Dokumentacja Techniczna
Władca Pierścieni: Konfrontacja

1. Informacje ogólne

1.1. Autor, data opracowania dokumentacji.

Autor: Rafał Nowicki
Data: 16.09.2012

1.2. Krótki opis programu.

Aplikacja *LotR: Confrontation* jest implementacją gry planszowej Władca Pierścieni – Konfrontacja, Reiner'a Knizia. Gra planszowa, jest swoistą interpretacją powieści Tolkiena. W grze wcielamy się w jedną ze stron – Drużyny Pierścienia lub sił mroku. Drużyna Pierścienia stara się, aby Frodo dotarł do Mordoru, a mroczny władca Sauron ze wszystkich sił próbuje odzyskać pierścień, zanim Frodo dotrze do Góry Przeznaczenia.
Każdy z graczy kontroluje 9 postaci, z których każda ma unikalne zdolności (9 postaci Drużyny jest kontrolowanych przez jednego gracza, 9 największych popleczników Saurona przez drugiego gracza). Następuje rozstrzygająca konfrontacja między siłami światła i mroku w Śródziemiu.

2. Charakterystyka programu

2.1. Przeznaczenie programu

Program służy celom rozrywkowym. Dzięki zastosowaniu w aplikacji trybu turowego hot-seat możliwe jest rozgrywanie partii na jednym komputerze.

2.2. Funkcje

Integakcja z użytkownikiem odgrywa bardzo ważną rolę w każdej tego typu aplikacji, dlatego zadbałem o to aby ta interakcja była możliwie na jak najwyższym poziomie.

- Ciekawy i estetyczny interfejs graficzny użytkownika z wieloma trybami rozgrywki;
- W różnych trybach gry użytkownik ma doczynienia ze wskazówkami, które sprawiają że interfejs graficzny jest jeszcze bardziej przyjazny użytkownikowi;
- Dzięki opracowanemu systemowi błędów i uwag, gracz który wykonuje czynności nie zgodne z zasadami gry jest na bierząco o tym informowany;
- Dzięki algorytmowi inteligentnego pozycjonowania pionów na planszy zawsze jest porządek a w zależności od tego ile pionków aktualnie znajduje się w danej krainie są one unikalnie pozycjonowane;
- Specjalnie zaprojektowany algorytm rejestrowania i analizowania ruchów w trakcie tury pozwala użytkownikowi na przesuwanie pionów po całej planszy, a w razie popełnionego błędu gracz może cofnąć wszytskie swoje dotychczasowe ruchy w turze (jeżeli nie pamięta pierwotnego położenia piona);
- System zapisu i odczytu stanu gry.

3. Struktura programu

3.1. Opis plików zewnętrznych (organizacja, użycie).

Większość plików zewnętrzne z których korzysta aplikacja to obrazki – tła oraz przygotowane wcześniej w programie graficznym sprity. Aplikacja kożysta również z czcionki *Papyrus* oraz plików przygotowanych przez bibliotekę pickle przeznaczonych m.in. do odtwarzania stanu gry.

3.2. Globalne struktury danych (opis, przeznaczenie).

Korzystając z dobrodziejstw języka programowania Python w programie używane są głównie:
LISTY:

- live / b_live / w_live – w tych strukturach danych przechowywane są informacje o dostępnych jednostkach każdego z graczy – mroku oraz światła. W podlistach przechowywane są takie informacje jak: identyfikator jednostki jako integer, nazwa jednostki wraz z identyfikatorem drużyny jako string, kraina w której aktualnie jednostka się znajduje – string, współrzędne środka piona na ekranie - krotka oraz cyfra mówiąca o sile jednostki – integer;
- dead / b_dead / w_dead – podobnie jak wyżej z tą różnicą że przechowywane są informacje o nie dostępnych jednostkach;
- cards / b_cards / w_cards – w tych strukturach przechowywane są informacje o dostępnych kartach. W podlistach przechowywane są identyfikatory kart oraz ich znaczenie;
- wasted_cards / wasted_b_cards / wasted_w_cards – podobnie jak wyżej z tym że tutaj przechowywane są zużyte karty;
- p_movement – tutaj przechowywane są informacje o jednostkach (tak jak w live../dead..) chronologicznie w kolejności w jakiej zostały przesunięte;
- who – lista jednostek pomiędzy którymi będzie toczona walka (tak samo jak w live../dead..);
- lands_names – lista ze wszystkimi nazwami krain na mapie. Nazwy krain są zapisane jako ciągi znaków;
- errors – jest to lista błędów i uwag, jej elementami są liczby całkowite;

SŁOWNIKI:

- areas – kluczami są specyficzne nazwy (stringi) obszarów w GUI a wartościami krotki o wartościach kolejno współrzędna x, y , szerokość, wysokość;
- buttons – podobnie jak wyżej, kluczami są nazwy (stringi) przycisków w GUI a wartościami krotki;
- land – podobnie jak wyżej, kluczami są nazwy (stringi) wszystkich krain na mapie a wartościami krotki;
- pawns_qu – w tym słowniku kluczami są nazwy (stringi) wszystkich krain na mapie a wartościami są nazwy jednostki wraz z identyfikatorem drużyny jako string

3.3. Podział na moduły, schemat komunikacji między modułami.

W programie współpracuje ze sobą 6 modułów – lotr, events, logic, display, images, variables. Moduły wywoływane są przez lotr – plik główny, z wyjątkiem images, który to jest wywoływany przez display. Modułom events i logic przekazywane są zmienne globalne z modułu variables.

3.4. Wykaz używanych modułów systemowych.

sys, pygame, pickle

4. Opis modułów

4.1. Informacje ogólne (zwięzła charakterystyka modułów).

- lotr – tutaj jest główny program;
- events – obsługa zdarzeń;
- logic – obsługa zasad gry;
- images – tworzy i udostępnia tekstury;
- variables – tworzy i udostępnia zmienne globalne;

4.2. Opis funkcjonalny modułu

4.2.1. Przeznaczenie modułu.

- lotr – tutaj znajduje się główny program, w którym inicjalizowane są prawie wszystkie moduły używane w programie, sprawdzany jest odczyt pliku pickle oraz śledzenie czasu. W pętli zaś, wywoływane są trzy główne funkcje kolejno z modułów events, logic i display;
- events – tutaj sprawdzane są wszystkie zdarzenia które mają miejsce w programie i w zależności od czynności wywoływane są konkretne instrukcje;
- logic – większość fukcji w tym module weryfikuje wykonane przez użytkownika ruchy, inne zajmują się m.in. przygotowaniem programu do zmiany użytkownika lub trybu;
- images – tutaj tworzone są wszytskie tekstury w grze;
- variables – tutaj tworzone są wszystkie zmienne globalne;

4.2.2. Sposób wykorzystania modułu

- EVENTS:
  - **get()** – głowna funkcja klasy Events()
    - we: brak
    - wy: brak
  - **quit()** – funkcja kończąca działanie programu
    - we: brak
    - wy: brak
  - **inteligent_positioning(land_name)** – pozycjonuje piony na konretnej krainie
    - we: nazwa krainy, na której ma pozycjonować
    - wy: brak
  - **insert(land_name, pos)** – funkcja zapisuje nową pozycje piona oraz nazwę krainy do której go przeniesiono
    - we: nazwa krainy, współrzędne jako krotka
  - **quantum()** – funkcja sprawdza gdzie podniesiony pion był wcześniej na mapie i kasuje jego ostatnią pozycję
    - we: brak
    - wy: brak
  - **rec_area()** – funkcja sprawdza czy współrzędne kursora myszy pokrywają się z zadanym prostokątem
    - we: krotka z 4 argumentami – współrzędna x, y, szerokość oraz wysokość prostokąta
    - wy: True – jeżeli kursor pokrywa się z zadanym prostokątem, False w przeciwnym przypadku

- **on_pictures()** – funkcja sprawdza czy współrzędne kursora znajdują się na jednym z obrazków na górze oraz na dole ekranu
  - we: brak
  - wy: True, jeżeli współrzędne kursora pokrywają się z obszarem obrazka, False w przeciwnym wypadku
- **on_button(name)** – funkcja sprawdza czy współrzędne kursora znajdują się na przycisku o zadanej nazwie
  - we: nazwa przycisku
  - wy: True, jeżeli współrzędne kursora pokrywają się z obszarem przycisku, False w przeciwnym wypadku
- **on_area(name)** – funkcja sprawdza czy współrzędne kursora znajdują się na obszarze o zadanej nazwie
  - we: nazwa obszaru
  - wy: True, jeżeli współrzędne kursora pokrywają się z konkretnym obszarem, False w przeciwnym wypadku
- **in_land(name)** – funkcja sprawdza czy współrzędne kursora znajdują się na obszarze krainy o zadanej nazwie
  - we: nazwa krainy
  - wy: True, jeżeli współrzędne kursora pokrywają się z konkretnym obszarem, False w przeciwnym wypadku
- **on_pawn()**– funkcja sprawdza czy współrzędne kursora znajdują się na obszarze któregoś z pionów
  - we: brak
  - wy: True, jeżeli współrzędne kursora pokrywają się z konkretnym obszarem, False w przeciwnym wypadku
- **enemy_on(land_name)** – funkcja sprawdza czy w krainie o zadanej nazwie jest przynajmniej jeden pion drugiego gracza
  - we: nazwa krainy
  - wy: True, jeżeli w krainie o zadanej nazwie jest przynajmniej jeden pion drugiego gracza, False w przeciwnym wypadku
- **possible_move(land_name, pawns_qu)** – funkcja sprawdza czy gracz może przesunąć piona do zadanej krainy
  - we: nazwa krainy do które gracz przesunął piona, ilość pionków które aktualnie stoją w tej krainie
  - wy: brak

- <u>LOGIC:</u>

  - **game()** – główna funkcja klasy Logic()
    - we: brak
    - wy: brak
  - **reset()** – funkcja która przywraca pierwotny stan aktualnej tury
    - we: brak
    - wy: brak
  - **confrontation()** – główna funkcja sterująca trybem konfrontacji
    - we: brak
    - wy: brak
  - **confirmation()** – funcja zarządzająca zmianą trybów po kliknięciu przycisku kontynuuj w trybie zmiany tury lub w trybie rozstrzygnięcia walki
    - we: brak
    - wy: brak
  - **verification()** – funkcja sprawdza czy spełnione zostały warunki zakończenia tury

        - we: brak
        - wy: brak
    - **analyze_move()** – funkcja analizuje kolejność poprzestawianych pionów na mapie
        - we: brak
        - wy: True, jeżeli pionki zostały przestawione dobrze, False w przeciwnym przypadku
    - **correct_movement(character, from_where)** – w zależności od jednostki I krainy w jakiej ta jednostka stała na początku tury, zwraca listę krain do których jednostkę można przesunąć
        - we: nazwa jednostki, wraz z identyfikatorem drużyny jako string, nazwa pierwotnej krainy
        - wy: lista krain do której można się poruszyć
    - **change_seat()** – funkcja która manipuluje wartościami zmiennych aby przygotować GUI i logikę do zmiany gracza i trybu
        - we: brak
        - wy: brak
    - **change_pawns_pos(pawns)** – funkcja zmienia współrzędne pionów na ekranie w celu prawidłowego wyświetlenia dla kolejnego gracza
        - we: informacje o jednostkach które wciąż są w grze
        - wy: brak
    - **lands_pos()** – funkcja zmienia współrzędne krain na ekranie w celu prawidłowego wyświetlenia dla kolejnego gracza
        - we: brak
        - wy: brak
- <u>DISPLAY:</u>

    - **things()** – główna funkcja klasy DrawGameplay()
        - we: brak
        - wy: brak
    - **menu_screen()** – funkcja drukuje na ekran wszystkie potrzebne elementy trybu menu
        - we: brak
        - wy: brak
    - **tour_screen()** – funkcja drukuje na ekran wszystkie potrzebne elementy trybu zmiana tury
        - we: brak
        - wy: brak
    - **victory_screen()** – funkcja drukuje na ekran wszystkie potrzebne elementy trybu zmiana tury – rozstrzygniecie walki
        - we: brak
        - wy: brak
    - **map_screen()** – funkcja drukuje na ekran wszystkie potrzebne elementy trybu mapy
        - we: brak
        - wy: brak
    - **confrontation_screen()** – funkcja drukuje na ekran wszystkie potrzebne elementy trybu konfrontacji
        - we: brak
        - wy: brak
    - **still_stuff()** – funkcja drukuje na powierzchnię rozgrywka wszystkie stałe elementy interfejsu

        - we: brak
        - wy: brak
    - **selection()** – funkcja drukuje na ekran okrąg przy obrazku postaci lub karty która jest aktualnie zaznaczona
        - we: brak
        - wy: brak

4.3. Sytuacje niepoprawne

4.3.1. Błędy i uwagi (opis błędu, warunki jego powstania).

- 101 – wszytskie piony nie są rozstawione
- 102 – piony są źle rozstawione (któryś z pionów znajduje się w górach lub powyżej)
- 103 – gracz postawił piona w krainie wroga
- 104 – maksymalna liczba postaci w danej krainie została osiągnięta
- 201 – informacja o konfrontacji
- 202 – w krainie jest więcej niż jeden pion wroga, należy wybrać piona do konfrontacji
- 203 – postać do konfrontacji została wybrana
- 204 – nie prawidłowy ruch
- 205 – informacja o omijanej kolejce