Informatyka II st. PGK sem. III 28.05.2019

Snake
Inżynieria Oprogramowania w Grach

Zespół:
Laura Dymarczyk
Agata Dziurka
Kamil Karpiński
Adrian Rupala

Koncept gry

Nasz projekt ma na celu zaprojektowanie i stworzenie gry w węża (ang. Snake), jest to nasza interpretacji klasycznej gry istniejącej od 1976 roku.

Gra polega na poruszaniu tytułowym wężem po prostokątnej planszy, na której pojawiają się punkty (jabłka), które są pożywieniem naszego zwierzątka. Pożywienie pozwala wężu się rozrastać, a celem gry jest zapełnienie całej przestrzeni planszy ciałem naszego węża.
W grze można ponieść porażkę poprzez dobicie do zewnętrznych krawędzi planszy, tzw. “ścian”.

Wzorce projektowe

W inżynierii oprogramowania **wzorcem projektowym** nazywa się uniwersalny, sprawdzony w praktyce, słowny opis rozwiązania jednego z często pojawiających się, powtarzalnych problemów projektowych. Służy on do zobrazowania powiązań i zależności między klasami oraz obiektami i ułatwia tworzenie, modyfikację, a nawet i pielęgnację kodu źródłowego.

Wzorce projektowe można podzielić na następujące grupy:

- *Konstrukcyjne (kreacyjne)* - opisują proces tworzenia nowych obiektów. Ich zadaniem jest tworzenie, inicjalizacja oraz konfiguracja obiektów, klas i innych typów danych.
- *Strukturalne* - opisują struktury powiązanych ze sobą obiektów.
- *Operacyjne (czynnościowe)* – opisują zachowanie i odpowiedzialność współpracujących ze sobą obiektów.
- *Sekwencjonowania* - rozwiązują problem znajdowania zależności pomiędzy występowaniem określonych zdarzeń w czasie.

Wykorzystane w projekcie wzorce projektowe

Wzorce konstrukcyjne:

- *Prototyp* - użyty przy tworzeniu instancji jabłek i kolejnych segmentów węża.
- *Singleton* - użyty do stworzenia instancji planszy, której pojedynczy obiekt jest wystarczający dla działania całej rozgrywki, używany do stworzenia aktualnego stanu poruszania się węża oraz obiekt służący do zliczania punktów w rozgrywce.

Wzorce strukturalne:

- *Kompozycja* - użyty do reprezentowania węża poprzez liniową strukturę rekurencyjną. Pomaga to przy przesuwaniu całego ciała węża w sposób ciągły.

Wzorce operacyjne:

- *Stan* - wąż może znajdować się w jednym z czterech stanów poruszania się: do góry (przyrost y), do dołu (zmniejszanie y), w lewo (zmniejszenie x) lub w prawo (przyrost x). Istnieje również stan rozgrywki, który może być wartością wygraną, przegraną bądź w toku.
- *Komenda* - pozwala zmienić stan ruchu węża, reaguje na wydarzenia klawiatury.
- *Obserwator* - Istnieją dwie instancje obserwatorów. Jeden z nich obserwuje węża względem jabłka i daje im sygnał, gdy spotkają się w tym samym polu. Drugi obserwator obserwuje węża i ścianę, wysyła on sygnał do węża, aby poinformować go o zderzeniu.

Wzorce sekwencjonowania:

- *Metoda aktualizująca* - zliczanie wykorzystane podczas kalkulowania punktów.
- *Renderer* - wizualizacja stanu gry, podpięcie assetów, obsługa efektów graficznych.

Opis realizacji gry

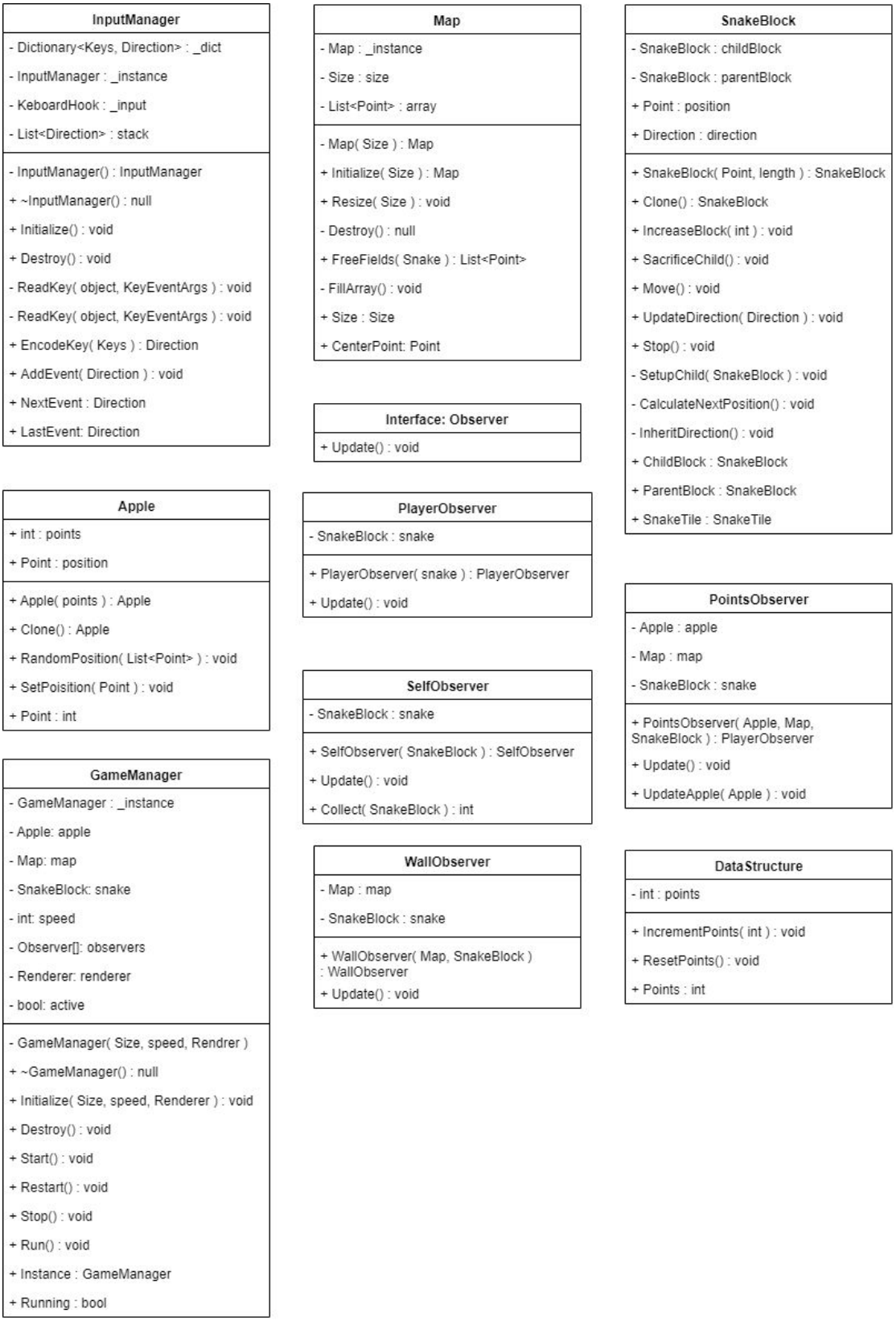
Głównymi instancjami gry są: wąż, plansza oraz stan gry.

Wąż składa się z segmentów, zewnętrznie, z punktu widzenia programu, dostępna jest głowa węża, która posiada w sobie połączone liniowo kolejne elementy ciała stworzenia. Podczas poruszania się postaci głowa korzysta z danych stanu węża, który mówi wężu, które koordynaty ulegają zmianie i w jaki sposób. Reszta ciała natomiast dziedziczy koordynaty po swoim rodzicu (wchodzi na miejsce poprzedniego elementu).

Plansza i stan gry są zaimplementowane jako singletony, co wymusza istnienie tylko jednej instancji obiektu klasy. Plansza może mieć wprowadzane różne wymiary, a wąż pojawia się zawsze w środku planszy. Zdefiniowany został domyślny rozmiar węża (3 jednostki na planszy), który jest do zmiany tylko w kodzie źródłowym (nie ma możliwości ingerowania w ten parametr z zewnątrz kodu).

W grze zaimplementowano czterech różnych obserwatorów, jeden z nich sprawdza, czy wąż uderzył w ścianę, jeżeli tak się stanie, wysyłany jest sygnał do stanu gry i gra kończy się przegraną. Kolejny obserwator pilnuje, czy wąż nie zjada sam siebie, jeżeli tak, to wysyła informacje do węża, aby zmniejszył swoją długość do miejsca ugryzienia, a licznik punktów zmniejsza się o konkretną ilość. Następny obserwator obserwuje eventy klawiatury i informuje stan ruchu węża o zmianach jego wartości. Ostatni obserwator zwraca uwagę na pozycję głowy węża względem jabłka, gdy te zaczną się pokrywać zostaje wysłana informacja do węża i jabłka, wąż zwiększa swoją długość o 1 pole planszy, a jabłko znika i pojawia się kolejne w niezajętym miejscu na planszy.

Diagramy klas



Uruchamianie gry

Wymagane oprogramowanie

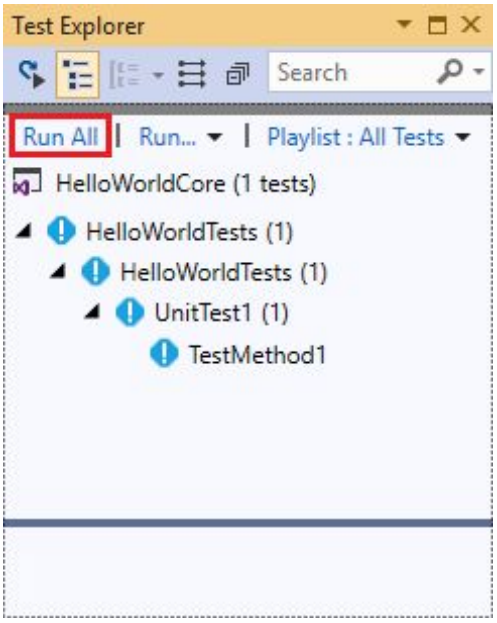
System Operacyjny: Windows
Microsoft Visual Studio 2019 wersja 16.1.1
Microsoft .NET Framework wersja 4.8.03752
Biblioteka KeyboardHook:
http://www.mediafire.com/file/1777b9cwl4rh5hf/GlobalKeyBoardHook.cs

Instrukcja uruchomienia gry

1. Przejdź do folderu projektu a następnie do ścieżki *…\Snake\bin\Release*
2. Uruchom aplikację **Snake.exe**.

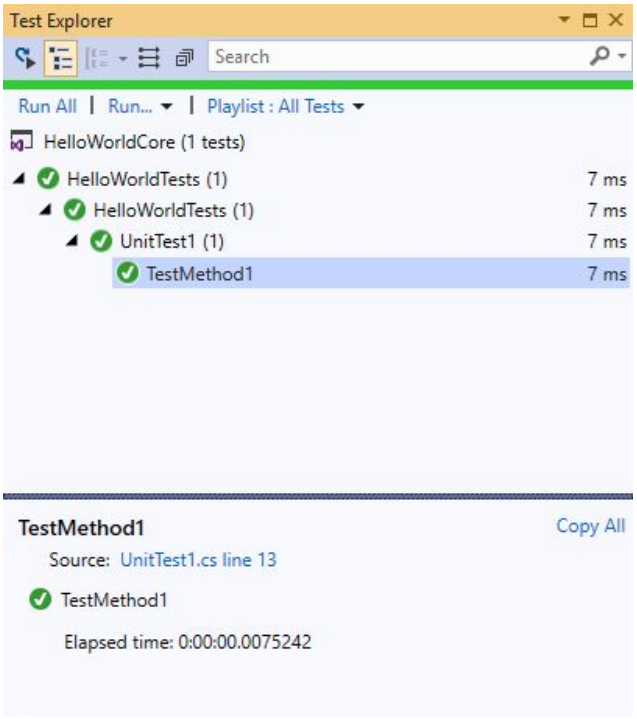
Uruchamianie testów jednostkowych

1. Otwórz **Eksplorator testów**, wybierając *tests* > *Windows* > *Eksplorator testów* na pasku menu u góry.
2. Uruchom testy jednostkowe, klikając pozycję **Uruchom wszystkie**.



Rysunek 1. Wskazanie pozycji “Uruchom wszystkie” Eksploratora testów
Źródło: https://docs.microsoft.com/pl-pl/visualstudio/test/getting-started-with-unit-testing?view=vs-2019

3. Po zakończeniu testów, zielony znacznik wyboru wskazuje, że test zakończony pomyślnie.
   Ikona z czerwonym symbolem "x" oznacza, że test nie powiódł się.



Rysunek 2. Zobrazowanie wyglądu zakończonych pomyślnie testów
Źródło: https://docs.microsoft.com/pl-pl/visualstudio/test/getting-started-with-unit-testing?view=vs-2019