Instrukcja projektowa; projekt numer 2

Podstawy Programowania 2020/21, kierunek Informatyka

autor: Robert Ostrowski[1]
wersja z dnia: 15.11.2020 r.

Projekt Robot Unicorn Attack

Cel

Celem projektu jest implementacja gry „Robot Unicorn Attack” i spełnienie swoich życzeń (o zaliczeniu części projektowej). Gra polega na sterowaniu skokami i zrywami jednorożca wiecznie galopującego w jedną stronę i zdobyciu jak największej liczby punktów poprzez utrzymanie go przy życiu jak najdłużej . Wybrane funkcjonalności/elementy gry, które należy zaimplementować podane są poniżej.

Uproszczeniem w stosunku do oryginalnej gry jest ograniczenie wyglądu etapu do prostokątnych elementów i poziomych powierzchni, po których porusza się jednorożec (chyba, że implementowane są wymagania z gwiazdką).

Opisy gry można znaleźć na niniejszych stronach:
https://en.wikipedia.org/wiki/Robot_Unicorn_Attack
https://gamefaqs.gamespot.com/flash/996571-robot-unicorn-attack/faqs/59954
Sama gra dostępna jest tutaj:
https://unicorn.jocke.no/ (uwaga flash!)

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
```

1 Uwaga: W razie niejasności lub niejednoznaczności w poniższym opisie proszę kontaktować się z autorem instrukcji; konsultacje odbywają się w środy 18-20, link znajduje się na stronie enauczania.

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. Wersja do wymagań podstawowych jest bardzo uproszczona, w szczególności sterowanie można wykorzystać do testowania kolejnych wymagań. Brak któregokolwiek z poniższych elementów skutkuje otrzymaniem 0 pkt. z tego projektu.

1. Przygotowanie graficznej oprawy gry: obrys planszy, przygotowanie miejsca na wyświetlanie dodatkowych informacji: czasu, który minął od początku etapu.
   Implementacja klawiszy sterujących:
   a. *Esc*: wyjście z programu – program jest natychmiast kończony,
   b. *n*: nowa rozgrywka.

2. Implementacja mechaniki poruszania jednorożcem przy pomocy klawiszy kierunkowych. Pozycja jednorożca powinna być ograniczona rozdzielczością gry, a nie pozycją bloków z których zbudowana jest scena. Nie jest wymagana ‘fizyka’ skoków. Pozycja jednorożca na ekranie jest zawsze przy jego lewej krawędzi.

3. Implementacja jednego etapu gry. Scena tego etapu gry powinna posiadać przeszkody, które jednorożec musi omijać (należy zaimplementować wykrywanie kolizji jednorożca z takimi przeszkodami). Scena powinna być przynajmniej kilkukrotnie szersza niż szerokość okna oraz należy zaimplementować możliwość przemierzania sceny, poprawnie wyświetlając bieżący (widoczny dla gracza) jej fragment na ekranie monitora. Scena zawija się poziomo, tworząc nieskończoną pętlę.

4. Nie ma konieczności implementacji efektu osiągania końca etapu (porażka), ale należy poprawnie mierzyć czas przeznaczony na rozgrywkę w tym etapie. Scena na

całej swojej szerokości posiada podłoże (brak możliwości wypadnięcia jednorożca poza scenę), a wszystkie przeszkody mają formę prostokątów „leżących” na wspomnianym podłożu (brak „platform” zawieszonych nad podłożem).

Wymagania nieobowiązkowe (10 pkt.)

1. **(1 pkt)** Implementacja domyślnego sterowania jednorożcem, zastępującego sterowanie bazowe z podpunktu 2 wymagań obowiązkowych:
   a. Przełączenie trybu sterowania powinno nastąpić po naciśnięciu przycisku **‘d’**.
   b. Jednorożec porusza się w prawo automatycznie, nabierając prędkości z czasem.
   c. **‘z’** - skok. Podczas skoku można wykonać zryw lub kolejny skok (ang. *double jump*). Przytrzymanie klawisza zwiększa wysokość i zasięg skoku do pewnej ograniczonej wartości. Jednorożec odzyskuje pełnię sprawności do wykonywania kolejnych skoków po dotknięciu ziemi.
   d. **‘x’** - zryw (ang. *dash*). Jednorożec może wykonać szybki ruch poziomo, chwilowo przerywając utratę bądź nabieranie wysokości. Podczas zrywu jednorożec nie może wykonać innej akcji, jednak zakończenie wykonania zrywu pozwala jednorożcowi na dodatkowy skok, nawet jeśli wyczerpał już zdolność podwójnego skoku.
   e. Parametry ruchu jednorożca i sterowania powinny być łatwe do zmiany aby osiągnąć płynne sterowanie.

2. **(2 pkt)** Zaawansowany wygląd etapu gry (poziome platformy):

   a. Etap składa się z wiszących platform, na które jednorożec może wskoczyć – w tym celu należy poprawnie zaimplementować wykrywanie kolizji (jednorożec stojący na platformie z niej nie spada), i po których powierzchni porusza się jednorożec.
   b. Etap jest jest nie tylko dłuższy, ale i wyższy niż szerokość okna. Dopóki jednorożec nie zbliża się do górnej lub dolnej krawędzi planszy to pozostaje na środku wysokości ekranu. Plansza demonstracyjna powinna zawierać co najmniej kilka platform na różnych wysokościach, w tym przynajmniej 2 platformy z których jedna znajduje się nad fragmentem drugiej.
   c. Przeszkody z podpunktu 4 wymagań obowiązkowych mogą znajdować się na platformach.
   d. **Uwaga**: **uproszczeniem w stosunku do oryginalnej gry!** Jeśli nie są implementowane wymagania “z gwiazdką” plansza może składać się z prostokątnych bloków, a wszystkie powierzchnie do galopu mogą być płaskie i poziome.

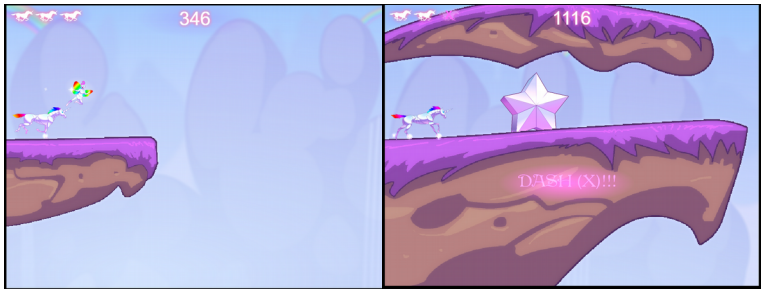
3. **(1 pkt)** Przeszkody i kolizje
    a. Dodanie przeszkód na platformach, w formie terenu o wyraźnie większej wysokości.
    b. Dodanie przeszkód-stalaktytów nie mających styku z podłożem. Należy poprawnie zaimplementować możliwość kolizji jednorożca z takimi obiektami uniemożliwiając jednorożcowi przejście przez taki obiekt.
    c. **Uwaga: uproszczenie w stosunku do oryginalnej gry!** Kształt przeszkód można przybliżyć korzystając z prostokątnych bloków budujących planszę.

4. **(1 pkt)** Śmierć, liczba żyć i menu:
    a. Wyświetlanie na ekranie pozostałej liczby żyć jednorożca w formie graficznej.
    b. Utrata życia po czołowym zderzeniu z terenem.
    c. Upadek z platformy i wypadnięcie poza dolną krawędź ekranu, etap nie posiada podłoża.
    d. Kontakt z gwiazdą bez zrywu (jeśli dojdzie do kontaktu z gwiazdą podczas zrywu jednorożec przez nią przechodzi, patrz punkt 6).
    e. Utrata życia powinna wyświetlić zapytanie o kontynuację i (jeśli zaimplementowano punkt 6) liczbę zdobytych punktów.
    f. Utrata wszystkich żyć powinna spowodować wyświetlenie menu głównego lub (jeśli zaimplementowano punkt 8) wcześniej zapytanie o zapis wyniku.

5. **(2 pkt.)** Animacje (niektóre podpunkty wymagają implementacji innych wymagań):
    a. Animacje biegu, skoku i opadania jednorożca.
    b. Animacje eksplozji gwiazd przez które jednorożec przeszedł podczas zrywu i zniszczenia jednorożca w przypadku kolizji.
    c. Zdobyte punkty bonusowe powinny być na chwilę wyświetlane na ekranie.
    d. Wyskakujące z dolnej krawędzi ekranu delfiny - coraz więcej delfinów przybywa podziwiać postępy gracza po osiągnięciu kolejnych kamieni milowych.
    e. **Uwaga:** szybkość animacji nie powinna być zależna od szybkości

komputera (przy założeniu, że spełnia on minimalne wymagania gry)!

6. **(1 pkt)** Liczenie punktów:
   a. Stale zwiększana liczba punktów na podstawie dystansu przebytego przez jednorożca.
   b. Za rozbijanie gwiazd. Gwiazdy to przeszkody, które można rozbić wchodząc w kolizję z nimi podczas zrywu. 100 punktów za pierwszą gwiazdę i o 100 więcej za każdą kolejną. Pominięcie gwiazdy powoduje reset wartości kolejnej do początkowych 100 punktów.
   c. Za zbieranie wróżek. Wróżki to postaci poruszające się losowo w zadanym obszarze, kolizja z wróżką powoduje jej ‘zebranie’. 10 punktów za pierwszą wróżkę i o 10 więcej za każdą kolejną. Pominięcie wróżki powoduje reset wartości kolejnej do początkowych 10 punktów.



   d. Wróżki i gwiazdy pojawiają się z pewnym prawdopodobieństwem w miejscach określonych przez parametry etapu. Gdy etap zawinie się pozycje bonusów nie powinny być zawsze takie same.
   e. Suma zdobytych punktów powinna być wyświetlane na ekranie.

7. **(1 pkt)** Zapamiętywanie najlepszych wyników:
   a. Po zakończeniu gry powinno być możliwe wpisanie swojego pseudonimu i wyniku do pliku
   b. Z poziomu menu można wyświetlić posortowaną listę wyników.
   c. Liczba wyników na ekranie powinna być ograniczona.
   d. Jeśli nie wszystkie wyniki mieszczą się na ekranie program powinien umożliwić dotarcie do nich np. poprzez przełączanie się między

stronami lub przewijanie listy.

8. **(1 pkt)** Kodowanie wyglądu etapu gry w pliku.
    a. Należy zaprojektować własny (edytowalny, np. w edytorze tekstowym) format pliku etapu gry. Format taki powinien być znany studentowi, tak aby był w stanie wytłumaczyć oraz dokonać wskazanych edycji etapu podczas odpowiedzi. Podczas startu danego etapu, powinien nastąpić odczyt wyglądu tego etapu z pliku. Plik powinien zawierać przynajmniej informacje o położeniu wszystkich przeszkód znajdujących się w tym etapie oraz wymiarach etapu.
    b. Dodatkowo w pliku powinny znajdować się informacje o wszystkich elementach o które rozszerzono implementację podczas realizacji wymagań nieobowiązkowych.
    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 – czy liczba obiektów w pliku nie przekracza tej podanej w preambule.)

Wymagania „z gwiazdką” (3 pkt.)

Uwaga: poniższe wymagania rozszerzają podpunkty: wygląd etapu gry, kolizje, kodowanie wyglądu etapu gry i są oceniane wyłącznie gdy zaimplementowane zostały wszystkie pozostałe punkty.

1. **(1.5 pkt)** Jeszcze bardziej zaawansowany wygląd etapu gry (nieregularne platformy)
    a. Elementy z których składa się etap gry mogą mieć nieregularne kształty, niekoniecznie ograniczone poprzez dostępne bitmapy. W szczególności powierzchnia platformy po której porusza się jednorożec może być określona poprzez funkcję. Do znalezienia odpowiednio wyglądających funkcji można posłużyć się na przykład https://mycurvefit.com/ i przyjąć w programie reprezentację wielomianową funkcji.
    b. Jeśli podczas zrywu jednorożec napotyka powierzchnię po której może się poruszać zakrzywioną lekko w górę, wtedy kontynuuje ruch po tej powierzchni zamiast prosto poziomo. Zryw z powierzchni nachylonej w dół odrywa jednorożca od podłoża
    c. Wyświetlany na ekranie wielokąt reprezentujący platformę może być jedynie

przybliżeniem ‘rzeczywistej’ trajektorii galopującego po jego powierzchni jednorożca.

1. **(1.5 pkt)** Dynamiczne tworzenie etapu:
    a. Dostępne do użycia kształty platform powinny być zapisane są w pliku określającym wygląd gry.
    b. Gdy przewidziany odcinek etapu kończy się przerwą między platformami pozycja kolejnej platformy powinna być wylosowana tak, aby jednorożec miał szansę skończyć skok na nowej platformie (przestrzeń losowania ograniczona jest możliwościami jednorożca, nie umiejętnościami gracza).

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.
- Projekt może być napisany w sposób obiektowy, ale całkowicie zabronione jest używanie biblioteki standardowej C++ (w tym typu string, cin, cout, vector itp.) (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. Stałe jednostki w programie powinny być opisane odpowiednimi komentarzami, na przykład:

```
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
```