Zaawansowane programowanie w C++

Projekt: Gra inspirowana grą Angry Birds

Przygotował: Łukasz Kowalczyk Prowadzący: Rafał Biedrzycki

Wprowadzenie

Celem projektu jest zaimplementowanej gry, która będzie wariancją gry Angry Birds. Jest to gra w której z lewej strony planszy są wystrzeliwane obiekty, tak aby zniszczyć wszystkie obiekty z prawej strony planszy chronione elementami pasywnymi. Użytkownik dobiera kąt i siłę z jaką będą wystrzelone obiekty, tak aby uzyskać pożądany efekt używając minimalnej liczby obiektów z lewej strony planszy, których liczba jest ograniczona.
W realizowanej wersji będzie zrealizowany tryb dwuosobowy, turowy. W pierwszej turze pierwszy zawodnik będzie ustawiał obiektu do zniszczenia i elementy pasywne, przy czym będzie miał ustalone ograniczenia co do ich umiejscowienia. Natomiast drugi gracz będzie próbował zniszczyć te obiekty (jedynie atakując obiekty gracz będzie mógł zdobyć punkty), ilość strzałów jakie będzie mógł wykonać gracz będzie ograniczona. W następnej turze gracze zamieniają się rolami.

Używanie aplikacji

Wymagania

Kompilacja aplikacji wyamga zainstalowania trzech repozytoriów w systemie:

libbox2d-dev
libsfml-dev
ibboost1.55-dev

Kompilacja:
Instrukcja kompilacji I uruchomienia znajduje się w pliku README.md .

Zrealizowana funkcjonalność

Z dokumentacji wstępnej został zrealizowana funkcjonalność, polegająca na stworzenie fundamentów pod grę.

Fundamenty zrealizowane:
- Zarządzanie światem – została stworzona hierarchia klas odpowiedzialna za część logiczną i wizualną świata. Klasa główna zarządzająca światem pozwala na poprawę stworzenie obiektów, I następnie rysowanie ich i aktualizowanie ich stanu logicznego. Klasa główna zaimplementowane ma odbieranie komunikatów od tych obiektów. Jedynymi komunikatami w tym momencie zrealizowanymi są komunikaty o niszczeniu obiektu. Klasa ta po zakończeniu pracy poprawnie usuwa wszystkie obiekty. Przez tą klasę również są przekazywane komendy sterowania.

    Obiekty które należą do tej hierarchii, mogą ze sobą oddziaływać fizycznie.
- Implementacja wzorca projektowego słuchacza dla nasłuchiwania wydarzeń z klawiatury I myszki. Implementacja tego wzorca pozwala na łatwe dotłaczanie nowych komponentów które reagują na polecenia użytkownika.

- Sterownia katapultą – zostało również zaimplementowane sterownie katapultą w której użytkownika może ustawić kąt oraz siłę wystrzału, przy ciągłym poglądzie wizualnym aktualnych parametrów strzału. Implementacja zawiera również tworzenie nowego obiektu pocisku I odpowiednie rozpędzanie.

- Została zrealizowania konfiguracja projektu pod systemy Windows I Ubuntu/Mint.

Problemy przy realizacji

W projekcie tym nie zostało zrealizowanych większość funkcjonalności zapowiedzianych w dokumentacji wstępnej, głównym powodem ich nie zrealizowania był rozpad zespołu i ich zbyt duża ilość na jedną osobą.

Lista zadań

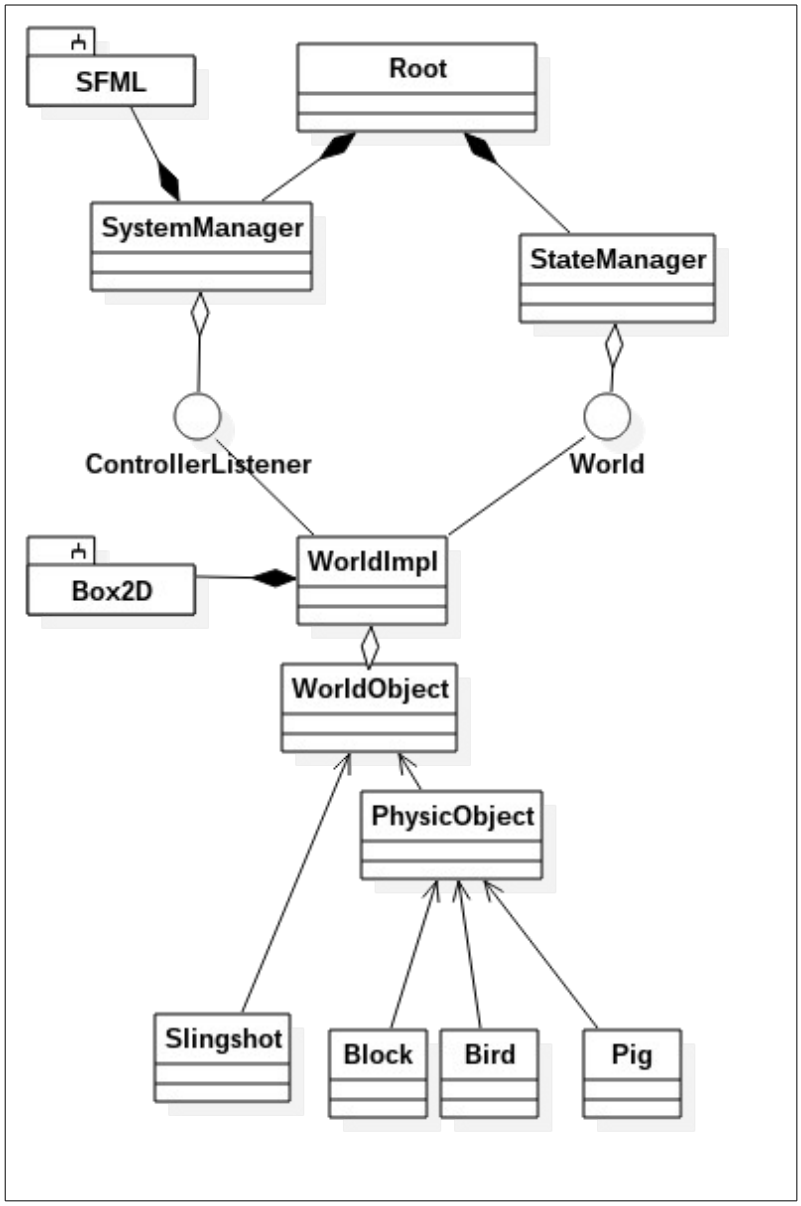
Funkcjonalność aplikacji	Zadanie do wykonania w ramach funkcjonalności	Przewidywany czas zadania/czas ralizacji
Ekran główny	Menu wyboru opcji rozgrywki	4h/(brak)
	Menu zarządzania rozgrywką	4h/(brak)
Ogólne	Implementacja kamery	3h/(brak)
	Implementacja systemu zarządzającego zasobami graficznymi	3h/(brak)
	Stworzenie klasy do zarządzania scena i na tej scenie obiektami Poza samym zarządzaniem powinna ona pozwolić na renderowanie sceny	12h/(18h)
	Integracja tej klasy z silnikiem fizycznym	5h/(5h)
Tryb ustawienia elementów	Interfejs pokazujący pozostałe elementy do ustawienia na planszy.	1h/(brak)
	Automatyczna modyfikacja mapy po prawidłowym dodaniu elementu.	2h/(brak)
	Uwzględnienie ograniczeń dla rozmieszczenia elementów.	3h/(brak)
	Implementacja sterowania elementami	5h/(brak)
	Umożliwienie graczowi testowania stabilności stworzonej konstrukcji	2h/(brak)
Tryb niszczenia elementów	Tryb użytkownika pokazujący pozostałe pociski do wystrzelenia.	1h/(brak)
	Animacje towarzyszące wystrzeleniu obiektów	3h/(brak)
	Implementacja sterowania proca	5h/5h
	Animacje niszczenia	3h/(brak)
	Implementacja niszczenia elementów przy uderzeniu	5h/(brak)

Interfejs podsumowania	Informacja o punktacji, zarządzanie zmianą tur.	2h(brak)

Łączny przewidywany czas	63h//28h

Do tego czasu pracy należy również doliczyć 10 godzin poświęconych na stworzenie szkieletu i konfiguracje projektu.

Diagram klas



Opis funkcji poszczególnych klas znajduję się w kodzie.

Testy

Kod nie został pokryty testami jednostkowymi.
Cały kod składa się z 1167 linii kodu.