

Sprawozdanie

Temat projektu

Gra w UNO

Autor: Bartłomiej Zientek

Data: 27.06.2024

Opis tematu

UNO to popularna gra karciana, która jest znana na całym świecie. Jej celem jest pozbycie się wszystkich swoich kart jako pierwszy. Gra ta jest prosta ale oferuje dużo zabawy i strategicznego myślenia. Talia UNO składa się z 108 kart w czterech kolorach (czerwonym, niebieskim, zielonym i żółtym) oraz kart specjalnych, takich jak „Skip”, „Reverse”, „Draw Two”, „Wild Draw Four”. Wszystkie karty oprócz kart specjalnych, są numerowane od 0 do 9 w każdym z czterech kolorów.

Karty specjalne

- "Skip" (Pominięcie) – następny gracz traci kolejkę.
- "Reverse" (Zmiana kierunku) – zmienia kierunek gry.
- "Draw Two" (Dobierz dwie) – następny gracz dobiera dwie karty i traci kolejkę.
- "Wild" (Dzika karta) – pozwala graczowi wybrać dowolny kolor do kontynuacji gry.
- "Wild Draw Four" (Dzika karta, dobierz cztery) – pozwala graczowi wybrać kolor oraz wymusza na następnym graczu dobranie czterech kart.

Rozgrywka

1. Rozdanie: Każdy gracz otrzymuje 7 kart. Reszta kart tworzy stos do dobierania, a pierwsza karta ze stosu dobierania jest odkrywana i umieszczana na stosie odrzutów.
2. Przebieg gry: Gracze na zmianę kładą karty na stosie odrzutów, starając się dopasować kartę do tej, która jest na wierzchu stosu odrzutów, pod względem koloru lub liczby. Można również użyć kart specjalnych.
3. UNO: Gdy gracz ma tylko jedną kartę, musi krzyknąć "UNO". Jeśli tego nie zrobi i zostanie na tym przyłapany przed kolejną turą, musi dobrać dwie karty.
4. Zakończenie gry: Gra kończy się, gdy jeden z graczy pozbędzie się wszystkich swoich kart. Punkty są liczone na podstawie kart, które pozostają w rękach przeciwników.

Punktacja

-Karty numerowane (0-9) – wartość równa numerowi karty.

Karty specjalne:

- "Skip" – 20 punktów.
- "Reverse" – 20 punktów.
- "Draw Two" – 20 punktów.
- "Wild" – 50 punktów.
- "Wild Draw Four" – 50 punktów.

Analiza tematu

Doprecyzowanie tematu

Tematem projektu jest implementacja gry karcianej UNO, popularnej gry towarzyskiej, która jest rozgrywana przy użyciu specjalnej talii kart. Celem projektu jest stworzenie gry, która umożliwia rozgrywkę między graczem oraz komputerowym przeciwnikiem, z wykorzystaniem różnych mechanizmów programowania obiektowego i współczesnych technologii programistycznych.

Uzasadnienie wyboru klas

- BasePlayer i Player: Klasy te reprezentują graczy w grze. Klasa `Player` dziedziczy po `BasePlayer`, co umożliwia łatwe dodanie kolejnych typów graczy (np. przeciwnika komputerowego reprezentowanego przez klasę `Opponent`).
- BaseCard i Card: Klasy reprezentują karty używane w grze. `BaseCard` stanowi bazową klasę, a `Card` zawiera szczegółowe właściwości kart, takie jak kolor, typ akcji, itp.
- Deck: Klasa ta reprezentuje talię kart, umożliwiającą tasowanie, dobieranie kart i inne operacje związane z zarządzaniem talią.
- DiscardPile: Klasa reprezentuje stos kart odrzuconych, co jest kluczowe dla mechaniki gry, umożliwiając odkładanie kart i zarządzanie kolorami kart na wierzchu stosu.
- ActionType i CardColor: Są to klasy wyliczeniowe, które definiują możliwe typy akcji kart i kolory kart, co zapewnia lepszą czytelność i łatwiejsze zarządzanie tymi właściwościami w kodzie.

Uzasadnienie wyboru bibliotek

- Raylib: Biblioteka ta została wybrana ze względu na swoją prostotę i łatwość użycia w tworzeniu grafiki 2D. Raylib wspiera szybkie tworzenie prototypów oraz zapewnia funkcje niezbędne do renderowania elementów graficznych gry, co jest kluczowe dla interaktywności i atrakcyjności wizualnej gry UNO.
- Regex (wyrażenia regularne): Wyrażenia regularne są używane do walidacji i przetwarzania danych wejściowych od użytkowników. Umożliwiają one efektywne sprawdzanie poprawności danych, co jest niezbędne np. przy wprowadzaniu nazw graczy czy przy analizie tekstowych komend gry.
- Filesystem: Biblioteka ta umożliwia operacje na systemie plików, takie jak zapisywanie loginów graczy, odczytywanie zapisanych gier, itp. Dzięki temu gra może mieć funkcjonalność zapisywania i wczytywania stanu rozgrywki, co jest ważne dla użytkowników chcących kontynuować grę później.
- Threads (wątki): Wątki są używane do zarządzania równoczesnymi operacjami, co kolejno jest przydatne np. przy implementacji sztucznej inteligencji przeciwnika komputerowego, która może wymagać równoczesnego przetwarzania w tle bez zakłócania głównego wątku gry.
- Wykorzystano również moduły: Strukturyzacja kodu w moduły pozwala na lepszą organizację i modularność aplikacji. Moduły umożliwiają podział kodu na logiczne części, co zwiększa jego czytelność i ułatwia zarządzanie dużymi projektami.

Specyfikacja zewnętrzna

Instrukcja dla użytkownika

Gra UNO to interaktywna gra karciana, której celem jest pozbycie się wszystkich kart z ręki przed innymi graczami. Poniższa instrukcja przeprowadza przez proces logowania, rozpoczęcia gry oraz samej rozgrywki.

1) Logowanie

Po uruchomieniu gry następuje przeniesienie do ekranu logowania. Na ekranie pojawi się formularz logowania, gdzie należy wpisać swoje dane logowania. Jeśli dane logowania są poprawne, następuje przeniesienie do menu głównego. Jeśli dane logowania są niepoprawne, następuje prośba o ponowne wpisanie danych.

2) Menu główne

Po poprawnym zalogowaniu pojawi się menu główne z opcjami do wyboru.

- Start

- Wyjście

Wybór opcji "Start" spowoduje rozpoczęcie gry. Wybór opcji "Wyjście" spowoduje zakończenie działania aplikacji.

3) Rozgrywka

Po wybraniu opcji "Start" gra się rozpoczyna.

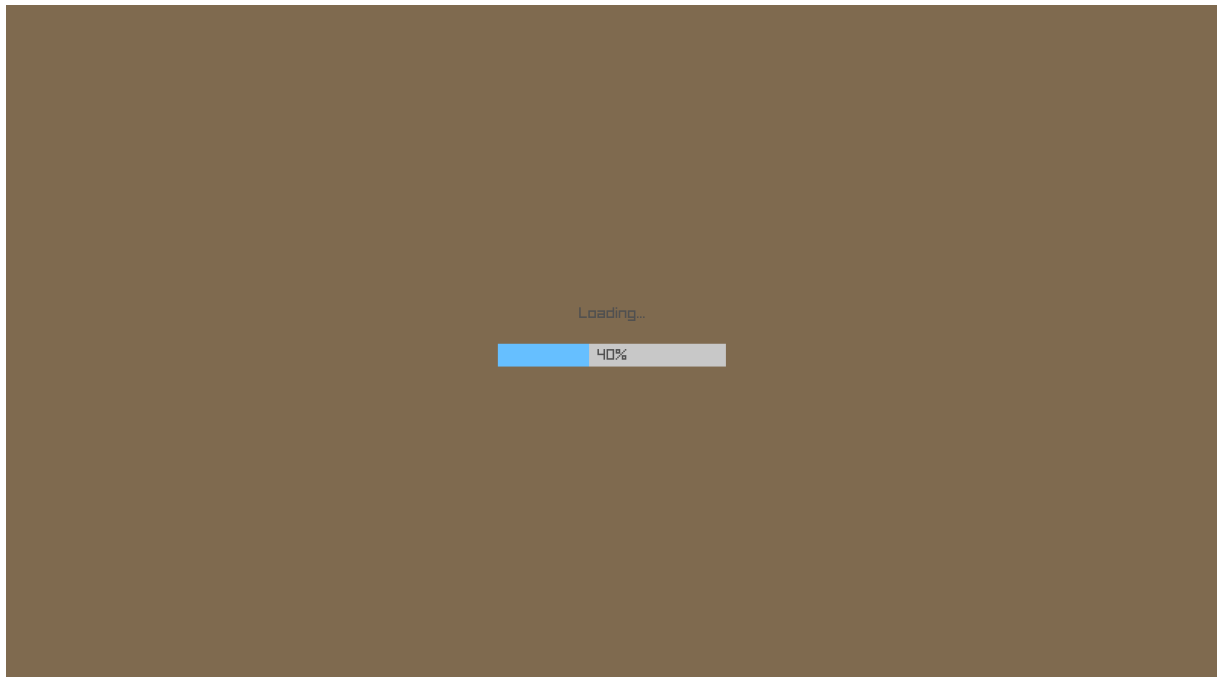
Wykonanie ruchu poprzez zagranie karty z ręki zgodnie z zasadami gry UNO. W przypadku braku możliwości zagrania karty, należy dobrać kartę z talii. Po wykonaniu ruchu, wprowadzane są odpowiednie zmiany do ręki kart oraz na planszę gry. Gdy zostanie użyta karta akcji `Wild`, pojawiają się 4 przyciski do zmiany koloru karty. Gdy zostanie użyta karta `Stop` następuje zablokowanie przeciwnika a następnie możliwe jest wykonanie kolejnego ruchu. Po wykonaniu ruchu przeciwnik komputerowy wykonuje swój ruch zgodnie z zasadami gry UNO. Po ruchu przeciwnika, wprowadzane są odpowiednie zmiany do jego ręki kart oraz na planszę gry.

4) Sprawdzenie warunków zwycięstwa

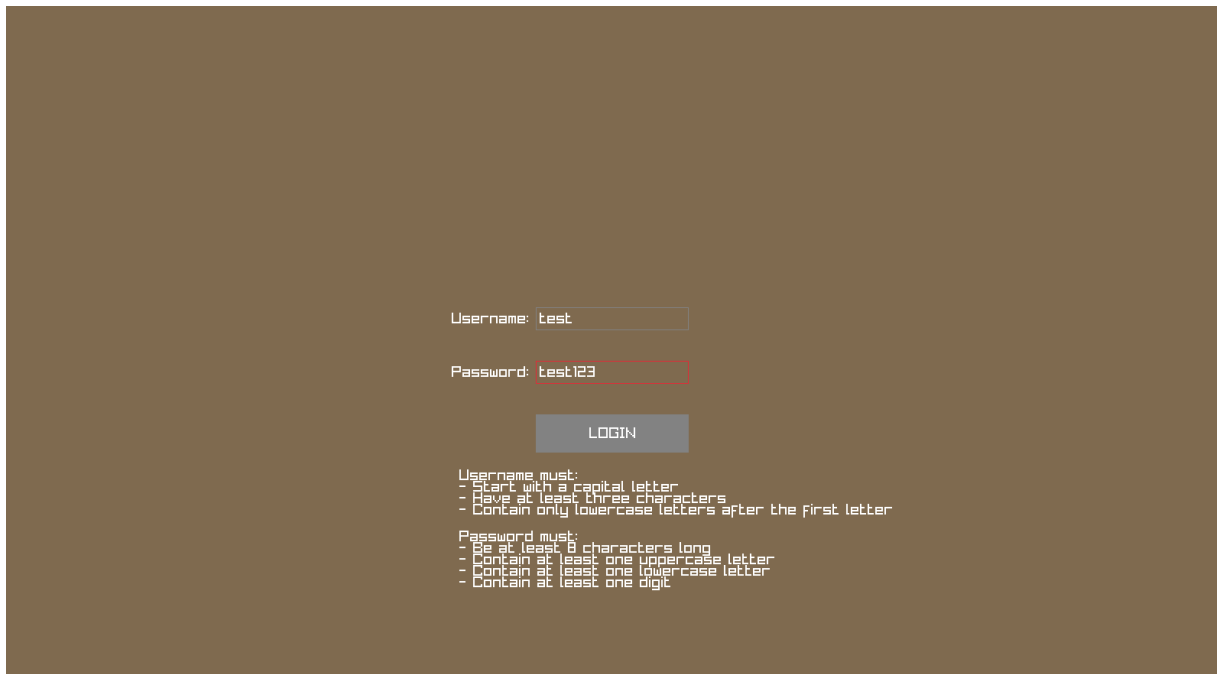
Gra sprawdza, czy któryś z graczy (użytkownik lub przeciwnik) wygrał, pozbywając się wszystkich kart z ręki. W przypadku zwycięstwa jednego z graczy, wyświetlana jest plansza końcowa z wynikami gry. Ekran pozwala na przejście do menu gry lub wyłączenie aplikacji.

Zrzuty ekranu:

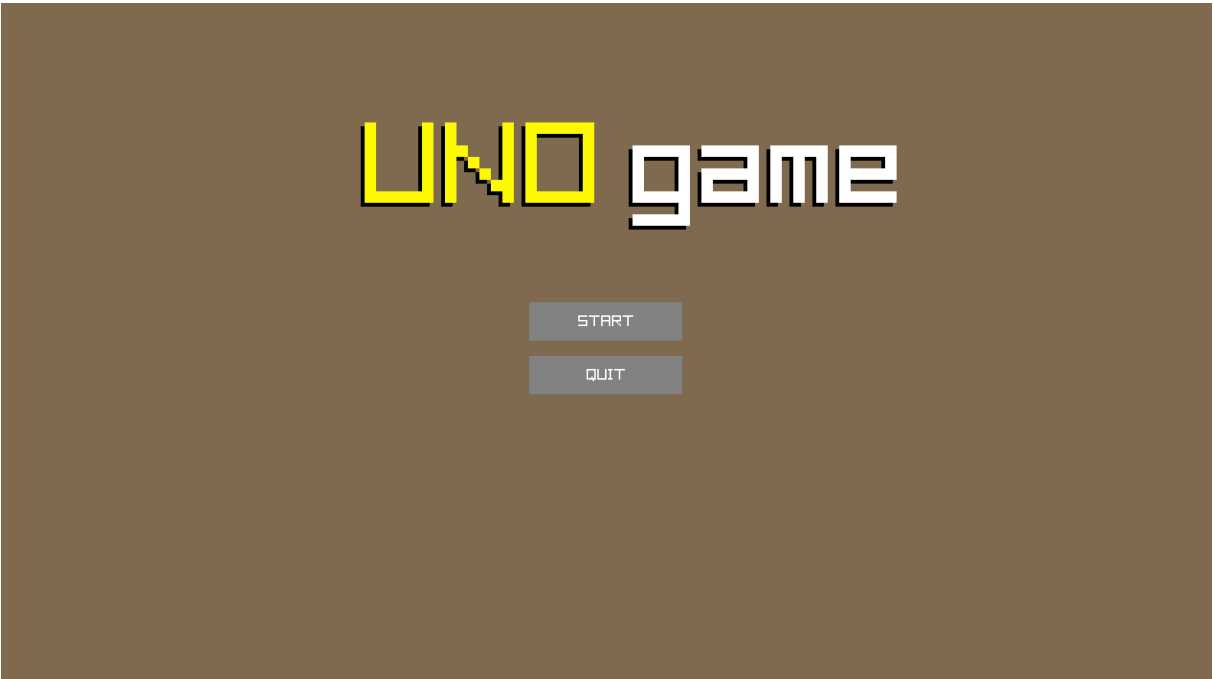
Ładowanie



Login

A screenshot of a login form on a dark brown background. The form consists of two input fields: "Username: test" and "Password: test123". The "Password" field has a red border, indicating it is required or has an error. Below the fields is a gray button labeled "LOGIN". Under the button, there are two sections of validation rules. The first section, "Username must:", lists three rules: "Start with a capital letter", "Have at least three characters", and "Contain only lowercase letters after the first letter". The second section, "Password must:", lists three rules: "Be at least 8 characters long", "Contain at least one uppercase letter", and "Contain at least one digit".

Menu

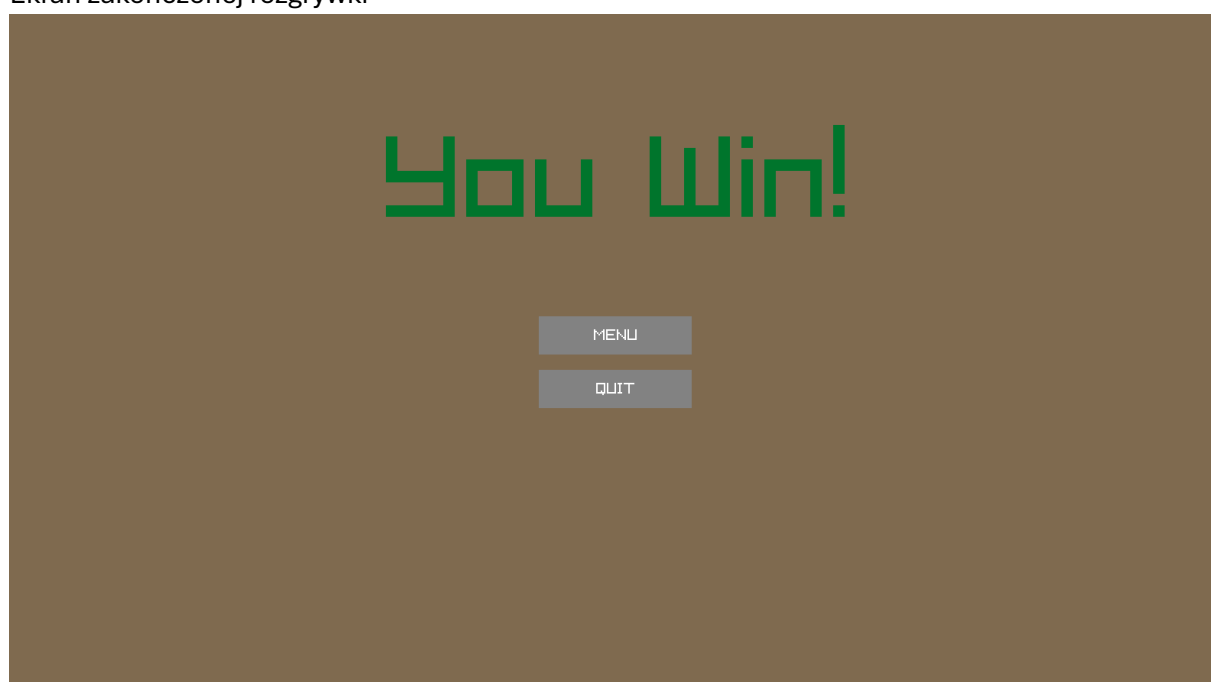


Rozgrywka



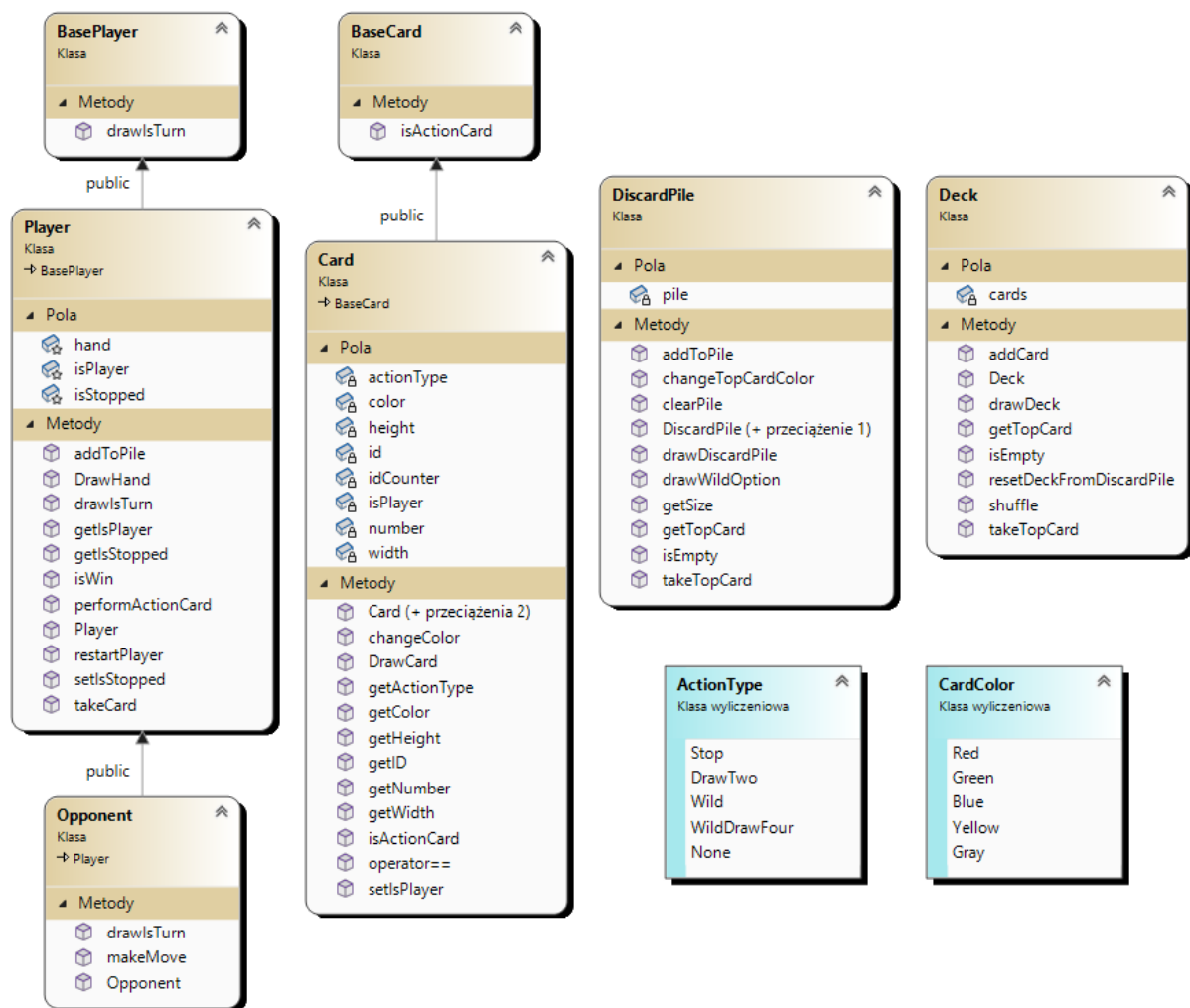


Ekran zakończonej rozgrywki



Specyfikacja wewnętrzna

Diagram klas



Omówienie najważniejszych klas

Klasa `Player` reprezentuje gracza w grze karcianej, który może być albo rzeczywistą osobą, albo komputerowym przeciwnikiem. Gracz ma swoją rękę, na której przechowuje swoje karty, oraz zestaw metod pozwalających na interakcję z talią kart (`Deck`) oraz stosem kart odrzuconych (`DiscardPile`). Klasa zawiera metody do dobierania kart, wyrzucania kart na stos oraz zwracania informacji, czy jest rzeczywistym graczem, czy komputerowym przeciwnikiem. Konstruktor klasy umożliwia utworzenie obiektu z odpowiednią rolą (gracz lub przeciwnik).

Klasa `Card` reprezentuje podstawową kartę w grze, która ma określony kolor (`CardColor`) i numer. Każda karta posiada również swoje wymiary (szerokość i wysokość), unikalny identyfikator oraz informację, czy należy do gracza, co może wpływać na sposób wyświetlania karty w interfejsie użytkownika. Karty mogą być rysowane na ekranie przy użyciu metody `DrawCard()`, która umożliwia wizualizację karty z różnymi opcjami, jak np. rysowanie karty od tyłu czy określenie jej pozycji. Karty mogą być również porównywane między sobą przy użyciu operatora porównania, co jest istotne dla logiki gry, na przykład przy sprawdzaniu, czy gracz może zagrać daną kartą.

Omówienie najważniejszych struktur danych

Struktura `CardColor`` jest wyliczeniem, które reprezentuje cztery możliwe kolory kart w grze: czerwony, zielony, niebieski i żółty.

Struktura `ActionType`` jest wyliczeniem definiującym różne rodzaje akcji, które mogą być związane z kartami. Może to być zatrzymanie gry, odwrócenie kolejności, zmuszenie przeciwnika do dobrania kart, dzika karta oraz dzika karta z dobieraniem czterech kart.

Dziedziczenie

W diagramie klas widać, że struktura gry UNO jest zorganizowana hierarchicznie za pomocą dziedziczenia:

- `Player`` dziedziczy po `BasePlayer``, co umożliwia rozszerzenie funkcjonalności bazowego gracza o bardziej specyficzne zachowania. Przykładem może być dodanie metod takich jak `DrawHand()` czy `isWin()`, które są specyficzne dla klasy `Player``.
- `Opponent`` dziedziczy po `Player``, co pozwala na ponowne wykorzystanie metod i pól z klasy `Player``, a także dodanie specyficznych dla przeciwnika metod jak `makeMove()`.
- `Card`` dziedziczy po `BaseCard``, co umożliwia wykorzystanie ogólnych właściwości kart, takich jak `isActionCard()`, i dodanie specyficznych właściwości, takich jak `color`` czy `actionType``.

Polimorfizm

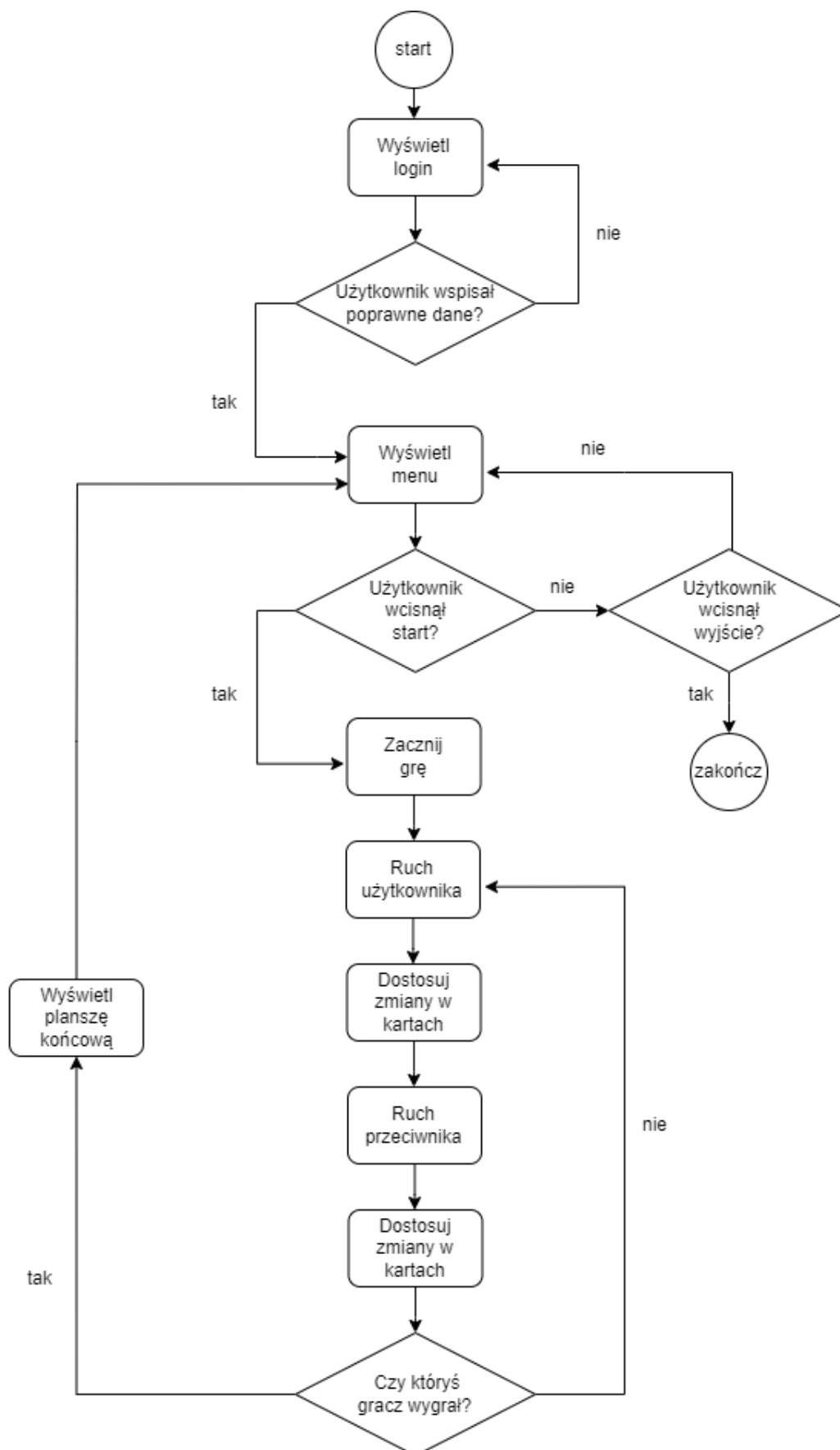
Polimorfizm w projekcie jest wykorzystany głównie poprzez dziedziczenie klas `Player`` i `Opponent`` oraz `BaseCard`` i `Card``. Pozwala to na traktowanie obiektów klas pochodnych jako obiekty klas bazowych. Przykładowo, metoda `drawIsTurn()` może być wywołana na obiekcie typu `BasePlayer``, niezależnie od tego, czy jest to instancja `Player`` czy `Opponent``. Podobnie metoda `isActionCard()` może być wywołana na obiekcie typu `BaseCard``, niezależnie od tego, czy jest to instancja `Card``.

Inne techniki programowania obiektowego

- Kapsułkowanie: Każda klasa ma swoje pola i metody, które operują na tych polach, zapewniając kontrolę nad tym, jak dane są przechowywane i modyfikowane.
- Abstrakcja: Bazowe klasy `BasePlayer`` i `BaseCard`` definiują ogólne właściwości i zachowania graczy oraz kart, które są następnie rozszerzane przez klasy pochodne.
- Modułowość: Struktura kodu w klasach i metodach pozwala na podział funkcjonalności gry na niezależne moduły, co zwiększa czytelność i utrzymanie kodu.

Wykorzystanie tych technik umożliwia stworzenie elastycznego i rozszerzalnego systemu, który jest łatwy do utrzymania i rozbudowy.

Ogólny schemat działania programu



Testowanie i uruchamianie

Podczas testowania i uruchamiania gry UNO, napotkano kilka istotnych problemów, które wymagały dokładnego zdiagnozowania i naprawienia. Poniżej przedstawiono najważniejsze z nich:

Mechanika przeciwnika

- Opis problemu: W trakcie testowania gry okazało się, że przeciwnik komputerowy nie zawsze wykonywał swoje ruchy poprawnie. Często zdarzało się, że przeciwnik nie używał kart zgodnie z zasadami gry UNO lub w ogóle nie wykonywał ruchu.

- Rozwiązanie: Poprawiono algorytm odpowiedzialny za podejmowanie decyzji przez przeciwnika, zapewniając, że będzie on wybierał kartę do zagrania na podstawie aktualnego stanu gry i zgodnie z zasadami UNO. Przeprowadzono wiele testów, aby upewnić się, że przeciwnik podejmuje logiczne i zgodne z regułami gry decyzje.

Błędy związane z wektorami

- Opis problemu: Częstym źródłem błędów były operacje na wektorach, które przechowywały między innymi karty graczy i przeciwnika. Błędy te były wynikiem odwoływania się do nieaktualnych danych, które zostały zmienione w innym miejscu kodu.

- Rozwiązanie: Zidentyfikowano miejsca, gdzie dochodziło do modyfikacji wektorów, i upewniono się, że są one odpowiednio zsynchronizowane. Wprowadzono mechanizmy zabezpieczające przed odwoływaniem się do nieaktualnych lub usuniętych elementów wektora. Dodatkowo, zastosowano odpowiednie techniki zarządzania pamięcią, aby uniknąć problemów związanych z wskaźnikami.

Wyświetlanie kart

- Opis problemu: Prawidłowe wyświetlanie kart zarówno dla gracza, jak i przeciwnika komputerowego stanowiło wyzwanie. Często zdarzało się, że karty były wyświetlane nieprawidłowo, były nakładły się na siebie lub były niewidoczne.

- Rozwiązanie: Poprawiono algorytmy odpowiedzialne za rysowanie kart na ekranie. Upewniono się, że każda karta jest wyświetlana w odpowiednim miejscu, z odpowiednią kolejnością i bez nakładania się na inne karty. Przeprowadzono wiele testów wizualnych, aby upewnić się, że interfejs użytkownika działa poprawnie zarówno dla gracza, jak i dla przeciwnika komputerowego.

Podsumowanie

Testowanie i uruchamianie gry UNO pozwoliło na wykrycie i usunięcie wielu błędów, co znacząco poprawiło stabilność i funkcjonalność aplikacji. Dzięki dokładnym testom i wprowadzonym poprawkom udało się zapewnić płynną rozgrywkę oraz zgodność z zasadami gry

UNO. Pomimo napotkanych trudności, proces testowania przyczynił się do znacznej poprawy jakości końcowego produktu.