

Wektory, macierze, przekształcenia. To co najważniejsze już mamy. Jednak aby was pognębić jeszcze trochę to muszę wam napisać, że to jeszcze nie wszystko ;). Aby cokolwiek zobaczyć na ekranie w przestrzeni 3D i aby miało to jakiś sens będzie nam jeszcze potrzebna wiedza o kilku specjalnych przekształceniach. W lekcjach o Direct3D mówiłem już o nich kilka razy, w tutorialach a zwłaszcza w tym o vertex shaderach też sporo wspominałem. Jak łatwo się domyślić dzisiaj będziemy mówić od trzech przekształceń, które pozwolą przenieść nam wytwory naszej wyobraźni na ekran komputera i rozkoszować się nimi do woli. Mowa będzie o przekształceniach świata, widoku i projekcji:

- Przekształcenie świata

Kiedy projektujemy pojedynczy model (nie całą scenę) w jakimś programie graficznym to często tworzymy model w środku układu współrzędnych. Większość programów domyślnie tak robi, bo po prostu łatwiej i prościej jest takim obiektem manipulować i w wielu przypadkach podawać różne jego wymiary. Kiedy tworzymy scenę, bierzemy nasze wcześniej przygotowane modele i umieszczamy je gdzieś w przestrzeni aby stworzyły nam (nie zawsze) coś sensownego. Często często okazuje się, że modele te są albo za duże albo za małe. Aby scena wyglądała jako tako nieraz trzeba także model obrócić albo przesunąć. Cała ta procedura dostosowywania modeli do sceny będzie nam odzwierciedlać w grafice 3D tak zwane przekształcenie świata. Każdy, nowo tworzony model w programie graficznym posiada tak jakby swój własny układ współrzędnych i jest on umieszczony w jego środku (przeważnie), ze względu o których napisałem powyżej. Jeśli model umieszczamy w scenie i coś tam z nim robimy to jego współrzędne niewątpliwie się zmieniają. Weźmy najprostszą kulę. Jeśli stworzymy ją w programie to najkorzystniej będzie umieścić jej środek w punkcie (0, 0, 0) prawda? Określenie jej promienia to już najprostsza i najwygodniejsza sprawa. Teraz wstawiamy naszą kulkę na scenę, ale nie ma ona leżeć na środku ale na przykład gdzieś zupełnie z boku, tak że prawie jej nie widać. Należy wszystkie punkty tworzące kulę przesunąć w odpowiednie miejsce (tak samo jak obrócić czy przeskalować). Można to zrobić w pętli, biorąc każdy punkt i wykonując na nim pożądane operacje, ale czy potrzebnie? Nie na darmo mówiliśmy w poprzednim tutorialu o przekształceniach podstawowych. Jak wiemy możemy sobie wszystkie współrzędne punktów pomnożyć przez macierze, które dokonają nam na nich przekształceń. Co lepsze, możemy sobie macierze przekształceń poskładać, tak aby kilka przekształceń wykonać za jednym zamachem. Większość pakietów graficznych oferuje nam do dyspozycji pewne mechanizmy, które w bardzo łatwy sposób umożliwiają nam dokonanie takich operacji.

Każdemu obiektowi na scenie możemy stworzyć taką macierz i przez proste wywołanie funkcji możemy się rozkoszować ruchem naszego obiektu po scenie. Każdy punkt obiektu zostanie automatycznie przez pakiet przekształcony przez zadane przez nas przekształcenie praktycznie bez naszego udziału i zostanie umieszczony we wskazanym przez nas miejscu, oczywiście jeśli chcielibyśmy robić to sami to też nic nie stoi nam na przeszkodzie. Ogólnie więc mówiąc przekształcenie świata polega na tym, że współrzędne lokalne obiektu (te, w których został on stworzony) zostają przekształcone na współrzędne świata, które obowiązują na naszej scenie tak, żeby obiekt ten mógł znaleźć się w określonym miejscu o właściwym czasie. Przekształcenie świata obejmuje trzy podstawowe operacje jakie możemy wykonać na obiekcie, czyli przesuwanie, skalowanie i obroty. Mając nasz obiekt, możemy stworzyć macierz przekształceń i pomnożyć wszystkie wierzchołki obiektu, żeby dokonać przekształcenia. W pakietach graficznych takich jak OpenGL czy Direct3D stosuje się funkcje, które pobierają taką macierz i odpowiednio przekształcają przez nią wszystkie punkty tworzące nasze obiekty. Tutaj uwidacznia się jeszcze jedna zaleta tworzenia obiektów w środku układu współrzędnych. Przy aplikowaniu im przekształcenia świata (umieszczaniu ich na scenie) najłatwiej jest określić takie przekształcenie, ponieważ po wczytaniu na scenę obiekt będzie na samym jej środku.

Aby bardziej sobie to uzmysłowić weźmy sobie mały przykład, może nie najlepszy, ale myślę, że dosyć wyrazisty. Mamy na scenie Słońce, Ziemię i Księżyc. Jak to wszystko się porusza mam nadzieję, że wszyscy wiedzą. Model każdego z obiektów mamy stworzony w ten sposób, że jego środek znajduje się w punkcie (0, 0, 0) i ma jakiś tam promień. Teraz wczytujemy je na scenę po kolei. Wyobraźmy sobie, że chcemy oglądać nasz układ patrząc na Słońce, które pozostanie nieruchome, wokół niego będzie krążyć Ziemia a wokół niej z kolei księżyc. Każdemu z obiektów musimy przypisać zatem inne przekształcenie świata. Słońca nie będziemy ruszać, bo nie będzie takiej potrzeby, więc przypiszemy mu takie przekształcenie, które nie spowoduje zmian w jego współrzędnych. Czyli musimy pomnożyć przez jakąś macierz, a jaką? Będzie to macierz jednostkowa, którą powinniście już pamiętać. Macierz taka ma wartości 1 tylko na swojej przekątnej i jak ktoś nie wierzy, że po pomnożeniu przez macierz współrzędnych nie dostaniemy tego samego to niech popatrzy:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \cdot x + 0 \cdot y + 0 \cdot z + 0 \cdot 1 \\ 0 \cdot x + 1 \cdot y + 0 \cdot 1 + 0 \cdot 1 \\ 0 \cdot x + 0 \cdot y + 1 \cdot z + 0 \cdot 1 \\ 0 \cdot x + 0 \cdot y + 0 \cdot z + 0 \cdot 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Nic się nie zmieni? Więc w naszej aplikacji zaaplikujemy obiektowi Słońca taką właśnie macierz świata. Kolej na Ziemię. Założyliśmy sobie, że ma się obracać wokół Słońca. Tutaj mamy kolejny dowód na to, że dobrze było sobie zamodelować ją w środku układu współrzędnych. Obracanie wokół środka układu mamy przecież świetnie opanowane, prawda? Wystarczy więc tylko przesunąć Ziemię na odpowiednią odległość od Słońca, zapewne przeskalować do właściwych stosunków i obrócić, nie zapominając, że Ziemia obraca się też dookoła własnej osi ;). Nic prostszego! Tworzymy naszą macierz świata dla Ziemi:

Sz - macierz skalowania Ziemi,
 Tz - macierz przesunięcia
 Rs - macierz obrotu wokół Słońca,
 Rz - macierz obrotu wokół własnej osi.

Macierz wynikowa:

$$W = Sz * Rz * Tz * Rs$$

Pamiętać musimy koniecznie o kolejności przekształceń. Dla naszego przykładu odpowiednią kolejnością będzie najpierw skalowanie Ziemi (w środku układu współrzędnych, że nie zmieniło to żadnych odległości !), obracanie wokół własnej osi, następnie przesunięcia i na końcu obrót. Gdybyśmy najpierw obrócili naszą Ziemię, wokół Słońca która znajduje się nadal w środku układu współrzędnych a potem przesunęli to Ziemia będzie się nam owszem obracać, ale w miejscu i to w dziwny i nieprzewidywany sposób ! Tak przygotowaną macierz aplikujemy naszemu obiektowi symbolizującemu zieloną planetę. Dla macierzy obrotu oczywiście z każdą liczoną klatką przygotowujemy odpowiednie kąty obrotu, które odzwierciedlą faktyczną szybkość obrotów wokół własnej osi i wokół Słońca. Nie muszę chyba mówić jak bardzo ważna jest tutaj synchronizacja czasowa, ale to temat na zupełnie oddzielny artykuł.

Ale to jeszcze nie koniec, czas na Księżyc ! Nie wiem czy on obraca się wokół własnej osi, ale dla ułatwienia ;) założmy sobie, że tak. Jakie macierze będą nam potrzebne ? Księżyc obraca się wokół Ziemi więc na pewno będzie potrzeba przesunąć go na odpowiednią odległość, założyliśmy sobie, że obraca się wokół własnej osi no i założmy jeszcze, że wokół Ziemi też, choć tutaj to mam potężne wątpliwości, no ale jak szaleć to szaleć ;) . Cały czas również obraca nam się Ziemia wokół Słońca, więc o tym też nie można zapominać. Potrzebne macierze więc:

Sk - macierz skalowania księżyca,
 Tk - macierz przesunięcia względem Ziemi,
 Rk - macierz obrotu wokół własnej osi,
 Rkz - macierz obrotu wokół Ziemi,
 Rs - macierz obrotu Ziemi wokół Słońca,
 Tz - macierz przesunięcia Ziemi względem Słońca.

No i kolej na macierz świata:

$$W = Sk * Rk * Tk * Rkz * Tz * Rs$$

Wzór strasznie skomplikowany, ale ze zrozumieniem nie powinno już być raczej wielkich problemów. Najpierw odpowiednio skalujemy (Sk) księżyc w środku układu (żeby nie pozmieniać odległości !), następnie obracamy go wokół własnej osi (Rk), potem przesuwamy na odległość, w jakiej powinien znajdować się od Ziemi (Tk). Mając taki układ możemy go obrócić wokół naszej planety (Rkz). Nie zapominamy jednak ciągle, że Ziemia obraca się wokół Słońca, więc księżyc również na tym cierpi. Musimy całość tego przesunąć na odległość Ziemi od Słońca (Tz) i obrócić (Rs). Uff ! Mam nadzieję, że wszystko zostało zrozumiane ;) . Oczywiście to co opisałem nie jest w żadnym razie układem optymalnym, to tylko ilustracja do czego może służyć macierz świata, jak taką macierz skonstruować i jak się nią posługiwać. Ale ogólną ideę mam nadzieję, że rozumiecie.

- Przekształcenie widoku

Mając gotowy przykład z naszym mini - układem planetarnym wyobraźmy sobie teraz, że siedzimy w statku kosmicznym obcych i przylatujemy sobie z jakiejś odległej galaktyki, żeby sprawdzić jak radzi sobie ta szalona rasa ludzi. Wlatujemy do naszego układu i oglądamy go sobie dokładnie z każdej strony. Do tej pory widzieliśmy wszystko z góry, wszystko było proste i ładne. Ale co będzie jeśli zapagniemy sobie na przykład usiąść gdzieś z boku i popatrzeć na to wszystko ? Ano wtedy w grafice 3D przychodzi nam z pomocą właśnie przekształcenie widoku. Cóż to jest takiego.

Ano przekształcenie widoku powoduje to, że niejako możemy umieścić obserwatora (czyli de facto nas samych :)) w naszym świecie, w dowolnym jego punkcie i będziemy mogli sobie spojrzeć w dowolną stronę naszego świata, każdemu obiektowi przyjrzeć się bliżej. Mówiąc naukowo i w odniesieniu do wierzchołków zostaną one przekształcone ze współrzędnych świata do współrzędnych kamery. Efekt patrzenia z dowolnego miejsca w dowolnym kierunku będziemy mogli zatem osiągnąć stosując na naszej scenie właśnie przekształcenie widoku a dokładniej mówiąc mnożąc wszystkie wierzchołki na scenie przez macierz widoku, która będzie symbolizować to przekształcenie. Macierz widoku będzie przekształcać wszystkie współrzędne ze świata do współrzędnych kamery bazując na podstawie położenia samej kamery (które to położenie będzie środkiem układu współrzędnych kamery) oraz kierunku jej patrzenia.

Istnieje wiele sposobów wyznaczania takiej macierzy. W każdym jednak z nich musimy określić położenie samej kamery jak i punktu na który ona patrzy. Położenie to będziemy określać oczywiście we współrzędnych świata, czyli de facto powiemy gdzie naprawdę znajduje się nasza kamera w stosunku do obiektów. Macierz widoku będzie obracać i przesuwać obiekty na scenie, tak aby umieścić je w układzie współrzędnych kamery (to znaczy tak zmienić współrzędne obiektów, żebyśmy mieli wrażenie, że patrzymy na nie z określonego miejsca). Jednym ze sposobów jest stworzenie takiej macierzy, na którą będą się

składać macierz przesunięcia i macierze obrotów wokół każdej z trzech osi. Ogólny wzór na macierz widoku przy takim podejściu przyjmie postać następującą:

$$V = T * R_x * R_y * R_z$$

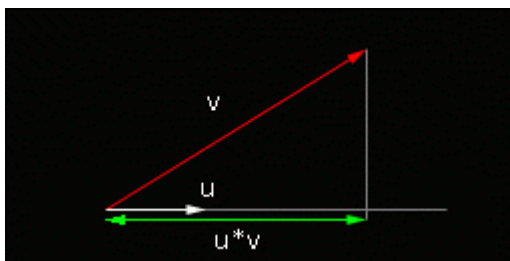
V to oczywiście nasza szukana macierz widoku, **T** to macierz przesunięcia a kolejne **R**-y to macierze obrotów dla poszczególnych osi. Macierze przesunięcia i obrotów są tworzone na podstawie jak już wspomniałem wcześniej położenia kamery we współrzędnych świata. Zaczniemy od przesunięcia - jeśli kamera jest na przykład położona w punkcie (10, 20, 100). Wtedy każdy obiekt będziemy musieli przesunąć dokładnie o -10 jednostek wzdłuż osi X, -20 wzdłuż osi Y i -100 wzdłuż osi Z. Dlaczego ? Ano wyobraźmy sobie, że w tym punkcie, w którym mamy kamerę mamy też jakiś inny punkcik, który możemy zobaczyć. Jeśli na ten punkt popatrzymy się przez kamerę to gdzie on się znajdzie ? No chyba oczywiste ! W samym środku kamery, prawda ? A więc de facto w punkcie (0, 0, 0) :, jasne ? Mam nadzieję, że oczywiste dla wszystkich. Teraz obroty.

Je tworzy się ze względu na to, jak osie układu współrzędnych kamery są ustawione względem osi układu współrzędnych świata, czyli tak naprawdę będzie tutaj brał udział kierunek patrzenia kamery i kąty jakie tworzy z osiami układu współrzędnych świata. Dla przykładu, jeśli kamera, o której mówiliśmy wcześniej jest skierowana w ten sposób, że patrzy bezpośrednio w dół, wtedy jej oś Z jest obrócona względem osi Z układu świata o 90 stopni. Macierze obrotów obracają nam obiekty o daną wielkość ale odwrotnie. Czyli jeśli obrócimy kamerę o 60 stopni w górę to faktycznie musimy obrócić obiekty o 60 stopni w dół aby osiągnąć zamierzony efekt. Jeśli nie wierzycie to wyobraźcie sobie, że chodzimy sobie po budynku i nagle patrzymy w górę. Co dzieje się z obiektami ? Nagle znikają ale... na dole ekranu ! Więc tak naprawdę to obracamy obiekty a nie kamerę.

Innym sposobem tworzenia macierzy widoku jest sposób bardziej bezpośredni, z którego korzystają między innymi funkcje z Direct3D. Skorzystamy tutaj z własności macierzy ortogonalnej, o której już była mowa. Aby stworzyć macierz widoku skorzystamy ze współrzędnych kamery umieszczonych we współrzędnych świata, aby określić położenie osi układu kamery. Nazwijmy sobie dla ułatwienia sprawy punkt na który patrzymy punktem celu kamery. Najpierw policzymy sobie wektor, który łączy położenie kamery z punktem celu właśnie poprzez odjęcie od celu punktu położenia kamery. Otrzymamy w ten sposób pewien wektor, nazwijmy go **n**. Następnie wykonajmy iloczyn wektorowy tego wektora z osią **Y** układu współrzędnych świata. W jego wyniku otrzymamy wektor prostopadły do tych, które biorą udział w naszym iloczynie (co wiemy) i będzie on skierowany w prawo od płaszczyzny jaką tworzą te dwa wektory (radzę sobie przypomnieć lekcję o wektorach :). W następnej kolejności normalizujemy go i otrzymujemy wektor, nazwijmy go **u**. Ale to nie koniec, bo będziemy jeszcze potrzebować do naszych rozważań jeszcze jednego wektora. Stwórzmy iloczyn wektorowy wektorów **n** i **u**, czyli tego, który determinuje kierunek patrzenia kamery i wektora skierowanego w prawo. Po wykonaniu tego działania dostaniemy wektor skierowany w górę, prostopadły do płaszczyzny tworzonej przez wektory **n** i **u** - nazwijmy go sobie **v**. Wektory **n**, **u** i **v** określają nam w tym momencie układ współrzędnych kamery w stosunku do układu współrzędnych świata, co łatwo sobie chyba wyobrazić. Aby zrozumieć na czym polega cała idea tworzenia macierzy w ten sposób będzie nam potrzebna wiedza o dwóch rzeczach. Po pierwsze jak tworzy się ogólnie macierz widoku i to omówiliśmy sobie powyżej oraz jak działa macierz ortogonalna i to też powinniśmy wiedzieć z poprzednich lekcji.

W ostatnim wierszu macierzy widoku powinniśmy mieć współrzędne oznaczające wektor przesunięcia wszystkich punktów świata aby znalazły się one we współrzędnych kamery. Są nam potrzebne trzy składowe - **x**, **y** i **z**. Skąd je policzyć ?

Wróćmy na chwilę do lekcji o wektorach i do pewnej sztuczki związanej z iloczynem skalarnym wektorów. Jeśli wykonaliśmy iloczyn dla takich dwóch wektorów, z których jeden był wektorem jednostkowym to otrzymywaliśmy wtedy kąt pomiędzy wektorami, pomnożony przez długość wektora, który nie był wektorem jednostkowym. W rzeczywistości, jeśli pamiętamy rysunek rzutowaliśmy niejako wektor nie jednostkowy na prostą, która zawierała wektor jednostkowy i tam powstawał odcinek będący długością wyliczoną z iloczynu skalarnego. Tę właśnie sztuczkę zastosujemy teraz tutaj:

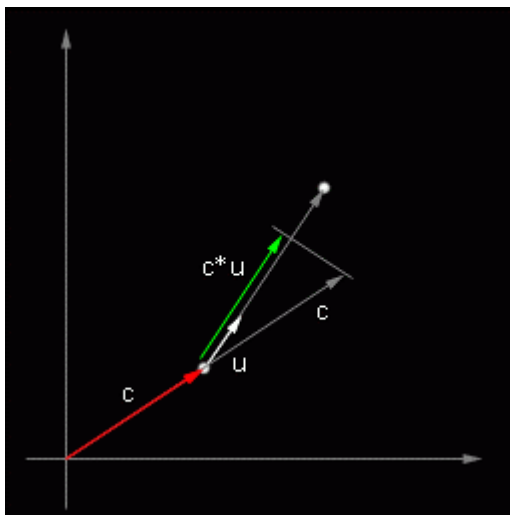


Wektor biały to wektor jednostkowy **u**, czerwony to drugi z wektorów biorących udział w iloczynie skalarnym **v**, natomiast zielony wyznacza wartość jaką będzie miał odcinek po zrzutowaniu wektora nie jednostkowego na prostą zawierającą ten drugi.

Jak powiedzieliśmy sobie, potrzebujemy trzech składowych, aby móc przesunąć wszystkie punkty do układu współrzędnych kamery. Wiemy też, że potrzebujemy wektorów jednostkowych. Te ostatnie weźmiemy oczywiście z układu kamery - będą to wektory **n**, **u** i **v**, które obliczaliśmy kilka linii wyżej. Wszystkie one będą znormalizowane, czyli dokładnie właśnie takie, jakich potrzebujemy. Teraz chcemy wiedzieć o ile przesuwać punkty w osiach X, Y i Z aby znalazły się one w układzie

kamery.

Całą operację może pokazać sobie na przykładzie składowej x , bo nie ma chyba większego sensu robić tego dla wszystkich, nadmienić należy, że wektory i składowe będą powiązane w sposób następujący ze sobą - odległość x policzymy z wektora \mathbf{u} , y z wektora \mathbf{v} a składową z z wektora kierunkowego czyli \mathbf{n} . Ale powracając do składowej x - w układzie kamery za tę oś jest odpowiedzialny wektor \mathbf{u} , więc posłużmy się nim. Przekształcanie do układu współrzędnych kamery polega jak wiemy na tym, żeby tak poustawiać punkty, żeby robiły wrażenie widzianych z jakiegoś punktu różnego od $(0, 0, 0)$. I wróćmy do przykładu z punktem, który leży w tym samym miejscu co kamera. Jeśli przepuścić go przez macierz widoku to powinien on mieć dokładnie współrzędne $(0, 0, 0)$, bo leży w tym samym miejscu co kamera, o tym już mówiliśmy. Pamiętajmy na pewno sztuczki z przesuwaniem wektorów w przestrzeni i wiemy, że nie wpływa to na iloczyn skalarny, prawda? Przesuńmy więc sobie wektor oznaczający położenie naszej kamery (\mathbf{c}) tak, aby jego początek pokrywał się z układem współrzędnych kamery.



Teraz zgodnie regułami sztuczek na wektorach rzutujemy sobie wektor \mathbf{c} na prostą zawierającą wektor jednostkowy \mathbf{u} . Otrzymana długość będzie dokładnie tym, czego szukamy - ilością jednostek, o jakie należy przesunąć punkt aby znalazł się on w układzie współrzędnych świata. Zauważmy też, że punkt niejako cofamy do środka naszego układu, więc musimy odwrócić zwrot tego wektora, o który będziemy cofać, w macierzy ogólnej będzie to bardzo dobrze widać. Trzeba przyznać, że ten pomysł podoba mi się o wiele bardziej niż składanie macierzy z transformacji i poszczególnych obrotów. Oczywiście analogicznie będzie dla pozostałych składowych, mam nadzieję, że będziecie potrafili sobie to wyobrazić bez większych problemów.

Teraz czas na resztę składowych w macierzy, czyli na obroty. Albo możemy pomnożyć poszczególne macierze obrotów i złożyć naszą macierz w całość albo możemy zauważyć pewną rzecz. Wektory układu kamery, które wyznaczyliśmy w jakiś tam sposób wyznaczają nam orientację kamery w przestrzeni względem układu świata. Mając układ współrzędnych i wektory kamery będziemy widzieć bardzo dokładnie jak kamera jest położona w świecie, gdzie patrzy, gdzie ma górę i tak dalej. Wektory te mamy znormalizowane. Wróćmy zatem na chwilę do lekcji o przekształceniach i macierzach ortogonalnych. Czy już wiecie skąd weźmiemy część macierzy odpowiedzialną za obroty? Skoro nasze wektory będą znormalizowane (jednostkowe) i będą w stosunku do siebie ortogonalne to będziemy mogli zamiast sinusów i cosinusów wstawić współrzędne tych wektorów, bo wartości kolejnych ich współrzędnych przecież będą ściśle odpowiadać wartościom w kolumnach macierzy. Macierz na przekształcenie widoku zatem przyjmie postać:

$$\begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ -(\mathbf{u} \cdot \mathbf{c}) & -(\mathbf{v} \cdot \mathbf{c}) & -(\mathbf{n} \cdot \mathbf{c}) & 1 \end{bmatrix}$$

- Przekształcenie projekcji

No i już prawie docieramy do końca w naszych rozważaniach. Potrafimy umieścić obiekty w świecie, potrafimy przekształcić je tak aby widzieć je w zasadzie z dowolnego miejsca. Ale wszystko to ma jeden mały szkopuł - dzieje się w świecie 3D. Ktoś powie, no oczywiście że tak, przecież nam o to właśnie chodzi. No i po trochę będzie mieć rację, ale nie możemy zapominać, że nasz monitor ma jedną bardzo istotną wadę. Jego ekran jest płaski i nie ma trzeciego wymiaru! Czy więc w tym miejscu zakończy się nasza przygoda z grafiką 3D? Postarajmy się aby tak nie było.

Wyobraźmy sobie zatem, że mamy taką wirtualną kamerę, którą możemy w zasadzie umieścić sobie w świecie w dowolnym

miejscu, skierować w dowolną stronę. Kameralą to możemy poruszać w dowolny sposób tworząc animacje lub oglądać obiekty z różnych stron. Aby jednak było można wykorzystać naszą kamerę do czegoś konkretnego musimy sobie powiedzieć o czymś takim jak rzut. Z czym to się je? Definicja mówi, że rzut przekształca punkt w n -wymiarowym układzie współrzędnych na punkt w układzie, którego wymiar jest mniejszy niż n . W naszym ogólnie przyjętym przypadku będziemy sobie przekształcać prawie realny świat 3D na płaski ekran monitora czyli na układ o wymiarze 2D. Niewątpliwie więc nasze kombinowanie z punktami będzie powiązane z rzutem. Rzuty (bo jest ich wiele) możemy podzielić na kilka rodzajów, w zależności od parametrów. W grafice komputerowej zajmować się będziemy głównie tak zwanymi planarnymi rzutami geometrycznymi. Rzuty te charakteryzują się tym, że rzutowanie naszych punktów z przestrzeni 3D będzie się odbywać na płaszczyznę 2D a promienie rzutujące będą liniami prostymi. Co to promienie rzutujące i płaszczyzna wyjaśni się już za moment a teraz zajmijmy się rodzajami rzutów. Płaskie rzuty o których tutaj mówimy można podzielić na dwa podstawowe - perspektywiczne i równoległe. Różnica pomiędzy nimi tkwi w tym, gdzie leży środek rzutowania, czyli punkt z którego wychodzą promienie rzutujące. Jeśli środek rzutowania jest w skończonej odległości od płaszczyzny na którą rzutujemy to taki rzut będziemy nazywać perspektywnym - czyli będziemy mieli do czynienia ze zjawiskiem perspektywy na scenie, ale o tym dalej. Jeśli środek rzutowania będzie leżał w odległości nieskończonej (w nieskończoności) od płaszczyzny to taki rzut nazwiemy równoległym.

Jeśli do obiektu zastosujemy rzut perspektywny to otrzymamy to co dobrze znamy z fotografii czy choćby z tego co widzimy ma co dzień, czyli tak zwany skrót perspektywy. Im dalej będziemy od środka rzutowania będzie obiekt tym jego obraz na płaszczyźnie rzutowania będzie mniejszy - myślę że nie trudno to sobie chyba wyobrazić, prawda? Natomiast rzut równoległy daje obraz mniej realistyczny - nie ma skrótu perspektywy i wielkości obiektów nie zmieniają się. Rzuty takie można stosować do pomiaru odległości na przykład, zresztą też go doskonale znacie na przykład z rysunku technicznego jak sądzę. Jakby tego było mało poszczególne rzuty możemy podzielić na jeszcze więcej kategorii, o których jednak tylko może wspomnijmy. Perspektywiczne rzuty mogą dzielić się na jedno-, dwu- i trzypunktowe, w zależności od tego ile mają tak zwanych podstawowych punktów zbieżności, które jak łatwo się domyśleć są miejsca gdzie zbiegają się linie. Nazywają się podstawowymi ponieważ punkty te mieszczą się na osiach układu współrzędnych i stąd może ich być co najwyżej trzy. My w naszych rozważaniach raczej będziemy się zajmować tylko pierwszym przypadkiem, czyli rzutami jednopunktowymi perspektywnymi. Natomiast rzuty równoległe są bardziej skomplikowane - w zależności od kierunku rzutowania, to znaczy kąta pod jakim promień rzutujący przecina płaszczyznę rzutowania możemy wyróżnić rzuty prostokątne lub ukośne. Rzuty prostokątne znamy też doskonale z lekcji rysunku technicznego - widok z boku, z góry, z przodu plus dodatkowo rzuty aksonometryczne, w tym izometrie - a może niektórzy z was mieli też styczność z rzutami ukośnymi i takimi terminami jak rzut ukośny wojskowy, kawaleryjski i tym podobne. Żeby usystematyzować trochę ten podział może króciutkie podsumowanie:

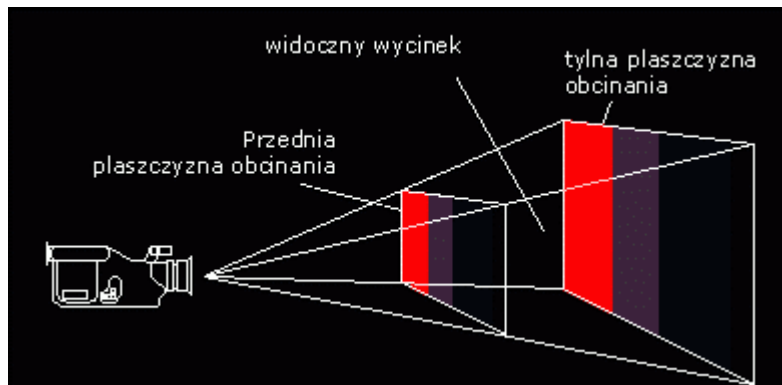
Rzuty:

- Perspektywiczne
 - Jednopunktowe
 - Dwupunktowe
 - Trzypunktowe
- Równoległe
 - Prostokątne
 - widok z boku
 - widok z przodu
 - widok z góry
 - Aksonometryczne
 - Izometryczne
 - Inne
 - Ukośne
 - Wojskowe
 - Kawaleryjskie
 - Inne

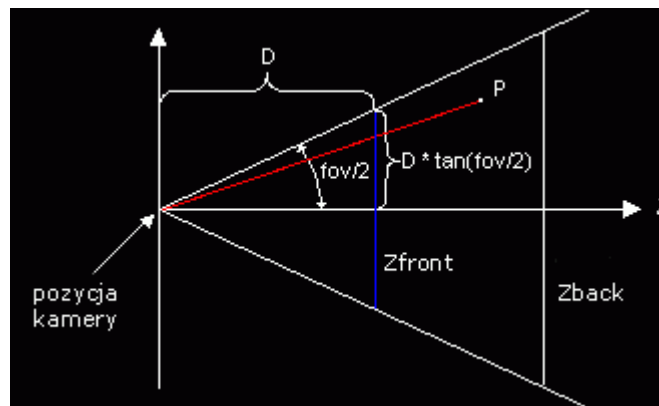
Sposobów tworzenia macierzy projekcji jest kilka, każdy pakiet do tworzenia grafiki 3D stosuje swój własny, ale nie będziemy ich wszystkich tutaj omawiać, skupimy się może na tych, które są zawarte w SDK do DirectX. Oczywiście ogólne zasady są podobne do wszystkich i może od nich zaczniemy. W większości naszych przykładów używać będziemy oczywiście rzutowania perspektywnego, żeby otrzymać złudzenie świata widzianego naszymi oczami i dlatego przede wszystkim omówimy sobie tworzenie macierzy projekcji dla tego typu rzutu.

Z punktu widzenia Direct3D przekształcenie projekcji składa się z dwóch oddzielnych przekształceń - perspektywy i skalowania. Jak doskonale wiemy z własnego doświadczenia człowiek nie ma oczu wokół głowy i widzi tylko wycinek przestrzeni. Wycinek taki charakteryzuje się tym, że obiekty widzimy w zasadzie tylko w pewnym zakresie, od tych które są najbliżej nas do tych, które leżą czasem nawet bardzo daleko. W zasadzie pole widzenia nie jest ograniczone ale różne czynniki, takie jak atmosfera, przeszkody terenu i tym podobne powodują, że nie widzimy dalej niż na kilkaset do kilku tysięcy metrów, ale może nie kłómy się o szczegóły. W komputerowej rzeczywistości nasz świat zawsze będzie ograniczony i to z kilku powodów - mocy obliczeniowej nie wystarczającej do zwizualizowania nieskończonej ilości obiektów, skończonych wielkości typów liczb i tym podobnych rzeczy. Biorąc rzecz praktycznie, niewiele będzie nam to

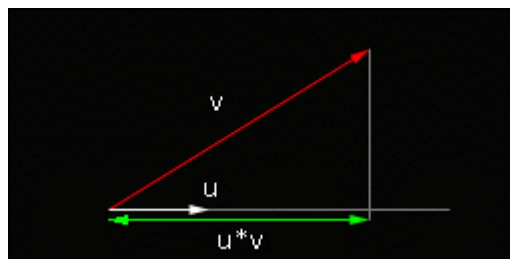
przeszkadzać i możemy sobie spokojnie założyć, że nasz widzialny wycinek może być ograniczony. Łatwo też myśleć sobie wyobrazić to, że im bliżej będzie do naszego oka, tym wycinek widzialny będzie mniejszy. Może ten rysunek wyjaśni wam poglądowo o co chodzi:



Da się tutaj zauważyć, że widzialny obraz nie zaczyna się koniecznie od samego położenia naszego oka (kamery). Wyobraźmy sobie na przykład sobie, że stoimy w bardzo ciemnym pokoju i patrzymy na świat przez otwarte okno. Stoimy w pewnej odległości od okna. To co widzimy rozpoczyna się dopiero w momencie kiedy kończy się pokój i jest ściana z oknem a kończy gdzieś na horyzoncie, albo czymś, co nam przysłoni widok. Jeśli podejmiemy bliżej okna to nasz widoczny wycinek trochę się będzie zmieniał (wypróbujcie w wolnej chwili). Zauważmy również ważną rzecz - przednia płaszczyzna obcinania zaznaczona na rysunku będzie naszą płaszczyzną, na którą będziemy rzutować wszystkie obiekty, które znajdują się w widocznym wycinku czyli będzie naszą płaszczyzną rzutowania lub krótko mówiąc rzutnią. Pokażmy sobie może jeszcze jeden rysunek, na którym będziemy sobie analizować nasze obliczenia:



Każdy rodzaj rzutu, o których wspominałem wyżej można określić za pomocą macierzy 4x4 a jak tę macierz wyznaczyć? Przypatrzmy się bliżej rzutowi perspektywicznemu i założmy sobie, że gdzieś w widocznym wycinku mamy punkt **P**. Punkt **P** ma zostać zrzutowany na płaszczyznę rzutowania (kolor niebieski). Zauważmy na rysunku powyżej, że kamera leży w odległości **D** od początku tego kawałka naszej przestrzeni, którego używaliśmy podczas tworzenia macierzy widoku. Tam wszystkie nasze punkty zostały tak przekształcone, że znalazły się w układzie współrzędnych kamery. W tym układzie teraz określamy sobie granice naszego widocznego wycinka. Korzystając z różnych matematycznych sztuczek i kombinowania dojdziemy w końcu do tego, że współrzędne naszego punktu P_p (zrutowanego punktu **P** na naszą rzutnię) trzeba będzie podzielić w jakiś sposób przez wielkość **D**. Dla nas będzie to po prostu współczynnik o jaki będziemy skalować nasze obiekty - wiemy doskonale, że im dalej będziemy od naszego obiektu tym będzie on mniejszy, wydaje nam się to zupełnie naturalne. Macierz, która spowoduje nam takie zachowanie wierzchołków a co za tym idzie i całych obiektów będzie wyglądać tak:



Ponieważ nasza płaszczyzna rzutowania leży w odległości **D** od naszego oka więc aby wszystko nam grało koniecznym będzie jeszcze przesunięcie wszystkich punktów o tę właśnie odległość, tak aby przednia płaszczyzna obcinania znalazła się

w punkcie, w którym aktualnie położona jest kamera. Macierz przesunięcia potrafimy sobie wyznaczyć, bo jest zupełnie banalna i wygląda dla naszego przypadku tak:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -D & 1 \end{bmatrix}$$

Mnożąc teraz macierz przesunięcia (aby punkty i płaszczyzna rzutowania znalazły się nam we właściwym miejscu) przez macierz perspektywy (aby obiekty zrobiły się większe lub mniejsze) otrzymamy tak zwaną połączoną macierz projekcji. Możemy sobie to wyobrazić w ten sposób, że nasz widoczny wycinek jest przesuwany do środka układu współrzędnych a następnie od strony przedniej płaszczyzny obcinania jest on powiększany tak aby współrzędne rogów ograniczających przednią jak i tylną płaszczyznę obcinania mieściły się w zakresie od -1 do 1. Przednia płaszczyzna obcinania będzie leżała w punkcie 0 jeśli chodzi o współrzędną z natomiast tylna w punkcie o współrzędnej z równej 1.

Ta macierz zbudowana w oparciu o odległość kamery od bliższej płaszczyzny obcinania będzie nam skalować i przesuwać obiekty, tak żeby wszystko ładnie wyglądało. Jednak nie bierze ona pod uwagę dwóch ważnych aspektów sprawy. Po pierwsze wartości z naszych punktów będą się mieścić w bardzo małym przedziale bo od 0 do 1 co może bardzo utrudnić potem nam przeprowadzenie porównań odległości i prawidłowe malowanie niewidocznych ścian. Po drugie - wróćmy jeszcze do naszego przykładu z oknem. Zauważmy, że okno nasze może być różnej wielkości, przy czym nie zmienia ono swego położenia względem nas ! Okno więc może być szersze i wyższe a co to dla nas oznacza ? To tak samo jak oglądać film w dobrym kinie i w słabym telewizorze - w kinie na obrazie panoramicznym zobaczymy o wiele więcej niż na obciętym do granic możliwości obrazie telewizyjnym. U nas, jeśli zmieniać będziemy rozmiary naszego okna zmieniać się będzie na pewno nasz widoczny wycinek, prawda ? A więc będziemy mogli widzieć na przykład więcej obiektów albo większą część sceny, oczywiście wszystko w granicach rozsądku, ale pole do popisu mamy dosyć spore. Niestety nasza przed chwilą wyprowadzona macierz nie uwzględnia takiego typu działań, w żaden sposób nie możemy w niej określić wielkości naszego okna a jak łatwo się przekonać nie będzie to nic innego jak kąt naszego widzenia. Jeśli nasze okno byłoby odpowiednio duże to moglibyśmy niemal widzieć całą przestrzeń od lewa do prawa nieograniczeni żadnym wycinkiem czyli obraz w zakresie prawie 180 stopni. Powstaje więc pytanie jak to przenieść do przekształcenia projekcji żeby miało wpływ na nasz obraz. Ponieważ myślimy o rozszerzaniu naszego pola widzenia no to pewnie przyjdzie nam dokonać jakiegoś skalowania. A dlaczego ? No popatrzmy - obraz nam się poszerza, widzimy coraz szerzej i więcej, ale niestety (!) nasz monitor jak miał te -naście cali tak ma dalej i nic a nic nie chce się rozciągnąć. Skoro nie może góra do Mahometa to Mahomet przyjdzie do góry i aby uzyskać efekt "rozciągania" obrazu my bezczelnie go po prostu zmniejszymy. Ponieważ rozmiary naszego okna będzie można zmieniać zarówno w pionie i poziomie więc skalować też będziemy w tych samych kierunkach. Pamiętajcie macierz skalowania ? Dwa pierwsze składniki na przekątnej służyły za współczynniki skalowania dla osi X i Y a my oznaczmy sobie je u nas w i h (jak szerokość i wysokość z angielskiego). O ile należy przeskalować naszą scenę wraz ze zmianą rozmiarów ? Przypatrzmy się naszemu rysunkowi prezentującemu tę operację powyżej, jak widać szerokość przedniej płaszczyzny obcinania można policzyć z następującego wzoru:

$$D \cdot \tan\left(\frac{fov}{2}\right)$$

Ponieważ my będziemy zwiększali (lub zmniejszali) nasz kąt widzenia więc współczynnik skalowania dla osi **Y** trochę prostszy niż to co powyżej przyjmie postać:

$$h = \cot\left(\frac{fov_h}{2}\right)$$

I chyba nie muszę dodawać, że dla osi **X** będzie bardzo podobnie:

$$w = \cot\left(\frac{fov_w}{2}\right)$$

Co natomiast z odległością **Z** ? Napisałem, że przedział od 0 do 1 będzie stanowczo za mały, żeby dobrze sobie potem radzić z wykrywaniem niewidocznych ścian, zwłaszcza dla z-buforów o małej rozdzielczości. Przekształcenie projekcji najpierw powodowało to, że współrzędne z obiektów były ściskane do przedziału (0, 1) więc teraz należałoby zadbać o to, żeby te współrzędne jakoś poszerzyć. do tego celu posłuży nam właśnie czynnik **Q**:

$$Q = \frac{z_f}{z_f - z_n}$$

gdzie **Zf** to dalsza płaszczyzna obcinania a **Zn** bliższa.

Równania w postaci macierzowej dla wymienionych przeze mnie operacji można zapisać jako:

$$\begin{bmatrix} w & 0 & 0 & 0 \\ 0 & h & 0 & 0 \\ 0 & 0 & Q & 1 \\ 0 & 0 & -Q \cdot Z_n & 0 \end{bmatrix}$$

W ostatnim wierszu tej macierzy projekcji mamy jeszcze przesunięcie o wartość **-Zn*Q**. Spowoduje to przesunięcie wszystkich punktów o przeskalowane odległości w tył, aby środek naszej sceny znajdował się w środku układu projekcji.

Czasem zdarza się, że zamiast podawać kąty widzenia w poszczególnych osiach my podamy sobie szerokość naszego okna w jednostkach występujących na przykład na scenie. Całe przekształcenie projekcji tak przekształca nam dane aby zmieściły się one w sześcianie o współrzędnych od -1 do 1 w osiach **X** i **Y** oraz 0 do 1 w osi **Z**. Robi się to po to aby łatwiej można było zaimplementować algorytmy obcinania w tak zwanej kanonicznej bryle widzenia 0 bryłą tą jest sześcian o którym właśnie mowa. Obcinanie w kanonicznej bryle widzenia jest o wiele prostsze i co najważniejsze szybsze niż w każdej innej.

Czasem w naszej aplikacji zdarza się, że wygodniej nam będzie podać wymiary okna, przez jakie patrzymy na nasz świat niż kąt patrzenia. Mając odległość do przedniej płaszczyzny obcinania **Zn** będziemy mogli sobie jednak w łatwy sposób wyznaczyć czynniki skalujące nasze obiekty w osiach **X** i **Y** ze względu na wielkość okna. Kiedy nie będziemy posługiwać się kątami możemy posłużyć się prostymi zależnościami:

$$w = 2 \cdot \frac{Z_n}{V_w} \quad h = 2 \cdot \frac{Z_n}{V_h}$$

Zn to oczywiście odległość od przedniej płaszczyzny obcinania, **Vw** i **Vh** to szerokość i wysokość naszego okna wyrażona we współrzędnych układu kamery.

I to byłoby w zasadzie na tyle z zastosowań macierzy i przekształceń w 3D dla osiągnięcia jakiś bardzo specyficznych efektów, w zasadzie takich które potrafią dać nam naprawdę wrażenia podobne do tych, z prawdziwego świata. Niektóre rzeczy mogą wam się wydać zakrecone, ze szczególnym wskazaniem na macierz projekcji, ale tak jak napisałem większość omówionego tutaj materiału znajduje taką właśnie realizację w bibliotece Direct3D. Nic oczywiście nie stoi na przeszkodzie, żebyście sobie mogli napisać własne sposoby przekształcania sceny, niekiedy może nawet bardziej optymalne niż te, przedstawione tutaj. Nam to na razie w zupełności wystarczy do dalszego działania i teraz będziemy mogli się zająć naprawdę bardziej interesującymi rzeczami w grafice 3D niż tylko obracające się sześciany. Mam nadzieję, że większość tego materiału jest dla was w miarę zrozumiała, dziękuję za uwagę i zabieram się za następną artykulik ;). Do zobaczenia następnym razem...