/Audio.hpp>
```

Konstruktor:

```
Interfejs(sf::Font & czcionka, sf::Texture tex [], sf::Texture tex2 [], sf::Texture & tex3, Gracz * gra=nullptr);
```

Czcionka – czcionka tekstów interfejsu

Tex[] – tablica tekstur kart pól

Tex2[] – tablica tekstur kart specjalnych

Tex3 – tekstura interfejsu przelewu

Gra – wskaźnik na graczy obecnych w grze

Publiczne parametry klasy:

bool przelewAktywny – zmienna ustawiana na wartość true jeżeli gracz jest w trakcie przelewu

sf::Sprite kartaSpecjalna[4] – tablica elementów kart specjalnych takich jak kolokwium czy odpowiedź

PolePlanszy polaNormalne[28] – tablica elementów pól możliwych do kupienia przez gracza

Gracz * gracz – wskaźnik na graczy obecnych w grze

bool pauza – zmienna która pauzująca grę na 2 sekundy gry ustawiona jest na „true”

int turaGracza – zmienna przechowująca informacje o aktualnej turze gracza

Metody publiczne:

void draw(sf::RenderTarget &target, sf::RenderStates states) const;

void sprawdzPrzyciski(sf::Music & muzyka) – funkcja która po kliknięciu na przycisk wyciszenia muzyki, wyłącza ją.

muzyka – muzyka która zostanie wyciszona

int sprawdzPola() – funkcja obsługująca wyświetlanie na środku ekranu szczegółowej karty pola na które najedziemy myszką

void przelejPieniadze() – funkcja wyświetlająca i obsługująca interfejs przelewu.

void aktualizujStanKonta() – funkcja wyświetlająca stan konta każdego z graczy w polu interfejsu

void wyswietlajKarty() – funkcja wyświetlająca karty kupione przez każdego z graczy w odpowiednim polu interfejsu

Użyte dodatkowe czcionki:

- Work Sans by Wei Huang (SIL Open Font License)
- Monopoly Regular by Hyun S. Choi (Freeware)

Muzyka w tle: Gothic 2 Noc Kruka – Jarkendar Theme