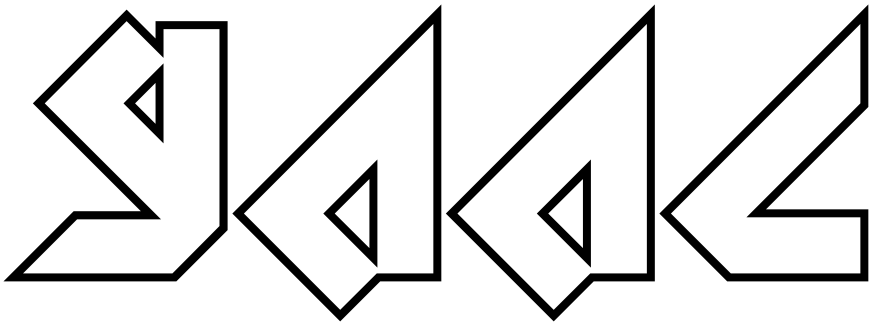
Design and Technical Document for game



Yet Another Arkanoid Clone

Maciej Czekański

IS2009 @ EAIE

AGH

Opis gry

Cel gry

Gra polega na zdobyciu jak największej ilości punktów poprzez odbijanie piłeczki paletką i burzenie ścian. Gracz ma jednak ograniczoną ilość żyć – za każdym razem, kiedy ostatnia kulka wyleci poza planszę, gracz traci jedno z nich. Brak żyć oznacza przegraną. Gracz, który uzyska dostateczną ilość punktów, zostanie zapisany na liście Najlepszych Wyników.

Rodzaje klocków:

Rodzaj	Ilość uderzeń potrzebnych do zburzenia	Ilość przydzielonych punktów
Zwykły	1	10
Kamienny	2	50
Stały	∞	0

Rodzaje bonusów:

- Przyspieszenie – przyspiesza jedną z kulek 1.5-krotnie
- Zwolnienie – zwalnia jedną z kulek 0.75-krotnie
- Rozdzielenie – rozdziela jedną z kulek na dwie osobne
- Wydłużenie – wydłuża paletkę
- Skrócenie – skraca paletkę
- Ognista kula – niszczy klocki nie odbijając się od nich

Sterowanie:

- Lewo/prawo – poruszanie paletką
- Góra/dół oraz Enter – poruszanie się po menu
- F1 – włączenie/wyłączenie licznika FPS
- F2 – włączenie/wyłączenie muzyki

Użyte oprogramowanie

Kompilator i edytor

Projekt jest skompilowany przy użyciu bardzo dobrego i darmowego kompilatora FPC (Free Pascal Compiler). Potrafi on generować szybki kod maszynowy, dostosowany do dzisiejszej architektury procesorów. Niestety, dołączony do niego edytor tekstu nie jest zbyt wygodny, więc zastąpiłem go przez Programmer's Notepad. Dzięki możliwości podpięcia własnych narzędzi pod dany skrót klawiszowy w tym programie miałem możliwość szybkiej i wygodnej kompilacji całego projektu.

Edytor graficzny

Większość grafik użytych w projekcie to zasługa darmowego programu Paint.NET. Pomimo, że nie jest on tak rozbudowany jak np. GIMP, to cenię go za wysoką wygodę użytkowania.

Edytor dźwiękowy

Niektóre dźwięki trzeba było poddać lekkiej obróbce, inne trzeba było przekonwertować z formatu mp3 na wav. Posłużył mi do tego darmowy program Audacity.

Inne

- **ttf2pcx** – program konwertujący czcionki w formacie TTF na format PCX obsługiwany przez Allegro

Użyte biblioteki zewnętrzne

W grze zostały użyte następujące biblioteki:

- **Windows** – funkcje związane z tworzeniem i obsługą okna. Biblioteka dostarczona jest razem z FPC.
- **SysUtils** – funkcje ogólnego zastosowania oraz do operacji na tekście. Również dostarczana z FPC.
- **Allegro.pas** (oraz pochodne) – biblioteka pozwalająca na wyświetlanie grafiki dwuwymiarowej, odtwarzanie dźwięków oraz obsługę klawiatury i myszy. Została ona wybrana ze względu na prosty interfejs i przystępną dokumentację.
  Strona internetowa projektu: http://allegro-pas.sourceforge.net/

Zasoby:

- Grafika – w większości własne; animacje oraz tło pochodzą z paczki 110MB spritów znalezionej na forum strony www.gamedev.net
- Dźwięki oraz muzyka – pobrane z serwisu www.soundsnap.com
- Czcionka Arkanoid – pobrana ze strony www.fontspace.com a następnie przerobiona z formatu TTF na format PCX

Napotkane problemy

Część z napotkanych przeze mnie problemów dotyczyła przeciętnej znajomości języka Pascal. Z pomocą przyszła jednak obszerna dokumentacja kompilatora FPC, do której często zaglądałem.

Inny problem wynikał z użycia biblioteki Allegro.pas w wersji beta. Program crashował się w pewnym bez słowa komunikatu, plik z logiem również się nie

tworzył. Po dogłębnej analizie okazało się, że źródłem problemów była źle zaimportowana funkcja al_getb. Co ciekawe, bliźniacze wersje tej funkcji(al_getr oraz al_getg) działały bez zarzutu. Aby pozbyć się tego problemu konieczna była edycja kodu źródłowego biblioteki i jej rekompilacja. W dalszej części realizacji projektu ponownie natrafiłem na podobny problem. Tym razem jednak część potrzebnych mi funkcji występujących w „natywnym" Allegro nie była w ogóle zaimportowana. Po raz kolejny konieczna była edycja i rekompilacja źródeł biblioteki.

Kod gry

Kod gry w celu uniknięcia bałaganu i zachowaniu modularności został podzielony na kilka mniejszych części:

Utility.pas

Moduł ogólnego przeznaczenia. Znajdują się tu funkcje do dokładnego odmierzania czasu za pomocą sprzętowego zegara oraz logger zapisujący informacje na konsolę systemową i do pliku. Logger okazał się szczególnie przydatny przy debugowaniu gry.

Math.pas

Definicje typów takie jak wektor czy prostokąt oraz funkcje do operacji na nich. Zawiera również funkcje służące do wykrywania kolizji między różnymi obiektami geometrycznymi za pomocą SAT (Separating Axis Theorem). Do wykrywania kolizji między prostymi obiektami (np. prostokąt-koło) zostały napisane prostsze i szybsze funkcje.

Warto zwrócić szczególną uwagę na klasę TVector2. Definiuje ona wektor dwuwymiarowy oraz różne operacje na nim, takie jak dodawanie, odejmowanie, mnożenie przez skalar, ale również iloczyn skalarny i wektorowy, projekcja wektora na wektor czy odbicie. Dzięki przeładowaniu podstawowych operatorów, operacje na tej klasie są niezwykle intuicyjne.

Pisząc moduły Utility oraz Math dbałem o to aby zachować wysoki współczynnik *reusability*[1]. Dzięki temu bez żadnych modyfikacji kodu moduły te mogą zostać użyte przy innym projekcie.

Arkanoid.pas

Ponieważ zarówno gra jak i edytor korzystają z tych samych stałych oraz typów, wygodnie było „wyciągnąć wspólny czynnik przed nawias", czego efektem jest ten właśnie plik.

[1] Możliwość ponownego użycia danego kawałka kodu

Editor.pas

Program do tworzenia plansz. Jego możliwości zostały ograniczone do minimum, ale mimo to można bardzo szybko stworzyć w nim ciekawą planszę. Plansza zapisywana jest w folderze Levels w pliku o pierwszej wolnej nazwie pasującej do maski %d.lvl, gdzie %d to numer planszy.

List.pas

Definicje list podwójnie wiązanych przechowujących obiekty gry takie jak TBall, TBrick itp. Ostatecznie w grze nie zostały wykorzystane, gdyż prostsze rozwiązanie polegające na przechowywaniu tych obiektów w statycznych tablicach i oznaczanie ich jako żywe/martwe okazało się szybsze.

YAAC.pas – główny plik zawierający pełną mechanikę gry.

Skrótowy opis działania gry

Pierwszą funkcją wywołaną w programie jest funkcja Init. Jej głównym zadaniem jest inicjalizacja trybu graficznego oraz wczytanie wszystkich zasobów. Jeżeli któryś z tych etapów się nie powiedzie, funkcja zwraca błąd, a logger zapisuje informacje o błędzie do pliku.

Następnym etapem po udanej inicjalizacji jest nieskończona pętla, w której wywoływane są trzy główne funkcje.

Pierwsza z nich (UpdateTime), jak sama nazwa wskazuje, służy do aktualizowania czasu gry. W każdej klatce gry pobierany jest obecny czas, od którego następnie odejmuje się czas poprzedniej klatki. Daje to wartość nazwaną w kodzie mDeltaTime. Jest to po prostu czas, który upłynął od rysowania poprzedniej klatki. Dzięki tej wartości możliwy jest ruch obiektów ze stałą prędkością, niezależną od maszyny na której uruchomiono grę. Drugim zadaniem tej funkcji jest aktualizacja licznika FPS.

Drugą funkcją jest funkcja Update. To w niej znajduje się cała logika gry. Zajmuje się ona nie tylko aktualizacją pozycji danych obiektów, ale również wykrywa kolizje między nimi i reaguje na nie.

Ostatnia funkcja to Draw. Zajmuje się ona tylko i wyłącznie odrysowaniem całego ekranu gry.

Po zakończeniu gry, czyli kiedy zmienna gGameRunning przyjmie wartość false, pętla główna zostaje przerwana i wywołana zostaje funkcja DeInit. Jej zadaniem jest zwolnienie wszystkich zasobów wczytanych na początku działania gry.

Kompilacja projektu

Aby skompilować projekt należy w linii poleceń wpisać:

```
FPC -Mobjfpc -Fu"ścieżka_do_allegro_lib" -O1 -Twin32 yaac.pas
```

Wymagany jest oczywiście kompilator FPC. Poprawioną bibliotekę Allegro dołączyłem do projektu.