Kompozycja obrazu Porter-Duff - część 1

Hans-Joerg Frieden

(c) Polski Portal Amigowy (www.ppa.pl)

Wstęp

Chcę przygotować serię krótkich artykułów poświęconych kompozycji obrazu Porter/Duff, w których zaprezentuję jak działa, jak jest wykorzystywana przez AmigaOS 4.1 oraz, być może, poruszyć kilka kwestii związanych z Cairo. Wydaje mi się, że wiele osób ma problem ze zrozumieniem idei kompozycji Porter/Duff, a także sposobu jej wykorzystywania.

Ten artykuł stanowi część pierwszą. Skupię się w nim na podstawach, tzn. do jego zrozumienia nie będzie wymagana wiedza programistyczna. Rzucimy okiem co to jest i jak się z tego korzysta.

Wprowadzenie

Kompozycja obrazu Porter-Duff (dalej określana jako KOPD) to technika wykorzystywana w produkcji filmów, której zadaniem jest pogrupować wygenerowane obrazy w warstwy lub połączenie wygenerowanych komputerowo obrazów z zapisem z taśmy filmowej. Po raz pierwszy wykorzystano ją podczas produkcji filmu Star Trek II przez LucasFilm, a obecnie stała się dość popularna na rynku komputerów desktop.

Krótko mówiąc - KOPD "komponuje" obraz źródłowy z obrazem przeznaczenia. Działanie jest zbliżone do pracy blittera - zasadniczo operacja ta zastępuje piksele w miejscu docelowym, w zależności od pikseli u źródła (i celu). Niektóre blittery obsługują nawet coś, co nazwano "transparents blits", tzn. zastępują tylko te piksele w miejscu docelowym, które nie są zerem (tj. nie są przezroczyste) w źródłowej bitmapie.

Blitter znajdujący się w Amidze klasycznej był o tyle lepszy niż większość obecnie stosowanych, teoretycznie nowszych, odpowiedników, że potrafił obsługiwać trzy niezależne części (zazwyczaj określane jako źródło, miejsce docelowe i maska) i co więcej, korzystał z wzornika (stencil), aby wycinać fragmenty obrazu źródłowego i zastępować je przez docelowe (oczywiście wśród wielu innych rzeczy - blitter Amigi potrafił w zasadzie wyliczać funkcje binarne oparte na funkcjach w postaci kanonicznej zwane minterms).

W pewnym sensie, KOPD to w porównaniu z blitterem Amigi jeden krok dalej. Zamiast zezwalać na pracę na wzornikach w trybie pojedynczego bitu, KOPD umożliwia programiście/grafikowi przypisanie wartości "wypełnienia" dla każdego piksela. Co to oznacza? Załóżmy, że posiadamy bitmapę, która przedstawia namalowany na niej czerwony trójkąt. Na jego krawędzi, jeśli jest to figura matematycznie poprawna, każdy piksel byłby tylko częściowo pokryty kształtem trójkąta. Reszta piksela byłaby niepokryta.

Gdybyśmy wmalowali ten trójkąt w normalną bitmapę, uzyskalibyśmy coś, co określa się mianem "postrzępienia" (z ang. "jaggies") lub "aliasingiem" - po prostu piksel nie może być tylko w części pomalowany kolorem. Musi być albo pokolorowany (w tym przypadku czerwony) lub niepokolorowany. Efekt aliasingu staje się bardziej wyraźny wraz ze zmniejszaniem rozdzielczości, ale nawet w wysokich rozdzielczościach nadal jest widoczny. Są sposoby na to, aby się tego ustrzec (zwane "anti-aliasingiem"), lecz zazwyczaj działają one dobrze jedynie dla podstawowych rodzajów kształtów, takich jak linie, łuki i tym podobne (AmigaOS, na przykład, wykorzystuje wspomniane techniki do "wygładzania czcionek").

Jeśli założymy, że każdemu pikselowi naszego kształtu przypisana jest także wartość, określająca stopień jego wypełnienia (tj. w ilu procentach pokryty jest on kolorem obrazu a w ilu przezroczysty), wówczas możemy wykorzystać tę informację, aby wymieszać czerwony trójkąt z kolorem, który jest na naszej bitmapie docelowej. Na przykład, jeśli chcemy wkomponować ten trójkąt w białe tło, możemy określić kolor każdego piksela przy pomocy wartości wypełnienia piksela źródła oraz koloru miejsca docelowego. W ten sposób, piksele na krawędzi trójkąta rozjaśnią się, tj. informacja o wypełnieniu pozwoli nam oszacować wpływ wartości koloru ze źródła i miejsca docelowego na wyliczenie najbardziej optymalnego koloru dla piksela. Ten proces zazwyczaj określa się mianem alpha blending, jako że wartość wypełnienia nazywana jest kanałem alpha (nieprzypadkowo, jest to również rozwinięcie litery A występującej w skrócie ARGB,

Kompozycja obrazu Porter-Duff - część 1

Hans-Joerg Frieden

(c) Polski Portal Amigowy (www.ppa.pl)

oznaczającym tryby koloru). "Alpha blending" to jedynie wybrany przykład KOPD, która obejmuje znacznie więcej.

Na początek - nie istnieje żadna przeszkoda, uniemożliwiająca zastosowanie wyżej opisanego procesu i przypisanie wartości wypełnienia pikselom z bitmapy docelowej. Nie musimy się ograniczać do pracy wyłącznie z całkowicie nieprzezroczystymi miejscami docelowymi. Może to nie jest ważne, ale wynik operacji komponowania może stanowić podstawę dla kolejnej operacji. Dla przykładu załóżmy, że chcemy opisać czerwony trójkąt anti-aliasowanym tekstem. Jak wspomniałem wcześniej, informacja wypełnienia dla łuków i linii daje się w prosty sposób wyliczyć. Czcionki składają się właśnie z takich kształtów, więc wartość wypełnienia jest prosta do uzyskania. Czcionka może być więc wkomponowana w nasz czerwony trójkąt bez większego kłopotu.

Rozważmy inny przykład. Chyba każdy spotkał się z często stosowanym w filmach efektem, który sprawia wrażenie jakbyśmy patrzyli przez teleskop. Aby stworzyć takie wrażenie, zazwyczaj przycina się wyświetlany obraz do kształtu koła. Podobnie rzecz się ma w przypadku efektu graficznego, dającego wrażenie oglądania filmu przez duże, wycięte w formie szablonu litery (na przykład tytuł filmu). O ile łatwo stworzyć taki "wycięty" okrąg w przypadku efektu teleskopu, o tyle stworzenie "wyciętej" czcionki może być już nieco trudniejsze. Co więcej, możemy nie chcieć tworzyć dodatkowej bitmapy, lecz użyć już istniejącej, która zawiera wyrenderowany tekst oraz wartości wypełnienia. W tym przypadku, wykorzystanie zwykłego "alpha blendingu" nie przyniesie zamierzonego efektu - moglibyśmy otrzymać dokładną odwrotność tego, co zamierzamy osiągnąć: czarne litery na wierzchu obrazu docelowego.

KOPD pracuje z wykorzystaniem operatorów. Operator to reguła mówiąca o tym, w jaki sposób wyliczyć piksel wynikowy i wartość wypełnienia opierając się na obrazie źródłowym i docelowym. Te reguły określają sposób, w jaki bitmapy źródłowa i docelowa wzajemnie na siebie oddziałują. Najczęściej stosowanym operatorem jest Source_Over_Destination (źródło nad przeznaczeniem - opisaliśmy go powyżej). Szczegółowy opis operatorów leży poza zakresem tematycznym tego wprowadzenia. Więcej informacji na ten temat można znaleźć w dokumentacji KOPD, którą można pobrać stąd.

Zaimplementowana w AmigaOS 4.1 KOPD potrafi więcej niż "tylko" komponowanie. Dostępne są dwa podstawowe tryby działania: tryb Blit oraz tryb trójkąta. Tryb Blit działa jak blitter, tj. wskazujesz dwie bitmapy (źródłową i docelową), a silnik kompozycji nakłada źródło na przeznaczenie przy pomocy określonego operatora i określonego zestawu współrzędnych. Dodatkowo może opcjonalnie przeskalować bitmapę korzystając z dowolnych parametrów, jak również przyciąć ją do określonego prostokątnego kształtu na obrazie przeznaczenia.

Tryb trójkąta jest znacznie bardziej elastyczny. Zamiast docelowych współrzędnych, programista może wskazać silnikowi kompozycji listę trójkątów. Każdy trójkąt składa się z trzech wierzchołków, z czego każdy z nich posiada własne współrzędne (włączając w to współrzędną głębokości na potrzeby zachowania perspektywy). Każdy trójkąt jest nakładany przy pomocy operatorów KOPD, lecz programista posiada cały czas kontrolę nad ich rozmieszczeniem i przypisaniem. W prosty sposób można, np. obracać, ścinać lub zniekształcać bitmapę. Efekty typu osławiony "desktop cube" (przełączanie się pomiędzy programami w efektownym, trójwymiarowym oknie), okna "jiggly" (pochylone, pozwijane okna) i inne nie stanowią już problemu i bez kłopotu można je implementować.

Cairo

Cairo Graphics API to API grafiki wektorowej, która pozwala programiście w pełni skorzystać z dobrodziejstw grafiki nierastrowej i która posiada wbudowane procedury komponowania obrazu dla niektórych funkcji (np. anti-aliasingu tekstu i kształtów). Obsługuje wiele urządzeń wyjścia, tzn, że ten sam kod, który odpowiedzialny jest za kreślenie np. tekstu w procesorze tekstu może również generować wynik na drukarkę przez PostScript lub do pliku PDF.

Cairo działa w oparciu o ścieżki. Ścieżka to dowolnie określony kształt będący zbiorem połączonych linii i łuków. Ścieżka może być zamknięta lub nie. Po jej zdefiniowaniu może stanowić ona "wzornik" lub "wypełnienie". W pierwszym

Kompozycja obrazu Porter-Duff - część 1

Hans-Joerg Frieden

(c) Polski Portal Amigowy (www.ppa.pl)

przypadku, wzdłuż stworzonej ścieżki przesunie się wyimaginowany pędzel/wzornik, nakładając kolor na tło znajdujące się pod ścieżką. Druga z opisywanych możliwości oznacza wypełnienie obszaru ograniczonego ścieżką przy pomocy koloru lub wzorcem. Stworzone elementy mogą być następnie ze sobą skomponowane.

Cairo i kompozycje - Aplikacje

Pytanie, które się nasuwa, to: po co to wszystko? Z całą pewnością przezroczyste okna i zaokrąglone rogi to niezły bajer, ale to żadna kosmiczna, zrywająca czapki z głów technologia. W takim razie przyjrzyjmy się kilku przykładowym aplikacjom, które mogą z niej skorzystać.

Rzecz oczywista: Mozilla Firefox

Od wersji 3.0 Mozilla Firefox całkowicie została przepisana pod Cairo. Bez cienia wątpliwości, szybka i wydajna, a zarazem wspomagana sprzętowo implementacja Cairo to dla AmigaOS 4.1 milowy krok zbliżający ten system do tego, aby wspomniana przeglądarka internetowa zawitała pod nasze strzechy. Poza Firefoksem jest jeszcze kilka innych aplikacji, które wykorzystują Cairo, jak chociażby OpenOffice, czy ClassPath.

Lepsze GUI

Ogólnie rzecz biorąc, wykorzystanie Cairo i KOPD pozwala na posiadanie lepszego interfejsu użytkownika. Powód jest prosty - z racji filozofii swojego działania, polegającej na rysowaniu na bitmapach nie wyświetlanych na ekranie, a następnie komponowaniu ich do widocznego okna, opisywana technika jest pozbawiona migotania - nie da się zauważyć, że coś jest rysowane. Po prostu widzisz końcowy efekt. Sprawia to, że znika problem uciążliwego migotania podczas rysowania animowanych elementów.

Co więcej, dzięki temu można zrobić większy użytek z animacji w interfejsie użytkownika. Wiele osób może to traktować jako zwykłe bajery, lecz praktycznie rzecz biorąc wszystkie nowoczesne interfejsy użytkownika robią z tego użytek. Załóżmy, że mamy przycisk "ikonifikacji" na belce okienka i w niego kliknęliśmy. Jeśli masz kilka ikon na blacie, zaczniesz szukać tej, do której zikonifikowało się twoje okno. Niemniej jednak, jeśli efekt ikonifikacji polega na "przeskalowaniu/przekomponowaniu" okna w ikonę, natychmiast zauważysz, gdzie się ona znajduje.

Animacja pozwala także bardziej uwidocznić jakiś element. Mając szereg obiektów w oknie, wykorzystując skalowalne GUI (Cairo) i kompozycję, możesz przybliżyć obiekt, nad którym znajduje się kursor, dzięki czemu, być może, ujawni on dodatkowe informacje. Jeśli elementy interfejsu użytkownika określone są jako grafika wektorowa, możemy je dowolnie skalować - powiększanie przyciągnie do nich uwagę, pomniejszanie zaś zapewni nam więcej miejsca na ekranie, gdy elementów przybędzie.

Kompozycja zapewnia także użytkownikom większą kontrolę nad wyglądem ich interfejsu. Z reguły, skórki dla GUI składają się z zestawu bitmap, które są zazwyczaj ładnie do siebie dopasowane. Dla przykładu, element "checkboksa" jest zazwyczaj zrobiony w taki sposób, że pasuje do wzorku tła. Kompozycja pozwala grafikowi lub projektantowi GUI na zdefiniowanie kanałów przezroczystości dla elementów graficznych GUI, a program sam może te elementy sobie poskładać. Sprawia to, że GUI jest nie tylko bardziej elastyczne, ale pozwala użytkownikowi na dowolną zmianę określonych elementów (na przykład tła). Zestaw Qt4 autorstwa TrollTech robi z tego spory użytek pozwalając na zmianę wyglądu aplikacji za pomocą arkuszów stylów.

Wpływ na AmigaOS 4.1

Oczywiście my jeszcze do tego nie doszliśmy, ale AmigaOS 4.1 i nowe technologie w nim wprowadzone stanowią podwaliny dla nowszych, bardziej zaawansowanych interfejsów użytkownika, które możemy mieć w przyszłości. Dla twórców oprogramowania, Cairo i KOPD daje ogromne możliwości polepszania wyglądu ich aplikacji. Czy będą "odbajerzone"? Bardzo prawdopodobne, że tak. Ale jeśli spojrzysz z innej strony, to sama nazwa "interfejs użytkownika" już sugeruje, że są to bardzo ważne elementy systemu. W końcu jest to bezpośredni łącznik pomiędzy Tobą, czyli użytkownikiem, a systemem. Dobry interfejs użytkownika jest bardzo ważną częścią pracy z komputerem.

Tłumaczenie na podstawie oryginału: Sebastian Rosa, korekta Konrad Czuba.