1 Temat

Komputerowa wersja gry logicznej ”Khet”dla dwóch osób.

2 Analiza tematu

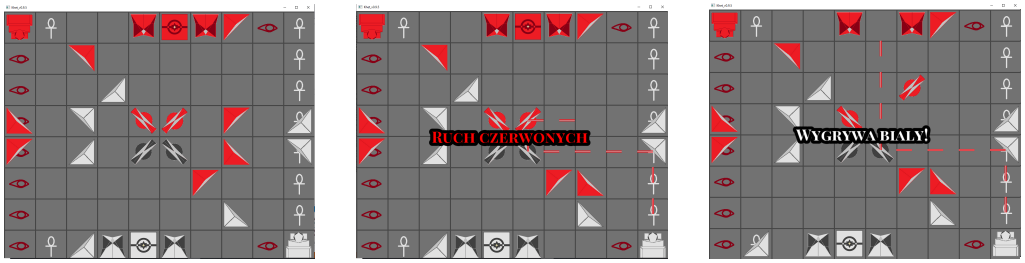
Zagadnienie opiera się na stworzeniu wirtualnego odpowiednika popularnej gry planszowej, umożliwiającego pełną rozgrywkę z użyciem wyłącznie komputera. Pomocne w operowaniu figurami okazały się biblioteki *vector*,*typeinfo* oraz *algorithm* a także *stdexcept* do obsługi potencjalnych błędów związanych z wczytywaniem z pliku. Do stworzenia interfejsu graficznego wykorzystano bibliotekę *SFML*, w szczególności moduły *SFML/Graphics* oraz *SFML/Window*, wystarczające do stworzenia miłej dla oka, prostej oprawy graficznej. Typ gry w założeniu pozwala na wykorzystanie mechanizmów dziedziczenia/poliformizmu co sprawia, że jest dobrym wyborem na projekt wykorzystujacy programowanie obiektowe.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z pliku .exe. Po uruchomieniu wyświetlane jest menu, w którym użytkownik może wybrać rozpoczęcie nowej gry lub wczytanie poprzedniej. Opcje zapisu gry lub kontynuuowania są widoczne ale ich funkcjonalność zostaje odblokowana dopiero po rozpoczęciu pierwszej rozgrywki w bierzącym uruchomieniu programu.

Rysunek 1: Widok menu gry

Wciśnięcie myszką przycisku rozpoczynającego nową lub poprzednio rozgrywnaną grę, spowoduje uruchomienie planszy z odpowiednio rozstawionymi figurami. Co turę gracze zostają informowani który z nich powinien wykonać ruch. W momencie trafienia Faraona dowolnego koloru gra zostaje zakończona a na ekranie pojawia się komunikat który z graczy zwyciężył. Wciśnięcie klawisza esc w dowolnym momencie spowoduje powrót do menu gry gdzie użytkownik będzie miał możliwość rozpoczęcia nowej rozgrywki, kontynuowanie bądź zapisanie toczącej się gry lub wczytanie poprzednio zapisanej.



(a) okno nowo rozpoczętej gry (b) komunikat o końcu tury (c) komunikat o wygranej

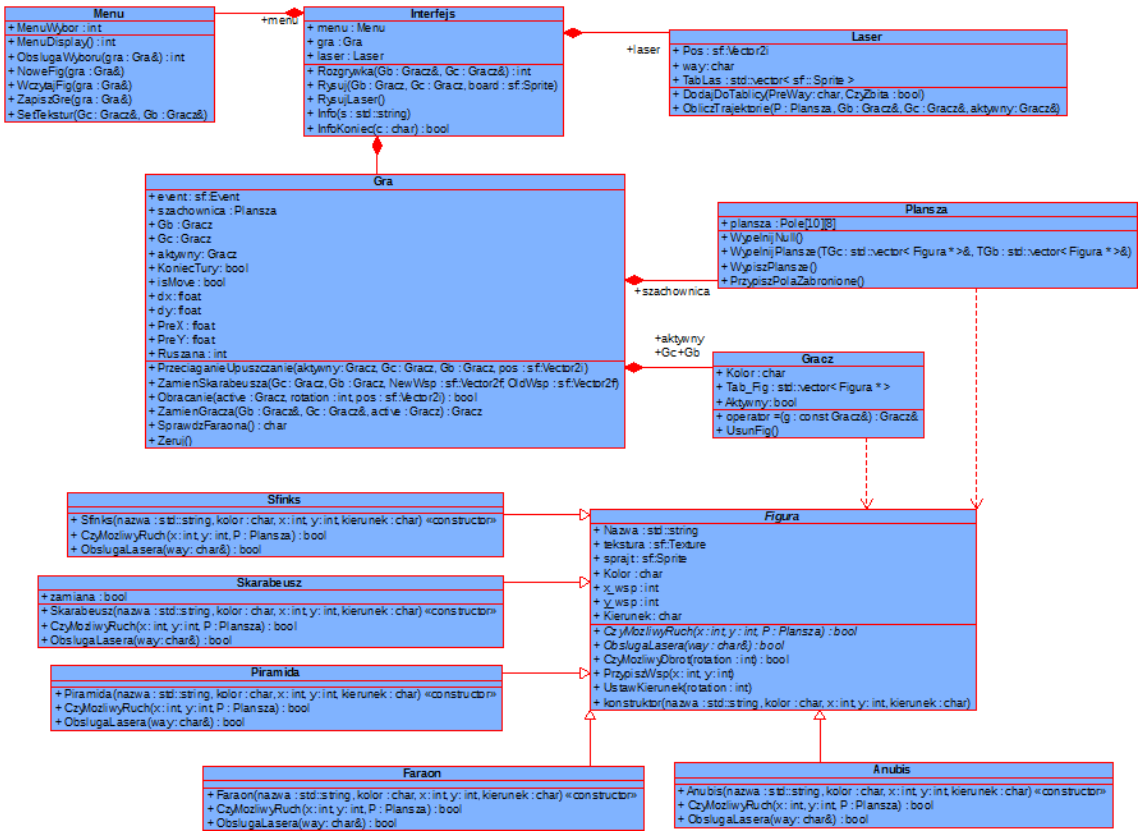
Rysunek 2: Zrzuty ekranu rozgrywki

Gra toczy się według oficjalnych zasad (link). Gracze na zmianę wykonują swoje posunięcia rozpoczynając od figur białych. Wszystkie ruchy odbywają się za pomocą myszki, wciśnięcie i przytrzymanie lewego przycisku pozwala przenosić figurę, która po „upuszczeniu" zostaje na wskazanym polu oraz jest wyśrodkowywana. Za obracanie figur po pojedynczym kliknięciu odpowiadają: prawy przycisk myszy (obrót figury w prawo [90 stopni]) oraz środkowy przycisk myszy (obrót w lewo [-90/270 stopni]). Próba wykonania ruchu zabronionego (jak nałożenie figury na figurę, ruch o zbyt wiele pól lub niedopuszczalny kierunek obrotu Sfinksa) zostanie wykryta przez program i ruch nie zostanie wykonany (gracz zachowuje swoją turę). Laser odpowiedzialny za zbijanie figur (jego źródłem jest odpowiedni Sfinks) uruchamia się każdorazowo, po skończonej turze aktywnego gracza. W momencie trafienia Faraona gra zostaje zakończona a po komunikacie o wygranej zostajemy przekierowani do menu.

4 Specyfikacja wewnętrzna

W programie rozdzielono interfejs (obsługę rozgrywki) od logiki aplikacji. Projekt jest zorientowany obiektowo i stworzony w języku C++.

4.1 Opis klas, typów i funkcji



Rysunek 3: Diagram klas

Możliwe ruchy każdej z figur są określone w wirtualnych funkcjach dziedziczących klas Piramida, Anubis, Faraon itp, które są potomkami nadrzędnej klasy abstrakcyjnej Figura. Metody klasy Gra są odpowiedzialne za prowadzenie rozgrywki, interpretacje i przypisywanie ruchów użytkownika do pamięci programu. Klasa Interfejs zapewnia komunikacje użytkownika z programem, zajmuje się wyświetlaniem grafik figur, planszy i komunikatów oraz obsługuje działające w pętli wczytywanie ruchów myszy czy wciskanych klawiszy. W reakcji na daną czynność użytkownika wywołuje odpowiednie metody klasy Gra, po czym przenosi ich wynik na ekran. Klasa Menu zawiera metody zajmujące się wyświetlaniem głównego menu gry oraz obsługą występujących w nim przycisków. Klasa Gracz reprezentuje gracza. Każdemu z nich jest przypisany kontener przechowujący jego figury - obiekty klas dziedziczących po klasie Figura oraz pole informujące o tym czy to jego tura. Klasa Plansza została użyta do reprezentacji planszy gry. Jej główna składowa to tablica

typu Pole, przechowuje w odpowiednich indeksach wskaźniki na obiekty klasy Figura oraz informacje o typie pola. Klasa Laser we współpracy z klasą Plansza odpowiada za obsługę wiązki wystrzeliwanej przez Sfinksa. Zadaniem jej metod jest stworzenie graficznej reprezentacji lasera, obliczanie jego trajektorii oraz przekazywanie informacji o ewentualnych zbitych figurach.

Szczegółowy opis klas, typów i funkcji zawarty jest w załączniku.

4.2 Ogólna struktura programu

Włączenie programu tworzy okno gry oraz obiekt klasy Interfejs, po czym w funkcji main() w głównej pętli programu wywołuje metodę MenuDisplay() odpowiedzialną za wyświetlenie menu i określenie wyboru gracza. Następnie metoda ObslugaWyboru(), wywołuje odpowiednie funkcje tworzące zestaw figur. Są one inicjowane wartościami standardowymi (NoweFig()) lub wczytanymi z pliku save.txt (WczytajFig()). Figury zostają przypisane do utworzonych obiektów klasy Gracz oraz wczytuje się im konkretne tekstury (SetTekstur()). Po uruchomieniu nowej gry lub wczytaniu poprzedniej zostaje wywołana metoda kierująca rozgrywką Rozgrywka(). Następnie program w nieskończonej pętli wczytuje akcje graczy do zmiennej typu Event z biblioteki SFML. W zależności od ruchu gracza wyłapuje go i wykonuje odpowiednie operacje z użyciem metod Obracanie() i PrzeciaganieUpuszczanie (). Za sprawdzenie poprawności wykonywanych ruchów są odpowiedzialne metody CzyMozliwyRuch() oraz CzyMozliwyObrot(). Po każdym dopuszczalnym ruchu wywoływana jest metoda WypelnijPlansze() przypisująca figury do konkretnych pól tablicy typu Pole, metoda klasy Laser obliczająca jego trajektorię (ObliczTrajektorie ()) oraz wyświetlana graficzna reprezentacja wiązki(RysujLaser()). Trafione figury zostają usuwane z kontenera, który je przechowuje a jej pozycja w tablicy reprezentującej plansze zostaje ustawiona na nullptr. Potencjalne zekończenie gry monitoruje metoda SprawdzFaraona (). Następnie następuje zmiana aktywnego gracza (ZamienGracza()) o czym program komunikuje wyświetlając w oknie gry odpowiedni komunikat metodą Info (). Grafiki aktywnych figur wraz z planszą są wyświetlane w utworzonym oknie po każdym przejściu pętli za pomocą metody Rysuj(). W przypadku chęci zapisania rozgrywki, ustawienie figur oraz informacja o aktywnym graczu są zapisywane metodą ZapiszGre() do pliku:

`save.txt`

4.3 Istotne struktury danych i algorytmy oraz techniki obiektowe

Jednym z przykładów struktur jest zdefiniowana na potrzeby programu struktura Pole, z polem informującym czy może stać na nim figura o danym kolorze (char) oraz wskaźnikiem na przechowywaną figurę (Figura*). Dwuwymiarowa tablica typu Pole reprezentująca planszę używana będzie potem w klasie Plansza. Program zawiera również kontenery STL typu Vector, m.in. przechowujące obiekty klasy Figura przypisane do każdego z graczy a poruszanie się po nich jest usprawnione dzięki użyciu iteratorów. To rozwiązanie pozwala w prosty sposób zapewnić możliwość poruszania się tylko aktywnemu graczowi. Figura jest usuwana z kontenera w przypadku jej zbicia. W realizacji projektu pomocne okazały się narzędzia RTTI, w szczególności wyrażenia *typeid* oraz *dynamic_cast*, w celu uzyskania informacji o dokładnym rodzaju figury, lub obsługi niektórych ruchów. Obsługa potencjalnych błędów związanych z brakiem lub uszkodzeniem pliku z zapisem rozgrywki jest rozwiązana z użyciem mechanizmu wyjątków.

5 Testowanie

Program został przetestowany poprzez rozegranie serii parti w różnych scenariuszach oraz zakończeniach. Została także sprawdzona reakcja programu na wielokrotne rozpoczynanie gry, wykonywanie zapisu lub niedozwolonych posunięć czy też próby wczytywania gry ze zmodyfikowanego lub usuniętego pliku. Wykryto tendencję do niereagowania na wciśnięcie klawisza *esc* w celu wyjścia z/do menu. Błąd ten został naprawiony - nie zdarza się praktycznie w ogóle i nie powoduje wyłączenia programu, w przypadku ewentualnego wystąpienia jest natomiat wymagane ponowne wciśnięcie klawisza. W projekcie zauważono również sporadyczne samoistne zamknięcie programu (nie zwracające kodu błędu) rozpoczynając grę po raz pierwszy od uruchomienia programu. W takim wypadku potrzebne jest ponowne uruchomienie.