Game Saper

https://github.com/GrzegorzM82/SaperGame

**WSTĘP**

Saper jest grą logiczną, która powstała w 1981 roku. Od tego czasu jest dostępna w każdej wersji systemu Windows.

**CELE**

1. Stworzenie gry komputerowej Saper zgodnie z zasadami gry pierwowzoru
2. Opracowanie 4 trybów rozgrywki:   
   - początkujący  
   - zawansowany  
   - ekspert  
   - tryb dowolnego doboru poziomu trudności
3. Program sam powinien rozpoznawać i nie dopuszczać do wystąpienia błędów
4. Gram ma mieć przejrzyste GUI

**ZASADY GRY**

Gra polega na odkrywaniu na planszy poszczególnych pól w taki sposób, aby nie natrafić na minę. Na każdym z odkrytych pól napisana jest liczba min, które bezpośrednio stykają się z danym polem (od jeden do ośmiu; jeśli min jest zero to na polu nie ma wpisanej liczby). Należy używać tych liczb by wydedukować gdzie schowane są miny.

**INTERFEJS UŻYTKOWNIKA**

**a)**

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, numer

Opis wygenerowany automatycznie

Gra zawiera 4 typy rozgrywki:

-Początkujący – plansza 8×8 pól, 10 min

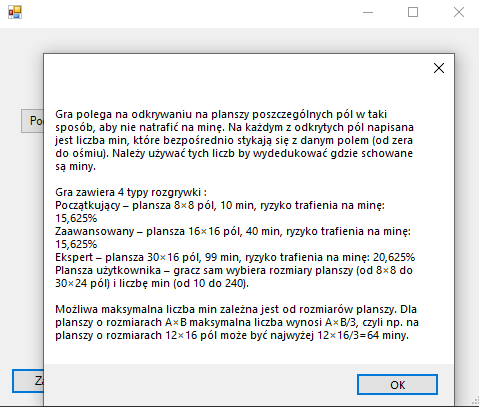
-Zaawansowany – plansza 16×16 pól, 40 min

-Ekspert – plansza 30×16 pól, 99 min

-Plansza dowolna – gracz sam wybiera rozmiary planszy (od 8×8 do 30×24 pól) i liczbę min (od 10 do 240).

Dodatkowo możliwa maksymalna liczba min zależna jest od rozmiarów planszy. Dla planszy o rozmiarach A×B maksymalna liczba wynosi A×B/3, czyli np. na planszy o rozmiarach 12×16 pól może być najwyżej 12×16/3=64 miny.

**b)**



Po naciśnięciu przycisku „Zasady” pojawią się wszystkie zasady gry Saper

**c)**

Po wciśnięciu przycisku „Graj” wszystkie dane podane przez użytkownika są sprawdzane.  
W przypadku pozostawieniu jakiegoś wolnego pola wyskoczy komunikat:  
**Obraz zawierający tekst, zrzut ekranu, Prostokąt, Czcionka

Opis wygenerowany automatycznie**

W przypadku wpisania nieprawidłowo ilości kolumn wyskoczy komunikat:  
Obraz zawierający tekst, zrzut ekranu, Prostokąt, Czcionka

Opis wygenerowany automatycznie

W przypadku wpisania nieprawidłowo ilości wierszy wyskoczy komunikat:  
Obraz zawierający tekst, zrzut ekranu, Czcionka, Prostokąt

Opis wygenerowany automatycznie

W przypadku wpisania nieprawidłowo ilości bomb wyskoczy komunikat:

Obraz zawierający tekst, zrzut ekranu, Czcionka, Prostokąt

Opis wygenerowany automatycznie

W przypadku prawidłowych wszystkich danych wejściowych uruchomi się plansza z gra

**d)**

Obraz zawierający tekst, zrzut ekranu, Prostokąt, kwadrat

Opis wygenerowany automatycznie

Pojawia się nowe okno gdzie w lewym górnym rogu są pokazane obecnie wybrane ustawienia, na środku jest licznik eksplozji .

**HIERARCHIA**

void main()

{

dołącza bibliotekę Win Form App

wstępnie konfiguruje i wywołuje element klasy do obsługi gry(MyGame.h)

}

public ref class MyGame:

{

private:

int randField(int numberEnd) //generuje liczbę z zakresu od 0 do numberEnd-1;

void ShowSettingsForm()//tworzy nowy element klasy MyMenu, wywołuje go i pobiera informacje podane w menu przez użytkownika

void InitializeGame()//tworzy interfejs graficzny gry

void GenerateBoard() //generuje logiczna plansze do gry i wypełnia ją bombami

void Button\_Click(Object^ sender, EventArgs^ e) //po kliknieciu przycisku na planszy sprawdza czy znajduje się tam mina, czy wolne pole

}

public ref class MyMenu :

{

private:

void InitializeComponent(void)// generuje interfejs graficzny dla menu gry

System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) //po wciscieniu przycisku wypełnia domyślne dla dla trybu początkującego

System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) //po wciscieniu przycisku wypełnia domyślne dla dla trybu zawansowanego

System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) //po wciscieniu przycisku wypełnia domyślne dla dla trybu ekspert

System::Void button7\_Click(System::Object^ sender, System::EventArgs^ e) //po wciscieniu przycisku wypełnia domyślne dla dla trybu indywidualnej preferencji

System::Void button8\_Click(System::Object^ sender, System::EventArgs^ e) //po wcisnieciu przysiku “Graj”, Wczytywane są ustawienia gry, jeśli są nieporawne, to zostaje wyświetlony odpowiedni komunukat

void numberTextBox\_KeyPress(Object^ sender, KeyPressEventArgs^ e) //zapobiega wpisywaniu liter do pól z wpisywaniem rozmiarów planszy

}

**OPIS REALIZACJI**

**Platforma testowa i oprogramowanie:**

* Komputer stacjonarny
* Visual Studio 2022
* Microsoft Visio
* .NET Framework
* CLI ( clr )
* Biblioteka (WinFormApp)

**Realizacja:**

1. Plan pracy
2. Schemat blokowy
3. Instalacja bibliotek do GUI
4. Napisanie części logicznej do obsługi Gry w konsoli
5. Dodanie nowych okien GUI do projektu
6. Usunięcie części realizowanej w konsoli
7. Zaprojektowanie GUI do Menu i Gry

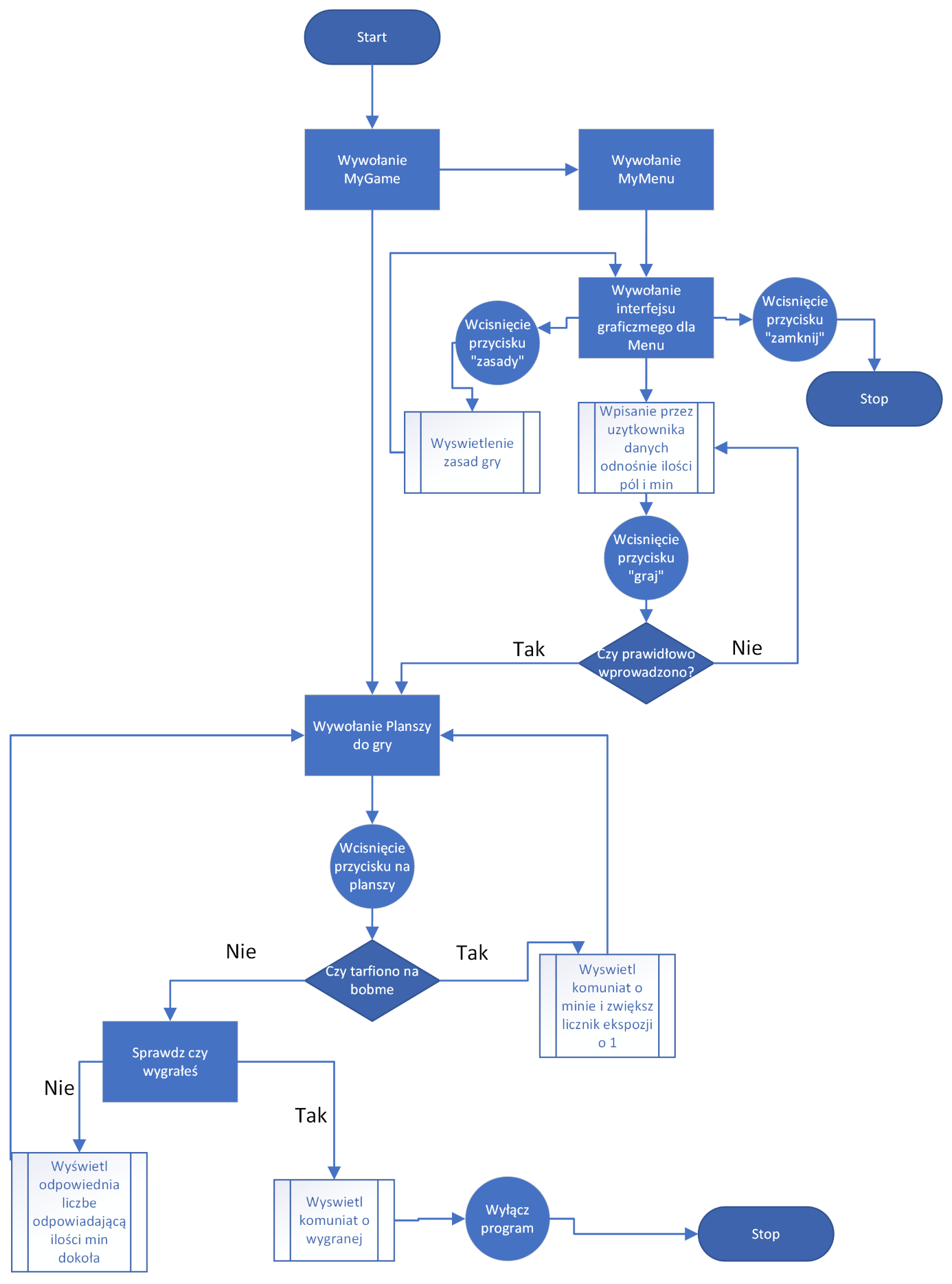
**Lista Bugów rozwiązanych:**

- problem z generowaniem dynamicznej tablicy( rozwiązanie vector)

- problem z opisanie współrzędnych na planszy(linearyzacja tablicy)  
- problem z algorytmem wstawiającym bomby do tablicy(zastosowanie rand)

-problem z połączeniem części logicznej z GUI(powrót do array z vector, biblioteka do GUI nie wspiera dynamicznego zmiany rozmiaru tablicy)  
-problem z zabezpieczeniem danych wejściowych przed nieprawidłowymi danymi(dodanie dodatkowej funkcji)

-problem z dodaniem kolejnych kart w GUI(stworzenie osobnych klas dla dwóch obiektów)  
-problem z algorytmem wykrywania zwycięstwa(ze względu na błędne początkowe założenia na początku były zliczania kliknie podwójne, które później zostały wyeliminowane.



**BIBLIOGRAFIA**

PROGRAMOWANIE W JĘZYKU C++ WPROWADZENIE DLA INŻYNIERÓW   
Autorstwa: Bogusław Cyganek

<https://gra-saper.pl/>

<https://pl.wikipedia.org/wiki/Saper_(gra_komputerowa)>

<https://www.youtube.com/watch?v=ymTCXDGakj4&list=PL2i17lRog5pBwTyF6Lhvzc-tkuySQNsfK&index=1>