Implementacja gry planszowej "Chińczyk"

Adam Hyjek
nr indeksu: 234987
Politechnika Wrocławska
Wrocław, Polska

Bartosz Rodziewicz
nr indeksu: 226105
Politechnika Wrocławska
Wrocław, Polska

I. Streszczenie

Celem projektu jest stworzenie aplikacji pozwalającej na rozegranie gry planszowej "Chińczyk" (znanej także jako "Ludo" oraz "Człowieku, nie irytuj się") na pojedynczym komputerze. W skład aplikacji umożliwiającej przeprowadzenie rozgrywki we wspomnianej grze wchodzi zarówno logiki sterującej grą, jak i graficzny interfejs użytkownika, co umożliwi komfortową rozgrywkę w czasie rzeczywistym. Aplikacja zostanie zaimplementowana w języku GDScript zaprojektowanym specjalnie na potrzeby silnika Godot [1].

II. Wstępny opis słowny

Zadaniem aplikacji jest umożliwienie rozegrania partii gry "Chińczyk". W tym celu po jej uruchomieniu ekranie wyświetla się plansza zawierająca składająca się z czterdziestu pól, czterech domków oraz czterech schowków. Ponadto na planszy znajduje się 16 pionków, po 4 w każdym z 4 kolorów reprezentujących poszczególnych graczy oraz kostka do gry. Na ekranie wyświetla się także krótka instrukcja do obsługi programu oraz wskazanie na gracza mającego właśnie kolejkę [2].

Aplikacja umożliwia kolejnym graczom wykonanie swojej kolejki poprzez wykonanie rzutu kostką oraz, w przypadku posiadania legalnego ruchu, wybór pionka, którym zostanie wykonany ruch. Ponadto program umożliwia przeprowadzenie podstawowych działań składających się na reguły gry takich jak bicie, wprowadzanie pionków do domków, wyprowadzanie pionków ze schowka, ponowienie rzutu po wyrzuceniu "szóstki", czy wykrycie końca rozgrywki w momencie wprowadzenia wszystkich pionków do domku przez jednego z graczy. Dodatkowo aplikacja powinna implementować funkcjonalności usprawniające doświadczenie użytkownika. Jedno z takich funkcjonalności jak menu ustawień umożliwiające dostosowanie ustawień graficznych, zmianę liczby graczy, czy zmianę sposobu sterowania. Kolejnymi funkcjami możliwymi do zaimplementowania są obsługa myszy do przesuwania pionków, animacja ich przesuwania, a także możliwość gry z komputerem. Ponadto aplikacja powinna posiadać menu początkowe oraz ekran zakończenia gry wskazujący zwycięzcę.

III. Słownik pojęć z dziedziny problemu

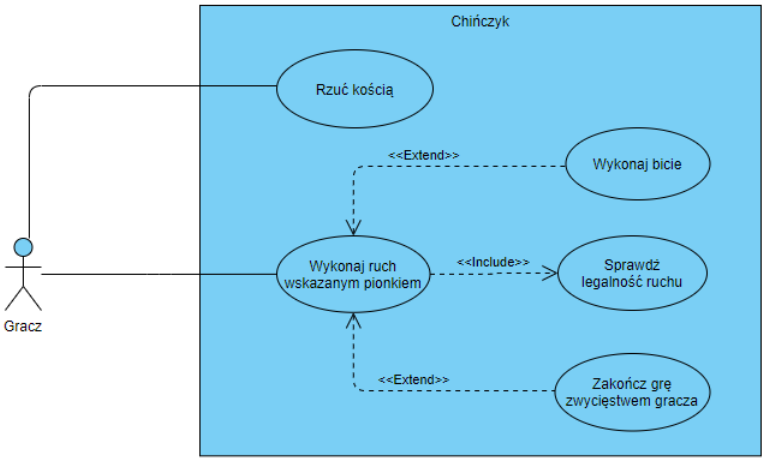
Słownik został opracowany na podstawie zasad gry opisanych w [2].

- Bicie - Sytuacja, w której pionek jednego koloru wkracza na pole zajmowane przez pionek innego koloru. W takiej sytuacji pionek, który uprzednio zajmował dane pole zostaje "zbity" i powraca do schowka.
- Domek - Cztery pola będące docelową lokalizacją pionków danego gracza. Gracz ma prawo wprowadzić pionka do domku dopiero po okrążeniu nim całej planszy, do domku jednego gracza nie mogą wejść pionki innego gracza (koloru). W momencie, w którym jeden z graczy zapełni swój domek wszystkimi swoimi pionkami wygrywa grę.
- Gracz - Jedna z czterech osób uczestniczących w grze. Każdy gracz posiada swój unikalny kolor i cztery pionki. Podczas swojej kolejki gracz rzuca kością i decyduje, który pionek zostanie ruszony. W grze uczestniczy czterech graczy.
- Kolejka - Moment, w którym dany gracz może wykonać swój ruch. Gracze wykonują swój ruch po kolei, zgodnie ze wskazówkami zegara. Kolejka składa się z rzutu kostką i przesunięcia pionka. W momencie, gdy gracz wyrzucił na kości "szóstkę" powtarza ruch przed zakończeniem kolejki.
- Kość - Służy do losowego określenia liczby pól, o które gracz przesunie pionka. Efektem rzutu kością jest liczba oczek z zakresu 1-6.
- Kolor - Unikalny identyfikator każdego z graczy określający jego pionki, schowek, pole startowe oraz domek.
- Pionek - Figura, którą porusza gracz podczas swojej kolejki w celu przesunięcia jej do domku. Każdy z graczy ma cztery pionki o własnym kolorze i po umieszczeniu wszystkich w domku wygrywa grę.
- Plansza - Miejsce rozgrywania gry. Składa się z czterdziestu pól, w tym czterech startowych, czterech domków oraz czterech schowków.
- Pole - Część planszy, po której można przesuwać pionki w celu okrążenia planszy i umieszczenia ich w domku. Na planszy znajduje się 36 zwykłych pól oraz 4 pola startowe.
- Pole startowe - Cztery pola, na których umieszcza się

pionki po wyprowadzeniu ich ze schowka. Każde z pól posiada własny kolor odpowiadający kolorowi gracza posiadającego schowek.
- Schowek - Miejsce startowe wszystkich pionków. Celem gracza jest wyprowadzenie tych pionków poprzez wyrzucenie "szóstki" na kości i umieszczenie pionka na odpowiednim polu startowym. Istnieją cztery schowki, po jednym dla każdego gracza.
- Ruch - Moment przesunięcia pionka po rzucie kością. Zazwyczaj gracz w jednej kolejce wykonuje jeden ruch, wyjątkiem jest sytuacja, w której gracz wyrzucił "szóstkę" - wtedy po wykonaniu ruchu w nagrodę ponownie rzuca kością i wykonuje dodatkowy ruch przed zakończeniem kolejki.

IV. Analiza wymagań użytkownika

Aplikacja ma za zadanie umożliwienie przeprowadzenia rozgrywki w grę "Chińczyk". W tym celu z punktu widzenia użytkownika aplikacja musi wykonywać dwie czynności: rzut kością do gry oraz wykonanie ruchu wskazanym przez gracza pionkiem o liczbę oczek określoną wykonanym przed ruchem rzutem. W ramach każdego wykonywanego ruchu program musi sprawdzić jego legalność i zabronić wykonywania ruchów niedozwolonych, takich jak ruch pionkiem ze schowka w momencie wyrzucenia mniejszej liczby oczek niż 6. Oprócz tego gra musi być wstanie wykryć i wykonać sytuację bicia pionka (gdy pionek gracza najedzie na pole, na którym znajduje się pionek przeciwnika) i osiągnięcie warunku zwycięstwa w postaci umieszczenia wszystkich pionków w domku. W formie graficznej wymagania te obrazuje diagram przypadków użycia:



Aktor w postaci gracza może wykonać dwie podstawowe czynności: rzut kością lub wykonanie ruchu wybranym pionkiem. Integralną częścią wykonywanego ruchu jest sprawdzenie jego legalności, stąd związek zawierania pomiędzy omawianymi przypadkami użycia. W przypadku wykonywania bicia oraz zakończenia gry, te przypadki użycia nie będą występować przy każdym wykonaniu ruch, więc są z nim w związku rozszerzenia.

V. Modele systemu z różnych perspektyw

Model systemu zakłada istnienie czterech klas reprezentujących cztery kluczowe pojęcia ze słownika opisującego dziedzinę problemu. Są to: plansza, kość do gry, gracz oraz pionek. Taki podział umożliwia reprezentację wszystkich kluczowych pojęć w możliwie najprostszej formie.

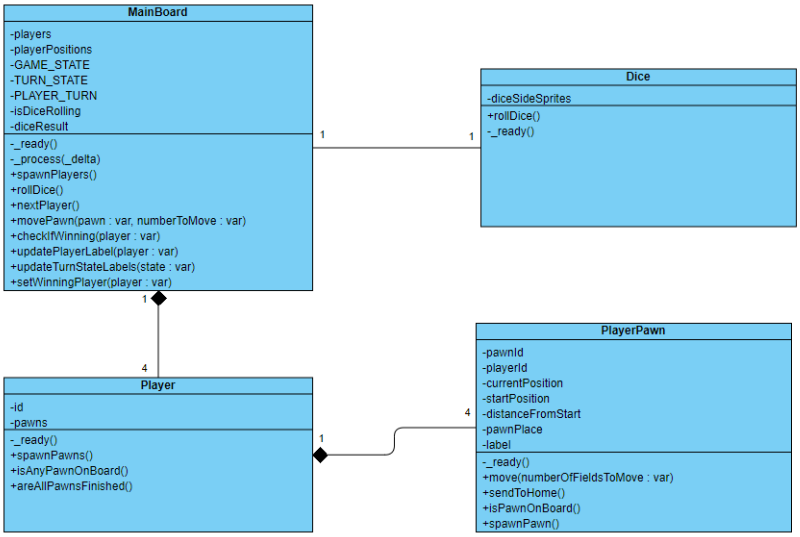
Plansza do gry jest rozbudowaną klasą opisującą i przeprowadzającą rozgrywkę. Zawiera ona wszystkich graczy oraz położenia ich pionków, a także ma dostęp do kości do gry. Ponadto zawiera on zmienne niezbędne do kontrolowania stanu rozgrywki określające stan gry, stan tury, rezultat ostatniego rzutu kostką i wskazującą gracza oczekującego na swoją kolejkę.

Klasa (Dice) opisująca kość do gry jest prostą klasą przechowującą wygląd kości i umożliwiającą symulowanie jej działania poprzez losowanie liczby z zakresu od 1 do 6 za pomocą funkcji (rollDice()). Ponadto klasa ta posiada standardową funkcję (_ready()) wywoływaną w momencie pojawienia się kości na ekranie po raz pierwszy.

Kolejną klasą jest przedstawiająca gracza klasa (Player). Każdy gracz reprezentowany jest poprzez (id), a ponadto przechowuje informacje o własnych pionkach. Do zadań tej klasy należy zainicjowanie pionków za pomocą funkcji (spawnPawns()) i kontrola ich położenia - funkcja (isAnyPawnOnBoard()) służy do określenia, czy któryś z czterech pionków wyszedł ze schowka, a funkcja (areAllPawnsFinished()) sprawdza warunek zwycięstwa, czyli czy wszystkie pionki znajdują się w domku.

Ostatnią niezbędną klasą jest klasa (PlayerPawn) uosabiająca pionki posiadane przez graczy. Zakłada się, że każdy z pionków ma przypisany do siebie numer (pawnId) oraz id kontrolującego go gracza ((playerId)). Ponadto każdy z pionków musi mieć określoną aktualną pozycję (zmienna (currentPosition)) oraz początkową pozycję ((startPosition)). Zmienna pawnPlace określa status pionka (w schowku, na planszy, w bazie), a zmienna label przechowuje graficzną reprezentację obiektu. Do określenia pozycji na planszy wykorzystywana jest zmienna (distanceFromStart), która określa ile pól przebył pionek. Dzięki znajomości całkowitej liczby pól, które pionek musi przebyć przed dotarciem do domku zawsze jesteśmy w stanie precyzyjnie określić jego pozycję na tej trasie.

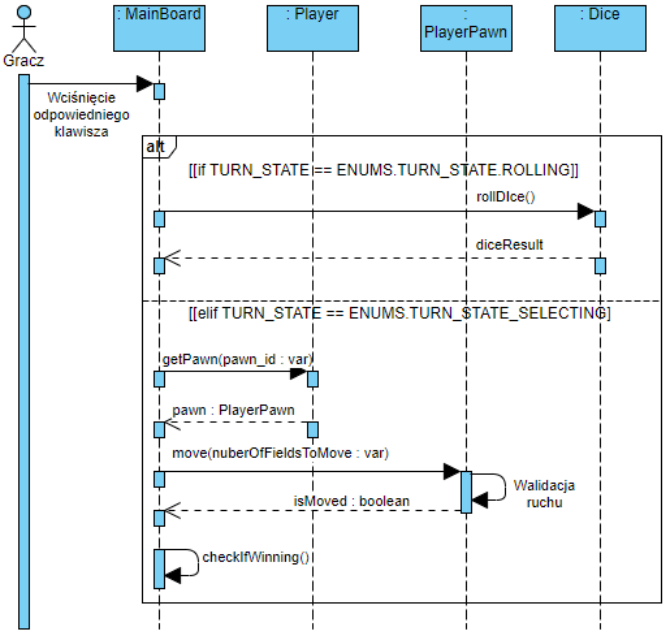
Graficzną reprezentację powyższego opisu stanowi diagram klas:



Główną klasą kontrolującą stan rozgrywki jest klasa (MainBoard), w związku z tym posiada ona najwięcej relacji z pozostałymi klasami. Jest ona w stanie komunikować się z klasą (Dice) dzięki związkowi asocjacji o pojedynczej krotności, a ponadto zawiera w sobie obiekty klasy (Player). W związku z tym obie klasy łączy związek agregacji całkowitej - obiekt klasy (MainBoard) zawiera cztery obiekty klasy (Player). Ponadto w aplikacji istnieje związek agregacji całkowitej pomiędzy klasami (Player) oraz (PlayerPawn) - każdy obiekt klasy (Player) zawiera cztery obiekty klasy (PlayerPawn)

Główną funkcjonalnością programu jest przeprowadzanie rozgrywki w grę "Chińczyk". W tym celu aplikacja musi przyjmować od użytkownika sygnały wejściowe w postaci odpowiednich klawiszy i odpowiednio na nie reagować w zależności od stanu przeprowadzanej rozgrywki. Do monitorowania stanu rozgrywki służą zmienne każdy obiekt klasy (Player) zawiera cztery obiekty klasy (GAME_STATE), (TURN_STATE) oraz (PLAYER_TURN). Pierwsza zmienna odpowiada za ogólny stan rozgrywki, taki jak: gra jeszcze nie rozpoczęta, gra w trakcie, czy gra zakończona. Zmienna (PLAYER_TURN) określa zawodnika, którego kolejka właśnie trwa, z kolei zmienna (TURN_STATE) reprezentuje moment tury, w którym obecnie znajduje się program. Jest ona kluczowa dla działania całej aplikacji, ponieważ znacząco wpływa na zachowanie danej iteracji pętli głównej.

Poniżej przedstawiono diagram sekwencji modelujący pojedynczą iterację pętli głównej odpowiedzialnej za działanie programu:



W przypadku gdy stan tury, reprezentowany przez zmienną (TURN_STATE) odpowiada wartości (ENUMS.TURN_STATE.ROLLING) program jest w fazie poprzedzającej rzut kością. W związku z tym obiekt klasy (MainBoard) komunikuje się z obiektem klasy (Dice) poprzez wywołanie funkcji (rollDice()) zwracającej wynik rzutu kością. Warto wspomnieć, że warunkiem wykonania tej części kodu jest wciśnięcie przez użytkownika klawisza odpowiedzialnego za interakcję, domyślnie jest to spacja. Z kolei jeżeli zmienna (TURN_STATE) odpowiada wartości (ENUMS.TURN_STATE.SELECTING) program reaguje na klawisze 1-4 odpowiadające poszczególnym pionkom posiadanym przez gracza. Po dokonaniu wyboru obiekt klasy (MainBoard) rozpoczyna komunikację z odpowiednim obiektem klasy (Player) w celu otrzymania pionka, który ma zostać przesunięty. Następnie obiekt klasy (MainBoard) wywołuje funkcję (move) z argumentem oznaczającym wyrzuconą liczbę oczek, która waliduje żądany ruch i zwraca zmienną binarną określającą, czy ruch jest legalnych i czy zakończył się sukcesem. W takim przypadku następuje sprawdzenie warunku zwycięstwa i zmiana stanu tury.

VI. Kwestie implementacyjne

Projekt został zaimplementowany na bazie silnika Godot Engine, który jest złożonym, wieloplatformowym narzędziem do tworzenia gier 2D oraz 3D. Umożliwia on w prosty sposób wyeksportować tworzony projekt na najpopularniejsze obecnie systemy operacyjne takie jak Windows, Linux, macOS, Android, iOS, a ponadto na platformy webowe, na przykład HTML. Silnik Godot tworzy kompleksowe środowisko wspierające pełny cykl

tworzenia gier, dzięki czemu jest to jedyne narzędzie potrzebne do stworzenia gry "Chińczyk". Godot umożliwia m.in. tworzenie grafiki przy pomocy biblioteki OpenGL ES, wspiera wielokanałowe systemu dźwiękowe a także posiada system ułatwiający tworzenie GUI [1].

Wybranym językiem programowania został język GDScrpit, który jest językiem skryptowym zintegrowanym z silnikiem Godot. Umożliwia on tworzenie aplikacji wykorzystujących ten silnik wykorzystując przy tym minimalną możliwą ilość kodu co zwiększa produktywność i ułatwia jego naukę. GDScript jest dynamicznie typowanym językiem upraszczający tworzony kod i nie wymagającym kompilacji do jego testowania [1].

VII. Podsumowanie i dyskusja krytyczna

W ramach tego projektu udało nam się zrealizować i zaimplementować niezbędne funkcjonalności systemu umożliwiającego zagranie w grę "Chińczyk". Jedyną rzeczą, której nie udało nam się zaimplementować było ponowienie rzutu kością po wyrzuceniu "szóstki". Pomimo, że projekt można by rozszerzyć o pewne dodatkowe funkcjonalności takie jak menu ustawień, animację pionków, czy możliwość sterowania myszką to nie wpłynęłyby one znacząco na zasadnicze zachowanie systemu, a jedynie polepszyły doświadczenie użytkownika.

W procesie tworzenia aplikacji zaznajomiliśmy się z tematyką programowania gier, a także poznaliśmy interesujące, otwartoźródłowe środowisko do ich tworzenia - Godot Engine. Implementuje on specjalny język GDScript usprawniający tworzenie gier komputerowych, dzięki czemu po krótkim zapoznaniu się z jego składnią i dokumentacją silnika mogliśmy skupić się na modelu samej aplikacji.

Kod źródłowy

Kod źródłowy zaiplementowanej gry jest dostępny w repozytorium pod adresem: https://github.com/baatochan/LudoGame

Literatura

[1] Dokumentacja silnika Godot: https://docs.godotengine.org/pl/latest/
[2] Strona na Wikipedii objaśniająca zasady gry "Chińczyk": https://pl.wikipedia.org/wiki/Chińczyk_(gra_planszowa)