Łukasz Reszka
Id: 300257

Plants vs Zombies – podsumowanie projektu

1. Realizacja projektu i wykorzystane narzędzia

Tworząc swój projekt korzystałem z następujących narzędzi:

- Eclipse IDE
- Maven
- repozytorium git (oraz z gitlaba)
- biblioteka JUnit

Dodatkowo wykorzystałem zależności Mavena do ściągnięcia silnika do gier Slick2D, który stał się szkieletem całej mojej gry. Projekt napisałem w języku Java. Do testów użyłem biblioteki JUnit. Etapy mojej pracy umieszczałem sukcesywnie w repozytorium. Wykorzystana w grze grafika i muzyka zostały znalezione w Internecie i ewentualnie poddane obróbce w różnym stopniu.

2. Struktura gry

Całą grę podzieliłem na 3 następujące pakiety:

- *core* – wymuszony przez wybrany silnik. Każda gra z wieloma „stanami" korzystająca ze Slick2D musi zawierać klasę dziedziczącą po StateBasedGame. W mojej grze jest to klasa Core. Zarządza ona ogólnymi ustawieniami okienka gry (wymiary, nazwa na górnym pasku) i samej gry (dodawanie wszystkich „stanów" gry – widoku menu głównego, planszy i instrukcji)
- *menu* – zawiera klasę Button (służy do obsługi przycisków służących do poruszania się w obrębie gry), klasę Instruction (obsługuje wyświetlanie instrukcji i powrót do

menu głównego) oraz Menu (obsługuje menu główne wraz z przyciskami i muzyką)

- *game* – stanowi prawdziwe serce gry. Sama klasa Game obsługuję uaktualnianie i renderowanie poszczególnych elementów gry i obiektów innych klas z pakietu game. Dodatkowo uruchamia specjalny podkład muzyczny, sprawdza koordynaty myszy (wykorzystywanej w grze) i status wciśnięcia/przytrzymana jej lewego klawisza oraz resetuje grę przy powrocie do menu głównego. Pozostałe klasy w pakiecie to BuyHeroButton (obsługa kupowania roślinek), Hero (obsługa postaci roślinki), Lawn (obsługa całego trawnika – głównego elementu planszy gry), PieceOfLawn (obsługa pojedynczego elementu trawnika), RandomSunsGenerator (generowanie spadających słoneczek), Sun (obsługa postaci słoneczka i jego ewentualnego spadania) oraz SunCounter (obsługa wyświetlacza aktualnej ilości słoneczek gracza).

3. Zaimplementowane test jednostkowe

Za pomocą biblioteki JUnit zaimplementowałem testy sprawdzające poprawność działania metod reset, removeSuns i addSuns klasy SunCounter. Testy te polegały na porównywaniu aktualnej liczby słoneczek (wartości zmiennej suns_number) z wartością oczekiwaną, wynikającą z działania powyższych metod.

4. Wybór silnika

Wybrany przeze mnie silnik do gier Slick2D cechuje prostota i intuicyjność, co zaważyło na jego wyborze. Stworzenie gry z wieloma widokami-stanami opierało się tylko na stworzeniu początkowej klasy i dodaniu wszystkich stanów. Natomiast w klasach reprezentujących stany uaktualnianie i wyświetlanie

elementów odbywają się w dwóch metodach – render i update. Dzięki Slick2D ominąłem szereg trudności związanych z bezpośrednio zaimplementowaną obsługą zdarzeń; wiedziałem również, gdzie konkretnie wywoływać poszczególne metody, gdyż wszelkie pętle gry i inne mechanizmy były już zaimplementowane. Slick2D posiadał również gotowe metody związane z ładowaniem i wyświetlaniem obrazków i animacji oraz odtwarzaniem muzyki.

Przed dokonaniem wyboru silnika do gry przeglądałem się różnym opcjom. Jedną z nich była biblioteka JavaFX. Nie zdecydowałem się jednak na nią ze względu na wyższy poziom skomplikowania i większy nakład pracy związany z jej wykorzystanie. Korzystając z JavaFX stanąłbym przed problemami, których nie istniały przy korzystaniu ze Slick2D. Oprócz zagłębiania się w różne klasy dotyczące poszczególnych kontrolek i ich metod, musiałbym samodzielnie zaimplementować system obsługi zdarzeń oraz rozwiązać problem związany z pętlą gry (np.: korzystając z klasy Timeline). Wszystkie wymienione przeszkody z pewnością mocno rozbudowałyby kod projektu. Pod względem ładowania muzyki i grafiki JavaFX nie odbiega od silnika Slick2D.

Inną rozważaną przeze mnie opcją był jMonkeyEngine. Pod względem systemu obsługi zdarzeń miałem takie same zastrzeżenia co do JavaFX (konieczność zastosowania ActionListener, AnalogListener itd.). Zniechęciła mnie również złożoność implementacyjna związana z jMonkeyEngine, w porównaniu do prostoty oferowanej przez Slick2D. Bez wątpienia to właśnie Slick2D wydał mi się najbardziej intuicyjny ze wszystkich przeglądanych opcji.

5. Wnioski z realizacji

Pomimo stworzonego planu sprintów i wyznaczonych terminów, nie udało mi się ukończyć projektu na czas. Przyczyn tego stanu rzeczy dopatrzyć się można w kilku obszarach:

- wpływ niewziętych po uwagę czynników zewnętrznych – obciążenie związane ze studiowaniem w skutecznym sposób uniemożliwiło mi realizację poszczególnych sprintów w terminie,
- komplikacje i trudności związane z niepoznanymi wcześniej narzędziami, których użycie było narzucone i często wymagało całkiem dogłębnego poznania tematu
- brak wcześniejszego doświadczenia związanego z tworzeniem gier i językiem Java.