Instrukcja do projektu 2

Podstawy Programowania 2025/26, Informatyka

author: Robert Ostrowski[1]
data: 26/11/2025

Beat This Project

Cel projektu

Celem projektu jest zaimplementowanie gry typu beat'em up dla jednego gracza z bocznym przewijaniem ekranu (Wikipedia). Przykłady gier tego gatunku można znaleźć na wspomnianej stronie. Postać przemierza planszę i pokonuje przeciwników, którzy utrudniają ukończenie projektu i kursu. Poniżej wymieniono wybrane funkcjonalności/elementy gry, które należy zaimplementować.

Możliwa jest również zmiana motywu gry na mniej brutalny (ataki mogą zamiast tego reprezentować akcje zgodne z opisem mechaniki gry). Niniejszym linkowana seria artykułów na temat pokrewnego gatunku bijatyk zawiera dodatkowe informacje, które mogą okazać się pomocne.

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

[1] Uwaga: W razie niejasności lub niejednoznaczności w poniższym opisie proszę kontaktować się z autorem instrukcji pod adresem robert.ostrowski@pg.edu.pl; dalsze informacje kontaktowe znajdują się na stronie enauczania.

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.

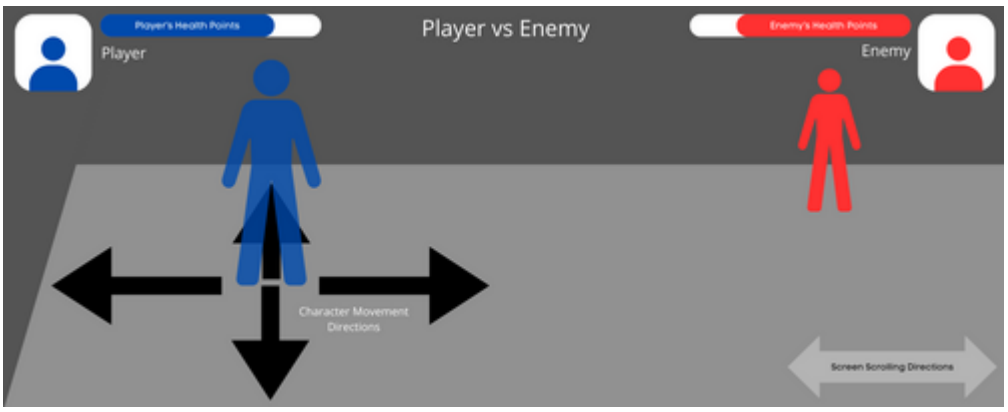
Wymagania obowiązkowe (5 punktów)

Wszystkie wymienione tutaj elementy muszą zostać zaimplementowane. Wersja wymagań podstawowych jest bardzo uproszczona; w szczególności może służyć jako poligon do testowania dalszych wymagań. Brak któregokolwiek z poniższych elementów skutkuje przyznaniem 0 punktów za projekt.

1. Przygotowanie oprawy graficznej gry: zarys sceny, przygotowanie miejsca do wyświetlania dodatkowych informacji (np. czasu, jaki upłynął od rozpoczęcia etapu, akcji podejmowanych przez gracza). Implementacja klawiszy sterujących:
   a. ***Esc***: wyjście z programu – program kończy się natychmiast;
   b. ***'n'***: nowa gra.

2. Implementacja jednego etapu gry. Scena powinna mieć podłogę o ograniczonym rozmiarze i tło. Rozmiar sceny nie powinien być ograniczony rozmiarem okna, tzn. kamera powinna podążać za postacią (należy pozostawić wystarczająco dużo miejsca, aby zachować rozsądne pole widzenia).

3. Implementacja mechaniki poruszania postacią za pomocą strzałek lub klawiszy WSAD. Pozycja postaci powinna być ograniczona przez podstawową rozdzielczość gry, a nie przez pozycję bloków, z których zbudowana jest scena.



4. Na bieżąco należy mierzyć i wyświetlać czas który upłynął od uruchomienia gry. Należy wyświetlać również informację o zaimplementowanych wymaganiach (obowiązkowych i opcjonalnych).

Wymagania nieobowiązkowe (10 pkt.)

A. **(1 pkt)** Implementacja skoków i ataków:
   a. Po naciśnięciu odpowiedniego przycisku postać powinna wykonać akcję, co najmniej:
      i. żwawą (light) akcję lub atak (*akcja X*) – szybsza
      ii. silną (heavy) akcję lub atak (*akcja Y*) – wolniejsza
      iii. skok
   b. Wysokość skoku powinna być ograniczona. Powinna istnieć możliwość przeskakiwania nad przeciwnikami (jeśli punkt E jest zaimplementowany). Parametry ruchu i sterowania postacią powinny być łatwe do zmiany, aby zapewnić płynne sterowanie.
   c. Jeśli animacje (punkt D) nie są zaimplementowane, wystarczy zastąpić sprite'a postaci lub zmienić jego kolor na określoną liczbę klatek podczas akcji A i B.

B. **(1pkt)** Combo input
   a. Zaimplementuj bufor wejściowy dla poleceń (przykład pomysłu można znaleźć tutaj)
   b. Co najmniej 3 dodatkowe akcje powinny być dostępne poprzez połączenie podstawowych danych wejściowych w następujący sposób:
      i. naciśnięcie przycisku akcji X lub Y trzy razy z rzędu
      ii. określona sekwencja przycisków akcji X i Y
      iii. naciśnięcie przycisku akcji X lub Y w powietrzu
   c. Gra nie musi zapewniać możliwości dodawania kombinacji przez użytkownika, kod programu powinien umożliwiać jednak proste dodawania kolejnych kombinacji przez programistę.
   d. Naciśnięcie klawisza trybu deweloperskiego powinno wyświetlić bufor, jego zawartość i nazwę aktualnie wykonywanej akcji.

C. **(1 pkt)** Hitboxy:
   a. Zaimplementuj obiekt gry, z którym można wchodzić w interakcję (zastąpiony przez wrogów w punkcie E).
   b. Akcja X powinna mieć inny obszar trafienia i prędkość niż akcja Y. Wszystkie akcje combo z punktu B powinny skutkować akcją o innych cechach.
   c. Trafienie obiektu bądź przeciwnika powinno przyznawać określoną liczbę punktów, a każde kolejne trafienie (bez bycia trafionym) w określonym przedziale czasowym zwiększa mnożnik przyznawanych punktów.
D. **(2 pkt.)** Animacje (wymaga punktów A i C opisanych powyżej):
   a. Animacje chodzenia, bycia uderzonym oraz akcji postaci i wrogów. Liczbę klatek ograniczyć można do minimum, jeśli zachowana będzie czytelność.
   b. Aktualny mnożnik punktów powinien pojawiać się i rosnąć na ekranie w rzucający się w oczy sposób.

   c. Uwaga: szybkość animacji nie powinna zależeć od szybkości komputera (zakładając, że spełnia on minimalne wymagania gry). Naiwne rozwiązanie z punktu A.c jest niewystarczające!
E. **(1 pkt)** Przeciwnicy
   a. W każdym etapie pojawia się ograniczona liczba przeciwników co najmniej 2 typów.
   b. Jeden typ przeciwnika postępuje zgodnie z prostym schematem zachowania: zbliża się do postaci gracza i próbuje ją uderzyć. Drugi typ powinien zachować dystans, ustawić się w jednej linii z graczem i wykonać szarżę.
   c. Trafienie przeciwnika powinno spowodować krótkotrwałe przerwanie jego akcji (oszołomienie, stun).
F. **(2 pkt.)** Konfiguracja etapu w pliku:
   a. W pliku należy zawrzeć następujące elementy konfiguracji etapu:
      i. szerokość i pozycja przeszkód: etap powinien mieć różną szerokość w różnych punktach(szerokość obszaru, po którym porusza się postać gracza nie powinna być stała) i zawierać dodatkowe przeszkody utrudniające ruch gracza.

ii. rozmieszczenie i liczba przeciwników (jeśli punkt E nie jest zaimplementowany, należy zamiast tego rozmieścić nieruchome cele-atrapy).

b. Po pokonaniu wrogów program powinien wykryć wejście na dany obszar, zakończyć etap i zasygnalizować to zdarzenie.
c. **Uwaga**: program nie powinien nakładać limitów na maksymalną liczbę obiektów różnego typu znajdujących się w pliku. Oznacza to, że program analizuje zawartość pliku i następnie przydziela pamięć wystarczającą na przechowanie danych o wszystkich obiektach znajdujących się w pliku. Format kodowania tych informacji w pliku należy dobrać samodzielnie, co oznacza, że dla pewnego ułatwienia wczytywania opisu etapu z pliku można zdecydować się na taki format, w którym na początku pliku znajdują się informacje (preambuła) o liczbie poszczególnych obiektów, a następnie znajduje się sam opis planszy. W ten sposób można wczytać preambułę, zaalokować pamięć na podstawie znajdujących się tam danych a następnie wczytać etap. (Przy takim rozwiązaniu należy dodać sprawdzenie poprawności danych – program powinien rozpoznać niezgodność preambuły z zawartością)

G. **(1 pkt)** Menu i interfejs:
   a. Po uruchomieniu gry powinno pojawić się menu umożliwiające graczowi wybór wszystkich opcji: wyjście, sprawdzenie wyników (punkt H) oraz wybór etapu (punkt F).
   b. Wybranie opcji, która nie jest zaimplementowana, powinno wyświetlić komunikat informujący o jej niedostępności.
   c. Tekst wprowadzany przez gracza (np. podczas pisania pseudonimu) powinien być zawsze widoczny, a klawisz ***backspace*** powinien umożliwiać usuwanie liter.
   d. Wyświetlanie na ekranie paska punktów życia postaci gracza w formie graficznej. Po utracie wszystkich punktów życia wyświetl zapytanie o kontynuowania gry lub powrotu do menu, oraz wyświetl (jeśli punkt F jest zaimplementowany) liczbę zdobytych punktów.
   e. Po dotarciu do końca etapu należy przejść do następnego (wcześniej należy przygotować prosty sposób zademonstrowania tej funkcji).

H. **(1 pkt)** Zapisywanie najlepszych wyników:
   a. Po zakończeniu gry gracz powinien mieć możliwość wprowadzenia pseudonimu i wyniku do pliku.
   b. Liczba wyników zapisanych w pliku nie powinna być ograniczona.
   c. Gracz może przeglądać posortowaną listę wyników z menu lub po naciśnięciu określonego przycisku.
   d. Liczba wyników wyświetlanych na ekranie, w przeciwieństwie do tych zawartych w pliku, powinna być ograniczona. Jeśli nie wszystkie wyniki mieszczą się na ekranie, program powinien umożliwiać graczowi dotarcie do nich, np. poprzez przełączanie się między stronami lub przewijanie listy.
   e. Jeśli punkt C nie zostanie zaimplementowany, zamiast wyników można rejestrować czas.

Wymagania „z gwiazdką” (3 pkt.)

Uwaga: poniższe wymagania są oceniane dopiero po spełnieniu wszystkich pozostałych punktów.

I. **(1,5 pkt)** Ruch wymuszony (artykuł)
   a. Daj przeciwnikom dodatkowy, silny atak, który wyrzuci gracza w powietrze.
   b. Podobnie, użycie konkretnych ruchów lub zadanie wielu ciosów przeciwnikowi powinno odepchnąć go lub wyrzucić w powietrze po osiągnięciu określonego progu.
   c. Uderzenie w ścianę powinno skutkować odbiciem.
   d. Powinna istnieć możliwość żonglowania (utrzymywania w powietrzu) przeciwnikiem poprzez wielokrotne uderzanie.

J. **(1,5 pkt)** Kodowanie sterowania, akcji i kombinacji w grze w pliku:
   a. Zaprojektuj własny (edytowalny, np. w edytorze tekstu) format pliku opisu dostępnych akcji i sterowania. Format ten powinien być Ci znany, abyś mógł wyjaśnić i wprowadzić wskazane zmiany podczas prezentacji.
   b. Plik powinien określać: sposób wykonania danego ruchu, pola trafień i wartości obrażeń. Ponadto powinna istnieć możliwość udzielenia dostępu do nich określonym typom przeciwników.

Uwagi końcowe

Konfiguracja programu powinna umożliwiać łatwą zmianę wszelkich parametrów, nie tylko tych wyraźnie wskazanych w powyższym opisie. Przez łatwą zmianę rozumiemy modyfikację stałej w programie.

Projekt może być napisany w sposób obiektowy, ale całkowicie zabronione jest używanie następujących elementów biblioteki standardowej C++: typu string, cin, cout, vector i pozostałych kontenerów. (Uwaga: typu string z biblioteki C++ nie należy mylić w biblioteką string.h z C – można używać funkcje znajdujące się w string.h)

Obsługa plików powinna być zrealizowana przy użyciu biblioteki standardowej C (rodzina funkcji f???? - np. fopen, fread, fclose itd.)

Każdy fragment przedstawionego do oceny kodu powinien być napisany samodzielnie przez studenta.

Szybkość działania programu powinna być niezależna od komputera, na którym uruchomiono program.

Jakość kodu będzie oceniana zgodnie z ogólnymi wymaganiami dostępnymi

na stronie kursu.