Gra Arkanoid dla dwóch graczy w języku Ada 2005.

Krzysztof Wesołowski, Automatyka i Robotyka rok II, grupa III

04 XII 2008

Wstęp

Cel programu

Motywem przewodnim projektu była prosta gra komputerowa, realizująca prostą mechanikę starej gry Arkanoid. W celu podniesienia jej atrakcyjności miała ona umożliwiać rywalizacje dwóch graczy. Dodatkowo zadbano o ciekawa wizualizacje z zastosowaniem stosunkowo nowoczesnych technik oraz o możliwość walki z komputerem.

Dodatkowe wymagania

Program jest zaliczeniem projektu z przedmiotu "Programowanie Systemów Czasu Rzeczywistego", dlatego głównym wymaganiem była wielowątkowa konstrukcja programu, mająca na celu poznanie i zaprezentowanie możliwości języka natywnie wspierającego wielowątkowość. Kolejnym założeniem było wykorzystanie mechanizmów języka Ada w celu stworzenia stabilnego i niezawodnego programu.

Realizacja projektu

Analiza problemu

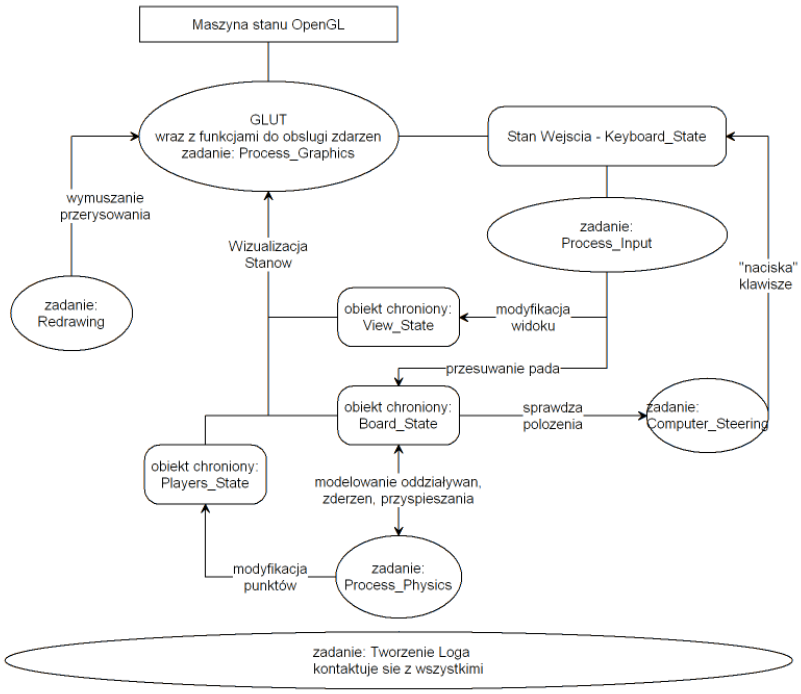
W trakcie rozważań na temat możliwych struktur programu, mających na celu maksymalizacje zarówno wygody jego tworzenia jak i niezawodności zdecydowałem się na podzielenie programu na jak najmniejsze jednostki logiczne, w celu odseparowania niezależnych od siebie części, zapewnienia ich wymienialności oraz przejrzystości struktury.

Zastosowany schemat implementacji

W celu usystematyzowaniu struktury zdecydowałem się na zastosowanie wzorca MVC, w którym następuje bardzo czytelny podział odpowiedzialności miedzy częściami programu, i który jest wyjątkowo wygodny do implementacji w języku Ada. Zbiór 2 obiektów chronionych i zadanie modelujące fizykę realizują Model, na który składają się ustawienia graczy i stan planszy do gry. Za realizacje widoku odpowiada obiekt chroniony przechowujący ustawienia widoku i osobny wątek, generujący wizualizacje modelu z wykorzystaniem bibliotek Open GL. Ostatnia część schematu czyli kontroler jest realizowana przez kolejny wątek, obsługujący wejście i odpowiednio na nie reagujący. Część implementacji nie mogla spełnić w pełni tych założeń, na wskutek ograniczeń narzuconych przez bibliotekę odpowiedzialna za wizualizacje.

Struktura Logiczna programu

Poniżej zamieszczam schemat obiektów, zadań i komunikacji miedzy nimi.



Szczegóły implementacji

Przechowywanie danych - typy

W celu przechowywania informacji o stanie gry stworzyłem znaczną ilość typów. Te wykorzystywane przez konkretną część programu czy też bardziej złożone znalazły się w odpowiednio właściwej części programu lub w odrębnym pliku zawierającym opis typu i kilka funkcji na nich działających. Pozostałe typy, a zwłaszcza wyspecjalizowane typy podstawowe znalazły się w pakiecie Common_Data, odpowiedzialnym za przechowywanie danych wykorzystywanych w większej części programu.

Przechowywanie danych - obiekty chronione

Jak już wcześniej wspomniałem informacje o modelu są przechowywane w obiektach chronionych. Ma to na celu umożliwienie wydajnej pracy wielu wątkom współpracującym jednocześnie. Drugim rozważanym rozwiązaniem było zastosowanie Zadania- Serwera Danych, pomysł ten został zarzucony z powodu zysku wydajności: możliwość równoległego odczytu z zadania za pomocą funkcji pozwala wykonywać więcej operacji jednocześnie, podczas gdy wszystkie spotka-

nia byłyby dla takiego serwera blokujące. Za dane opisujące bieżący stan naszej gry odpowiadają 3 obiekty chronione:

Board_State odpowiedzialny za przechowywanie informacji na temat planszy - przechowuje wszystkie klocki, informacje o kulce i padach - odbijakach. Player_State odpowiedzialny za informacje o graczach, zdobyte punkty i konfiguracje sterowania, oraz View_State przechowujący dane na temat kątów i odległości widoku od planszy. Poza nimi występuje jeszcze obiekt chroniony Keyboard_State przechowujący stan klawiatury.

Przetwarzanie danych - zadania

Na program składają się 2 zadania główne, pierwszy odpowiedzialny za wizualizację stanu gry, oraz drugie odpowiadające za realizację "fizyki". Poza tym występują jeszcze zadania pomocnicze, odpowiedzialne za przetwarzanie wejścia, wymuszanie odświeżania, logowanie informacji. Pierwsza warte rozszerzonego opisu zadanie jest elementem modelu: modyfikuje ono obiekt symulując upływ czasu, realizuje ruch padów i kulki, wykrywa kolizje obiektów odpowiednio na nie reaguje. Kolejne zadanie wykonuje zapętloną procedurę GlutMainLoop, która jest częścią biblioteki GLUT odpowiedzialnej za wyświetlanie grafiki. W trakcie inicjalizacji zadanie inicjalizuje okno, po czym wchodzi w ww zapętloną procedurę. Procedura ta w trakcie działania wywołuje funkcje wcześniej zdefiniowane jako przeznaczone do obsługi zdarzeń (callback). Z kolei obsługa zdarzeń często ingeruje w obiekt przechowujący stan klawiatury, lub też za pomocą menu wykonuje inne zmiany. Kolejne zadanie odczytuje stan klawiatury z obiektu chronionego Keyboard_State, i na jego podstawie modyfikuje położenie padów.

Wizualizacja wyników

Za wizualizacje stanu gry odpowiada zadanie Process_Graphics. Zadanie to zawiera funkcje rysującą bieżący stan gry na ekranie, funkcja jest wywoływana przez zewnętrzny Task wymuszający przerysowanie po zadanym w ustawieniach czasu. Pakiet zawierający to zadanie zawiera funkcje pomocnicze rysujące pojedyncze obiekty.

Integracja z światem zewnętrznym

Pierwszy kontakt programu z otoczeniem to wczytanie pliku zaraz po uruchomieniu. Wczytywana jest domyślna plansza którą w każdej chwili można zmienić rozpoczynając nową grę. Poza tym program za pomocą specjalnych procedur biblioteki GLUT pozwala sterować za pomocą klawiatury i JOYSTICKA. W trakcie działania wszystkie wydarzenie są zapisywane w pliku logującym uruchomienie programu, oraz zgodnie.

Podsumowanie

Instrukcja Obsługi

Program tuż po uruchomieniu wyświetla przykładową plansze (numer 0), pozwala tez sterować za pomocą kombinacji klawiszy A i D. Jest to tryb "DEMO", grywalny ale służący raczej do prezentacji. W każdej chwili działania gry można użyć menu, dostępnego pod prawym klawiszem myszy. Omówię po krotce dostępne opcje:

- Nowa Gra: Menu pozwalające rozpocząć całkowicie nową grę, z listy rozwijanej należy wybrać numer oczekiwanej planszy.
- Zatrzymaj: Zatrzymuje grę.
- Wznów: Wznawia grę.
- Sterowanie: Można zmienić sterowania dla każdego z obu graczy, należy wybrać spośród opcji
  - Klawiatura A-D,
  - Klawiatura J-L,
  - Pad Lewo-Prawo.
  - Dla gracza zielonego można też wybrać sterowanie przez komputer.
- Rozmiar Okna: Domyślnie gra wyświetla się w oknie, można dla wygody rozszerzyć ja na cały ekran.
- Zakończ: Pozwala wyjść z programu, potrzebne zwłaszcza w trybie pełnego ekranu.

Możliwość przyszłej rozbudowy

Gdyby dalsza rozbudowa miała być przeprowadzona, będzie ona bardzo łatwa, pod względem integracji z istniejącym kodem. Jako przykład może służyć dodana już po zamknięciu przeze mnie programu obsługa gracza komputerowego - jego dodanie sprowadzało się do stworzenia zadania, czytającego stan planszy z obiektu Board_State, oraz zapisującego sterowanie poprzez naciśnięcie wirtualnego przycisku na klawiaturze (obiekt Keyboard_State, metody Key_Down i Set_Key_Up). Kolejną możliwą rozbudową jest stworzenie bardziej złożonego systemu zapisu plansz, tak aby możliwe było tworzenie plansz o rożnych rozmiarach, zapisywanie w nich informacji o autorze czy tez wymaganiach. Wymagałoby to rozbudowy metod Load_Pattern_From_File, oraz ewentualnie rozszerzenie obiektu Board_State o dodatkowe zmienne, czy też wypisanie dodatkowych informacji w Wizualizacji.

Wnioski z przeprowadzonego projektu

W trakcie tworzenia projektu nauczyłem się systematycznego podejścia do projektowania złożonych projektów. Zdecydowanie doceniłem łatwość implementacji bardziej złożonych problemów za pomocą wielu niezależnych zadań - procesów. Nauczyłem się też adaptować założenia projektowe do możliwości wybranego sytemu/środowiska, oraz dostosowywać je w celu ułatwienia implementacji. Składnie i możliwości języka Ada ułatwiły taką implementacje, szczególnie poprzez dostarczanie struktur języka dedykowanych do synchronizacji dostępu do zasobów. Z kolei GLUT jako menedżer okna i system obsługujący wejścia był wspólnym, stosunkowo słabym ogniwem, ułatwił jednak wiele czynności, powodując że stworzenie wizualizacji 3D czy obsługa Joysticka stały się możliwe.