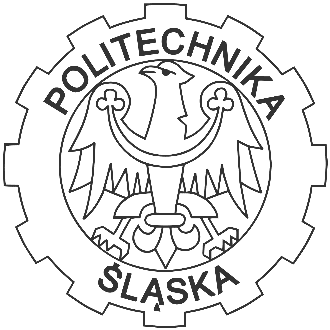
Katowice, 05.??.2016

Politechnika Śląska w Gliwicach  
Wydział Automatyki, Elektroniki i Informatyki



Laboratorium Programowania Komputerów

Temat projektu:  
Gra Klocki (Tetris)

|  |  |
| --- | --- |
| Autor | Piotr Wawrzyńczyk |
| Prowadzący | mgr inż. Maciej Długosz |
| Rok akademicki | 2015/2016 |
| Kierunek | Informatyka |
| Rodzaj studiów | SSI |
| Semestr | Letni |
| Termin laboratorium | Poniedziałki nieparzyste, 12.00 – 13.30 |
| Grupa/Sekcja | 3/3 |

1. Treść zadania

*Napisać program wzorowany na grze „Tetris”. Przesuwające się w dół elementy o różnych kształtach powinny być układane przez użytkownika w taki sposób, aby tworzyły ciągłe wiersze, które automatycznie znikają zwiększając pulę punkową użytkownika. Gra powinna mieć funkcję tworzenia historii wyników, która będzie przechowywana w pliku.*

1. Analiza problemu

Plansza gry jest reprezentowana przez tablicę dwuwymiarową X, Y, gdzie X i Y są odpowiednio szerokością i wysokością pola gry. W programie rozmiary planszy gry zostały powiększone, by nadmiarowe pola (dwa na osi odciętych i cztery na osi rzędnych) umożliwiały łatwiejsze sprawdzanie kolizji i poprawności obrotów. Rozmiar planszy nie zmienia się w czasie gry, więc tablica może być deklarowana statycznie. Co więcej, pola zapisane w tablicy dwuwymiarowej są jednoznacznie identyfikowane przez jej indeksy, więc odwoływanie się do nich jest uproszczone.

W rotacji klocków został wykorzystany algorytm obliczania położeń poszczególnych członów, szerzej opisany w dalszej części dokumentacji. Inną możliwością było wpisanie do kodu programu wszystkich kombinacji obrotu każdego z klocków, jednakże rozwiązanie to było by żmudne, czasochłonne i mało interesujące.

Zmiana rozmiaru planszy oraz skalowanie całego ekranu gry jest możliwe, ale tylko poprzez modyfikację odpowiednich stałych w kodzie. *Nie jest to możliwe w trakcie działania programu.*

Do obsługi grafiki została wykorzystana biblioteka SFML, ze względu na swoje możliwości, prostotę oraz istnienie odpowiedniego połączenia z językiem C.

1. Specyfikacja zewnętrzna

Celem gry jest zdobycie jak największej liczby punktów poprzez tworzenie wierszy złożonych z klocków na siatce kwadratowej. Gra zaczyna się na pustej planszy, na górze której pojawiają się przemieszczające się w dół czteroblokowe klocki, sterowane przez użytkownika. Utworzony wiersz znika, a klocki nad nim przesuwają się o jedno pole w dół. Gdy upuszczony klocek znajduje się w niewłaściwej pozycji (nachodzi na inny/pozycja jednego z jego elementów wychodzi poza planszę), gra kończy się. Gracz mam możliwość „podglądu” następnego klocka.

Sterowanie klockiem odbywa się przez klawisze strzałek w prawo i w lewo, powodujące przesuwanie się obiektu na osi x oraz klawisz strzałki w górę, powodujący rotację klocka w prawo. Można przyspieszyć spadanie klocka poprzez wciśnięcie klawisza strzałki w dół (przesuniecie o jedno pole w dół) lub klawisza Enter (klocek jest natychmiastowo spuszczany na pierwszą dozwoloną pozycję pod nim). Możliwe jest uzyskanie płynnego ruchu poprzez przytrzymanie klawiszy.

Pozostałe skróty klawiaturowe:

|  |  |
| --- | --- |
| Klawisz | Funkcja |
| R | Reset planszy |
| P/Spacja | Włączenie/Wyłączenie pauzę |
| H | Włączenie ekranu wyników |

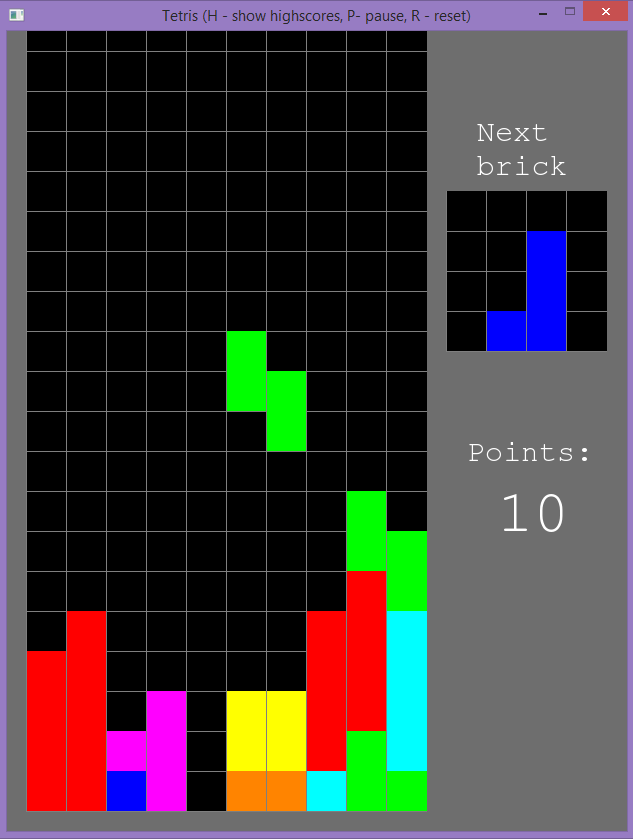
Stan pauzy oznacza wstrzymanie pętli gry i oczekiwanie na kontynuację gry. W tym czasie klocki nie przesuwają się w dół, a gracz nie może nimi manipulować. Pauza włącza się automatycznie, gdy okno gry przestanie być aktywne.

Rys. 1. Ekran przykładowej gry

Aktywny klocek

Następny klocek

Licznik punktów



Statyczne klocki

Lista wyników jest przechowywana w pliku binarnym. Na początku programu jest ona ładowana do listy jednokierunkowej, która następnie może być powiększana o kolejne elementy w miarę gry. Istnieje możliwość usunięcia listy dotychczasowych wyników. W czasie gdy wyświetlana jest lista wyników, istnieje możliwość przewijania jej poprzez wciskanie strzałek w górę i w dół. Domyślnie wyświetlane jest pięć rekordów.

W czasie uruchomienia programu ładowane są niektóre zasoby, lista wyników oraz konieczne do działania biblioteki. W razie nieuzyskania dostępu do któregokolwiek z nich, wyświetlany jest odpowiedni komunikat.

|  |  |  |
| --- | --- | --- |
| Komunikat | Źródło | Brakujący element w folderze programu |
| *Nie można uruchomić programu, ponieważ na komputerze nie znaleziono csfml*-(tu nazwa modułu)*-2.dll.* | Okno | Pliki biblioteki SFML  - csfml-graphics-2.dll  - csfml-system-2.dll  - csfml-window-2.dll |
| *Failed to load font "cour.ttf" (failed to create the font face)* | Konsola | Plik z czcionką  - cour.ttf |
| *Could not load scores list! (file: scores.dat)* | Konsola | Plik binarny z listą plików  - scores.dat |

1. Specyfikacja wewnętrzna

Projekt składa się z pliku main.c, define.h oraz sześciu dodatkowych plików źródłowych.

* main.c – główny plik, odpowiedzialny za wczytanie zasobów, uruchomienie gry oraz obsługę zdarzeń.
* define.h – plik nagłówkowy, będący bazą reszty plików. Znajdują się tu stałe i definicje procesora, spis kolorów oraz definicje wybranych struktur:

- fieldStruct

typedef struct{

sfVector2i pos;  
bool taken;  
sfRectangleShape \*rect;

}fieldStruct;

Struktura ta jest odpowiedzialna za reprezentowanie pojedynczego elementu planszy. Wektor dwuwymiarowy *pos* określa pozycję na planszy, wartość logiczna *taken* odpowiada za określanie dostępności komórki, a *\*rect* – jest to wskaźnik na kształt prostokąta z biblioteki SFML, używany do graficznej reprezentacji pola. Struktura ta jest również używana w wyświetlaniu kolejnego bloku.

- brickStruct

typedef struct{

char type;

sfVector2i poses[4];

sfRectangleShape \*elements[4];

sfVector2i pivot;

sfColor col;

}brickStruct;

Struktura ta jest odpowiedzialna za reprezentowanie opadającego bloku, złożonego z czterech bloków (*\*elements[4])*. Charakteryzują ją typ klocka (*type*), środek ciężkości, według którego będzie obracany (*pivot*), kolor klocka (*col*) oraz tablica pozycji jego poszczególnych elementów *poses[4]*, zapisanych jako dwuelementowe wektory liczb całkowitych.

* LinkedList.h oraz LinkedList.c – w nich zdefiniowana jest lista jednokierunkowa oraz podstawowe operacje wykonywane na niej.

typedef struct LinkedList{

int val;

struct LinkedList \*pNext;

}LinkedList;

Pole *val* jest wartością przechowywaną w elemencie, *\*pNext* – wskaźnikiem na kolejny element listy. Jeśli bieżący element jest elementem ostatnim, \**pNext* posiada wartość NULL.

1. Testowanie i wnioski

Testowanie programu składało się z dwóch części – pierwszym etapem była ocena *właściwiej obsługi sprawdzania obecności* plików oraz braku wycieków pamięci. Drugi etap polegał na testowaniu poprawności programu w działaniu, czyli w czasie gry sprawdzane było ogólne działanie mechaniki gry.

1. Źródła

* Oficjalna strona biblioteki SFML - <http://www.sfml-dev.org/>
* *Algorytm obrotu klocków -* [*http://gamedev.stackexchange.com/questions/17974/how-to-rotate-blocks-in-tetris*](http://gamedev.stackexchange.com/questions/17974/how-to-rotate-blocks-in-tetris)