Projekt programistyczny						
Rok akademicki	Termin	Rodzaj studiów	Kierunek	Prowadzący	Grupa	Sekcja
2014/2015	Wtorek 12:45 - 15:00	SSI	INF	dr inż. Arkadiusz Biernacki	GKiO3	1

Odpowiedzialność klas i interfejsy

Danmaku Shooter

Skład sekcji:

Buchała Bartłomiej

Forczmański Mateusz

Motyka Marek

Wudecki Wojciech

Model interakcji

Podmodel klas

1. **Okno gry**: utworzenie okna w języku WinAPI, inicjalizacja i konfiguracja Direct3D 9, implementacja zegara gry.

2. **Klasa TitleScreen**: narysowanie ekranu powitalnego, utworzenie menu z wszystkimi możliwościami, utworzenie klasy Menu z szczegółowymi możliwościami zmian, zapisywanie i odczytywanie ustawień z pamięci, utworzenie sprawnego przejścia z ekranu powitalnego do gry i na odwrót.

3. **Klasa Game**: narysowanie interfejsu gry, wyświetlenie wszystkich danych: liczby żyć i bomb, przechowywanych w klasie Player oraz wyniku i liczby grejzu, przechowywanych w klasie Game, usunięcie pocisków gracza i wrogów z pamięci gdy wyjdą poza planszę, umożliwienie zatrzymania i zakończenia gry, wejścia do menu oraz zapisania wyniku gry do pamięci.

4. **Klasa Sprite**: rysowanie sprajtów na ekranie gry, implementacja translacji, skalowania i rotacji, umożliwienie przechowywania i wyboru większej liczby tekstur.

5. **Klasa Input**: enkapsulacja wszystkich funkcji do obsługi klawiatury, synchronizacja z klasami integralnymi (gra oraz ekran powitalny). Stworzenie uniwersalnych klawiszy – *Shoot*, *Bomb*, *Focus* itp., które będą połączone z odpowiadającymi im wartościami z klasy Input.

6. **Klasa Player**: uniemożliwienie wyjścia graczowi poza planszę, utworzenie trybu *focus*, umożliwienie mu strzelania pociskami i bombami, narysowanie bomby i pocisków, synchronizacja z klasą Input.

7. **Klasy typu Bullet**: zrealizowanie ruchu pocisków zgodnie ze wskazanymi torami - domyślnie po prostym wektorze, z możliwością zmiany na wskazany tor, umożliwienie przyspieszenia, silna parametryzacja, narysowanie kilku kształtów pocisków w różnych kolorach.

8. **Klasy Pattern i Spellcard**: generowanie i układanie wzorów z pocisków, reagowanie na usunięcie pocisków gdy wyjdą poza planszę lub zostaną usunięte przez bombę, realizacja bonusów za pokonanie karty czarów bez straty życia.

9. **Klasa Hitbox**: implementacja, narysowanie sprajta hitboxa, zrealizowanie obsługi zdarzeń i styków dwóch hitboxów, wykrywanie *graze* dla hitboxów klas Player i EnemyBullet (poprzez ustawienie pewnej odległości) oraz obsługa utraty życia przy nałożeniu się hitboxów tych klas.

10. **Klasa Bomb**: implementacja, reakcja na akcję ze strony gracza, narysowanie efektu graficznego, integracja z Timerem (czas trwania bomby, a jej efekt na znajdujące się na mapie pociski/efekt graficzny).

11. **Klasy typu Bonus**: implementacja, narysowanie sprajtów, synchronizacja z klasami Player (zebranie, zwiększenie powera) oraz Game (zwiększenie wyniku), zwolnienie pamięci po zebraniu przez gracza lub wyleceniu z planszy.

12. **Klasa Enemy**: implementacja, umożliwienie strzelania i wykorzystywania patternów, realizacja wrażliwości na pociski gracza, narysowanie sprajtów wrogów.

13. **Klasa Stage**: przechowywanie wszystkich obiektów wrogów, decydowanie który jakiego wzoru pocisków używa, silna integracja z timerem – to Stage ma wiedzieć, w której sekundzie pojawiają się wrogowie, kiedy mają strzelić i zejść ze sceny, narysowanie tła.

Specyfikacja interfejsów

1. Drawable

- **Wykorzystywany w:** Klasie Sprite.
- **Założenie:** Obiekt klasy implementującej ten interfejs może zostać wyświetlany na ekranie
- **Funkcjonalności** Metoda Draw() będzie odpowiedzialna za przedstawienie elementu klasy na ekranie. Dodatkowo, implementacja tego interfejsu będzie oznaczała, że klasa będzie powiązana z plikami graficznymi (np. teksturami, modelami 3D).

2. IPattern

- **Wykorzystywany w:** wszystkich klasach typu Pattern (Wzorców).
- **Założenie:** Zawarcie opisu rozwiązania problemu dla klas typu Pattern.
- **Funkcjonalności:** Interfejs służy do automatyzacji procesu tworzenia nowych patternów, narzucajac sposób implementacji, modyfikacji i obsługi kodu źródłowego. Składa się z on 3 metod: inicjalizacji (Initialize()), aktualizacji stanu (Update()) i rysowania(Draw()). Pierwsza z nich określa, ile pocisków zostanie początkowo utworzonych , jakiego typu będą to pociski, ich grafiki itp. Aktualizacja stanu pozwala stwierdzić, jak zmieniają się pociski Patternu w czasie (ruch, obrót, skalowanie). Draw() odpowiada głównie za narysowanie pocisków, pozwalając jednak osobie implementującej na wprowadzenie drobnych zmian.

3. IException

- **Wykorzystywany w:** klasach wyjątków.
- **Założenie:** rozszerzenie funkcjonalności wyrzucanych wyjatków.
- **Funkcjonalności:** Klasy korzystające z tego interfejsu mają na celu zapobiec jakiemukolwiek błędowi związanemu z obsługą programu, przechwycić go, a następnie poinformować o tym użytkownika i wykonać czynność przewidzianą na wypadek błędu. Implementując ten interfejs, uzyskujemy dodatkowe możliwości podczas wyrzucenia wyjątku przez program: otrzymanie komunikatu i/lub wyświetlenie MessageBoxa.

Podział zadań

Student	***Klasy do zaprojektowania***
Bartłomiej Buchała	a) Klasa Player b) Klasa Game c) Klasy typu Bullet d) Klasa Bomb
Mateusz Forczmański	a) Model UML b) Okno Gry b) Klasy Pattern c) Klasa Spellcard d) Klasa Sprite
Marek Motyka	a) Klasa Input b) Klasa Hitbox c) Klasy typu Bonus
Wojciech Wudecki	a) Klasa TitleScreen b) Klasa Enemy c) Klasa Stage