Spora część ludzi nie ma w zwyczaju czytania wszelkich wstępów, gdyż szkoda im na to czasu i fatygi. W tym przypadku jednak radziłbym nie ignorować tej części, gdyż może to uchronić cię od niewłaściwego postępowania, które przy odrobinie pecha stanie się przyczyną mniejszych lub większych kłopotów.

Bez zbędnych słów przejdźmy zatem do meritum.

1

WPROWADZENIE

Nawet tysiącmilowa podróż
zaczyna się od jednego kroku.
Lao Tse

Witam serdecznie wszystkich razem i każdego z osobna. Niniejszy megatutorial ma za zadanie przybliżyć zwykłemu użytkownikowi komputera arkana programowania gier. Jeżeli czytasz te słowa, to najprawdopodobniej chcesz wiedzieć, w jaki sposób tworzone są gry, którymi pewnie nie raz bawiłeś się przez wiele godzin. I nie tylko chcesz wiedzieć, ale też umieć samemu stworzyć podobne produkcje.

Jeżeli tak, to dobrze trafiłeś.

Dla kogo jest ten tutorial?

Nie twierdzę, że potrafię nauczyć programowania gier każdego, kto chciałby taką umiejętność posiąść. Na pewno jednak będę starał się przekazywać swoją (i nie tylko swoją) wiedzę w sposób jak najbardziej jasny i klarowny. Wybaczcie mi, jeżeli nie zawsze będzie mi się to udawać.

Od czytelnika oczekuję jednak przynajmniej umiarkowanej znajomości obsługi komputera. Jeśli samodzielnie ściągnąłeś i odczytałeś plik z tym tutorialem, to twoje umiejętności są chyba wystarczające :)
Generalnie nie chciałbym, aby osoba czytająca ten tutorial robiła wielkie oczy na widok takich pojęć jak „okno”, „menu”, „przycisk” itp. Skoro spełniasz te warunki (a spełniasz? :D), możemy kontynuować.

Oprócz kompletnie zielonych w kwestii programowania gier czy programowania w ogóle, liczę że tutorial ten trafi również do programistów z pewnym, niekoniecznie dużym, doświadczeniem. Aby ułatwić im nieco życie i nie zmuszać do przeglądania całego tekstu w poszukiwaniu zagadnień, których nie znają, postaram się dać im pewne wskazówki, pozwalające szybko zorientować w swoim poziomie zaawansowania i przejść od razu do odpowiadającym im rozdziałów. Wszyscy jednak powinni doczytać tą część tutoriala do końca (zaczynając co najwyżej od następnego rozdziału).
Absolutni początkujący mogą pominąć ten fragment i iść do następnego podrozdziału.

Tak więc jeśli:

- Używałeś już kiedyś OpenGL, przejdź do Materiału Pomocniczego *Przejście z OpenGL na DirectX*

- Umiesz już programować w C++ i znasz Windows API, lecz nigdy nie kodziłeś gier i nie znasz żadnego graficznego API, przejdź do części IV, *DirectX*
- Umiesz w miarę dobrze programować w C++, lecz nigdy jeszcze nie tworzyłeś programów dla Windows, przejdź do części III, *Wprowadzenie do Windows API*
- Znasz podstawy C++ i chciałbyś kontynuować naukę, przejdź do części II, *Zaawansowane C++ i STL* lub do rozdziałów wcześniejszych, w zależności od swojego stanu wiedzy
- Nie znasz C++, ale znasz Pascala lub (lepiej) Object Pascala (Delphi), to zajrzyj do Materiału Pomocniczego *Przejście z Object Pascala na C++*
- Nie znasz żadnego języka programowania, przejdź do następnego podrozdziału :)

Wszyscy jednak powinni przeczytać do końca tą część tutoriala. Wyjdzie wam to na zdrowie :))

Kto może zostać koderem?

Chcesz więc opanować piękną acz niełatwą sztukę programowania? Chociaż prawie każdy może tego dokonać, niektóre cechy osobowości i pokrewne zainteresowania mogą być bardzo pomocne. Niektóre zaś okażą się pewną przeszkodą.

Zacznijmy od tego, że programista nie tylko pisze kod. Programista także, czy nawet przede wszystkim, **rozwiązuje problemy**. Używa on do tego narzędzi, jakimi są właśnie języki programowania.
Specyfika programowania polega na odpowiednim podejściu do rozwiązywania problemów, mianowicie na podziale ich na mniejsze fragmenty, wykonywane w określony sposób w określonym porządku – to jest właśnie **kod**. Dlatego też trzeba umieć rozbijać duże zagadnienia na mniejsze, wielki problemy na małe części. Trzeba mieć **umysł analityczny**.

Programista prawie cały czas poznaje nowe zagadnienia i techniki. Dlatego musi być **ciekawy świata** i autentycznie lubić to, co robi.

Pamiętajmy też, że koder programuje komputery, które w działaniu są przeraźliwie jednoznaczne, logiczne i przewidywalne. Należy więc umieć myśleć **logicznie** i formułować swoje koderskie pomysły w sposób jasny i jednoznaczny.

Wreszcie, należy posiadać dobrą pamięć **syntaktyczną** i **semantyczną**. Cóż znaczą te straszne słowa? :) Pamięć syntaktyczna to zdolność do jak najlepszego zapamiętywania reguł składniowych – chodzi tu oczywiście o języki programowania. Pamięć semantyczna to umiejętność przyswajania sobie znaczenia pojęć – nie można na przykład mylić procedury z klasą albo zmiennej z makrem.
Te dwa warunki są zazwyczaj najtrudniejsze do spełnienia :D

Dobry koder powinien mieć typ osobowości określany jako *teoretyczny*, a więc doceniać wartości intelektualne. Także typ *polityczny* bywa przydatny, szczególnie jako szef zespołu programistów.
OK. to tyle tytułem socjologicznej psychoanalizy :)

Dodatkowo, programista gier musi w znacznie większym stopniu niż inni koderzy znać się na matematyce i fizyce. Bez nich nie ma przecież mowy o grafice trójwymiarowej czy realistycznym symulowaniu rzeczywistości.

Zastanawiasz się teraz, czy spełniasz te warunki? Jeżeli naprawdę się zastanawiasz, to istnieje duże prawdopodobieństwo, że je spełniasz! :) Wcale nie żartuję – większość ludzi od razu odrzuciłaby je jako nieprzystające do ich osoby i powiedziała „To nie dla mnie”. Jeśli ty tak nie robisz, to jesteś usilnie proszony o dalszą lekturę :D

Natomiast jeżeli wydaje ci się, iż nie podpadasz pod jeden tylko warunek, to twoja decyzja powinna zależeć od tego, jakiego warunku nie spełniasz. W przypadku gdy chodzi o logiczne (racjonalne) myślenie, to nie martw się – znam kilku koderów, którzy doskonale obywają się bez niego :)
Bez umysłu analitycznego też da się żyć, ale z początku może być ci trudno samodzielnie wymyślać algorytmy. Mówi się, że wszystkiego można się nauczyć – nie wiem, czy zasada ta działa i w tym przypadku, ale zawsze możesz spróbować…
Niestety, brak dobrej pamięci syntaktycznej i semantycznej w zasadzie dyskwalifikuje kandydata na programistę. Jeśli mimo wszystko chcesz spróbować, pamiętaj że robisz to na własne ryzyko i że ostrzegałem :)

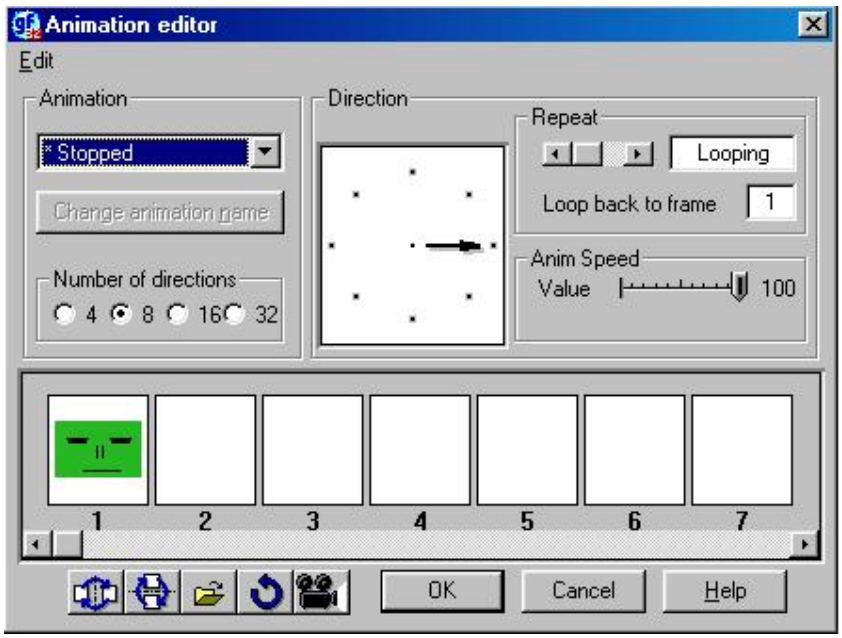
Cóż, czas na męską decyzję – czerwona czy niebieska pigułka? :) Daj sobie chwilę przerwy, zastanów, przemyśl wszystkie za i przeciw… Kiedy będziesz gotowy, to albo wciśniesz Alt+F4 i zapomnisz, że kiedykolwiek to czytałeś, albo przejdziesz dalej…

***

A więc wybrałeś czerwoną pigułkę… Zatem witaj na pustyni rzeczywistości ;)) Ale dosyć już tych żartów – niedługo zaczniemy całkiem poważną naukę i będziesz mógł zweryfikować swoją decyzję :) Zanim jednak to nastąpi, musisz otrzymać garść wskazówek, które pomogą ci przejść przez obraną drogę.
(Huh, jak to wszystko patetycznie brzmi ;D)

Clicki a sprawa polska

Swego czasu dużą popularność zyskały programy do łatwego tworzenia gier, jak Klik & Play, The Games Factory, Game Maker czy Multimedia Fusion. W założeniu mają one umożliwić kompletnym laikom tworzenie prostych gier bez konieczności programowania. Możliwe, że używałeś kiedyś tego rodzaju programów. Jeżeli tak, to jednocześnie dobrze i źle. Dobrze – gdyż przy okazji nauczyłeś się zapewne używania prostych funkcji (np. matematycznych) czy używania układu współrzędnych. Źle – bo programowanie może ci się na początku wydawać zbyt trudne i pracochłonne.



Screen 1. Okno programu The Games Factory - edytor animacji. Czyż nie wygląda żałośnie? :)
(screen pochodzi z serwisu Strefa Twórców Gier)

Zgadzam się jednak, że używanie programu Game Maker mogło ci przynieść więcej pożytku niż szkody. Zawarty w nim język skryptowy mógł pomóc nauczyć się podstaw programowania; zresztą, nawet nie używając go można się tam nauczyć tworzenia prostych algorytmów. No i nie ma tam takich ułatwień jak wbudowany ruch odbijany czy platformowy.

Jeżeli jednak chcesz na poważnie zająć się programowaniem gier, to powinieneś czym prędzej porzucić wszelkie tego rodzaju programy. Wierz mi, iż o wiele większe możliwości daje kodowanie. No i przynosi znacznie większą satysfakcję :)

Treść tutoriala

Zobaczmy więc, jak wygląda droga, który prowadzi nas do upragnionego celu, czyli nauczenia się programowania gier.

Zawartość kursu

Najpierw musisz poznać programowanie w ogóle. Nauczysz się kodować w języku C++, zaczynając od najbardziej elementarnych programów po coraz trudniejsze, poznasz większość możliwości tego języka.
Po takiej solidnej porcji wiedzy będziesz mógł przejść do pisania programów dla środowiska Windows. Nie będziemy się dokładnie wgłębiać we wszystkie aspekty tego niezwykle obszernego zagadnienia – poznasz to, co będzie ci niezbędne do przejścia do następnej fazy, czyli nauki DirectX.
Wtedy wreszcie zacznie się prawdziwe programowanie gier. Poznasz sposób tworzenia trójwymiarowych i dwuwymiarowych scen, stosowania oświetlenia, tekstur i wielu innych technik.

Niektórzy mogą się kłócić z zaproponowanym tu zestawem narzędzi (C++, Windows i DirectX). Odpowiem im, że tak naprawdę jest to sprawa drugorzędna – o wiele ważniejsze niż znajomość samego języka czy graficznego API jest orientowanie się w koncepcjach: programowania w ogóle i grafiki w szczególności.

Układ graficzny i zapis składniowy

To jest właśnie droga, którą będziemy podążać. Zanim jednak w nią wyruszysz, poznaj niektóre z drogowskazów, które dla ciebie zostawiłem. Mówiąc wprost, chodzi o układ graficzny tekstu - niektóre elementy są w nim bowiem specjalnie wyróżnione:

```
Tak będą oznaczane fragmenty kodów źródłowych (tzw. listingi).
```

To będą ważne uwagi oraz pojęcia i definicje, które powinieneś sobie przyswoić.

Tak będę zwracał na potencjalne niebezpieczeństwa (głównie chodzi o popełnianie trudnych do wykrycia błędów).

Te ramki to dodatkowe informacje dla zainteresowanych, opisujące dokładniej omawiane sprawy. Początkujący mogą je pominąć, szczególnie przy pierwszym czytaniu.

A to będą mniej lub bardziej związane z tematem dygresje lub inne, luźne uwagi.

Niekiedy będę też przedstawiał składnię jakichś konstrukcji w języku programowania. W takich fragmentach wyrazy pisane *kursywą* mają być zastąpione przez konkretne nazwy, zaś te napisane normalnie powinny pozostać bez zmian.
Pogrubione nawiasy kwadratowe **[ ]** zawierają natomiast nieobowiązkowe składniki danej konstrukcji, a symbol wielokropka . . . oznacza dalszy ciąg listy podobnych elementów.

Podsumowania, pytania i zadania

Na końcu większości rozdziałów zamieszczał będę krótkie podsumowanie oraz (niestety :D) ćwiczenia do wykonania dla ciebie. Cóż, prace domowe są nieuniknione :)

Podsumowanie

To dopiero pierwszy rozdział pierwszej (czy raczej zerowej) części, więc siłą rzeczy nie przedstawiłem tu jeszcze żadnych konkretnych informacji. Przypomnijmy jednak, co ciekawego udało się nam razem dokonać.

Przede wszystkim dowiedziałeś się, że ten tutorial przeznaczony jest dla osób pragnących nauczyć się programowania gier. Podjąłeś brzemienną w skutkach decyzję o podjęciu tego wyzwania – brawo :) Przeczytałeś też kilka uwag na temat programów do tworzenia gier i wreszcie poznałeś pokrótce treść i formę tutoriala.

Z następnego rozdziału dowiesz się natomiast, z jakich innych źródeł informacji powinieneś koniecznie korzystać podczas tego kursu, jak i w czasie codziennego kodowania.

Zapraszam więc wgłąb króliczej nory…