1. Pictorial Tetris

Autorzy:
- Michał Szczepańczyk – Programista, system budowania projektu
- Kamil Warchoł – Programista, testy, dokumentacja
- Adrian Wysocki – Programista, testy

2. Opis projektu

Projektem jest gra komputerowa przypominająca grę Tetris. Różnice polegają m.in. na tym, iż zamiast układania poziomych linii klocków układany jest wybierany przez gracza obrazek. Gracz kontroluje obrotem nadchodzących losowo wybieranych fragmentów obrazka oraz miejscem jego usadowienia. Gra polega na ukończeniu obrazka składając klocki w postaci kwadratów w jak najszybszym czasie.

Gra posiada 3 poziomy trudności, modyfikujące szybkość opadania klocków oraz punktację za czas ukończenia gry. Dodatkowo można modyfikować ilość klocków, na jakie obrazek zostanie podzielony – im ich więcej, tym trudniej i więcej punktów.

3. Założenia wstępne przyjęte w realizacji projektu

Program powinien pozwalać na wybór jednego z kilku zdefiniowanych obrazków. Powinna również istnieć możliwość ustalenia na ile kwadratów obrazek ma zostać podzielony oraz tempa ich ruchu. Kwadraty mogą przesuwać się w dół skokowo o odległość równą bokowi kwadratu.

W wersji poszerzonej program powinien:
a) gwarantować w miarę płynny ruch kwadratów,
b) kwadraty mogą być nie tylko odwrócone, ale również odbite symetrycznie wzdłuż pionowej lub poziomej osi.
c) obraz wzorcowy mógłby być wyświetlany w studni w postaci czarno białej, a spadające kwadraty budowałyby go w wersji kolorowej.

Należy zadbać, aby spadający kwadrat zawsze dał się ulokować we właściwym miejscu. To znaczy, że jeżeli na przykład mamy zbudowany dopiero najniższy wiersz obrazka, nie należy „spuszczać” kwadratu z jego wierzchołka.

Wszystkie założenia wstępne zostały zaimplementowane w projekcie. Jedynym wyjątkiem jest punkt b). Powodem tego były wykonane testy przez samych programistów oraz ich znajomych. Odbicie lustrzane praktycznie uniemożliwiało pomyślne ukończenie gry i zabijało przyjemność z rozrywki. Pomimo stosunkowo łatwej implementacji takiej funkcjonalności twórcy postanowili z niej zrezygnować zupełnie.

4. Analiza projektu

4.1. Specyfikacja danych wejściowych

Jednymi z danych wejściowych są pliki z obrazami w formacie jpg. Program domyślnie przechowuje w pamięci 4 obrazy i każdy z nich powinien znajdować się w katalogu „data” i nazywać się „imgX.jpg”, gdzie X jest indeksem tablicy, w której zostanie zapisane zdjęcie. Pozostałe dane wejściowe pochodzą od użytkownika i głównie polegają na jego sterowaniu w oknach menu oraz kontrolowaniu klocków.

4.2. Opis oczekiwanych danych wyjściowych

Z racji faktu, iż ten projekt był grą komputerową oczekiwanymi danymi było wyświetlanie gry.

4.2. Zdefiniowanie struktur danych

Większość logiki gry opiera się na stanach. Stan jest implementacją abstrakcyjnego obiektu stanu i definiuje aktualne zachowanie się gry. Rozumiane jest to poprzez aktualny stan tego, co jest wyświetlane, akcji, jakie można podjąć i ewentualnej aktualizacji stanu zgodnie z aktualnymi jego parametrami. Mniej formalnie: gracz będąc w menu głównym może wybrać odpowiednią opcję (aktualizacja widoku), uruchomić tą opcję (przejście do kolejnego stanu) lub ją anulować (przejście do poprzedniego stanu). Stany przechowywane są w strukturze stosu i działają praktycznie tak samo jak funkcje w każdym języku programowania: stan prowadzi do następnego, następny do kolejnego i wyjście z jakiegoś wraca do stanu, który go wywołał. Taka architektura ułatwia modyfikację i rozbudowę gry: użytkownik tworzy nowy obiekt stanu, implementujący interfejs stanu jako obiektu ogólnego i ustala, jakie inne stany i w jaki sposób przechodzą do nowo utworzonego stanu.

Oprócz tego zaimplementowany został widget przycisku, który jest widoczny w menu gry oraz przycisku z wyborem różnych ilości opcji.

4.3. Specyfikacja interfejsu użytkownika

Interfejs jest kontrolowany przez widok reprezentowany przez klasę View. Reszta wyjaśniona została w punkcie 4.2. na podstawie stanów gry.

4.4. Wyodrębnienie i zdefiniowanie zadań

Zadaniami gry było umożliwienie użytkownikowi wyboru zdjęcia do ułożenia, poziomu trudności, ilości klocków oraz rozpoczęcie właściwej rozgrywki.

4.5. Decyzja o wyborze narzędzi programistycznych

Do wykonania tego projektu wykorzystany został język C++ w wersji C++11, wykorzystujący wbudowaną bibliotekę STL oraz zewnętrzną bibliotekę graficzną SFML. Biblioteka SFML jest idealnym narzędziem do tworzenia prostych, dwuwymiarowych gier komputerowych (aczkolwiek wspiera programowanie gier 3D) a jej połączenie z językiem C++ daje w efekcie dobrze zoptymalizowane gry (porównując np. do silników graficznych jak Unity3D).

5. Podział pracy i analiza czasowa

Praca nad projektem została równomiernie rozłożona. Każdy członek zespołu aktywnie uczestniczył w zaimplementowaniu funkcjonalności i spełnieniu założeń projektu.

6. Implementacja

Zrozumienie kodu źródłowego i ewentualna modyfikacja gry jest możliwa dzięki załączonej dokumentacji wykonanej narzędziem DoxyGen. Oprócz tego cała dokumentacja znajduje się w plikach nagłówkowych.

7. Testowanie

Testy polegały głównie na częstym uruchamianiu gry i sprawdzaniu poprawności jej funkcjonowania. Oprócz szukania błędów brane pod uwagę były ogólne wrażenia i balans rozgrywki.

8. Wdrożenie, raporty i wnioski

Kompilacja gry polega na wygenerowaniu przez CMake odpowiednich skryptów do budowania statycznych projektów. Oprócz tego należy mieć zainstalowaną wersję biblioteki SFML 2.5.

Proces implementacji gry zmusił nas do dokładnego przemyślenia całej architektury i struktury całego programu. Testy dały często bardzo cenne spostrzeżenia na temat błędów w grze oraz samej rozgrywki. W przyszłości można zmodyfikować grę tak aby sama automatycznie skanowała katalog

z obrazkami w celu dynamicznego ich ładowania. Oprócz tego, wykorzystując architekturę stanów w stosunkowo łatwy sposób można grę rozbudowywać do jakiejkolwiek funkcjonalności.