Kierunek: Inżynieria i Analiza Danych

Gra karciana

Adam Wróbel, Mateusz Wrzosek, Julia Wronowska

10 lutego 2024

Spis treści

1 Wprowadzenie

Gra to prosta aplikacja napisana w języku C++ przy użyciu biblioteki wxWidgets. Celem gry jest stworzenie prostego interaktywnego środowiska, w którym gracz może współdziałać z przeciwnikiem, używając kart do atakowania, przestwania kart, obrony i leczenia.

2 Cel gry

Celem gry jest pokonanie jednego z przeciwników losowanych przez komputer, w tym przypadku królika, węża lub misia.

3 Mechanika Gry

3.1 Wybór Przeciwnika

Przy każdym uruchomieniu gry losowany jest przeciwnik spośród trzech dostępnych.

3.2 Wybór Karty

Gracz ma trzy karty do wyboru, z których każda ma różne efekty - atak, obrona, leczenie.

3.3 Walka

Gracz i przeciwnik wykonują ruchy na zmianę. Każda karta wybrana przez gracza wpływa na zdrowie i pancerz przeciwnika oraz zdrowie gracza.

3.4 Aktualizacja Stanu

Po każdej rundzie aktualizowany jest stan zdrowia i pancerza gracza i przeciwnika.

3.5 Zakończenie Gry

Gra kończy się, gdy zdrowie gracza lub przeciwnika spadnie do zera.

4 Wymagania Systemowe

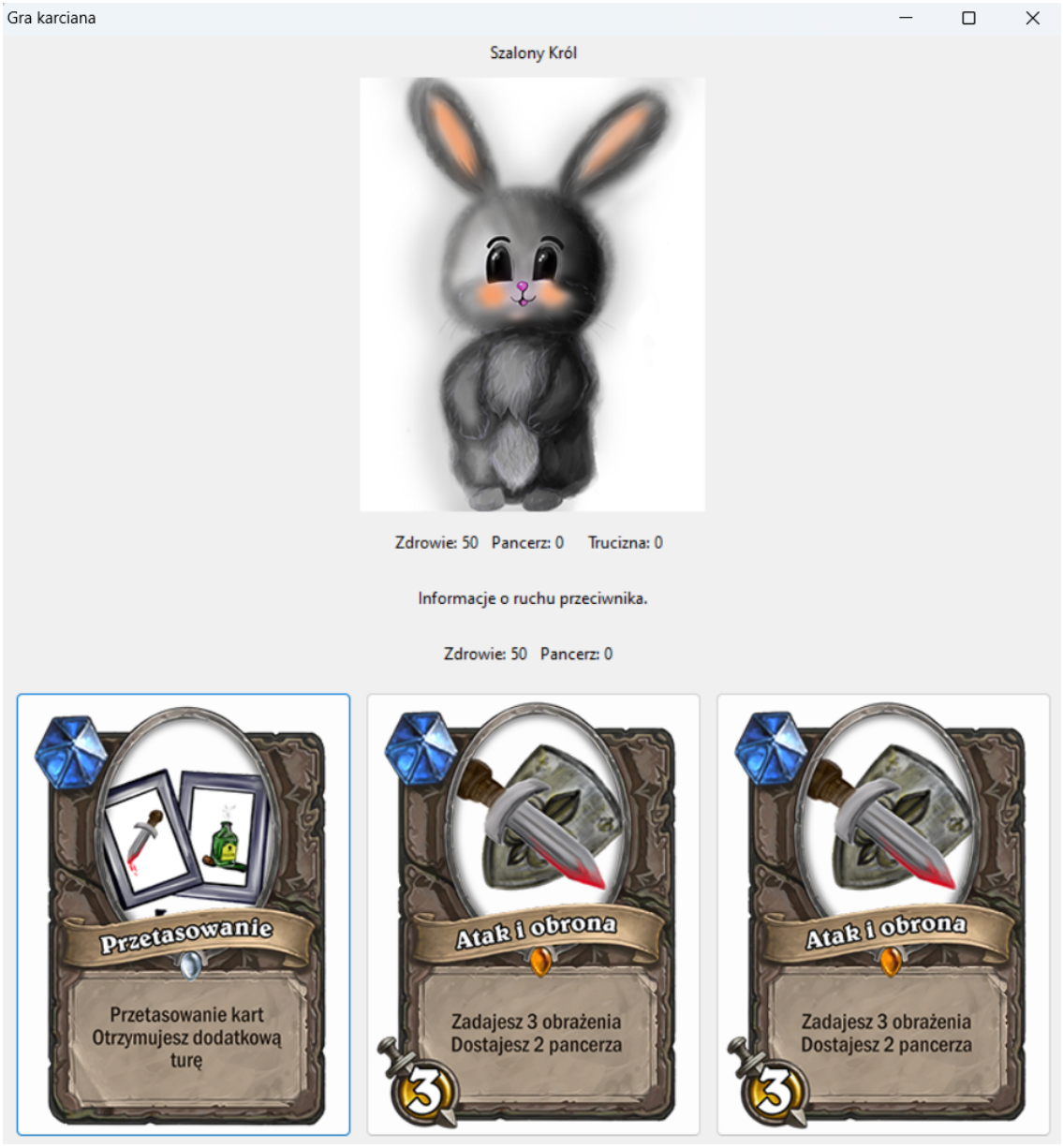
Aby uruchomić grę, wymagane jest środowisko wxWidgets oraz odpowiednie zasoby graficzne (pliki obrazów kart i przeciwników).

5 Elementy Interferfejsu Graficznego

Główne okno gry zawiera pola tekstowe, obrazy, przyciski i inne elementy interfejsu graficznego. Gracz może wybierać karty, atakować przeciwnika i obserwować aktualny stan zdrowia oraz pancerza.

6 Widok główny aplikacji

Aplikacja składa się z okienka z odpowiednim przeciwnikiem, wierszy z pokazanym zdrowiem i pancerzem. Wartości zdrowia i pancerza zmieniają się odpowiednio w zależności od wybranej karty. Najniżej położne są karty, które przedstawiają leczenie, tarcze, przetasowanie kart, atak, trucizna, atak-obrona.



Rysunek 1: Zrzut ekranu z gry.

7 Przeciwnicy

Aplikacja posiada trzech przeciwników, którzy są losowo wybierani przez komputer. Przeciwnik w grze reprezentuje postać, z którą gracz walczy. Przeciwnik ma zdrowie, atak i tarczę, które decydują o przebiegu walki.

7.1 Trudność pokonania przeciwnika

W zależności od danego przeciwnika, zdrowie jest indywidualnie do nich przypisane. Najprościej pokonać króilka, dla którego wyświetla się 50. Drugim przeciwnikiem jest wąż, który ma wartość: 75. Najtrudniejszym do pokonania jest Miś, którego wartość wynosi aż 100.

(a) Królik - Przeciwnik 1. (b) Wąż - Przeciwnik 2. (c) Miś - Przeciwnik 3.

Rysunek 2: Wygląd kart przeciwników

8 Karty do gry

Karty są elementem kluczowym w rozgrywce. Każda karta posiada wartości ataku, obrony, leczenia oraz inne atrybuty. Obrazy kart są wczytywane z plików graficznych.

8.1 Właściwości kart

- Przetasowanie-ta karta pozwala na przetasowanie kart gracza, co może być przydatne, gdy potrzebujesz nowych opcji na ręce.
- Obrona-karta obrony umożliwia zwiększenie pancerza gracza, chroniąc go przed obrażeniami przeciwnika
- Leczenie-karta leczenia pozwala na przywrócenie zdrowia gracza, co jest kluczowe w przypadku odniesienia obrażeń.
- Atak-karta ataku zadaje obrażenia przeciwnikowi, zmniejszając jego ilość zdrowia.
- Trucizna-karta trucizny zadaje przeciwnikowi obrażenia w ciągu kilku rund.
- Obrona-atak- karta łączy zarówno aspekt obrony, jak i ataku, zwiększając pancerz gracza i zadając obrażenia przeciwnikowi.

(a) Karta - Przetasowanie.

(b) Karta - Obrona.

(c) Karta - Leczenie.

(d) Karta - Atak.

(e) Karta - Trucizna.

(f) Karta - Atak i Obrona.

Rysunek 3: Opisy kart.

9 Struktura kodu

9.1 Struktura projektu

- `ProjektMain.cpp`: Główny plik źródłowy aplikacji, zawierający klasę `ProjektDialog`, która reprezentuje główne okno gry.
- `ProjektMain.h`: Plik nagłówkowy klasy `ProjektDialog`.
- `ProjektApp.h` i `ProjektApp.cpp`: Pliki reprezentujące klasę `ProjektApp`, dziedziczącą po `wxApp`, odpowiedzialną za inicjalizację aplikacji.
- `cards.h` i `cards.cpp`: Pliki zawierające implementację klasy `cards`, reprezentującej karty dostępne w grze.

9.2 Fragmenty kodu

ProjektMain.cpp

ProjektDialog::ProjektDialog(wxWindow* parent, wxWindowID id)

- Inicjalizuje interfejs graficzny gry, ustawiając okna, przyciski i obrazy.
- Przechowuje informacje o stanie gry, takie jak zdrowie, pancerz, obrazy kart itp.

```
ProjektDialog::ProjektDialog(wxWindow* parent,wxWindowID id)
{
    //(*Initialize(ProjektDialog)
    wxBoxSizer* BoxSizer1;
    wxBoxSizer* BoxSizer2;
    wxBoxSizer* BoxSizer3;
    wxFlexGridSizer* FlexGridSizer1;...
```

ProjektDialog::OnShowRandomImage()

- Losuje i ustawia losowe obrazki kart na przyciskach gracza.

```
void ProjektDialog::OnShowRandomImage() //Wylosowanie przeciwnika
{
    losowyIndexWew;
    StaticBitmap1->SetBitmap(Przeciwnicy[losowyIndexWew]);
}

void ProjektDialog::OnShowRandomImage2() //Ustawienie zdjęć
{
    auto seed = std::chrono::high_resolution_clock::now().time_since_epoch().count();
    std::mt19937_64 rng(seed);
    std::uniform_int_distribution<int> distribution(0, 5);;
    for(int i=0;i<3;i++)
    {
        int losowyIndex = distribution(rng);
        wxBitmapButton* wybranyPrzycisk;
        switch (i) {
            case 0:
                wybranyPrzycisk = BitmapButton1;
                break;
```

```
            case 1:
                wybranyPrzycisk = BitmapButton2;
                break;
            case 2:
                wybranyPrzycisk = BitmapButton3;
                break;
            default:
                break;
        }
        wybranyPrzycisk->SetBitmapLabel(Karty[losowyIndex]);
        wybranyPrzycisk->SetBitmapPressed(Karty[losowyIndex]);
        wybranyPrzycisk->SetBitmapCurrent(Karty[losowyIndex]);
        wybranyPrzycisk->SetBitmapHover(Karty[losowyIndex]);
        wybranyPrzycisk->SetBitmapSelected(Karty[losowyIndex]);
        wybranyPrzycisk->Update();
        this->Refresh();
}

}
```

ProjektDialog::OnBitmapButtonClick(wxCommandEvent& event)

- Obsługuje kliknięcia w przyciski kart gracza, wywołując odpowiednie funkcje zależne od rodzaju karty.

ProjektDialog::DecreaseHealth()

- Zmniejsza zdrowie przeciwnika po ataku gracza. Aktualizuje etykietę z informacją o zdrowiu i pancerzu.

```
void ProjektDialog::DecreaseHealth() //Obrażenia
{
    cards atak(6,0,0,0,0,0,"Atak.png");
    int attackValue=atak.GetAttack();
    int PancerzBlokada = PancerzPrzeciwnika - attackValue;
    if(PancerzBlokada <= 0)
    {
        Zycie += PancerzBlokada;
        PancerzPrzeciwnika = 0;
        StaticText3 -> SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
        StaticText2 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), Zycie));
        return;
    }
    else
    {
        PancerzPrzeciwnika=PancerzPrzeciwnika-attackValue;
        StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
        return;
    }
}
```

ProjektDialog::IncreaseArmor()

- Zwiększa pancerz gracza po użyciu karty obrony. Aktualizuje etykietę z informacją o pancerzu.

```
void ProjektDialog::IncreaseArmor() // Pancerz
{
```

```
    cards pancerz(0,5,0,0,0,0,"Obrona.png");
    int armorValue=pancerz.GetArmor();
    NaszPancerz += armorValue;
    StaticText5 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), NaszPancerz));
    return;
}
```

ProjektDialog::Healing()

- Uzdrawia gracza po użyciu karty leczenia. Aktualizuje etykietę z informacją o zdrowiu.

```
void ProjektDialog::Healing() //Leczenie
{
    cards leczenie(0,0,3,0,0,0,"Leczenie.png");
    int healValue=leczenie.GetHeal();
    if(NaszeZycie >= 50)
    {
        return;
    }
    NaszeZycie += healValue;
    if(NaszeZycie >= 50)
    {
        NaszeZycie = 50;
        StaticText4 ->  SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), NaszeZycie));
    }
    StaticText4 ->  SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), NaszeZycie));
    return;
}
```

ProjektDialog::Shuffle()

- Tasuje karty gracza po użyciu karty przetasowania.

```
void ProjektDialog::Shuffle() //Przetasowanie
{
    cards Przetasowanie(0,0,0,1,0,0,"Przetasowanie.png");
    bool shuffleValue=Przetasowanie.GetRefresh();
    if(shuffleValue==1)
    {
        OnShowRandomImage2();
        Tura = 1;
    }
    return;
}
```

ProjektDialog::AttackAndArmor()

- Obsługuje kartę łączącą atak i obronę. Zadaje obrażenia przeciwnikowi i zwiększa pancerz gracza.

```
void ProjektDialog::AttackAndArmor()
{
    cards atak_obrona(3,2,0,0,0,0,"AtakObrona.png");
    int attackValue=atak_obrona.GetAttack();
    int armorValue=atak_obrona.GetArmor();
    int PancerzBlokada = PancerzPrzeciwnika - attackValue;
```

```
    NaszPancerz += armorValue;
    StaticText5 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), NaszPancerz));
    if(PancerzBlokada <= 0)
    {
        Zycie += PancerzBlokada;
        PancerzPrzeciwnika = 0;
        StaticText3 -> SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
        StaticText2 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), Zycie));
        return;
    }
    else
    {
        PancerzPrzeciwnika=PancerzPrzeciwnika-attackValue;
        StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
        return;
    }
}
```

ProjektDialog::Poison()

- Obsługuje kartę trucizny. Dodaje truciznę przeciwnikowi.

```
void ProjektDialog::Poison()
{
    cards trucizna(0,0,0,0,0,2,"Trucizna.png");
    Trucizna += trucizna.GetTrucizna();
}
```

ProjektDialog::ObrazeniaPoison()

- Funkcja odpowiada za zadawanie obrażeń truciznę przeciwnikowi.
- Oblicza obrażenia związane z trucizną i aktualizuje etykiety na interfejsie użytkownika (GUI).

```
void ProjektDialog::ObrazeniaPoison()
{
    int PancerzBlokada = PancerzPrzeciwnika - Trucizna;
    if(PancerzBlokada <= 0)
    {
        Zycie += PancerzBlokada;
        PancerzPrzeciwnika = 0;
        StaticText3 -> SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
        StaticText2 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), Zycie));
    }
    else
    {
        PancerzPrzeciwnika=PancerzPrzeciwnika-Trucizna;
        StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
    }
    Trucizna--;
    StaticText6 -> SetLabel(wxT("Trucizna: ") + wxString::Format(wxT("%d"), Trucizna));
}
```

ProjektDialog::OnBitmapButtonClick(wxCommandEvent& event)

- Funkcja obsługuje zdarzenie kliknięcia na przycisku obrazkowym.
- Sprawdza, który przycisk został kliknięty, porównując obrazy na przycisku z zdefiniowanymi obrazami kart.
- W zależności od klikniętej karty wykonuje odpowiednie akcje, takie jak zadawanie obrażeń, zwiększanie pancerza, leczenie, tasowanie, atak i obrona, czy zadawanie trucizny.

```
void ProjektDialog::OnBitmapButtonClick(wxCommandEvent& event) //sprawdza które pole zostało klinkięte
{
    wxBitmapButton* clickedButton = dynamic_cast<wxBitmapButton*>(event.GetEventObject());
    if (clickedButton) // sprawdza które pole zostało klinkięte
    {
        int indeksPrzycisku = -1;
        for (int i = 1; i < 4; i++)
        {
            if (clickedButton == przyciskiKart[i])
            {
                indeksPrzycisku = i;
                break;
            }
        }

        if (indeksPrzycisku != -1) // sprawdza kartę na tym polu
        {
            wxBitmap bitmapOnButton = clickedButton->GetBitmapLabel();
            wxBitmap bitmapAtak = Karty[0];
            wxBitmap bitmapPancerz = Karty[1];
            wxBitmap bitmapLeczenie = Karty[2];
            wxBitmap bitmapTasuj = Karty[3];
            wxBitmap bitmapAtakObrona = Karty[4];
            wxBitmap bitmapTrucizna = Karty[5];

            if (bitmapOnButton.IsSameAs(bitmapAtak))
            {
                DecreaseHealth();
            }

            if (bitmapOnButton.IsSameAs(bitmapPancerz))
            {
                IncreaseArmor();
            }

            if (bitmapOnButton.IsSameAs(bitmapLeczenie))
            {
                Healing();
            }
            if (bitmapOnButton.IsSameAs(bitmapTasuj))
            {
                Shuffle();
            }
            if(bitmapOnButton.IsSameAs(bitmapAtakObrona))
            {
                AttackAndArmor();
            }
```

```
            if(bitmapOnButton.IsSameAs(bitmapTrucizna))
            {
                Poison();
            }
        }
    }
}
```

ProjektDialog::RuchPrzeciwnika()

- Funkcja implementuje ruch przeciwnika w grze.
- Losuje ataki przeciwnika na podstawie typu przeciwnika (krolik, waz, mis) i aktualizuje etykiety na interfejsie użytkownika.

```
void ProjektDialog::RuchPrzeciwnika() // ruch przeciwnika
{
    auto seed = std::chrono::high_resolution_clock::now().time_since_epoch().count();
    std::mt19937_64 rng(seed);
    std::uniform_int_distribution<int> distribution(0, 2);
    int LosowyAtak = distribution(rng);
    if(losowyIndexWew==0) //krolik
    {
        if(LosowyAtak==0)
        {
            NaszeZycie -= 4;
            StaticText4 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), NaszeZycie));
            StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 4 obrażenia."));
            return;
        }
        if(LosowyAtak==1)
        {
            int PancerzBlokada = NaszPancerz - 10;
            if(PancerzBlokada <= 0)
            {
                NaszeZycie += PancerzBlokada;
                StaticText4 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), NaszeZycie));
                StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 10 obrażeń."));
                return;
            }
            StaticText5 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzBlokada));
            StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 10 obrażeń."));
            return;
        }
        PancerzPrzeciwnika += 3;
        StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
        StaticText7 -> SetLabel(wxT("Przeciwnik otrzymał 3 pancerza."));
        return;
    }
    if(losowyIndexWew==1) //waz
    {
        if(LosowyAtak==0)
        {
            int PancerzBlokada = NaszPancerz - 5;
            if(PancerzBlokada <= 0)
```

```
        {
            NaszeZycie += PancerzBlokada;
            StaticText4 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), NaszeZycie));
            StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 5 obrażeń."));
            return;
        }
        StaticText5 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzBlokada));
        StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 5 obrażeń."));
        return;
    }
    if(LosowyAtak==1)
    {
        NaszPancerz = NaszPancerz/2;
        StaticText5 -> SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), NaszPancerz));
        StaticText7 -> SetLabel(wxT("Przeciwnik zmniejszył twój pancerz o połowę."));
        return;
    }
    PancerzPrzeciwnika += 4;
    StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
    StaticText7 -> SetLabel(wxT("Przeciwnik otrzymał 4 pancerza."));
    return;
}
if(losowyIndexWew==2) //mis
{
    if(LosowyAtak==0)
    {
        if(Zycie >= 100)
        {
            StaticText7 -> SetLabel(wxT("Przeciwnik uleczył się o 4."));
            return;
        }
        Zycie += 4;
        if(Zycie >= 100)
        {
            Zycie = 100;
            StaticText2 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), Zycie));
            StaticText7 -> SetLabel(wxT("Przeciwnik uleczył się o 4."));
            return;
        }
        StaticText2 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), Zycie));
        StaticText7 -> SetLabel(wxT("Przeciwnik uleczył się o 4."));
        return;
    }
    if(LosowyAtak==1)
    {
        int PancerzBlokada = NaszPancerz - 8;
        if(PancerzBlokada <= 0)
        {
            NaszeZycie += PancerzBlokada;
            StaticText4 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), NaszeZycie));
            StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 8 obrażeń."));
            return;
        }
        StaticText5 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzBlokada));
```

```
            StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 8 obrażeń."));
            return;
        }
        PancerzPrzeciwnika += 5;
        StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
        StaticText7 -> SetLabel(wxT("Przeciwnik otrzymał 5 pancerza."));
        return;
    }
}
```

ProjektDialog::InitializeGame()

- Inicjalizuje stan gry po pierwszej wygranej walce, ustawiając losowego przeciwnika, ich atrybuty i aktualizując etykiety GUI.

```
void ProjektDialog::InitializeGame()
{
srand(time(nullptr));
losowyIndexWew = rand() % 3;
OnShowRandomImage();
OnShowRandomImage2();

if(losowyIndexWew==0)
{
    Zycie = 50;
    StaticText2 ->  SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), Zycie));
    StaticText1 ->  SetLabel(wxT("Szalony Król"));
    StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
    StaticText6 ->  SetLabel(wxT("Trucizna: ") + wxString::Format(wxT("%d"), Trucizna));
}
if(losowyIndexWew==1)
{
    Zycie = 75;
    StaticText2 ->  SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), Zycie));
    StaticText1 ->  SetLabel(wxT("Rzeczny Wąż"));
    StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
    StaticText6 ->  SetLabel(wxT("Trucizna: ") + wxString::Format(wxT("%d"), Trucizna));
}
if(losowyIndexWew==2)
{
    Zycie = 100;
    StaticText2 ->  SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), Zycie));
    StaticText1 ->  SetLabel(wxT("Brat Niedzwiedź"));
    StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
    StaticText6 ->  SetLabel(wxT("Trucizna: ") + wxString::Format(wxT("%d"), Trucizna));
}
NaszPancerz = 0;
NaszeZycie = 50;
StaticText4 ->  SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), NaszeZycie));
StaticText5 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), NaszPancerz));
}
```

ProjektDialog::Gra(wxCommandEvent& event)

- Główna funkcja obsługująca rozgrywkę.

- Wywołuje funkcję obsługi kliknięcia na karcie, aktualizuje interfejs, wykonuje ruch przeciwnika, sprawdza warunki wygranej/przegranej i wyświetla stosowne komunikaty.

```
void ProjektDialog::Gra(wxCommandEvent& event) //działanie gry
{
    OnBitmapButtonClick(event);
    OnShowRandomImage2();
    if(Tura==1)
    {
        Tura = 0;
        return;
    }
    RuchPrzeciwnika();
    if(Trucizna > 0)
    {
        ObrazeniaPoison();
    }
    if(NaszeZycie<=0)
    {
        wxMessageBox("Przegrana! Zostales pokonany.", "Koniec gry", wxICON_ERROR | wxOK);
        InitializeGame();
    }
    if(Zycie<=0)
    {
        wxMessageBox("Gratulacje! Przeciwnik pokonany.", "Wygrana", wxICON_INFORMATION | wxOK);
        InitializeGame();
    }
}
```

10 Karty przeciwnika

Karty przeciwnika

1. ProjektDialog::RuchPrzeciwnika():

- Symuluje ruch przeciwnika, podejmując losowe decyzje dotyczące ataku, obrony lub uleczenia.

```
auto seed = std::chrono::high_resolution_clock::now().time_since_epoch().count();
std::mt19937_64 rng(seed);
std::uniform_int_distribution<int> distribution(0, 2);
int LosowyAtak = distribution(rng);
if(losowyIndexWew==0) //krolik
{
if(LosowyAtak==0)
{
NaszeZycie -= 4;
StaticText4 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), NaszeZycie));
StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 4 obrażenia."));
return;
}
if(LosowyAtak==1)
{
int PancerzBlokada = NaszPancerz - 10;
```

```
if(PancerzBlokada <= 0)
{
NaszeZycie += PancerzBlokada;
StaticText4 -> SetLabel(wxT("Zdrowie: ") + wxString::Format(wxT("%d"), NaszeZycie));
StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 10 obrażeń."));
return;
}
StaticText5 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzBlokada));
StaticText7 -> SetLabel(wxT("Przeciwnik zadał ci 10 obrażeń."));
return;
}
PancerzPrzeciwnika += 3;
StaticText3 ->  SetLabel(wxT("Pancerz: ") + wxString::Format(wxT("%d"), PancerzPrzeciwnika));
StaticText7 -> SetLabel(wxT("Przeciwnik otrzymał 3 pancerza."));
return;
```

11 Podsumowanie

Projekt stanowi dobrze zorganizowaną grę karcianą, gdzie gracz i przeciwnik używają kart do zadawania obrażeń i obrony. Interfejs graficzny ułatwia graczowi interakcję, a różnorodność kart dodaje element strategii. Kod jest czytelny i logiczny, oparty na dwóch kluczowych bibliotekach, co ułatwia rozwijanie i utrzymanie gry.

12 Podział pracy

- **Przygotowanie wyglądu okna dialogowego:**
  Adam Wróbel: 25%
  Mateusz Wrzosek: 25%
  Julia Wronowska: 50%
- **Stworzenie klasy cards**
  Adam Wróbel: 30%
  Mateusz Wrzosek: 60%
  Julia Wronowska: 10%
- **Funkcje opisujące działanie kart:**
  Adam Wróbel: 40%
  Mateusz Wrzosek: 50%
  Julia Wronowska: 10%
- **Funkcja dotycząca przeciwnika i jego ruchu:**
  Adam Wróbel: 100%
  Mateusz Wrzosek: 0%
  Julia Wronowska: 0%
- **Funkcja odpowiadająca za interakcje przycisków:**
  Adam Wróbel: 0%
  Mateusz Wrzosek: 100%
  Julia Wronowska: 0%
- **Funkcja odpowiadająca za działanie gry:**
  Adam Wróbel: 100%
  Mateusz Wrzosek: 0%
  Julia Wronowska: 0%
- **Przygotowanie dokumentacji:**
  Adam Wróbel: 5%
  Mateusz Wrzosek: 5%
  Julia Wronowska: 90%
- **Przygotanie plików instalacyjnych:**
  Adam Wróbel: 100%
  Mateusz Wrzosek: 0%
  Julia Wronowska: 0%