POLITECHNIKA LUBELSKA
WYDZIAŁ MATEMATYKI I INFORMATYKI TECHNICZNEJ

Kierunek: Inżynieria i Analiza Danych

Wisielec

Bartosz Oleszek, Katarzyna Zyśko, Zuzanna Winiarska

27 sierpnia 2024

Spis treści

Wprowadzenie

Wisielec to interaktywna gra słowno-logiczna, w której gracze testują swoją zdolność do odgadywania słów poprzez wprowadzanie pojedynczych liter. Gra została stworzona w języku C++ przy użyciu biblioteki wxWidgets, umożliwiającej dynamiczny interfejs graficzny. Inspiracją do tego projektu był klasyczny hangman (wisielec) dostępny na wielu platformach.

Cel projektu

Gra "Wisielec"to interaktywna aplikacja napisana w języku C++, wykorzystująca bibliotekę wxWidgets. Celem projektu jest dostarczenie użytkownikowi przyjemnej rozrywki, jednocześnie rozwijając umiejętności logicznego myślenia oraz słownikowe. Gra opiera się na odgadywaniu słów, co pozwala ćwiczyć zdolności związane z spostrzegawczością, dedukcją i umiejętnością radzenia sobie w sytuacjach, w których konieczne jest podejmowanie decyzji na podstawie ograniczonych informacji.

Cel Gry

Celem gry "Wisielec"jest dostarczenie rozrywki i jednocześnie rozwijanie umiejętności logicznego myślenia poprzez odgadywanie ukrytych słów. Gracz stawia sobie za zadanie odgadnąć słowo przed przekroczeniem limitu błędnych prób. Punkty są przyznawane za poprawne odgadnięcia, a statystyki gracza motywują do poprawy wyników. Gra łączy zabawę z edukacją, rozwijając spostrzegawczość i dedukcję.

Zasady Gry

- Gra prezentuje graczowi ukryte słowo za pomocą pustych miejsc reprezentowanych podkreśleniem "_". Gracz musi odgadnąć literę, wprowadzając ją do pola tekstowego.
- Po wprowadzeniu litery, program sprawdza, czy litera znajduje się w słowie. Jeśli tak, odkrywa odpowiednie miejsce w słowie, a gracz kontynuuje odgadywanie.
- Jeśli litera nie występuje w słowie, program rysuje element wisielca. Gracz ma ograniczoną liczbę prób, zanim zostanie ukazany cały wisielec, co oznacza przegraną.
- Gracz ma także możliwość skorzystania z przycisku podpowiedzi, który wyświetli jedną literę w słowie, jednak za cenę utraty punktów.
- Po zdobyciu 5 zwycięstw, gracz otrzymuje wirtualny brązowy puchar, co stanowi wyróżnienie za osiągnięcia w grze. Kolejne wyróżnienie, srebrny puchar, przyznawane jest po uzyskaniu 10 zwycięstw. Dla najbardziej wytrwałych graczy, po 15 zwycięstwach przewidziany jest złoty puchar, symbolizujący najwyższy poziom umiejętności w odgadywaniu słów.

Elementy Interfejsu Graficznego

Interfejs graiczny gry *"Wisielec"* (rysunek 1) został starannie zaprojektowany, aby zapewnić użytkownikowi intuicyjne i przyjemne doświadczenie podczas zgadywania słów. Składa się z kilku kluczowych elementów, które nie tylko ułatwiają rozgrywkę, ale także dostarczają istotnych informacji o stanie gry.

1. Okno główne

Okno główne to główny obszar interakcji w grze "*Wisielec*", gdzie odbywa się cała rozgrywka i umieszczone są wszystkie elementy interfejsu użytkownika. Dodatkowo, gracz może zminimalizować lub zamknąć okno za pomocą przycisków systemowych w prawym górnym rogu, co umożliwia mu chwilowe ukrycie gry lub zakończenie rozgrywki.

2. Pole zgadywanego słowa

W górnej części okna wyświetlane jest pole do zgadywania słowa, z serią pustych pól (w formie ciągu " _") zastąpionych literami po każdej poprawnej odpowiedzi gracza. To miejsce jest centralnym punktem interakcji, gdzie gracz monitoruje postępy w odgadywaniu słowa. Dodatkowo, można tu otrzymać podpowiedzi po naciśnięciu odpowiedniego przycisku.

3. Kategoria

Poniżej pola zgadywanego słowa znajdują się informacje o kategorii, do której należy słowo, oraz liczba zdobytych punktów przez gracza. Kategoria dostarcza dodatkowego kontekstu dotyczącego słowa, co może pomóc graczowi w dedukowaniu jego zawartości.

4. Liczba punktów

Liczba punktów, oznaczonych w interfejsie jako błękitne diamenty, motywuje graczy do poprawnego zgadywania słów i stanowi walutę, którą mogą wymieniać na podpowiedzi ułatwiające rozgrywkę. Jej wyświetlanie pozwala graczom śledzić swój postęp i podejmować decyzje dotyczące dalszej rozgrywki.

5. Obszar rysowania postaci wisielca

Pod informacjami o słowie, kategorii i liczbie punktów znajduje się obszar, gdzie rysowana jest postać wisielca. Stopniowo, w miarę postępu rozgrywki i popełniania błędów przez gracza, dodawane są kolejne elementy postaci, co tworzy wizualną reprezentację stanu gry. Gracz może obserwować rozwój postaci, co dodaje emocji i dynamiki do rozgrywki.

6. Przycisk podpowiedzi

Przycisk podpowiedź, wyświetlany w formie grafiki żarówki, znajduje się obok obszaru rysowania postaci wisielca. Kliknięcie tego przycisku wyświetla dodatkową wskazówkę, która może pomóc graczowi w odgadnięciu słowa. Nad przyciskiem wyświetlana jest również informacja o cenie podpowiedzi, co informuje gracza o koszcie skorzystania z tej opcji.

7. Pole do wpisywania odpowiedzi

W centralnej części interfejsu znajduje się pole tekstowe, które służy do wprowadzania odpowiedzi. To pole stanowi główne narzędzie interakcji, gdzie gracz aktywnie uczestniczy w rozgrywce, podejmując decyzje i wprowadzając litery w celu zgadywania słowa. Jest to kluczowy element interfejsu, który umożliwia graczowi bezpośrednie zaangażowanie się w rozgrywkę i prowadzenie procesu odgadywania.

8. Lista użytych liter

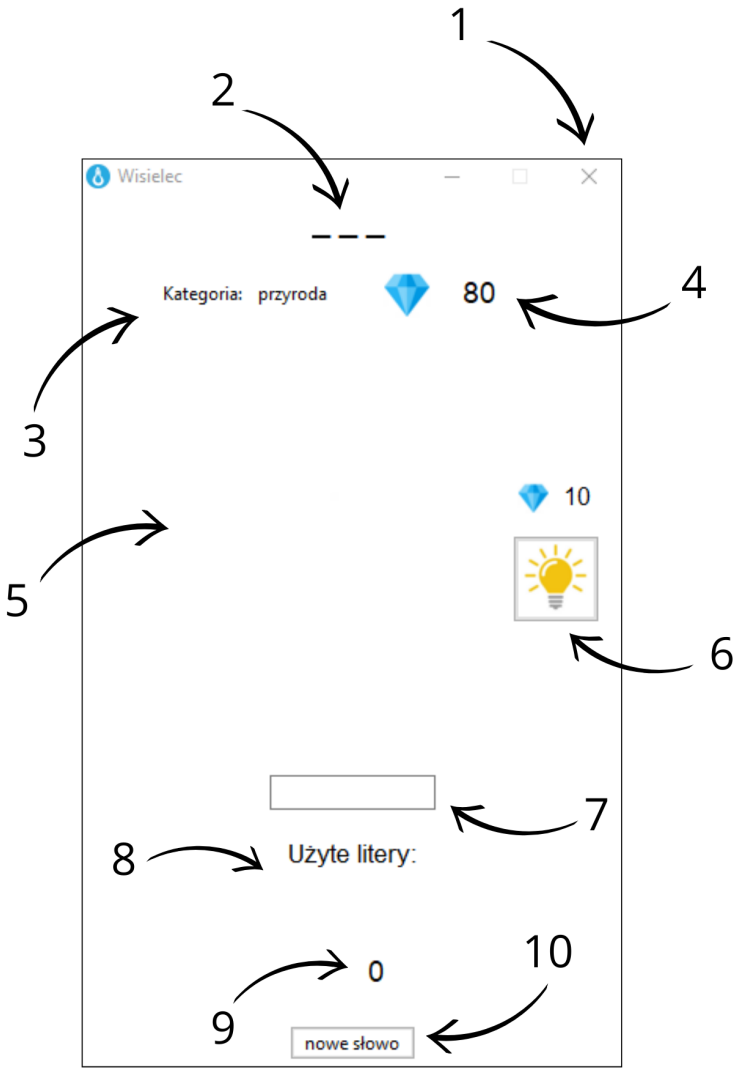
Pod polem do wpisywania odpowiedzi znajduje się obszar, w którym wyświetlana jest lista liter, które zostały już użyte w trakcie rozgrywki. To pomaga uniknąć powtórek i śledzić postępy.

9. Wyświetlanie pucharu i liczby zwycięstw

W dolnej części okna znajduje się obszar, w którym prezentowane są informacje o pucharze (gdy liczba zwycięstw ≥ 5) oraz liczbie wygranych. Kolor pucharu może się zmieniać w zależności od liczby zwycięstw gracza, co dodaje elementu motywującego do dalszej gry.

10. Przycisk "nowe słowo"

Na samym dole okna znajduje się przycisk ”nowe słowo”, który pozwala na rozpoczęcie nowej rundy. Po jego kliknięciu losowane jest nowe słowo do odgadnięcia, umożliwiając graczowi kontynuację rozgrywki.



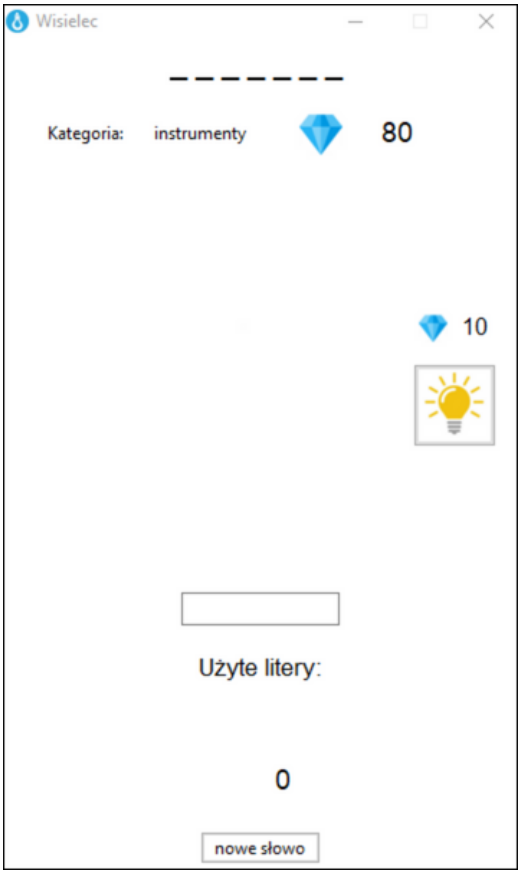
Rysunek 1: Interfejs gry

Rozgrywka

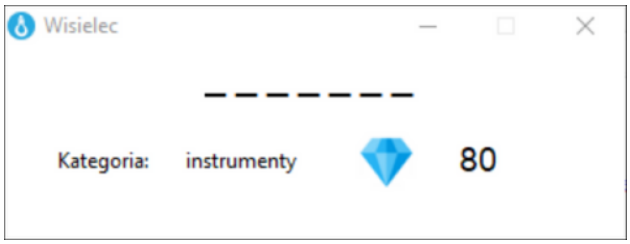
Rozgrywka gry "*Wisielec*" polega na odgadywaniu słowa poprzez wprowadzanie liter. Gracz musi uzupełnić puste pola literami, próbując odgadnąć słowo. Może również korzystać z podpowiedzi. Za każdym razem, gdy zgadnie literę, zostanie ona ujawniona. Za błędne odpowiedzi dodawany jest element postaci wisielca. Gracz ma ograniczoną liczbę prób. Po zakończeniu rundy może rozpocząć kolejną. Gra wymaga spostrzegawczości i szybkiego myślenia.

1. Rozpoczęcie gry

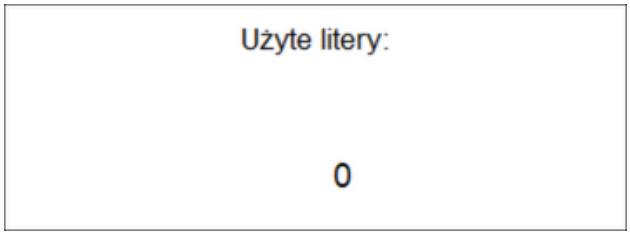
Gracz uruchamia grę "Wisielec", klikając na ikonę aplikacji lub wybierając ją z menu startowego urządzenia. Po uruchomieniu, na ekranie wyświetlony zostaje przejrzysty interfejs gry (rysunek 2a), przygotowany do rozpoczęcia rozgrywki. Interfejs zawiera pole tekstowe z pustymi miejscami na litery. Tutaj wyświetlane jest losowo wybrane słowo do zgadnięcia. Poniżej znajduje się kategoria wylosowanego słowa oraz liczba punktów dostępnych do wykorzystania na podpowiedzi, która startowo jest ustawiona na 80 (rysunek 2b). Przy każdej nowej grze zbiór użytych liter jest pusty, a liczba zwycięstw jest równa 0 (rysunek 2c).



(a) Początek rozgrywki



(b) Kategoria słowa i aktualna liczba punktów

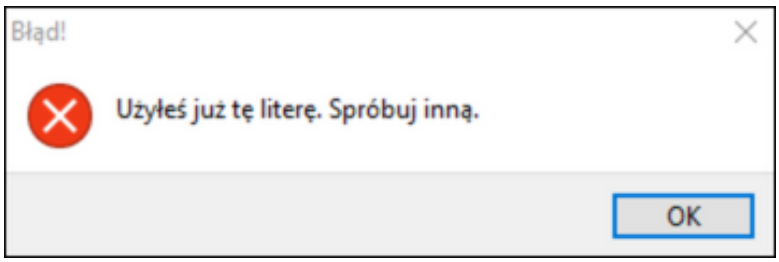


(c) Użyte litery i liczba zwycięstw

Rysunek 2: Rozpoczęcie gry

2. Zgadywanie liter

Gracz wprowadza litery do pola tekstowego, próbując odgadnąć słowo. Jeżeli słowo zostało już wykorzystane (znajduje się w zbiorze *"Użyte litery"*), pojawia się odpowiedni komunikat (rysunek 3). W przeciwnym przypadku następuje sprawdzenie, czy podana litera znajduje się w ukrytym słowie.



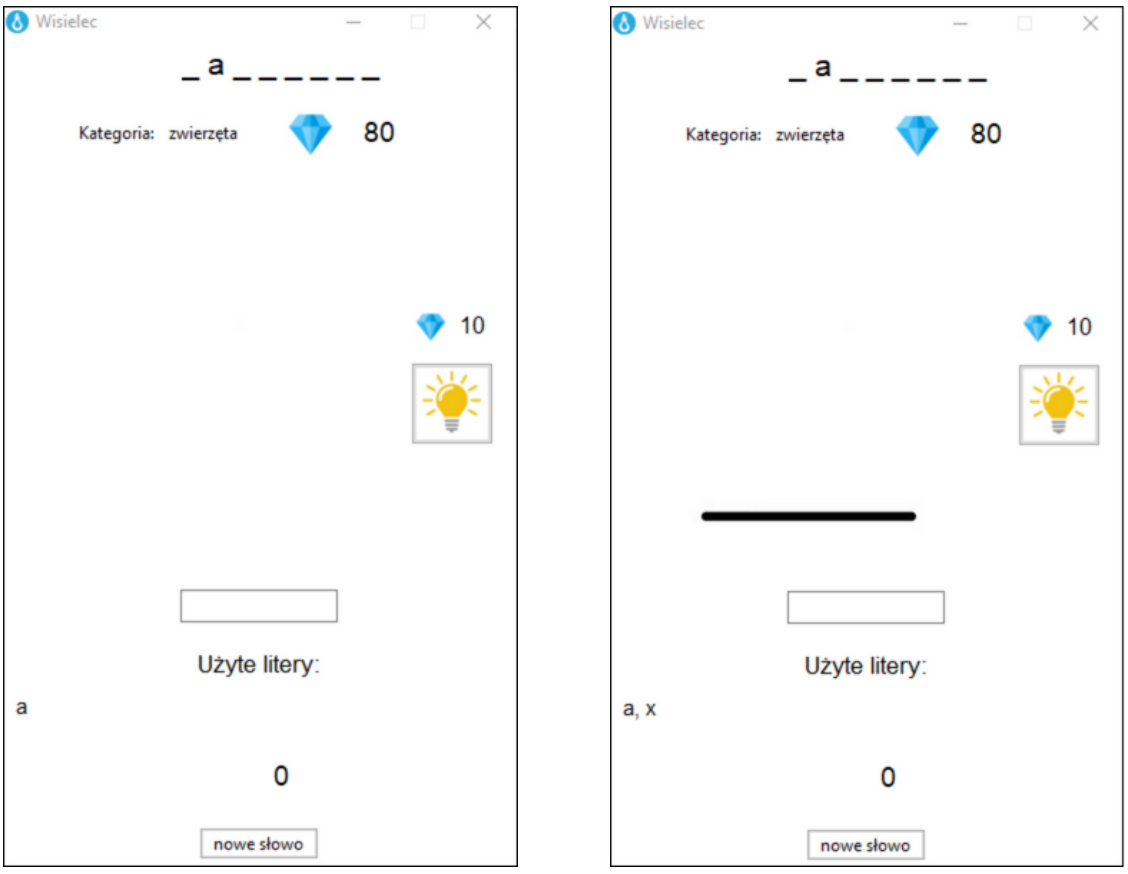
Rysunek 3: Komunikat o błędzie ponownego wpisania tej samej litery

Poprawna odpowiedź

Jeżeli litera występuje w słowie, wszystkie jej wystąpienia zostają odkryte na swoich odpowiednich pozycjach (rysunek 4a), a gracz może kontynuować zgadywanie kolejnych liter. Dodatkowo, zgadnięta litera pojawia się wśród liter już użytych.

Niepoprawna odpowiedź

Jeżeli litera nie występuje w słowie liczba dostępnych prób zostaje zmniejszona o jeden, co zwiększa trudność gry. Dodatkowo, rysowany jest kolejny element postaci wisielca (rysunek 4b), co stanowi wizualną reprezentację błędnej odpowiedzi. Ta dynamiczna zmiana wizualna motywuje gracza do większej ostrożności i trafności w kolejnych próbach. Ponadto, tak jak poprzednio, litera zostaje dopisana do zbioru użytych liter.



(a) Poprawna odpowiedź

(b) Błędna odpowiedź

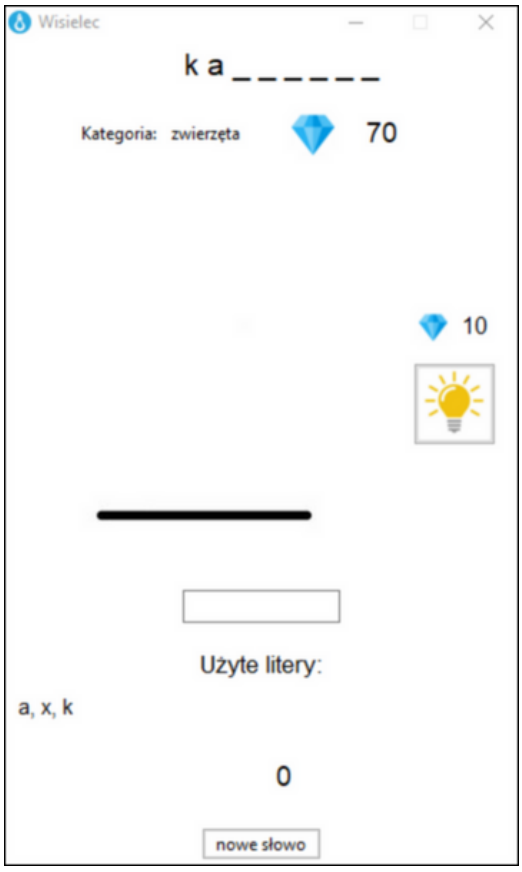
Rysunek 4: Sprawdzenie poprawności odpowiedzi

3. Wyświetlenie podpowiedzi (opcjonalne)

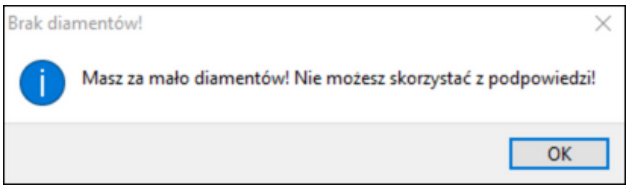
Jeżeli gracz ma ≥ 10 punktów, może skorzystać z podpowiedzi, co skutkuje zmniejszeniem aktualnego zasobu punktów o 10. Po jej użyciu jedna litera w zgadywanym słowie zostaje odsłonięta w każdym miejscu, w którym występuje w słowie (rysunek 5a) oraz zostaje dołączona do zbioru *"Użyte litery"*. Podpowiedź może znacząco ułatwić proces odgadywania słowa, pomagając graczowi dokładniej określić jego zawartość.

Jeżeli zaś gracz ma 0 punktów, po naciśnięciu przycisku oznaczającego podpowiedź, wyświetla się odpowiedni komunikat (rysunek 5b).

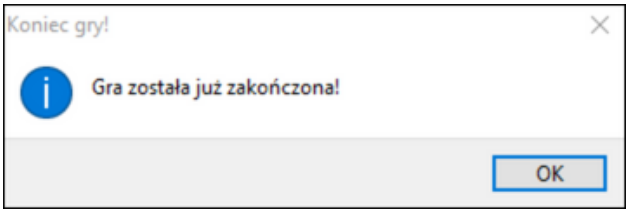
W przypadku, gdy słowo zostało już odgadnięte, a naciśnięto przycisk podpowiedzi wyświetli się komunikat o zakończonej grze (rysunek 5c).



(a) Podpowiedź



(b) Komunikat o zbyt małej liczbie punktów(diamentów)



(c) Komunikat o zakończeniu gry

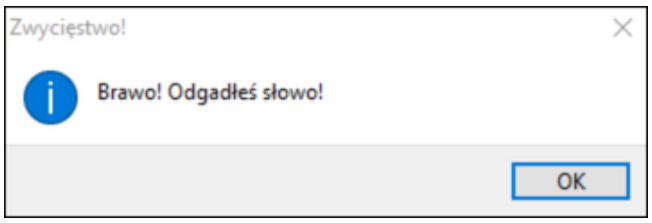
Rysunek 5: Działanie przycisku podpowiedzi

4. Zakończenie gry

Gra kończy się, gdy gracz odgadnie całe słowo (zwycięstwo) lub popełni maksymalną liczbę błędów (porażka).

Zwycięstwo

Gdy gracz odgadnie wszystkie litery w tajnym słowie, gra kończy się zwycięstwem. W tym momencie gracz otrzymuje komunikat o sukcesie (rysunek 6), liczba punktów zwiększa się o 10 (rysunek 7), a następnie gra może być kontynuowana (przez użycie przycisku *"nowa gra"*) lub zakończona.

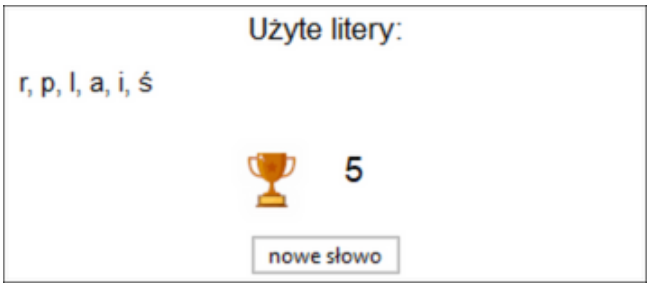


Rysunek 6: Komunikat o zwycięstwie

Rysunek 7: Zwiększenie liczby punktów (diamentów) o 10

Jednocześnie, zwiększana jest również liczba odniesionych zwycięstw (rysunek 8a). Jeżeli gracz wygrał po raz 5, to przy liczbie zwycięstw pojawia się puchar brązowy (najniższy poziom w grze) (rysunek 8b). Po 10 zwycięstwach puchar zmienia się na srebrny, a po 15 na złoty. Puchary odpowiadające kolejnym osiąganym poziomom w grze są przedstawione na rysunku 9.

(a) Zwiększenie liczby zwycięstw



(b) Wyświetlenie pucharu brązowego przy 5-tej wygranej

Rysunek 8: Zwiększanie liczby zwycięstw i wyświetlanie pucharów

(a) Puchar brązowy

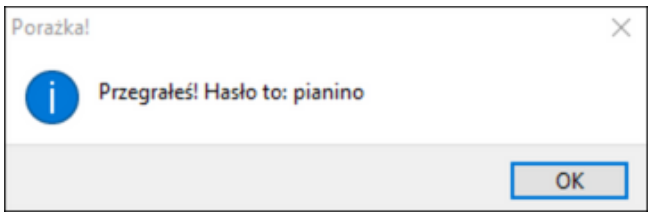
(b) Puchar srebrny

(c) Puchar złoty

Rysunek 9: Puchary odpowiadające kolejnym osiąganym poziomom w grze

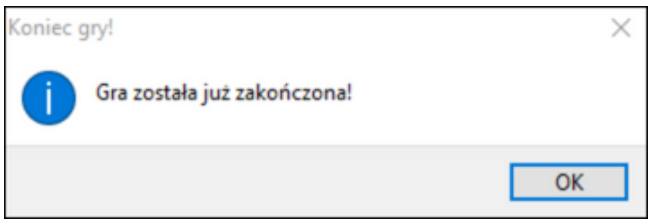
Porażka

Gdy gracz przegra w grze, czyli popełni maksymalną dopuszczalną liczbę błędów (10), następuje zakończenie gry. W tym momencie pojawi się komunikat informujący gracza o przegranej (rysunek 10). Gracz nie otrzymuje punktów za przegraną rundę. Gra może być kontynuowana (przycisk *"nowa gra"*) lub zakończona.



Rysunek 10: Komunikat o porażce

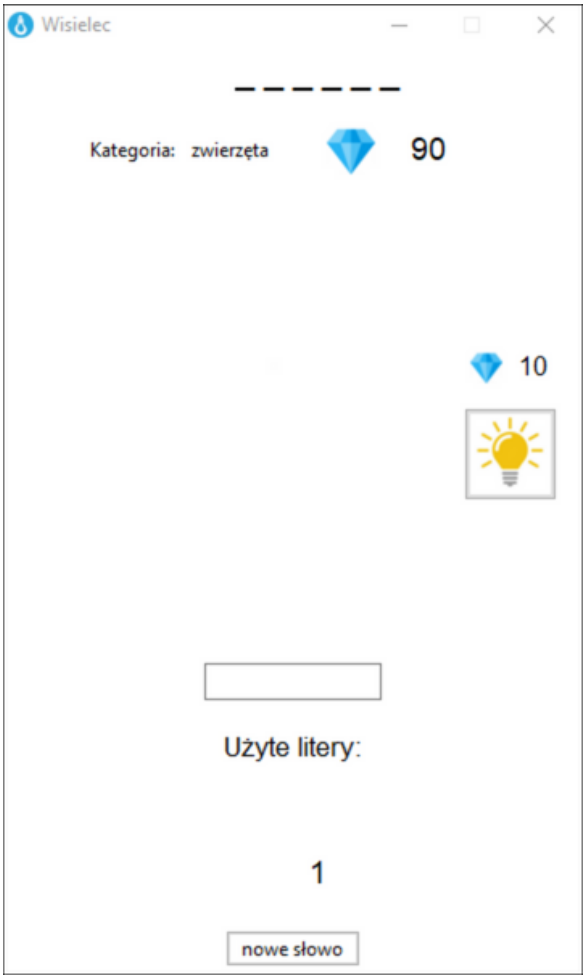
Zarówno w przypadku zwycięstwa, jak i porażki, gdy po wyświetleniu komunikatu o wyniku gry użytkownik spróbuje wpisać kolejną literę, wyświetli się komunikat o końcu gry (rysunek 11).



Rysunek 11: Komunikat o zakończeniu gry

5. Rozpoczęcie nowej rundy

Po zakończeniu aktualnej rundy, gracz może nacisnąć przycisk *"nowe słowo"*, aby rozpocząć kolejną rundę i odgadnąć nowe słowo. Liczba zwycięstw i punkty (diamenty) zostaną zapisane do kolejnej rundy (rysunek 12).



Rysunek 12: Nowa gra

Struktura Kodu

Kod źródłowy składa się z jednego pliku nagłówkowego i jednego pliku cpp.

wisielecMain.h

Plik nagłówkowy w języku C++ definiuje główną klasę dialogową (wwisielecDialog) dla aplikacji opartej na bibliotece wxWidgets. Oto kilka kluczowych elementów pliku:

- **Dołączenie bibliotek:** Plik dołącza standardowe nagłówki biblioteki C++ (<vector> i <fstream>) oraz definiuje makro, które przekształca tekst z kodowania UTF-8 na obiekt wxString w języku C++.

```
//...
#include <vector>
#include <fstream>
#include <wx/msgdlg.h>
//...
#undef _
#define _(s) wxString::FromUTF8(s)
//...
```

Listing 1:

- **Deklaracja klasy:** Jest to deklaracja klasy wwisielecDialog, dziedziczącej po klasie wxDialog, która reprezentuje główne okno dialogowe aplikacji.

```
class wisielecDialog: public wxDialog
{
    // ... (treść klasy)
};
```

Listing 2:

- **Metody publiczne:** Na początku deklarowany jest konstuktor i destruktor klasy wisielecDialog. Następnie dodaliśmy własne funckje umożliwiające lepszą przejżystość kodu i zmniejszenie redundancji.

```
//...
public:
wisielecDialog(wxWindow* parent,wxWindowID id = -1);
virtual ~wisielecDialog();
void InitializeGame();
void UpdateCupImage();
void LoadWordsFromFile(const wxString& fileName);
//...
```

Listing 3:

  - ***void InitializeGame();*** Inicjalizuje nową grę w wisielca, losując słowo i kategorię, resetując stan gry i aktualizując interfejs.
  - ***void UpdateCupImage();*** Aktualizuje obraz pucharu w zależności od liczby wygranych.
  - ***void LoadWordsFromFile(const wxString & fileName);*** Wczytuje słowa i kategorie z pliku tekstowego do dwóch list (wordsList i categoriesList). Każdy wiersz w pliku powinien zawierać parę kategoria-słowo, oddzieloną spacją.

- **Pola prywatne:** Stworzyliśmy kilka zmiennych, używanych przez naszą gre.

```
            //...
            private:
            wxString secretWord;
            wxString guessedWord;
            wxString usedLetters;
            wxString category;
            wxString remainingLetters;
            bool gameFinished;
            int incorrectGuesses;
            int currentHangmanImage = 0;
            int points = 80;
            int wins=0;
            std::vector<wxString> wordsList;
            std::vector<wxString> categoriesList;
            //...
```

Listing 4:

  - ***wxString secretWord;*** Przechowuje ukryte słowo, które gracz musi odgadnąć.
  - ***wxString guessedWord;*** Zawiera aktualne odgadnięte litery z ukrytego słowa w formie tekstowej, gdzie nieodgadnięte litery są zastępowane znakiem podkreślenia.
  - ***wxString usedLetters;*** Zawiera litery, które gracz już użył podczas gry.
  - ***wxString category;*** Przechowuje kategorię, do której należy ukryte słowo.
  - ***wxString remainingLetters;*** Zawiera litery, które pozostały do odgadnięcia w ukrytym słowie.
  - ***bool gameFinished;*** Wartość logiczna informująca o stanie gry (czy gra została zakończona).
  - ***int incorrectGuesses;*** Liczba nieudanych prób odgadnięcia litery.
  - ***int currentHangmanImage=0;***Numer aktualnego obrazu w grze w wisielca, domyślnie zadeklarowany jako 0.
  - ***int points = 80;*** Liczba punktów zdobytych w trakcie gry, domyślnie ustawione na 80.
  - ***int wins = 0;*** Liczba zwycięstw w grze, domyślnoie ustawiona na 0.
  - ***std::<wxString> wordsList;***Wektor przechowujący słowa, które można odgadnąć w grze.
  - ***std::<wxString> categoriesList;*** Wektor przechowujący kategorie odpowiadające słowom w wordsList.

wisielecMain.cpp

Plik *wisielecMain.cpp* zawiera implementację głównego okna aplikacji *"Wisielec"*. Znajdują się w nim definicje funkcji obsługujących zdarzenia interakcji użytkownika. Dodatkowo, plik zawiera definicje funkcji pomocniczych oraz konstruktor i destruktor klasy *wisielecDialog*, które odpowiadają za inicjalizację interfejsu użytkownika. Oto jego podstawowe elementy:

- **Początek pliku:** Na początku dołączamy dyrektywy #include używane do wczytania nagłówków z różnych bibliotek. Wśród nich znajdują się m.in. *wx/msgdlg.h* dla obsługi okien dialogowych wiadomości w wxWidgets, "wisielecMain.h"dla lokalnego pliku nagłówkowego związanego z głównym oknem aplikacji "wisielec", $<$ *sstream* $>$ dla operacji na strumieniach w C++ oraz $<$ *wx/log.h* $>$ dla logowania w aplikacji wxWidgets.

```
#include <wx/msgdlg.h>
//...
#include "wisielecMain.h"
#include <sstream>
#include <wx/log.h>
```

Listing 5:

- **Konstruktor klasy wisielecDialog:** Inicjuje elementy interfejsu użytkownika oraz przygotowuje stan gry do rozpoczęcia, ustawiając ikonę aplikacji oraz inicjując generator liczb pseudolosowych.

```
wisielecDialog::wisielecDialog(wxWindow* parent,wxWindowID id)
{
    //...
    SetIcon(wxICON(aaaa));
    this->FlexGridSizer1 = FlexGridSizer1;
    srand(time(0));
    InitializeGame();
}
```

Listing 6:

  - ***SetIcon(wxICON(aaaa));*** Ustawia ikonę okna dialogowego.
  - ***this->FlexGridSizer1 = FlexGridSizer1;*** Przypisuje wskaźnik do utworzonej siatki rozmieszczenia (*FlexGridSizer1*) do odpowiedniego pola (*FlexGridSizer1*) w obiekcie *wisielecDialog*.
  - ***srand(time(0));*** Inicjalizuje generator liczb pseudolosowych srand przy użyciu aktualnego czasu jako ziarna.
  - ***InitializeGame();*** Wywołuje metodę inicjalizującą grę.

- **OnButton1Click:** Obsługuje zdarzenie przetworzenia nowo wpisanej litery w polu tekstowym. Oto jego kolejne składowe:

  - Pobiera wartość wprowadzoną przez użytkownika do pola tekstowego *TextCtrl*1 i przypisuje ją do zmiennej *input* typu *wxString*. Ta wartość to litera wprowadzona przez użytkownika w celu odgadnięcia słowa w grze.

```
{
    wxString input = TextCtrl1->GetValue();
    //...
```

Listing 7:

– Ten fragment służy do obsługi wprowadzania liter przez użytkownika w polu tekstowym (*TextCtrl*1). Po pobraniu wartości wprowadzonej przez użytkownika do zmiennej *input*, kod sprawdza, czy gra nie została zakończona (!*gameFinished*) oraz czy wprowadzona wartość jest pojedynczą literą alfabetu. Jeśli tak, to sprawdzane jest, czy wprowadzona litera nie została wcześniej użyta. Jeśli nie, jest ona dodawana do listy użytych liter (*usedLetters*).

```
    //...
    if(!gameFinished)
    {
        if (input.length() == 1 && wxIsalpha(input[0]))
        {
            auto guessedLetter = input[0];

            if (usedLetters.find(guessedLetter) == wxString::npos)
            {
                if(usedLetters.length()==0)
                {
                    usedLetters = wxString(guessedLetter);
                }
                else
                {
                    usedLetters +=   ", " + wxString(guessedLetter);
                }
    //...
```

Listing 8:

– Sprawdzenie, czy wprowadzona litera jest częścią tajnego słowa (*secretWord*). Jeśli tak, aktualizowana jest wyświetlana odgadnięta część słowa (*guessedWord*), w której każda litera jest reprezentowana przez znak podkreślenia. Jeśli litera występuje w tajnym słowie, jej odpowiednie miejsce w *guessedWord* jest zastępowane wprowadzoną literą, a wartość zmiennej *correctGuess* jest ustawiana na *true*.

```
    //...
                bool correctGuess = false;
                guessedWord = StaticText1->GetLabel();
                for (size_t i = 0; i < secretWord.length(); ++i)
                {
                    if (secretWord[i] == guessedLetter)
                    {
                        guessedWord[i * 2] = guessedLetter;
                        correctGuess = true;
                    }
                }
            //...
```

Listing 9:

– Jeśli wprowadzona litera została odgadnięta poprawnie (czyli występuje w tajnym słowie), aktualizowane jest wyświetlane odgadnięte słowo (*guessedWord*) na etykiecie statycznej (*StaticText*1). Jeśli nie, nic nie jest aktualizowane.

```
                //...
                if (correctGuess) {
                    StaticText1->SetLabel(guessedWord);
                }
                //...
```

Listing 10:

– Jeśli wszystkie litery w tajnym słowie zostały odgadnięte (nie ma już znaków podkreślenia "_" w *guessedWord*), wyświetlane jest okno dialogowe gratulujące zwycięstwa. Dodatkowo, przycisk *"nowe słowo"* jest aktywowany, a punkty oraz liczba wygranych są zaktualizowane, a następnie wyczyszczony jest *TextCtrl*1, aby użytkownik mógł zacząć odgadywanie nowego słowa.

```
//...
if (guessedWord.find('_') == wxString::npos)
{
    wxMessageBox(_("Brawo! Odgadłeś słowo!"),
        _("Zwycięstwo!"), wxOK | wxICON_INFORMATION);
    TextCtrl1->Clear();
    gameFinished=true;
    wins+=1;
    UpdateCupImage();
    StaticText7->SetLabelText(wxString::Format(_("%d"),
        wins));
    points+=10;
    StaticText6->SetLabel(wxString::Format(_("%d"),
        points));
    return;
}
//...
```

Listing 11:

– Linia $TextCtrl1 -> Clear()$; czyści zawartość kontrolki *TextCtrl*1, która służy do wprowadzania liter przez użytkownika podczas odgadywania słowa. Po udanym odgadnięciu słowa, ta linia usuwa wprowadzoną literę, aby użytkownik mógł zacząć wprowadzać nową literę bez konieczności ręcznego usuwania poprzedniej.

```
//...
TextCtrl1->Clear();
//...
```

Listing 12:

– Ten fragment kodu obsługuje przypadki, gdy wprowadzona litera nie znajduje się w tajnym słowie. Gdy wprowadzona litera nie jest częścią słowa, zwiększa licznik niepoprawnych prób *incorrectGuesses*, co oznacza, że wisielec jest rysowany stopniowo. Następnie aktualizowany jest obraz wisielca poprzez zmianę mapy bitowej na kolejną fazę wisielca. Jeśli liczba niepoprawnych prób osiągnie 10, użytkownik otrzymuje komunikat informujący, że przegrał, a jednocześnie wyświetlone jest tajne słowo.

```
//...
if (!correctGuess)
{
    incorrectGuesses++;

    currentHangmanImage++;

    wxString imageName =
        wxString::Format("wisielec%d.jpg",
        currentHangmanImage);
    wxBitmap hangmanBitmap(imageName,
        wxBITMAP_TYPE_JPEG);

    if (hangmanBitmap.IsOk())
    {
        StaticBitmap1->SetBitmap(hangmanBitmap);
    }

    if (incorrectGuesses == 10)
    {
```

```
                        wxMessageBox(_("Przegrałeś! Hasło to: ") +
                            secretWord, _("Porażka!"), wxOK |
                            wxICON_INFORMATION);
                    }
                }
                //...
```

Listing 13:

– Aktualizowanie etykiety *StaticText3*, aby wyświetlić litery, które zostały już użyte przez gracza. Zawiera to zarówno poprawne, jak i niepoprawne próby zgadywania liter.

```
                //...
                StaticText3->SetLabel(usedLetters);
                }
                //...
```

Listing 14:

– Wyświetlenie komunikatu błędu, jeśli gracz próbuje użyć litery, którą już wcześniej użył. Następnie czyszczone jest pole tekstowe *TextCtrl1*, aby umożliwić graczowi wprowadzenie nowej litery.

```
                //...
                else
                {
                    wxMessageBox(_("Użyłeś już tę literę. Spróbuj
                        inną."), _("Błąd!"), wxOK | wxICON_ERROR);
                    TextCtrl1->Clear();
                }
                }
            }
            //...
```

Listing 15:

– Sprawdzenie, czy gra została już zakończona (*gameFinished*). Jeśli tak, wyświetlany jest komunikat informacyjny o zakończeniu gry. Następnie funkcja kończy swoje działanie.

```
            //...
            else
            {
                wxMessageBox(_("Gra została już zakończona!"),
                    _("Koniec gry!"), wxOK | wxICON_INFORMATION);
                return;
            }
        }
```

Listing 16:

- **InitializeGame:** Funkcja rozpoczyna nową grę, losując słowo i kategorię, resetując stan gry i przywracając ustawione w niej domyślne ustawienia. Oto opis jej kolejnych elementów:

  – Wczytywanie słów i kategorii z pliku *"slowa.txt"*, a następnie losowanie słowa i odpowiadającej mu kategorii.

```
    //...
    {
        LoadWordsFromFile("slowa.txt");

        size_t wordIndex = rand() % wordsList.size();
        secretWord = wordsList[wordIndex];
        category = categoriesList[wordIndex];
        //...
```

Listing 17:

– Resetowanie gry: ustawianie pustej zmiennej *guessedWord* na ciąg znaków "_" o długości równej tajnemu słowu, czyszczenie użytych liter, resetowanie liczby błędnych prób, ustawianie zmiennej *gameFinished* na fałsz (gra nie jest zakończona).

```
        //...
        guessedWord="";
        for (int i=0; i< secretWord.length();i++)
        guessedWord += "_ ";
        usedLetters.clear();
        incorrectGuesses = 0;
        gameFinished = false;
        //...
```

Listing 18:

– Aktualizacja etykiety tekstowej w interfejsie użytkownika: ustawienie wartości dla odgadniętego słowa (*guessedWord*), użytych liter (*usedLetters*), kategorii (*category*), liczby zwycięstw (*wins*), oraz punktów (*points*).

```
        //...
        StaticText1->SetLabel(guessedWord);
        StaticText3->SetLabel(usedLetters);
        StaticText5->SetLabel(category);
        StaticText7->SetLabelText(wxString::Format(_("%d"), wins));
        StaticText6->SetLabel(wxString::Format(_("%d"), points));
        //...
```

Listing 19:

– Ustawienie obrazu wisielca na początkowy, czyszczenie pola tekstowego oraz wczytanie obrazu wisielca o nazwie *"wisielec0.jpg"*.

```
        //...
        currentHangmanImage = 0;
        TextCtrl1->Clear();
        wxString imageName = wxString::Format("wisielec0.jpg");
        wxBitmap hangmanBitmap(imageName, wxBITMAP_TYPE_JPEG);
        //...
```

Listing 20:

– Ten fragment sprawdza, czy obraz wisielca został poprawnie wczytany, a następnie ustawia ten obraz w kontrolce statycznej (*StaticBitmap*1).

```
        //...
        if (hangmanBitmap.IsOk())
        {
            StaticBitmap1->SetBitmap(hangmanBitmap);
        }

    }
```

Listing 21:

- **OnButton2Click:** Funkcja obsługuje kliknięcie przycisku *"nowe słowo"* w interfejsie. Oto kolejne kroki tej funkcji:

  – Wywołanie metody *InitializeGame*(), która przygotowuje nową grę, losując nowe słowo, resetując stan gry i przywracając domyślny obraz wisielca.

```
    {
        InitializeGame();
    }
```

Listing 22:

- **LoadWordsFromFile:** Funkcja obsługuje kliknięcie przycisku *"nowe słowo"* w interfejsie. Oto kolejne kroki tej funkcji:
  - Funkcja otwiera plik tekstowy *fileName* i wczytuje z niego listę słów do odgadnięcia wraz z ich kategoriami. Następnie każde słowo i jego kategoria są dodawane do odpowiednich list *wordsList* i *categoriesList*. Jeśli format wiersza w pliku jest nieprawidłowy, generowany jest odpowiedni komunikat logu. W przypadku błędu otwarcia pliku, wyświetlany jest komunikat błędu, informujący użytkownika o problemie z otwarciem pliku.

```
    {
        std::ifstream file(fileName.ToStdString());
        if (file.is_open()) {
            std::string line;
            while (std::getline(file, line)) {
                std::istringstream iss(line);
                std::string category, word;

                if (iss >> category >> std::ws && std::getline(iss,
                    word)) {
                    wordsList.push_back(wxString::FromUTF8(word.c_str()));
                    categoriesList.push_back(wxString::FromUTF8(category.c_str
                } else {
                    wxLogMessage(_("Nieprawidłowy format wiersza:
                        %s"), wxString::FromUTF8(line.c_str()));
                }
            }
            file.close();
        } else {
            wxLogError(_("Nie udało się otworzyć pliku:%s"),
                wxString::FromUTF8(fileName.c_str()));
            wxMessageBox(wxString::FromUTF8(_("Nie udało się
                otworzyć pliku")) + fileName,
                wxString::FromUTF8(_("Błąd!")), wxOK | wxICON_ERROR);
        }
    }
```

Listing 23:

- **OnBitmapButton1Click:** Metoda obsługuje kliknięcie przycisku podpowiedzi.
  - Jeśli gra nie została zakończona i liczba błędnych prób jest mniejsza niż 10, funkcja sprawdza, czy sekretny wyraz jest pusty. Jeśli tak, wyświetla komunikat o błędzie informujący o braku słowa do odgadnięcia.

```
    {
        if(!gameFinished && incorrectGuesses<10)
        {
            if (secretWord.empty()) {
                wxMessageBox(_("Brak słowa do odgadnięcia!"),
                    _("Błąd!"), wxOK | wxICON_ERROR);
                return;
            }
            //...
```

Listing 24:

– Jeśli gracz ma co najmniej 10 punktów, funkcja sprawdza, które litery w sekretnym wyrazie jeszcze nie zostały odgadnięte. Następnie losuje jedną z tych liter jako podpowiedź dla gracza.

```
            //...
            if (points >= 10)
            {
                wxString remainingLetters;
                for (size_t i = 0; i < secretWord.length(); ++i)
                {
                    if (guessedWord[i * 2] == '_') {
                        remainingLetters += secretWord[i];
                    }
                }

                if (!remainingLetters.empty()) {
                    points -= 10;
                    StaticText6->SetLabel(wxString::Format(_("%d"),
                        points));

                    size_t randomIndex = rand() %
                        remainingLetters.length();
                    char randomLetter =
                        remainingLetters[randomIndex];
                    //...
```

Listing 25:

– Jeśli wylosowana litera nie została jeszcze użyta, dodaje ją do użytych liter. Następnie aktualizuje wyświetlane słowo, zastępując podkreślenia odgadniętymi literami.

```
                    //...
                    if (!usedLetters.empty()) {
                        usedLetters += ",␣";
                    }
                    usedLetters += wxString(randomLetter);
                    StaticText3->SetLabel(usedLetters);

                    for (size_t i = 0; i < secretWord.length(); ++i)
                        {
                        if (secretWord[i] == randomLetter) {
                            guessedWord[i * 2] = randomLetter;
                        }
                    }

                    StaticText1->SetLabel(guessedWord);
                    //...
```

Listing 26:

– Jeśli odgadnięto wszystkie litery w słowie, wyświetla się wiadomość o zwycięstwie, aktualizując wynik gracza i obrazek pucharu oraz dodając punkty. W przeciwnym razie, jeśli liczba błędnych prób osiągnie 10, wyświetla się wiadomość o przegranej.

```
                //...
                if (guessedWord.find('_') == wxString::npos) {
                    wxMessageBox(_("Brawo!␣Odgadłeś␣słowo!"),
                        _("Zwycięstwo!"), wxOK | wxICON_INFORMATION);
                    TextCtrl1->Clear();
                    gameFinished=true;
                    wins += 1;
                    UpdateCupImage();
                    StaticText7->SetLabelText(wxString::Format(_("%d"),
                        wins));
```

```
9                       points += 10;
10                      StaticText6->SetLabel(wxString::Format(_("%d"),
                            points));
11                      return;
12                  }
13                  else if (incorrectGuesses == 10)
14                  {
15                      wxMessageBox(_("Przegrałeś!␣Hasło␣to:␣") +
                            secretWord, _("Porażka!"), wxOK |
                            wxICON_INFORMATION);
16                      TextCtrl1->Clear();
17                      gameFinished = true;
18                  }
19                  //...
```

Listing 27:

– Jeśli wszystkie litery w słowie zostały już odgadnięte, wyświetla się komunikat o braku dostępnych podpowiedzi.

```
1               //...
2               } else
3               {
4               wxMessageBox(_("Wszystkie␣litery␣zostały␣już␣
                    odgadnięte!"), _("Brak␣podpowiedzi"), wxOK |
                    wxICON_INFORMATION);
5               }
6               //...
```

Listing 28:

– Jeśli liczba punktów gracza jest mniejsza niż 10, wyświetla się komunikat informujący go o braku wystarczającej liczby punktów do skorzystania z podpowiedzi.

```
1           //...
2           } else
3           {
4           wxMessageBox(_("Masz␣za␣mało␣diamentów!␣Nie␣możesz␣
                skorzystać␣z␣podpowiedzi!"), _("Brak␣diamentów!"), wxOK |
                wxICON_INFORMATION);
5           }
6           //...
```

Listing 29:

– Jeśli gra została już zakończona, wyświetla się komunikat informujący gracza o końcu gry.

```
1       //...
2           }
3       else
4       {
5           wxMessageBox(_("Gra␣została␣już␣zakończona!"), _("Koniec␣
                gry!"), wxOK | wxICON_INFORMATION);
6           return;
7       }
8   }
```

Listing 30:

- **UpdateCupImage:** Metoda aktualizuje obrazek pucharu na podstawie liczby wygranych gracza.

  Funkcja aktualizuje obrazek pucharu na podstawie liczby wygranych. Jeśli liczba wygranych wynosi od 5 do 9, ustawia obrazek pucharu pierwszego poziomu. Jeśli liczba wygranych wynosi od 10 do 14, ustawia obrazek pucharu drugiego poziomu. Jeśli liczba wygranych wynosi 15 lub więcej, ustawia obrazek pucharu trzeciego poziomu. Na koniec odświeża obrazek pucharu.

```
{
    if (wins >= 5 && wins < 10)
    {
        StaticBitmap3->SetBitmap(wxBitmap(wxImage(("puchar_1.png"))));
    }
    else if (wins >= 10 && wins < 15)
    {
        StaticBitmap3->SetBitmap(wxBitmap(wxImage(_T("puchar_2.png"))));
    }
    else if (wins >= 15)
    {
        StaticBitmap3->SetBitmap(wxBitmap(wxImage(_T("puchar_3.png"))));
    }

    StaticBitmap3->Refresh();
}
```

Listing 31:

Podział pracy

- **Przygotowanie wyglądu okna dialogowego:**
  Bartosz Oleszek 60%
  Katarzyna Zyśko 20%
  Zuzanna Winiarska 20%
- **Napisanie funkcji OnButton1Click:**
  Bartosz Oleszek 100%
  Katarzyna Zyśko 0%
  Zuzanna Winiarska 0%
- **Napisanie funkcji InitializeGame:**
  Bartosz Oleszek 0%
  Katarzyna Zyśko 0%
  Zuzanna Winiarska 100%
- **Napisanie funkcji OnButton2Click:**
  Bartosz Oleszek 0%
  Katarzyna Zyśko 0%
  Zuzanna Winiarska 100%
- **Napisanie funkcji LoadWordsFromFile:**
  Bartosz Oleszek 0%
  Katarzyna Zyśko 0%
  Zuzanna Winiarska 100%
- **Napisanie funkcji OnBitmapButton1Click:**
  Bartosz Oleszek 40%
  Katarzyna Zyśko 0%
  Zuzanna Winiarska 60%
- **Napisanie funkcji UpdateCupImage:**
  Bartosz Oleszek 20%
  Katarzyna Zyśko 20%
  Zuzanna Winiarska 60%
- **Przygotowanie dokumentacji:**
  Bartosz Oleszek 30%
  Katarzyna Zyśko 40%
  Zuzanna Winiarska 30%
- **Ogólnie:**
  Bartosz Oleszek 30%
  Katarzyna Zyśko 25%
  Zuzanna Winiarska 35%

Instrukcja instalacji

Instalator gry powstał przy pomocy aplikacji Createinstall.

Podsumowanie

Gra *"Wisielec"* wyróżnia się przyjemną oprawą graficzną oraz prostą, lecz angażującą mechaniką, co ułatwia szybkie zrozumienie zasad gry. Różnorodność losowanych słów do odgadnięcia, wraz z możliwością korzystania z podpowiedzi, dodaje głębi rozgrywce, zapewniając graczom różnorodne wyzwania. Dodatkowo, zdobywanie pucharów za osiągnięcie określonej liczby zwycięstw stymuluje do dalszej gry, dodając element rywalizacji i motywując do osiągania coraz lepszych wyników.

"Wisielec" nie tylko dostarcza rozrywki, ale także wspomaga rozwój umiejętności logicznego myślenia oraz szybkiego podejmowania decyzji w sposób przystępny i angażujący. Dzięki wbudowanym elementom edukacyjnym stanowi atrakcyjną propozycję dla graczy pragnących zarówno bawić się, jak i doskonalić swoje umiejętności.