Podręcznik do

próbka

/okładka zastępcza/

Adam Mierzwa

Słowo od autora

Czytasz poglądówkę, uciętą w miejscu, gdzie wciąż jeszcze jest szkicem. To demo książki, którą mam nadzieję ukończyć w tym roku.

Pierwsze dwie lekcje oparte są o analizę przykładów ze strony Hollywood i są moją propozycją do nauki „na sucho”. Bez posiadania Hollywood, w oparciu o dowolny edytor kodu. Wygaszacz „Aquarium” pokazuje bardzo dużo rzeczy; może nie przekrój języka, ale użyte rozwiązania takie jak pętla i tablica jak na pierwsza lekcję to wręcz przesada. Z drugiej strony nie mogę wykluczyć, że po podręcznik sięgnie osoba znająca inne języki programowania. Pokazanie wyłącznie prymitywnych elementów nie zainteresuje taką osobę, nie zachęci do sięgnięcia po Hollywood.

Jeśli czegoś nie zrozumiesz, to nie przejmuj się; jest jeszcze dużo stron przed nami dużo różnych przykładów. Krzywa uczenia się u mnie nie jest stroma, raczej pagórki.

Muszę opierać się na przykładach cudzych. Nawet jak szybko nauczyłbym się języka, to nie jest możliwe żeby od razu przeskoczyć na projektowanie własnych gier. Ciężko mi będzie przebić 200 stron, a oczekiwania są duże. Pasuje zrobić ponad 400, ale jak na jeden program zużywam 10-15 stron, to żeby zapełnić 400 musiałbym być ekspertem w tym języku, a nie zwykłym użytkownikiem Hollywood, Albo opisać kilkadziesiąt programów/gier, czyli też być ekspertem. Jest wyzwanie.

Pomysły na dalszy ciąg książki:

- gra logiczna
- gra typu bomberman (”magazynier”)
- chcę pokazać jak zrobić grę typu The Widow
- rozdział o własnym projekcie – pomysł, przeniesienie na papier koncepcji, zebranie materiałów, przygotowanie grafik, muzyki i na koniec przeniesienie tego w kod i testowanie.

Kompletny podręcznik w wersji dla patronów myślę, że zdążę przygotować do końca października. Potem będą poprawki o znalezione błędy i sugestie. Publiczna wersja do końca roku. Życzcie mi wytrwałości, a sobie cierpliwości.

Kontakt do mnie: adam_mierzwa@yahoo.com

Zawartość

O autorze Hollywood....4

Andreas Falkenhahn i Amiga....4

Andreas Falkenhahn i programowanie....4

O Hollywood....4

Rozwój Hollywood....4

Zalety i wady Hollywood....6

Hollywood a Lua....6

Zakup....7

Co jest czym....7

Objaśnienie podstawowych pojęć....8

Korzystanie z dokumentacji....10

Zbierając na bilet do Hollywood....11

Wygaszacz „Aquarium”....14

Cracktro„Fairlights”....31

Instalacja i konfiguracja....39

O autorze Hollywood

Andreas Falkenhahn i Amiga

Zaczął od Amigi 500, by poprzez CD-32 z przystawką SX-1 dojść do Amigi 1200 w wieży Micronik, z kartą Blizzard 68040. To był jego podstawowy zestaw w latach 1997-2001, z tym że to Amiga bez karty graficznej, Workbench o rozdzielczości 640x256 i 16 kolorów. Blizzard PPC z RTG ożywił jego zainteresowanie Amigą i przyszedł czas na poznanie Amigi NG. Mimo otrzymania Micro-A1 w 2005 roku podstawową maszyną deweloperską stał się Pegasos 2 z systemem MorphOS (który umożliwia także używanie AmigaOS 4). W planach była przesiadka na X5000, niestety nie istnieje żadna sensowna karta graficzna wspierana jednocześnie równie dobrze pod AmigaOS 4/MorphOS.

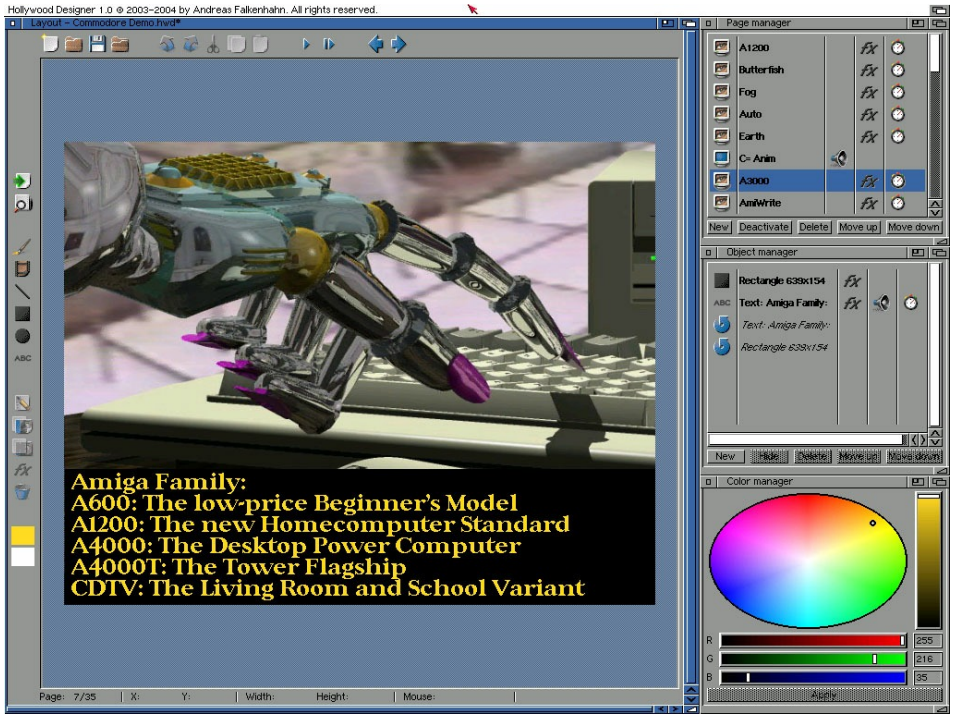
Andreas Falkenhahn i programowanie

Przeszedł pełną ścieżkę: AmigaBASIC, potem AMOS, potem Blitz Basic 2, potem 68k Assembler, potem AmigaE i wreszcie C. Stworzył sporo drobnego oprogramowanie – freeware (AHelp, CDTV Tools, Da Cool Booter, Da Cool Patcher, DMS2HD, SNES9x GUI) i shareware (Akiko, CD32 Games Install Kit, Da Cool Installer, NewInstaller Rainboot). Hollywood to największy projekt. Od wersji 1.0 (2002 r.) do 3.0 powstał w ledwie 2 lata i to zanim Falkenhahn był studentem. Pytany o motywy, dlaczego powstało Holywood, twierdził że zawsze chciał mieć własny język programowania.

O Hollywood

Rozwój Hollywood

Pierwszym dużym wyzwaniem było przeniesienie na platformę MorphOS (2002 r.), ponieważ Holywood został napisany w AmigaE i wiązało się to z przepisaniem do języka C. Trwało to ledwie 3 tygodnie, przy czym sama kompilacja to kilka wieczorów. Wbrew spekulacjom Falkenhaln pracuje sam, włączając w to obszerne testowanie, stąd każdej nowej odsłonie programu towarzyszy slogan „Airsoft Softwair, the hardest working men in code business”.



Historyczne wersje Hollywood, na górze prezentacja możliwości (2005 rok), na dole wersja 1.0 Designera

Pierwszy moduł który powstał to Malibu (2003 r.), początkowo płatny, który umożliwiał import projektów ze znanego programu do prezentacji Scala. Jednocześnie powstawały programy i dema prezentujące możliwości Hollywood.

Ważna data to 2004 rok, kiedy debiutował Designer, czyli typowy program prezentacyjny, będący w istocie nakładką na Hollywood.

Rdzeń programu został oddzielony od reszty, która jest zgodna z API systemu na który działa Hollywood. Obecnie Hollywood działa „na wszystkim” (systemy wywodzące się z Amigi jak i wiodące, takie jak Windows, Linuks i inne). Kolejne funkcjonalności (np. obsługa kolejnych formatów danych) realizowane są przez kolejne wtyczki.

Zalety i wady Hollywood

Niewątpliwą zaletą jest przenośność kodu. Także łatwość programowania, ponieważ jest to język skryptowy a nie kompilowany. Program jest znany, zakorzeniony na rynku i daje możliwość tworzenia własnych programów i gier nawet amatorom.

Wadą Hollywood jest niska wydajność. Także wysoka cena. Wiele osób narzeka na różne drobne błędy, ale kontakt z autorem pod względem poprawek i w ogóle wsparcia jest bardzo dobry.

Hollywood a Lua

Warto rozjaśnić tę kwestę. Lua to nie jest jakaś egzotyka jak ARexx, który jest właściwie językiem martwym. Lua została wprowadzona (przez Grzegorza Kraszewskiego) do systemu MorphOS. Niestety tu dobre wiadomości się kończą.

W Wikipedii można znaleźć informację, że Hollywood opiera się na języku Lua, ale na stronie Luy nie jest to wymienione (a przynajmniej ja nie znalazłem). W dokumentacji SDK do Hollywood w podrozdziale 2.16 „Differences between Hollywood and Lua” znajduje się informacja, że wirtualna maszyna Lua jest sercem Hollywood, ale zaraz potem wymieniona jest lista różnic. Jest ich dużo. Na tyle dużo, że jeśli znasz Luę, to musisz tam zajrzeć, a jeśli nie znasz – informacja o podobieństwie do języka Hollywood niech pozostanie ciekawostką. Nie warto uczyć się Luy z myślą o Hollywood, wprost rzecz ujmując.

Zakup

Jest możliwy tylko na stronie autora, przy czym preferowana jest płatność w usłudze PayPal. Oprogramowanie można nabyć w wersji do pobrania jak i (za dopłatą) na nośniku CD.

Co jest czym

Poniżej treść cennika (stan na czerwiec 2021 rok) z objaśnieniem co jest czym.

Hollywood 9.0 for Amiga, Windows, macOS, and Linux (79 euros)
Podstawowy program, dostajesz wersje na wymienione systemy

Hollywood 9.0 Upgrade for Amiga, Windows, macOS, and Linux (49 euros)
To samo co wyżej, ale opcja zniżkowa, jeśli masz wcześniejszą wersję.

Hollywood 9.0 Upgrade for Amiga (39 euros)
Hollywood 9.0 Upgrade for Linux (39 euros)
Hollywood 9.0 Upgrade for macOS (39 euros)
Hollywood 9.0 Upgrade for Windows (39 euros)
To są wersje tylko na pojedyncze platformy. Jest to opłacalne jeśli używasz Hollywood tylko pod jednym systemem.

Hollywood Designer 5.0 for Amiga (69 euros)
Hollywood Designer 5.0 Upgrade from any previous version (49 euros)
Nakładka na Hollywood, umożliwiająca tworzenie aplikacji w Hollywood bez pisania kodu. Taki graficzny generator prezentacji z ambicjami na coś więcej. Analogicznie jak wyżej, aktualizacja z poprzedniej wersji wychodzi taniej niż zakup pełnej.

Hollywood APK Compiler 3.0 Home Edition (49 euros)
Hollywood APK Compiler 3.0 Professional Edition (79 euros)
Hollywood APK Compiler 3.0 Upgrade Home to Professional Edition (39 euros)
To ci umożliwi kompilowanie apek na Androida. Natywnych, czyli użytkownik nie będzie potrzebował czegoś dociągać do odtwarzania apletów Hollywood. APK będzie działać jak każda inna aplikacja na smartfonach z Androidem. Ograniczenie wersji Home to niemożność edycji: własnego kodu w Javie, manifestu i skryptu Gradle.

Remedios 1.0 for macOS (49 euros)
Jest odpowiednikiem narzędzia powyżej, ale do kompilowania na urządzenia Apple z systemem iOS.

Przy zakupie na nośniku CD doliczana opłata to 4 euro wysyłka na terenie Niemiec, 5 euro na Europę, 9 euro na cały świat. Opłata zawiera koszt nośnika i wysyłki. Wszystkie ceny nie są obłożone dodatkowo VAT-em.

To co powyżej to jest płatne. Zakupu dokonasz na podstronie Purchase:

https://www.hollywood-mal.com/purchase.html

Oprócz tego na podstronie Download

https://www.hollywood-mal.com/download.html

będziesz mógł pobrać:

Plugins – wtyczki rozszerzające Hollywood głównie obsługę różnych formatów plików
Hollywood Player – odtwarzacze apletów
Example scripts – przykładowe skrypty, ale z programem masz je dostarczone i tak, i to o wiele więcej
SDK – przydatne jeśli zechcesz tworzyć własne wtyczki
Add-ons for third-party IDEs – dodatki pozwalające na prawidłowe podświetlanie składni w innych edytorach, które nie mają wbudowanej obsługi Hollywood
Translations – tłumaczenia

Objaśnienie podstawowych pojęć

Pewne pojęcia zakładam że znasz. Dlatego takie rzeczy jak wytłumaczenie co to jest procesor, pamięć RAM, różne podstawowe pojęcia z obsługi komputera – wspólnie sobie nawzajem darujemy. Natomiast jest sporo określeń, które znasz choćby z nazwy, ale warto tę wiedzę uporządkować już na początku. Żebyś miał pewność, że rozumiemy pod danym określeniem to samo.

Apka – zdrobnienie od „aplikacja”, przy czym dotyczy to raczej aplikacji na smartfona. Ostatnio coraz rzadziej używa się określeń „programy” i „gry”, wszyscy robią aplikacje. To brzmi bardziej profesjonalnie, a dla mnie pretensjonalnie, zwłaszcza że słowo aplikacja ma też inne znaczenia (to może być podanie o pracę albo podanie zastrzyku). Będę używał określeń wymiennie, ale za każdym razem chodzić będzie o to co napisałeś w Holywood i daje się uruchomić.

Skrypt – najprościej rzecz ujmując, to sposób zapisu programu w pliku tekstowym. Jest to rażące uproszczenie, ale dla naszych potrzeb wystarczy.

IDE – skrót od „Integrated Development Enviroment” czyli „zintegrowane środowisko programistyczne”. Oprócz edytora kodu najczęściej zawiera także kompilator, służący do łączenia plików programu i przekształcenie projektu w pliki działające u użytkownika. Mówiąc brzydko i obrazowo, twoje wypociny dzięki kompilatorowi dadzą się wypluć do pliku .exe, który będzie tym właściwym programem/grą. To od ciebie zależy, czy cała aplikacja będzie zapakowana w jeden plik wykonywalny, przykładowo .exe., czy też rozbita na katalogi i pliki. Oprócz edytora i kompilatora IDE może zawierać debugger, który ułatwia znalezienie błędów i poprawienie ich. Bardziej zaawansowane IDE pozwalają na poprawienie wyglądu kodu, integrację z GitHubem i mają wiele innych użytecznych funkcji.

SDK – skrót od „Software Develepment Kit” czyli zestaw dla programistów. Zawiera dokumentację, pliki nagłówkowe, dodatkowe biblioteki, przykładowe kody źródłowe.

Aplet – od słowa „applet”. W języku Java to niewielki program dający się osadzić na stronie www. W języku REBOL używane jest określenie „reblet”. W Hollywood aplet to jest aplikacja w formie, która wymaga posiadania w systemie odtwarzacza.

Lua – skryptowy język programowania, który stał się podstawą Hollywood.

Brush – ang. „pędzel”. W starych programach graficznych to słowo zapuściło korzenie. Użycie tego słowa w grafice w grze wywodzi się od tego, że malować można było nie tylko kolorem ale i teksturą. No i został nam taki archaizm, jak i „duszek”.

Sprajt – ang. „duszek”. W odróżnieniu od brusha jest żywy (porusza się w grze).

Zmienna – przechowuje wartość, którą można zmienić. Jest kilka podstawowych typów zmiennych, przedstawia je tabelka.

Typ zmiennej	Znaczenie	Przykłady
Boolean	typ logiczny	True (prawda) i False (fałsz)
Nil	typ pusty	nil
Number	liczba zmiennoprzecinkowa	3.141592
Integer	liczba całkowita	100
String	tekst	”tekst” ‘tekst’

Jest wiele więcej określeń, np. funkcja, pętla, identyfikator, ale to opiszę przy okazji omawianych przykładów. Wtedy będzie widoczne od razu ich zastosowanie, będzie można lepiej rozumieć jak działa program.

Może się zdarzyć, że ucząc się napotkasz na różne sprzeczności. Na przykład instrukcja `ElseIf` raz jest określana jako identyfikator (ang. identifier), gdzie indziej jak preprocesor (ang. preprocessor command), a kolega który programuje powie ci, że to instrukcja warunkowa albo instrukcja wyboru. W ogóle się tym nie przejmuj. Skup się na zrozumieniu, jak działa program.

Korzystanie z dokumentacji

Dokumentacja znajduje się na stronie Hollywood pod przyciskiem Help, bezpośredni link do podstrony:

https://www.hollywood-mal.com/help.html

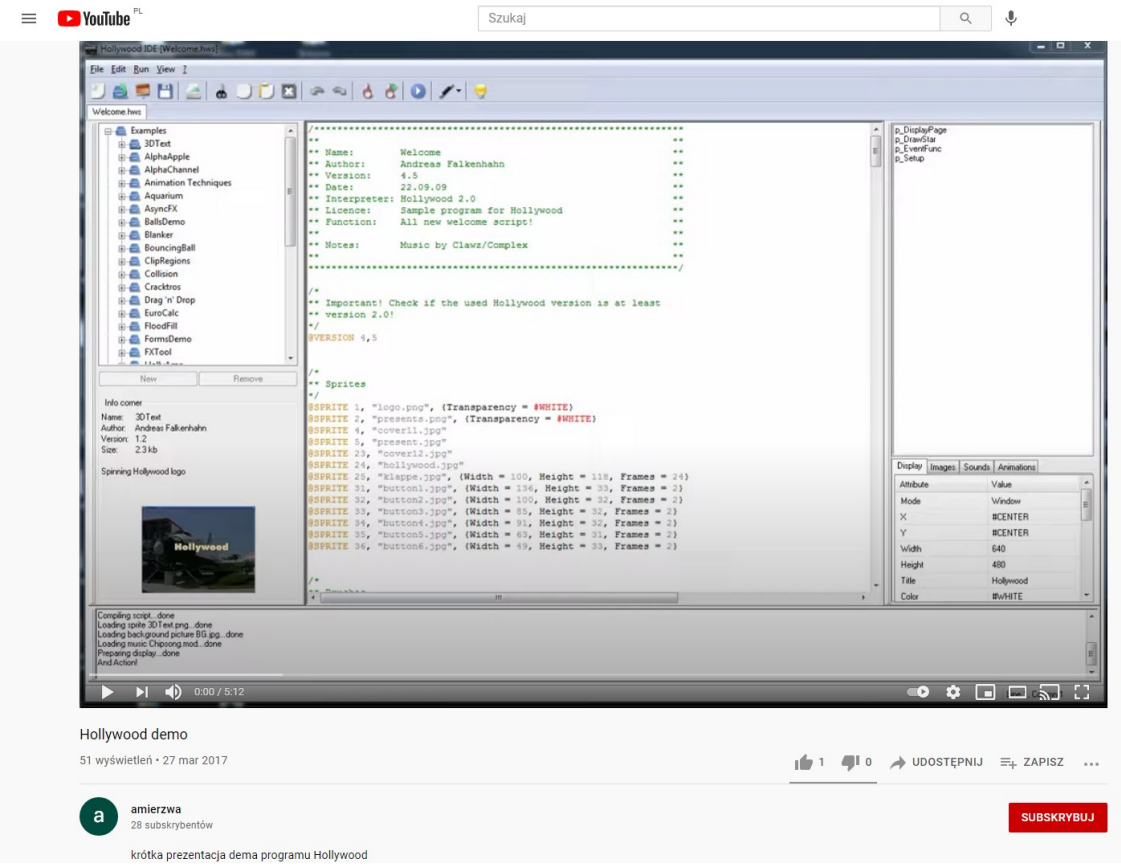
Jest dostępna jako PDF i jako HTML do przeglądania online. Korzystam z obu form. PDF pozwala mi szybciej coś wyszukać, natomiast forma online jest lepsza do przeklejania w narzędzie Google Translator, bo tekst nie jest po drodze łamany.

Do przeglądania PDF na pececie polecam lekki czytnik Sumatra (ciekawostka: autorem jest Adam Kowalczyk, postać która miała swoje 5 minut w historii Amigi nowej generacji).

Zachęcam cię do komentowania kodu. Będzie ci to pomocne na początku dla utrwalenia wiedzy, a później może się przydać gdybyś chciał komuś udostępnić źródła swoich programów i gier.

Zbierając na bilet do Hollywood

Kiedyś nagrałem filmik reklamujący Hollywood przy okazji jednej ze zbiórek na PPA:



Wtedy bawiłem się demem pod Windows i pokazałem z poziomu IDE uruchamiane przykłady. Wydawały mi się bardzo proste, a zwłaszcza wygaszacz ekranu „Aquarium”. Muszę się przyznać, że gdy zacząłem naukę Hollywood od niego, takim się nie okazał. Ale zaczynamy naszą wspólną naukę w miejscu, w którym ostatnio przerwałem. Dlatego pokażę najpierw „Aquarium”, gdyż uważam że naukę powinno zaczynać się od przykładów dołączonych do programu, po to one są. To także wyciągnięta dłoń do tych spośród nas, którzy jeszcze nie kupili Hoolywood – wskazany wygaszacz (a także kilka innych przykładów) można pobrać ze strony programu za darmo, a przeanalizować pod windowsowym (również darmowym) Microsoft Visual Studio.

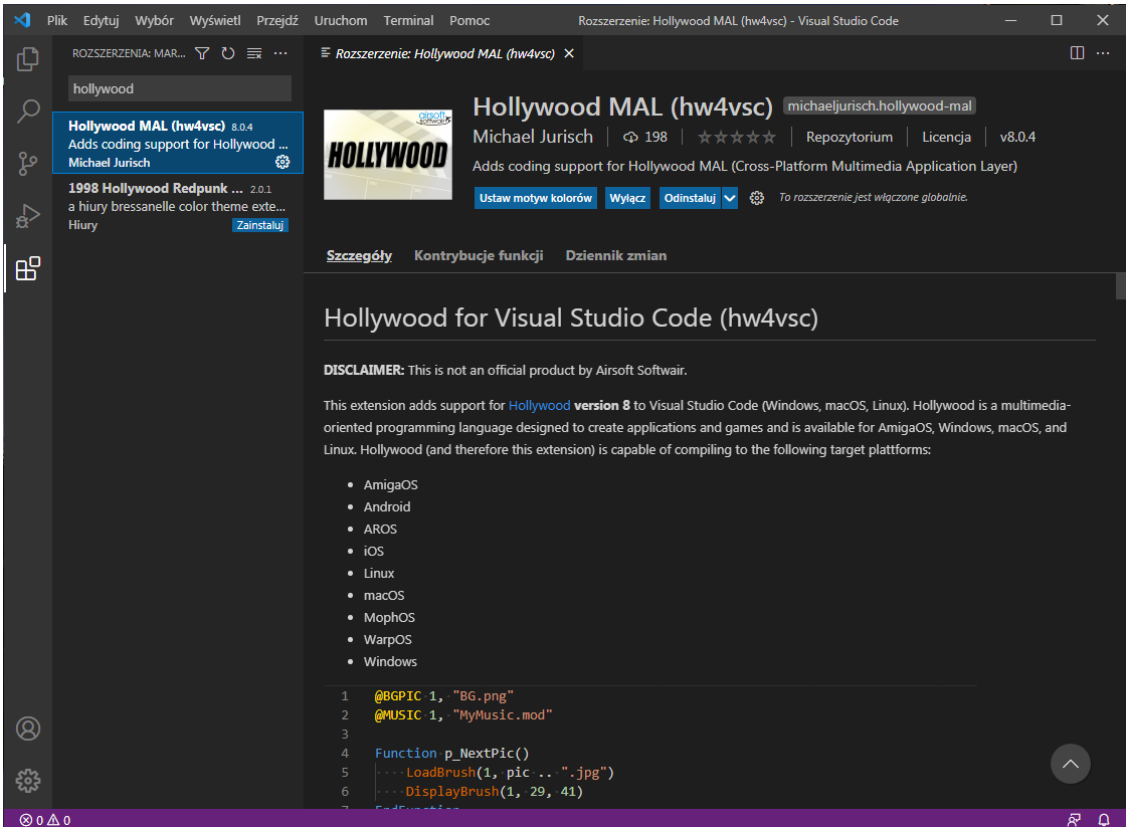
Jest pewna niedogodność – nie posiadając Hollywood nie będziesz mógł skompilować skryptu do działającego programu. Możesz to skompensować sobie oglądając wygaszacz w działaniu – na powyższym filmiku jest około trzeciej minuty. Nie powinien być to problem na tym etapie nauki, bo analizujemy ukończony, działający program. Archiwa z przykładami oprócz kodu zawierają użyte obiekty, będziesz widział na czym pracujemy.

Nie ma co dalej się ociągać – zaczynajmy!

Visual Studio Code

https://code.visualstudio.com/download

Visual Studio Code jest innym programem niż Visual Studio Community, nie tylko mniejszym ale i bardziej pasującym do naszych potrzeb. Ma też wtyczkę do kolorowania składni dla języka Hollywood.



Jeżeli chciałbyś bez instalacji VS Code zobaczyć możliwości tej wtyczki, to przedstawione są tutaj:

https://marketplace.visualstudio.com/items?itemName=michaeljurisch.hollywood-mal

```
34
35          For k = 1 To 5
36
37                  fish[k].c = Wrap(fish[k].c + 1, 1, 11)
38                  fish[k].x = fish[k].x + fish[k].zx
39                  fish[k].y = fish[k].y + fish[k].zy
40
41                  If fish[k].x > fish[k].xb
42
43                          fish[k].zx = -4 - Rnd(4)
44                          fish[k].xb = Rnd(900)
45                          fish[k].frame = 10
46
47                  ElseIf fish[k].x < -fish[k].width - 6
48
49                          fish[k].zx = Rnd(4) + 1
50                          fish[k].xb = Rnd(900)
51                          fish[k].frame = 0
52
53                  EndIf
54
55                  If fish[k].y > fish[k].yb
56
57                          fish[k].zy = -3 - Rnd(3)
58                          fish[k].yb = Rnd(900)
59
60                  ElseIf fish[k].y < -fish[k].height - 6
61
62                          fish[k].zy = Rnd(3) + 1
63                          fish[k].yb = Rnd(900)
64
65                  EndIf
66
67                  DisplaySprite(k, fish[k].x, fish[k].y, fish[k].frame + fish[k].c)
68          Next
69
```

Powyżej fragment kodu, który będzie pierwszą lekcją. Pokazuję jako przykład kolorowania składni w Visual Studio Code z podaną wtyczką.

Możesz korzystać z innego edytora kodu. Wspierane są na stronie programu w dziale Downloads pod pozycją „Add-ons for third-party IDEs”. Na tę chwilę (oprócz VS Code) wsparcie mają: Cubic IDE, UltraEdit, Vim i Notepad++.

14

Wygaszacz „Aquarium”

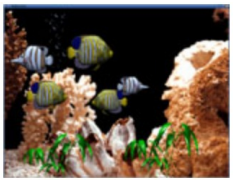
https://www.hollywood-mal.com/download.html

Sekcja „Example scripts”.

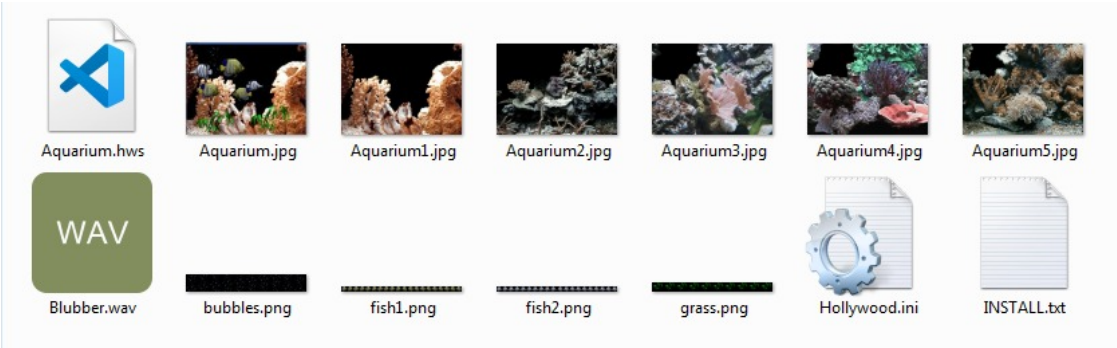
Aquarium

A Hollywood version of the famous Aquarium screensaver. Created by Telemar Rosenberger.

Download: Aquarium.zip (900 KB)



Poniżej zawartość katalogu po rozpakowaniu. Nie ma tam pliku wykonywalnego, są za to wszystkie pliki projektu. Holllywood.ini i INSTALL.txt są nieistotne.



Aquarium.hws – skrypt

Aquarium.jpg – zrzut ekranu działającego wygaszacza, nieużywany w projekcie

Aquarium1.jpg – Aquarium5.jpg – obrazki tła, będą one dobierane losowo przez program; to będzie ogólny widok naszego akwarium

Blubber.wav – plik dźwiękowy o długości 5 sekund, prezentuje dźwięk napowietrzacza w akwarium („bąbelków”)

bubbles.png – wspomniane bąbelki

fish1.png i fish2.png – prezentują rybki

grass.png – roślinka

Zajrzyjmy wreszcie do kodu:

```
1   /*****************************************************************

2   **          Aqarium © By T.Rosenberger 2003  for Hollywood 1.5   **

3   **                                                               **

4   **               Hollywood, and the World will never be the same **

5   **        http://abakus-design.amiforce.de/aquarium/aquarium.php **

6   *****************************************************************/
```

Pierwszy blok nie jest kodem. Wszystko to, co jest umieszczone pomiędzy `/*` a `*/` jest komentarzem. Czyli nie jest wykonywane jako program. Możesz tak obłożyć nieużywany fragment kodu, bądź zamieszczać komentarze do napisanego kodu. Tutaj zostało to wykorzystane do przedstawienia informacji o programie. Tak robisz, jeśli twój komentarz jest wieloliniowy. Pozostałe użyte „gwiazdki” `*` służą jedynie dla ozdoby, do wykreślenia kształtu prostokąta, tworzą ramkę dla tekstu. Jeżeli chcesz umieścić komentarz jednoliniowy (może zaczynać się w środku wiersza, zaraz po kodzie) to zacznij od znaku średnika `;`.

Następne linijki, zaczynające się od znaku „małpy” `@`, określają warunki początkowe, ale bardziej właściwe jest użyć określenie „definicje”. Definiujemy (określamy) co jest czym. W dokumentacji określane jest to jako „preprocessor command”.

```
7

8   @VERSION 2, 0
```

Mamy więc podaną wersję 2.0. To oznacza, że do skompilowania tego skryptu wymagany jest Hollywood w wersji minimum 2.0. Nie jest to istotna informacja o tyle, że w sprzedaży zawsze jest tylko najnowsza wersja, dostępne odtwarzacze apletów na wszystkie platformy również; z drugiej strony użytkownika nie obchodzi etap tworzenia, bo dostaje gotowy program. Niemniej ta deklaracja powinna znaleźć się na początku przed wszystkimi poleceniami. To może mieć znaczenie, jeśli wymieniałbyś się źródłami, a ktoś miałby starszą wersję Hollywood.

Poniżej wypisane są definicje użytych elementów. Zwróć uwagę, że każda kategoria (sample, sprajty) ma oddzielną numerację. Jeśli odwoływałbyś się później do konkretnego elementu, posłużysz się właśnie numeracją a nie nazwami plików na dysku. Jedynie na początku jest przedstawione powiązanie, na przykład że `SAMPLE 1` to plik Blubber.wav.

```
 9

10   @SAMPLE 1, "Blubber.wav"
```

Następne linijki określają sprajty.

Sprajt nr 1 to obrazek fish1.png. Jest określona przezroczystość, w naszym obrazku będzie to `#BLACK` (ang. „czerń”). Dalej podane są wymiary – `Width` to szerokość, `Height` to wysokość; `Frames` to dosłownie ramki, ale to chodzi o liczbę klatek, `FPR` (ang. „frames per row”) to liczba klatek w rzędzie. Zazwyczaj parametry `Frames` i `FPR` są takie same. Musi tak być, przykładowo przy różnych wartościach mógłby się wyświetlać wycinek paska z rybkami o długości innej niż ryba, na przykład pół ryby albo półtorej.

```
11   @SPRITE 1, "fish1.png", {Transparency = #BLACK, Width = 256, Height =
     192, Frames = 20, FPR = 20}

12   @SPRITE 4, "fish2.png", {Transparency = #BLACK, Width = 160, Height =
     120, Frames = 20, FPR = 20}

13   @SPRITE 6, "bubbles.png", {Transparency = #BLACK, Width = 82, Height
     = 120, Frames = 10, FPR = 10}

14   @SPRITE 7, "grass.png", {Transparency = #BLACK, Width = 320, Height =
     240, Frames = 10, FPR = 10}
```

Tu mała uwaga: przy składaniu tego podręcznika kod w dłuższych liniach jest złamany w sposób nie zawsze poprawny. Po prostu wynikło tak z szerokości strony.

Druga kwestia: ten wygaszacz ma niewiele elementów, ale gdybyś pisał własny program, warto chwilę przysiąść i wypisać sobie wcześniej, zanim zaczniesz tworzyć kod, co się w nim ma dziać, jakie rzeczy będą składać się na niego.

Jak widzisz obrazek zawiera fazy ruchu rybki – płynie ona w prawo, a potem w lewo. Obrazki pozornie są takie same. Na małej ilustracji tego nie widać, ale różne jest ułożenie płetw piersiowych, stopień ich pofalowania, rzucany przez nie cień.

To razem sprawia, że przesuwający się obrazek ryby wygląda naturalnie, bo rybka wachluje płetwami.

Już w tym momencie powinieneś sobie zdawać sprawę jak wiele pracy przed tobą od strony graficznej. Dlatego na początku tworzenia swojej gry posługuj się uproszczonymi obrazkami. Dopiero jak wszystko działa poprawnie, zrób ładne wersje. Podczas projektowania i tworzenia gry wiele się zmienia, szkoda marnować czas i energię na rzeczy, które lądują w koszu.

Kolejne użyte sprajty to fish2.png (druga rybka), bubbles.png (bąbelki powietrza) i grass.png (roślinka).

Może zauważyłeś, że nie jest podana ścieżka dostępu do plików. Jest tak dlatego, że projekt tego wygaszacza składa się z niewielu plików i wszystkie znajdują się w tym samym katalogu co plik skryptu.

```
16

17   @DISPLAY {Width = 800, Height = 600}

18

19   EscapeQuit(True)

20
```

Linia 17 Oznacza, że program uruchomi się w oknie o rozmiarach 800 na 600 pikseli.

Linia 19 określa, że wciśnięcie klawisza Esc pozwala wyjść z programu. To jest wygodne, ale jeśli stworzyłbyś nie wygaszacz, a np. program z polami do wpisywania danych, to dosyć łatwo można by było utracić je przypadkowo. W takiej sytuacji parametr dobrze jest przestawić z `True` na `False`, wtedy jednak musisz przewidzieć inny sposób wyjścia z programu. Jest to przykład zmiennej booleanowskiej, gdzie są tylko dwie możliwości „jest-nie ma” i to jest sprawdzane.

```
21   SetFontColor(#WHITE)

22   TextOut(#CENTER, #CENTER, "PLEASE WAIT...")

23
```

W linii 21 została ustawiona czcionka na kolor biały.

W linii 22 jest polecenie podania tekstu na ekran. W nawiasie jest określone, że ma to być na środek ekranu, a wyświetlany tekst to "PLEASE WAIT..." (”Proszę czekać...”).

Dalej w kodzie znajduje się długi blok funkcji (linie 24 do 79). Jest to pętla o nazwie `p_MainLoop()`. Przedrostek `p_` używaj zawsze jeśli tworzysz własną funkcję. To cię uchroni przed niespodziankami, gdyby w przyszłych wersjach Hollywood zostały wprowadzone nowe funkcje, ale o nazwach jakie ty używałeś. Wtedy program by działał inaczej niż zaplanowałeś. Użycie `p_` powoduje, że nazwy będą różne. Natomiast nazwa funkcji może być dowolna, chociaż też nie ma co wydziwiać; tu autor dał `MainLoop`, czyli po polsku „główna pętla”. Co ona robi?

Określa ona życie w akwarium – sposób pływania rybek, ruch bąbelków, falowanie roślin. Zwróć proszę uwagę na formatowanie bloków kodu, a ściślej wcięcia. Blok `Function` p_MainLoop() (linia 24) domykany jest przez `EndFunction` (linia 79), `BeginRefresh` (linia 26) domykany jest przez `EndRefresh` (linia 77), itp.

```
24  Function p_MainLoop()

25

26      BeginRefresh

27

28          ; bubbles are drawn every second time

29          call = 1 - call

30

31          If call = 1

32                  DisplaySprite(6, 140, 20)   ; bubbles

33          EndIf

34
```

Linia 26 rozpoczyna ustawienia odświeżania. Zaczynamy od bąbelków w linii 28. Tu pokażę coś – mam nadzieję – interesującego, jak własnym rozumowaniem (a zarazem przy błędnych założeniach) można dojść do podobnych rezultatów.

Ja kombinowałem tak: „dowiadujemy się” (linia 29), że `call=1` i od razu `call` odejmuje samą siebie. A jeśli `call = 1`, to `call = 1 - call` to będzie 1 = 1 - 1. Czyli wynik działania tej linii daje 0. Teraz za drugim razem będzie to już nie 1 = 1 - 1, bo `call` ma nową wartość 0. Będzie 0 = 1 - 0, co da wynik 1. Czyli wynikiem tego „równania” będzie uzyskiwanie na zmianę wartości 1 i 0.

Prawda okazała się inna. W Hollywood jeśli zmienna nie ma deklaracji (przypisanej wartości) jest zerowana (dostaje wartość początkową równą 0). Czyli w naszym przypadku `call = 1 - call` da 0 = 1 - 0 czyli wyjdzie 1; następnym razem jak podstawimy 1 to 1 = 1 - 1 da 0, czyli na zmianę będzie 1 i 0.

Teoretycznie wychodzi na to samo. W praktyce – warto sięgać do dokumentacji. Kombinowanie może być przyjemne, pogłębia twórcze myślenie, ale w tym konkretnym przypadku (świetna dokumentacja do Hollywood) wydłuża jedynie czas pracy.

Takie zmienne, które mogą zmieniać swoje wartości „w locie” nazywamy zmiennymi dynamicznymi. Zmienne zawierające ciągi tekstowe muszą kończyć się znakiem dolara `$`, a te które przechowują wartości zmiennoprzecinkowe znakiem wykrzyknika `!`.

Linię 31 należy rozumieć tak, że jeśli wartością jest 1 to wyświetlany jest sprajt z bąbelkami. Skąd to wiemy, że akurat ten sprajt? W linii 32 jest polecenie wyświetlenia sprajta nr 6 (pierwsza liczba w nawiasie). A to, że bubbles.png to sprajt 6, wiemy z linii 14, gdzie na początku programu zdefiniowaliśmy co jest czym.

Podsumujmy jak działa nasze `call`. Ponieważ wartości tej zmiennej dynamicznej przyjmą wartości 0 i 1 na zmianę, to bąbelki będą wyświetlane co drugie wykonanie głównej pętli programu. Jest to nawet określone w komentarzu w linii 28 („; bubbles are drawn every second time”).

Tu dla odprężenia i utrwalenia wiedzy pokażę dokładniej jak działa taka animacja, na przykładzie rzeczonych bąbelków. Poniżej obrazek bubbles.png.

Jak mu się przyjrzysz, to dopatrzysz się, że animacja składa się z 10 obrazków. Bąbelki co kolejny obrazek mają minimalne przesunięcia i dzięki różnicom między obrazkami powstaje wrażenie ruchu. Wczytany obrazek do jakiejkolwiek przeglądarki grafiki ujawni ci, że wymiary to 820 na 120. Czyli jak podzielimy dłuższy bok na 10 to wyjdzie, że wyświetlane bąbelki będą mieć 82 piksele szerokości i 120 pikseli wysokości. To jest wymiar obrazka, których komplet tworzy sekwencję animacji. Możesz sprawdzić w linii 13. Jest tam też podana liczba klatek, czyli 10, czyli też się zgadza, bo mamy 10 obrazków tworzących pasek.

A jak ustalane jest, gdzie bąbelki się pojawią, w którym dokładnie miejscu okna ekranu? To kolejne liczby. Przyjęta konwencja położenia to współrzędne lewego górnego rogu. Czyli obrazek będzie zaczynał się (patrz na linię 32) 140 pikseli od lewego boku ekranu i 20 pikseli od jego górnej krawędzi. Na kolejnej stronie pokazane jest całe akwarium. Poświęć chwilę uchwyceniu co i jak w nim jest.

Jeśli uważnie przyjrzysz się bąbelkom, to słupek pęcherzyków powietrza jest wyższy niż w oryginalnym obrazku. Dlaczego? Czy coś przegapiłem?

Nie, obrazek został przeskalowany. Jest to sporo dalej, w linii 92 (nowy wymiar to 92 na 190). Jak myślisz, dlaczego to zostało zrobione? Autorowi nie chciało się zmienić wyglądu tworząc nowy obrazek? Może i tak. Ale dlaczego bąbelki mają być takie wyciągnięte? Jako akwarysta podpowiem ci – pędząc ku górze, do powierzchni wody, zmieniają kształt właśnie w taki sposób. Pewnie wynika to z jakichś praw fizyki, ale miło, że autor zadbał o uchwycenie takich szczegółów.

Dalej zaczyna się ciężki kawałek kodu.

```
35          For k = 1 To 5

36

37                  fish[k].c = Wrap(fish[k].c + 1, 1, 11)

38                  fish[k].x = fish[k].x + fish[k].zx

39                  fish[k].y = fish[k].y + fish[k].zy

40

41                  If fish[k].x > fish[k].xb

42

43                          fish[k].zx = -4 - Rnd(4)

44                          fish[k].xb = Rnd(900)

45                          fish[k].frame = 10

46
```

To było pierwsze miejsce, gdzie samodzielnie nie potrafiłem ruszyć z miejsca. Nie mogłem zrozumieć o co tu chodzi. Jest tu podstawowa trudność, wspólna dla przykładów dołączonych do Hollywood. Komentarze kodu są lakoniczne, często wprowadzane jest coś i nie tłumaczone co to jest, skąd, po co.

Na następnej stronie będzie ciąg dalszy tego bloku, żeby już nie szatkować kodu podam go w jednym kawałku, a po nim spróbuję opisać jak to działa.

```
47                  ElseIf fish[k].x < -fish[k].width - 6
48
49                          fish[k].zx = Rnd(4) + 1
50                          fish[k].xb = Rnd(900)
51                          fish[k].frame = 0
52
53                  EndIf
54
55                  If fish[k].y > fish[k].yb
56
57                          fish[k].zy = -3 - Rnd(3)
58                          fish[k].yb = Rnd(900)
59
60                  ElseIf fish[k].y < -fish[k].height - 6
61
62                          fish[k].zy = Rnd(3) + 1
63                          fish[k].yb = Rnd(900)
64
65                  EndIf
66
```

```
67                  DisplaySprite(k, fish[k].x, fish[k].y, fish[k].frame
    + fish[k].c)

68          Next

69

70          ; plants are drawn every second time!

71

72          If call = 1

73                  DisplaySprite(7, 400, 400)   ; plant 1

74                  DisplaySprite(8, 40, 440)    ; plant 2

75          EndIf

76

77      EndRefresh

78

79  EndFunction

80
```

Uwaga: istnieje kilkadziesiąt instrukcji w Hollywood, których nazwy są zarezerwowane i nie mogą być używane jako nazwy zmiennych i funkcji. Pełna lista jest w dokumentacji w podrozdziale 7.2

Własna funkcja (linie od 24 do 79) zawiera w sobie procedury odświeżania (linie od 26 do 77). Odświeżanie bąbelków (linie od 28 do 33) mamy za sobą. Niestety kolejne elementy animowane mają inne procedury.

Linia 35 zaczyna się od `For`. To instrukcja rozpoczęcia pętli, po której ustawia się zmienną. Nasza zmienna to `k`, która przyjmuje wartości od 1 do 5.

`If`, `ElseIf`, `Else`, `EndIf`. Na moim etapie wiedzy to są dla mnie warunki do wykonania. `If` to takie „gdy cośtam, to...”. `ElseIf` to „chyba że nie, to wtedy”, `Else` to „jeśli żaden warunek nie został spełniony, to...” i `EndIf` „koniec gdybania”.

Dostajemy garść liter – mamy k, c, x, y, zx, zy, xb. Część z nich wyjaśniona jest w liniach 102-104. Uporządkujmy:

`x, y` – pozycja ryby, jej współrzędne
`zx, zy` – prędkość x i y
`xb, yb` – granice x i y

Do ustalenia: `c` i `Wrap`. Funkcja Wrap jest opisana w dokumentacji w rozdziale 36 „Math library”. Przyjmuję, że cała pętla określa sposób poruszania się ryb. Linie 37 do 39 to zawracanie, 41 do 53 to poruszanie się w osi x, a 55 do 65 w osi y. Linia 67 określa od której klatki ma być wyświetlana animacja rybki.

Ryby poruszają się w losowych kierunkach, przy czym przekroczenie dozwolonego obszaru nie jest możliwe. Po osiągnięciu granicy rybka zawraca. Jaki obszar jest dozwolony? `Rnd` (ang. „random”) zapewnia losowe ustalanie współrzędnej, określona jest wartość 900, która jest maksymalną. Okno programu ma 800 na 600 pikseli, czyli ryby mogą wypływać poza ekran. Tu jest pewna niekonsekwencja, bo o ile przy animacji bąbelków autor zbliżył się do realizmu (kształt unoszących się pęcherzyków ku górze), to tutaj ryby mogą wypływać poza granice okna czyli za szyby akwarium. Oczywiście ryby nie zawracają tylko poza ekranem, wylosowana wartość to może być np. 400 i każda inna dowolna w przedziale do 900.

W tym momencie możesz mieć szereg pytań, na przykład:
- jak zdefiniowane jest zawracanie?
- jak jest rozwiązana sprawa, żeby ryba nie mogła płynąć tyłem?
- jak to się dzieje, że ryby się nie zderzają?

W tej lekcji nie otrzymasz na nie odpowiedzi. Inna sprawa, że ja nie potrafię ich na tę chwilę udzielić. Pokazuję jak sam się uczyłem – taka była umowa ze społecznością, z tymi którzy ufundowali mi ten program. Nauka nie musi zaczynać się od „Hello World!”, ani też od jakichś fragmentów kodu. Jest masa podręczników pełnych świetnie wytłumaczonych fragmentów kodu, ale nigdzie nie pokazują kompletnego programu. Moje podejście jest inne. Próbuję rozwiązania problemów, dzielę się spostrzeżeniami, przemyśleniami. Rzeczy proste tłumaczę wnikliwie od podstaw, rzeczy trudniejsze rozbijam na kawałki, najtrudniejsze – opisuję lakonicznie, na ile zrozumiałem. W dalszej części książki będzie jeszcze klasycznie (instalacja, konfiguracja, na początek jakieś proste kawałki kodu), ale na razie oglądamy kompletny program, jak jest zbudowany i jak działa.

```
81   ; make more fish!

82   CopySprite(1, 2)

83   ScaleSprite(2, 160, 120)

84

85   CopySprite(1, 3)

86   ScaleSprite(3, 140, 100)

87

88   CopySprite(4, 5)

89   ScaleSprite(5, 120, 80)

90

91   ScaleSprite(1, 180, 140)  ; fish 1

92   ScaleSprite(6, 92, 190)   ; bubbles

93

94   CopySprite(7, 8)

95   ScaleSprite(8, 300, 200)  ; plant 2

96
```

„Make more fish” czyli jak zrobić więcej ryb. Masz tutaj `CopySprite`, które służy do kopiowania sprajtów, oraz `ScaleSprite` służące do ich skalowania (zmiany rozmiaru). Ale co oznaczają parametry w nawiasach? Może zwróciłeś na początku uwagę na dziury w numeracji sprajtów (linie 11-14), były wypisane tylko 1, 4, 6 i 7. Gdzie pozostałe? Właśnie tworzysz je teraz.

Linia 81 `CopySprite(1, 2)` oznacza, że sprajt 1 zostanie sklonowany i powstanie sprajt 2. W kolejnej linii zostanie ustalony rozmiar dla nowego sprajta.

`ScaleSprite` jest istotne o tyle, że utworzone kopie nie będą identyczne. Liczby w nawiasach oznaczają kolejno numer sprajta i nowe wymiary dla niego. Zwróć uwagę, że sprajt 3 ma podane inne wymiary iż sprajt 2. Używając tych samych grafik uzyskasz rybki i rośliny różnej wielkości.

Czyli fish1.png to będą sprajty 1, 2 i 3. Sprajt 4, czyli fish2.png, utworzy tylko jedną kopię, sprajt 5. Otrzymaliśmy z dwu gatunków ryb (dwóch grafik) sprajty od 1 do 5, które przełożą się na 5 sztuk różnej wielkości.

Zwróciłeś może uwagę, że pomimo utworzenia nowych sprajtów ich nazwa nie występuje w kodzie? Nie ma nigdzie `SPRITE 2` na przykład? To była dla mnie jedna z pierwszych trudności w zorientowaniu się w składni Hollywood. Jest tak dlatego, że te nowe sprajty nie powstały za pomocą deklaracji `@`, tylko w wyniku innych poleceń. Taki numer sprajta w nawiasie zawsze będzie poprzedzony poleceniem `CopySprite`, `ScaleSprite`, `DisplaySprite` (bądź innym ze „Sprite” w nazwie).

Teraz omówiłem linie 81-90, ale dlaczego w linii 91 sprajt 1 jest skalowany? Co było z nim nie tak, przecież to był pierwotny, najpierwszy sprajt?

Przyjrzyjmy się „dawcy”(plik fish1.png). Oryginał ma 5120 x 192 piksele. Pocięty na 20 klatek da wymiar dla jednej 256. Określmy z grubsza, że to długość ryby. Da się zauważyć, że pozostałe „klony” (sprajty 2 i 3) z tego samego obrazka mają długości 160 (linia 83) i 140 (linia 86). Tak więc skalowanie sprajta 1 do długości 180 (linia 91) jest zabiegiem kosmetycznym, mającym wyrównać wielkość rybek w stadzie. Zamiast 256, 160 i 140 mamy 180, 160 i 140.

Ta sama operacja skalowania została wykonana dla roślinki (sprajt 7 utworzył kopię – sprajt 8 o nowym rozmiarze).

Zamykając temat skalowania – warto pamiętać, że lepiej skalować z lepszej jakości (większej rozdzielczości obrazka) w dół niż odwrotnie.

Dalej (kod na kolejnej stronie) mamy ważny element uruchomienia akwarium. Nie rozumiem, czemu tablica definiująca początkowe ułożenie wszystkiego w akwarium jest po funkcji, która definiuje działanie wszystkiego. Może w Hollywood jest tak, że funkcja zanim zostanie wywołana musi być wpierw opisana?

Linia 97 to utworzenie tablicy o nazwie `fish`.

Linia 99 wygląda znajomo. Tu jednak, w odróżnieniu od pętli, ustalane są warunki początkowe. Start akwarium, gdzie pojawiają się ryby po uruchomieniu programu.

```
 97   fish = {}

 98

 99   For k = 1 To 5

100

101           ; init fish!

102           ; x, y: fish position

103           ; zx, zy: x and y speed

104           ; xb, yb: x and y borders

105

106           fish[k] = {x = Rnd(900), y = Rnd(900), c = 1, xb = Rnd(900),
      yb = Rnd(900),

107                   zx = IIf(Rnd(2) = 1, -4 - Rnd(4), Rnd(4) + 1), zy =
      IIf(Rnd(2) = 1, -3 - Rnd(3), Rnd(3) + 1),

108                   width = GetAttribute(#SPRITE, k, #ATTRWIDTH), height
      = GetAttribute(#SPRITE, k, #ATTRHEIGHT)}

109

110           fish[k].frame = IIf(fish[k].zx < 0, 10, 0)

111   Next

112

113   LeftMouseQuit(True)

114
```

To jest trudny fragment kodu dla mnie (na przykład jest wprowadzone jakieś `IIf`) nie rozumiem go jeszcze, więc nie wytłumaczę.

```
115   f$ = "Aquarium" .. Rnd(5) + 1 .. ".jpg"

116

117   LoadBGPic(2, f$)

118

119   DisplayTransitionFX(2, #RANDOMEFFECT, #RANDOMEFFECT, #FASTSPEED)

120

121   PlaySample(1,0)

122

123   SetInterval(1, p_MainLoop, 1000 / 25)

124

125   Repeat

126           WaitEvent

127   Forever

128
```

Linia 113 pozwala kliknięciem myszy zakończyć działanie skryptu.

Linia 115 to zmienna `f` (prawdopodobnie skrót od „file” czyli plik), która będzie pobierać wartości typu string (czyli tekstowe), dlatego ma na końcu symbol `$`. Będzie wybierała spośród plików z Aquarium i .jpg w nazwie. `Rnd` (ang. „random”) zapewni, że ten wybór będzie losowy. Masz tam cyfrę 5, ale obrazków jest 6, bo jest 6 takich plików – Aquarium.jpg, Aquarium1.jpg i tak dalej aż do Aquarium5.jpg.

Jak może pamiętasz plik Aquarium.jpg jest zrzutem ekranu z wygaszacza, więc przez to że zawiera animowane elementy nie może być losowany jako tło. Dlatego dane jest `+ 1 ...`. Dalej jest `LoadBGPic(2, f$)` to ciąg dalszy, ładuje wylosowany obrazek. Nie wiem co oznacza 2. Nie rozumiem tego (jeszcze) jak to działa, ale to odsiewa plik Aquarium.jpg (ten bez numeracji).

Linia 119 odpowiada za efekty przejścia. Nie wiem co oznacza 2, ani dlaczego domyślna wartość efektu `#RANDOMEFFECT` jest podana dwukrotnie. Przejście jest ustawione na największą szybkość zamiany.

Linia 121 odgrywa muzyczkę. Nie wiem co oznaczają wartości (1,0). Mamy tylko jeden dźwięk zdefiniowany w linii 10 jak SAMPLE 1 i przypuszczam, że to 1 oznacza jego odgrywanie. Na 0 nie mam pomysłów. Nie ma sytuacji, że on nie jest odgrywany.

`SetInterval` ustawia parametry pętli `p_MainLoop()`. Bez podania parametrów zajęła by całą moc procesora. Wartość 1000 odpowiada jednej sekundzie, 25 tutaj oznacza 25 wywołań na sekundę. `Repeat` określa, że program wykonuje się w kółko. `WaitEvent` oznacza gotowość na odebranie zdarzenia (np. wciśnięcie klawisza Esc), `Forever` oznacza, że jeśli nic się nie wydarzy, to program wykonuje się w nieskończoność.

Tu jest koniec programu.

Podsumowanie projektu:

- przygotowano elementy graficzne (4 sprajty i 5 teł, to nie są brushe) i dźwiękowe (jeden) niezbędne do wygaszacza
- napisano funkcje tworzące życie w akwarium
- sprajty zostały powielone z przeskalowaniem, co niewielkim nakładem pracy dało więcej różnorodności

Wygaszacz stał się ważną lekcją, bo nauczył cię rzeczy takich jak:

- tworzenia okna programu z warunkiem jego zamknięcia
- zwracanie uwagi na formatowanie kodu
- pokazał pętlę i tablicę, będące sercem programu
- pokazał w działaniu użyteczne funkcjonalności: tworzenie animacji, losowość, instrukcje warunkowe (”jeśli coś-tam – to coś-tam")

Dobra wiadomość jest taka, że wytłumaczone rzeczy nie będą tłumaczone powtórnie. Jak na taki drobiazg jak wygaszacz dużo czytania, ale zrozumienie tej lekcji zaprocentuje ci w kolejnych. Kolejna będzie lżejsza i oparta na krótszym kodzie, ale żeby przejść do niej musisz zrozumieć tę przynajmniej do linii 34.

Opisy są wnikliwe bo to początki. W kolejnych przykładach będę omawiał krytyczne dla nauki fragmenty kodu, wklejanie całości jest niemożliwe. Póki kod jest krótki to to robię, ale widziałem źródła gry napisanej w Hollywood, które miały kilkanaście tysięcy linii kodu, to tutaj się nie wydarzy.

Cracktro„Fairlights”

Słowo „cracktro” oznacza krótkie demo. Jest skróconą formą od „crack intro”. Pamiątka z czasów, gdy na złamanych przez piratów grach przedstawiały się grupy hakerskie. Nieraz to były reklamy BBS-ów, ale prawie zawsze oprócz tekstu były jakieś efekty, animacje. Z czasów Atari 8 bit pamiętam to jako coś, co próbowało zabić czas jak wgrywała się gra z taśmy magnetofonowej. „Crack intro” rozwinęło się w demoscenę.

Hollywood ma dołączone te małe dziełka, które zostały przepisane z oryginałów. Pokażę jedno jako przykład. Wybrałem cracktro grupy „Fairlights” do gry „Pinball Dreams”. Jedyne kryterium to mała objętość kodu.

```
 1  /*****************************************************************
 2  **                                                             **
 3  ** Name:        Fairlight generic cracktro                     **
 4  ** Author:      Mr.X of Barracuda                              **
 5  ** Version:     1.0                                            **
 6  ** Date:        13.01.10                                       **
 7  ** Interpreter: Hollywood 4.5                                  **
 8  ** Licence:     Sample program for Hollywood                   **
 9  ** Function:    Hollywood port of this nice cracktro           **
10  **                                                             **
11  ** Notes:       This cracktro was used in dozens of Fairlight  **
12  **              releases. I like its dark atmosphere a lot.    **
13  **                                                             **
14  ** History:                                                    **
```

```
15  **                                                          **
16  ** 1.0: (13.01.10)                                          **
17  **                                                          **
18  ** - initial release                                        **
19  **                                                          **
20  ***************************************************************/
21
22  /*
23  ** Important! Check if the used Hollywood version is at least
24  ** version 4.5!
25  */
26  @VERSION 4,5
27
28  /*
29  ** this cracktro actually uses hi-res
30  */
31  @DISPLAY {Width = 640, Height = 505}
32
```

Na razie nie ma tu nic nowego, tylko wstęp do programu, analogiczny do tego, który poznałeś przy okazji wygaszacza. Jest więc długaśny komentarz, wyczerpująco opisujący programik, jest zdefiniowana wielkość okna. Dalej już będzie ciekawiej, bo dojdą nowe rzeczy, głównie związane z tekstem.

```
33
34  /*
35  ** our data files
36  */
37  @BRUSH 1, "font.iff"
38  @BGPIC 2, "fairlight.iff"
39  @SAMPLE 1, "fairlight.wav"
40
41  /* make the text */
42  Function p_MakeText()
43
44     Local t$ =
45     "                  F  A  I  R  L  I  G  H  T                " ..
46     "                ----------------------------               " ..
47     "                                                           " ..
48     "                     PROUDLY PRESENTS :                    " ..
49
50     "           P  I  N  B  A  L  L    D  R  E  A  M  S         " ..
51     "         -------------------------------------------        " ..
52     "                                                           " ..
53     "Supplied and cracked by Black Shadow (Welcome back dude!Thanx!)".
```

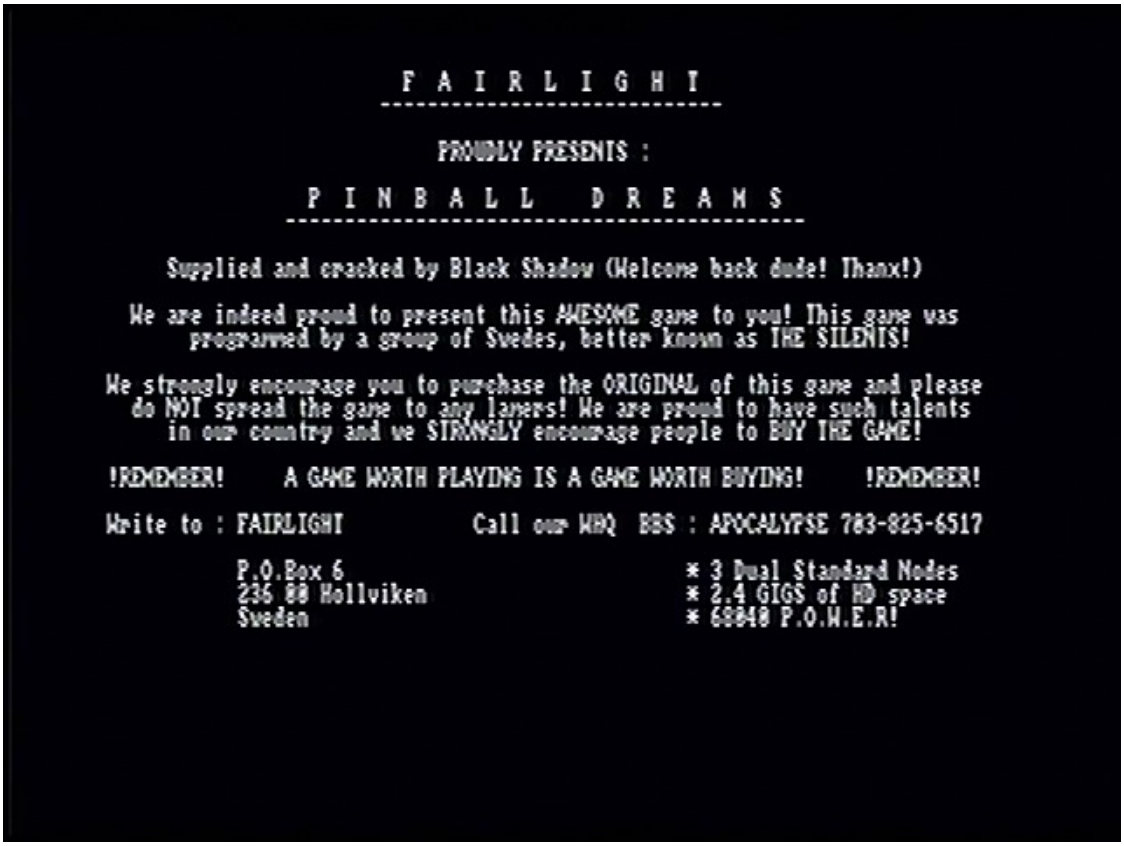
Powyżej obrazek fairlight.iff będący pierwszą rzeczą pokazująca się po uruchomieniu.

Linie od 34 do 39 to załadowanie plików. Jest ich tylko trzy: czcionka, obrazek tytułowy i dźwięk na 8 sekund. W liniach od 43 do 53 (także w 68) pousuwałem spacje, żeby nie zaburzyć czytelności przez łamanie wiersza.

Kod od 54 do 69 linii nie pokazałem, bo jest tak szeroki, że łamanie po wklejeniu tutaj kompletnie zabrało by sens prezentacji. Jak to wyszło na ekranie po uruchomieniu jest widoczne na kolejnej stronie. Rok 1992, oba graby z prawdziwej Amigi... No ale nie rozklejajmy się we wspomnieniach.

W linii 42 jest tworzona funkcja, która pobiera tekst bezpośrednio z pliku skryptu (linia 44). Sam tekst żeby był poprawnie pobrany musi być sformatowany w użyciem apostrofów. To znaczy musi wyglądać tak ‘Tekst’ albo tak "Tekst"; nie „Tekst” (ten ostatni to cudzysłów drukarski).

Czy coś rzuciło ci się w oczy? Powinno. Tekst nie jest sformatowany poprzez wycentrowanie, tylko z użyciem spacji. Wszystkie znaki spacji z obu stron napisów, w każdej linii, przełożyły się na odstępy od lewej i prawej strony ekranu.



Czy nie można było tego po prostu wycentrować? Zwróć uwagę, że centrowaniem nie dało by się uzyskać takich wcięć i odstępów jak w dolnej części tekstu. Nakład czasu na opracowanie eleganckiego rozwiązania często okazuje się niewart wysiłku. W końcu tu chodzi tylko o kilkanaście linijek napisów.

```
68      "    Sweden                                  * 68040 P.O.W.E.R!   "

69

70      CreateBrush(2, 74 * 8, 24 * 8)

71      SelectBrush(2)

72
```

Linia 68 to koniec napisu. Co się dzieje od linii 70? Tworzony jest brush nr 2 o szerokości 74 pikseli i wysokości 24 pikseli, który jest powielony 8 razy.

```
73      For Local k = 0 To StrLen(t$) - 1
74
75          DisplayBrushPart(1, (Asc(MidStr(t$, k, 1)) - 32) * 8, 0, x, y,
    8, 8)
76
77          c = c + 1
78
79          If (k <> 0) And (c % 74 = 0)
80              x = 0
81              y = y + 8
82          Else
83              x = x + 8
84          EndIf
85      Next
86
87      EndSelect
88
89      ; scale to hi-res
90      ScaleBrush(2, "100%", "200%")
91
92  EndFunction
```

Linia 71 to wybranie tego brusha, tak sekwencja kończy się w linii 87 poleceniem `EndSelect`. W środku jest pętla (pętle zazwyczaj są od `For` do `Next`).

SZKIC

```
 93
 94   p_MakeText()
 95
 96   PlaySample(1)
 97
 98   ; well the rest is actually very simple stuff :-)
 99
100   DisplayTransitionFX(2, {Speed = 2, type = #CROSSFADE})
101   Wait(2000, #MILLISECONDS)
102   DisplayTransitionFX(1, {speed = 5, type = #CROSSFADE})
103
104   DisplayBrushFX(2, #CENTER, #CENTER, {Type = #CROSSFADE, Speed =
      #SLOWSPEED})
105   WaitLeftMouse
106   CreateBrush(3, 640, 505)
107   DisplayBrushFX(3, #CENTER, #CENTER, {Type = #CROSSFADE, Speed =
      #SLOWSPEED})
108   Wait(25)
109   End
```

Instalacja i konfiguracja

Nadszedł dzień, kiedy doszła fizycznie płyta CD z Hollywood. Nabywca wersji fizycznej nie dostaje dostępu do wersji elektronicznej ani klucza (przynajmniej ja nie dostałem). Przy okazji wspomnę, że kupując wersję uaktualnienia poprzedni klucz przestaje działać, ani starsza wersja programu, ani starszy klucz nie są wymagane do instalacji wersji „update”,

/Zdjęcia płyty - zamknięte i otwarte pudełko/

/Zrzut z zawartością płyty spod Windows/

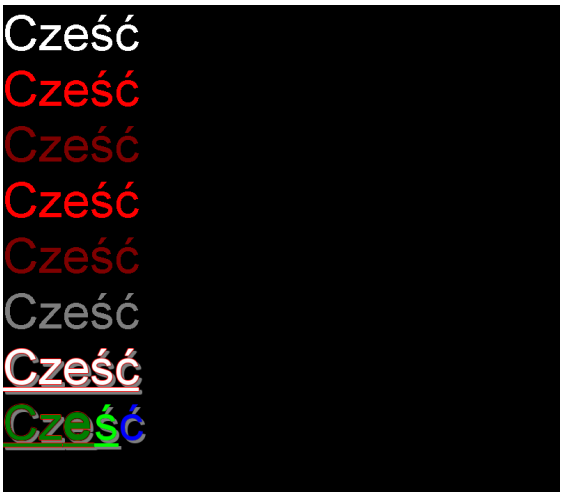
/Zrzuty z instalacji i konfiguracji spod Windows i AmigaOS 4/

/Zrzuty z MorphOS-a - FlowStudio/

/Musi być ujęte omówienie zawartości płyty. /

/Choćby lakoniczny opis rzeczy których nie ma na płycie, a można doinstalować./

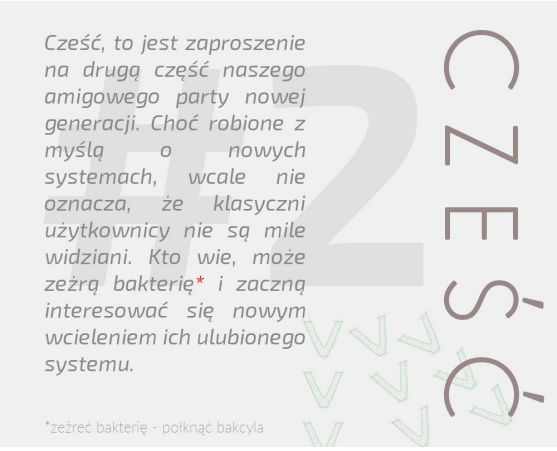
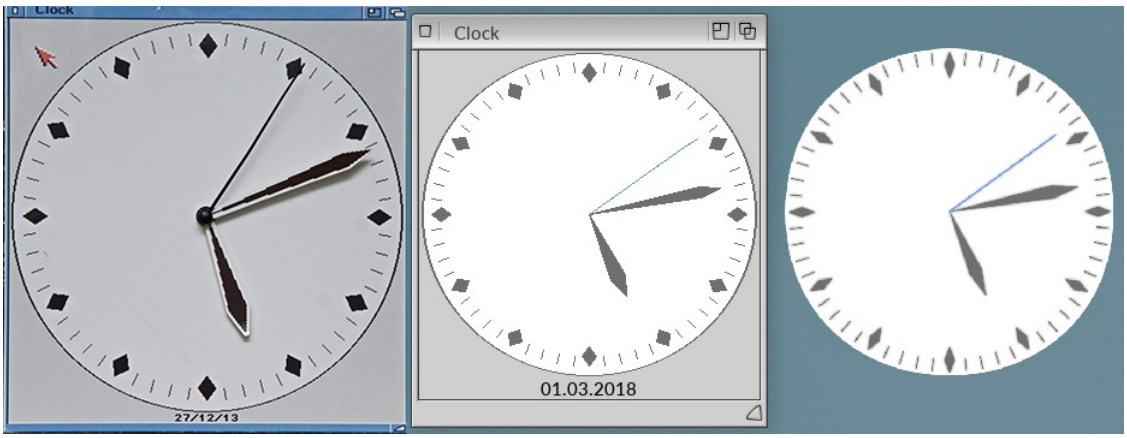
SZKIC



W oczekiwaniu na książkę możesz sięgnąć po starsze artykuły z czasopisma „Amiga NG”

Hollywood #1 – Tagać, tagać…
https://amiga.org.pl/archiwalne-artykuly--online/hollywood-1/

Hollywood #2 – E.M.M.A
https://amiga.org.pl/archiwalne-artykuly--online/hollywood-2-e-m-m-a/



Hollywood #3 – Czas
https://amiga.org.pl/archiwalne-artykuly--online/hollywood-3-czas/