Instrukcja projektowa; projekt numer 2

Podstawy Programowania 2015/16, kierunek Informatyka

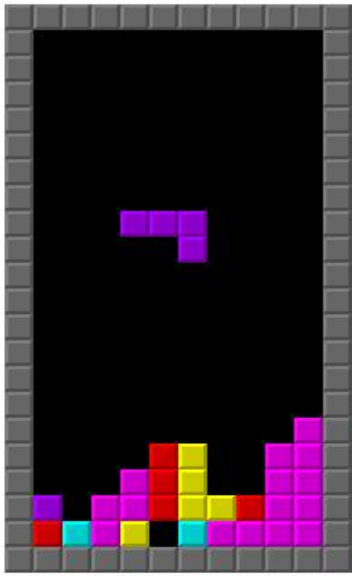
autor: Dariusz Dereniowski[1]
wersja z dnia: 26.10.2015 r.

Projekt TETRIS

Cel

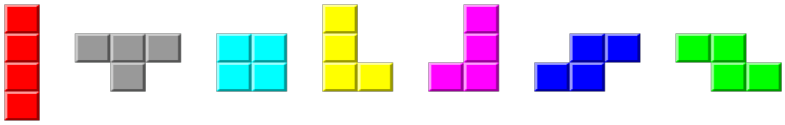
Celem projektu jest implementacja gry „Tetris”. Gra polega na kierowaniu ruchem (lewo, prawo, dół) opadających *klocków*. Każdy klocek składa się z czterech *kwadratów*. Opadanie klocka jest zakończone w momencie, gdy pod dowolnym z kwadratów składających się na ten klocek znajdzie się kwadrat należący do innego klocka. Zakończenie opadania danego klocka skutkuje dwoma zdarzeniami:

1) jeśli istnieje na planszy wiersz złożony w całości z kwadratów, to wiersz taki jest usuwany z planszy i wszystkie wiersze znajdujące się powyżej są przesuwane o jeden w dół;
2) pojawia się nowy (losowy) klocek u góry planszy, który rozpoczyna opadanie.



Ilustracja 1: (Źródło: https://simple.wikipedia.org/wiki/Tetris)

Dostępne są wszystkie rodzaje klocków (jest ich 7), które można utworzyć przy zachowaniu „spójności” z 4 kwadratów, mianowicie:



Ilustracja 2: (Źródło: https://pl.wikipedia.org/wiki/Tetris)

Szczegółowy opis gry można znaleźć na niniejszej stronie: https://pl.wikipedia.org/wiki/Tetris

1 Uwaga: W razie niejasności lub niejednoznaczności w poniższym opisie proszę kontaktować się z autorem instrukcji; pokój EA 209; konsultacje odbywają się w poniedziałki, w godz. 15:15-17:00

Środowisko programistyczne

Do instrukcji dołączony jest program startowy w którym zaimplementowano:

- obliczanie przyrostu czasu, co pozwala śledzić jego upływ
- wyświetlanie na ekranie plików graficznych w formacie BMP
- rysowanie piksela, linii, prostokąta
- wyświetlanie tekstu

Program działa w oparciu o bibliotekę SDL2 (2.0.3) – http://www.libsdl.org/. Jest ona dołączona do projektu startowego i nie trzeba pobierać jej źródeł.
Kompilacja pod systemem Linux wykonujemy za pomocą komendy (w systemie 32-bitowym):

```
g++ -O2 -I./sdl/include -L. -o main main.cpp -lm -lSDL2 -lpthread -ldl -lrt
```

oraz (w systemie 64-bitowym)

```
g++ -O2 -I./sdl/include -L. -o main main.cpp -lm -lSDL2-64 -lpthread -ldl -lrt
```

W celu pomyślnej kompilacji projektu startowego, w katalogu, w którym znajduje się plik main.cpp powinny znajdować się

- Bitmapy z wymaganymi rysunkami (cs8x8.bmp, eti.bmp). Uwaga na wielkość liter w nazwach plików!
- Plik libSDL2.a (libSDL2-64.a przy kompilacji 64 bitowej).
- Katalog sdl dołączony do projektu.

Do projektu dołączone zostały skrypty, które mogą być użyte do kompilacji (comp w środowisku 32-bitowym oraz comp64 w środowisku 64-bitowym).

Prezentacja programu (zaliczenie tej części projektu) odbywać się będzie w wybranym przez studenta środowisku spośród dwóch poniższych opcji:

- w systemie Linux. Student jest zobowiązany sprawdzić przed przybyciem na zaliczenie czy program poprawnie się kompiluje i uruchamia pod dystrybucją dostępną w laboratorium,
- w systemie Windows, w środowisku MS Visual C++ w wersji zgodnej z tą dostępną w laboratorium.

Uruchomienie programu podczas zaliczenia jest warunkiem koniecznym uzyskania punktów z projektu nr 2.

W programie nie należy używać biblioteki C++ **stl**.

Wymagania obowiązkowe (5 pkt.)

Wszystkie wymienione tutaj elementy należy zaimplementować. Brak któregokolwiek z poniższych elementów skutkuje otrzymaniem 0 pkt. z tego projektu.

(a) Przygotowanie graficznej oprawy gry: obrys planszy, rysowanie poszczególnych klocków (elementy te można rysować za pomocą figur geometrycznych lub przygotować pliki graficzne, które są odczytywane podczas uruchomienia programu). Klocki o różnych kształtach powinny mieć różne kolory. Implementacja klawiszy sterujących:

- *Esc*: wyjście z programu – program jest natychmiast kończony,
- *n*: nowa rozgrywka.

(b) Implementacja mechaniki opadania klocka oraz jego sterowanie w poziomie. Uwaga: w ramach wymagań obowiązkowych wystarczające jest aby opadanie klocków było „skokowe”, mianowicie co ustaloną jednostkę czasu (zakoduj tą jednostkę czasu jako stałą w programie, aby można było łatwo ten parametr zmodyfikować!) klocek jest przesuwany w dół o wartość równą szerokości boku kwadratu, z którego zbudowane są klocki. Sterowanie klocka w ramach niniejszego punktu powinno obejmować jego przesuwanie w lewo (*klawisz kierunkowy w lewo*) oraz prawo (*klawisz kierunkowy w prawo*), z uwzględnieniem kolizji z brzegami planszy.

(c) Implementacja kolizji klocka z innymi klockami znajdującymi się na planszy. Opadanie klocka powinno zostać zakończone gdy jeden z tworzących go kwadratów znajduje się bezpośrednio nad kwadratem innego klocka lub bezpośrednio nad dolnym brzegiem planszy. Także, przesuwanie klocka w bok jest możliwe tylko wówczas gdy nie występuje kolizja nie tylko z brzegiem planszy ale również z kwadratami innych klocków. W ramach tej funkcjonalności należy również zaimplementować rozpoznawanie końca gry, które następuje wówczas, gdy po zakończeniu opadania klocka w najwyższym wierszu planszy znajduje się kwadrat. Po zakończeniu gry program daje możliwość ponownej rozgrywki lub zakończenia programu.

Wymagania nieobowiązkowe (10 pkt.)

(e) (**1 pkt.**) Implementacja rotacji klocków (klawisz *spacja* oraz *klawisz kierunkowy góra* – oba te klawisze pełnią tą samą funkcję i oba powinny być uwzględnione w implementacji).

(f) (**1 pkt.**) Usuwanie pełnych wierszy. Program powinien w momencie zakończenia opadania klocka usunąć każdy wiersz wypełniony kwadratami. Usunięcie każdego wiersza powinno powodować przesunięcie o jeden wiersz ku dołowi wszystkich kwadratów znajdujących się nad usuwanym wierszem. W celu realizacji tego punktu należy zaimplementować (e).

(g) (**1 pkt.**) Implementacja etapowości gry. W implementacji gry należy określić pewien odcinek czasu *T*, po którym następuje zwiększenie prędkości opadania klocków. Dokładniej, przyspieszenie opadania następuje w momencie, gdy upłynęło *T* jednostek czasu od początku gry (pierwsze przyspieszenie) lub od poprzedniego przyspieszenia (każde kolejne przyspieszenie). Zdarzenie przyspieszenia tempa gry powinno nastąpić maksymalnie określoną liczbę razy (na przykład 10). Dodatkowo, wciśnięcie klawisza *s* powinno natychmiastowo podnosić prędkość opadania klocków „o jeden poziom” (nawet jeśli nie upłynął odcinek czasu *T*). Użycie klawisza *s* powinno zerować licznik czasu aktywacji kolejnego przyspieszenia. (Wymienione parametry powinny być stałymi, aby można było łatwo dokonać zmian ich wartości!) W celu realizacji tego punktu należy zaimplementować (e) i (f). Uwaga: jeśli w danym momencie rozgrywki wykonano do tej pory *x* zdarzeń przyspieszenia tempa gry, to mówimy, że gra znajduje się w *etapie x*. (Odwołujemy się do tego pojęcia w punkcie (i).

(h) (**1 pkt.**) Natychmiastowe opadanie klocka. Naciśnięcie *klawisza kierunkowego w dół*

skutkuje natychmiastowym przebyciem całej drogi klocka ku dołowi planszy (aż do wykrycia kolizji z kwadratami innych klocków lub dolnym brzegiem planszy). W celu realizacji tego punktu należy zaimplementować (e) i (f).

(i) (**1 pkt.**) Liczenie punktów. W momencie zakończenia opadania klocka gracz otrzymuje następującą liczbę punktów (liczba *x* oznacza number bieżącego etapu gry):
- 1200(*x* + 1) punktów – jeśli jednocześnie usuwane są po raz drugi z rzędu 4 wiersze z planszy,
- 800(*x* + 1) punktów – jeśli jednocześnie usuwane są 4 wiersze z planszy, lecz poprzednie zdarzenie usunięcia wierszy eliminowało mniej niż 4 wiersze,
- 400(*x* + 1) punktów – jeśli jednocześnie usuwane są 3 wiersze z planszy,
- 200(*x* + 1) punktów – jeśli jednocześnie usuwane są 2 wiersze z planszy,
- 100(*x* + 1) punktów – przy usuwaniu jednego wiersza z planszy.

Aktualna liczba punktów jest na bieżąco wyświetlana na ekranie. W celu realizacji tego punktu należy zaimplementować (e) i (f).

(j) (**1 pkt.**) Zapamiętywanie w pliku listy najlepszych wyników. Format zapisu do pliku należy wybrać samodzielnie. W momencie zakończenia gry program powinien poprosić użytkownika o podanie imienia, gdy uzyskany wynik kwalifikuje gracza do dopisania na listę najlepszych wyników. Program powinien także wyświetlać listę najlepszych wyników (jeśli nastąpiło dopisanie rozgrywki do listy, to wyświetlana jest uaktualniona wersja). Student prezentując program powinien posiadać przykładowy gotowy plik. Liczba pozycji na liście najlepszych wyników powinna być możliwa do łatwej modyfikacji w kodzie programu: należy zadeklarować stałą, która to określa. W trakcie trwania rozgrywki, jeśli bieżący wynik gracza kwalifikuje go na umieszczenie na liście najlepszych wyników, to program powinien wyświetlić taką informację na ekranie, wskazując które miejsce na liście najlepszych wyników zostało osiągnięte w danym momencie. W celu realizacji tego punktu należy zaimplementować (e), (f) i (i).

(k) (**1 pkt.**) Zapisanie stanu gry. Z tej opcji gracz może skorzystać co najwyżej trzy razy podczas każdej rozgrywki. Naciśnięcie przycisku *s* w trakcie gry powoduje, że stan gry zostaje zapisany do pliku bez przerywania rozgrywki, o ile skorzystano z tej funkcjonalności co najwyżej dwa w bieżącej rozgrywce (jeśli uprzednio zapamiętany był inny stan gry, to zostaje on utracony). Stan gry obejmuje:
- położenie wszystkich kwadratów znajdujących się na planszy (wraz z ich kolorami, a także położenie opadającego w danym momencie klocka),
- bieżącą liczbę punktów, jeśli zaimplementowano (i),
- bieżącą prędkość opadania klocka, jeśli zaimplementowano (g).

Naciśnięcie klawisza *l* w dowolnym momencie (w trakcie trwania rozgrywki lub po jej zakończeniu) powoduje odtworzenie ostatnio zapamiętanego stanu gry.

(l) (**2 pkt.**) Płynny ruch klocków. Opadanie klocków powinno być płynne, tzn. ich położenie powinno się zmieniać (zarówno na boki jak i w dół) o jeden piksel. Dokładniej, dotychczasowy ruch skokowy przesunięcia klocka o długość boku kwadratu (oznaczmy tą długość przez *x*) co *t* jednostek czasu powinien być zastąpiony *x* przesunięciami klocka, każde co *t*/*x* jednostek czasu. Uwaga: nie ma możliwości zatrzymania klocka na współrzędnych nie będących wielokrotnościami boku kwadratu. Jeśli zaimplementowano tą funkcjonalność, to implementacja „skokowego” ruchu klocków nie jest konieczna w celu

przyznania pięciu punktów za część obowiązkową projektu.

(m) (**0.5 pkt.**) Pauza. Wciśnięcie klawisza *p* powoduje zatrzymanie gry i wyświetlenie na ekranie napisu „Pauza”. Kolejne wciśnięcie klawisza *p* wznawia grę.

(n) (**0.5 pkt.**) Płynne natychmiastowe opadania klocka. W celu realizacji tego punktu należy zaimplementować (e), (f), (h) i (l). W wyniku naciśnięcia *klawisza kierunkowego w dół* klocek powinien *bardzo szybko*, ale płynnie, opaść ku dołowi zgodnie z regułami opisanymi w (h). Powyższa prędkość powinna być określona na stałe w programie.

Wymagania „z gwiazdką” (3 pkt.)

Tą część można zaimplementować tylko wówczas, gdy zaimplementowano wszystkie wcześniejsze elementy.

Płynna rotacja klocków. Rotacje klocków powinny odbywać się płynnie zgodnie z ustaloną prędkością i wokół „naturalnie” dobranej „osi obrotu”. Prędkość dokonywania obrotu powinna być parametrem łatwo modyfikowalnym w kodzie programu. Ostatnie w szczególności oznacza, że poszczególne obroty klocków nie powinny być uprzednio przygotowane w postaci gotowych plików graficznych, tylko ich obroty o określony kąt (zależny od prędkości obrotu) powinny być wyliczane podczas działania programu.

Uwagi końcowe

- Wymagania dotyczące szaty graficznej: wystarczające jest użycie dowolnych bitmap (dobranych właściwie pod względem rozmiaru).
- Konfiguracja programu powinna umożliwiać łatwą zmianę wszelkich parametrów, nie tylko tych wyraźnie wskazanych w powyższym opisie. Przez łatwą zmianę rozumiemy modyfikację stałej w programie.
- Szybkość działania programu powinna być niezależna od komputera, na którym uruchomiono program. Stałe jednostki w programie powinny być opisane odpowiednimi komentarzami, na przykład:

```
const double PREDKOSC_KLOCKA = 40.0;  // piksele na sekundę
const int SZEROKOSC_PLANSZY = 320;    // piksele
const double POZYCJA_X_NAPISU = 60.0; // procent szerokości ekranu
const double POZYCJA_Y_NAPISU = 5.0;  // procent wysokości ekranu
```