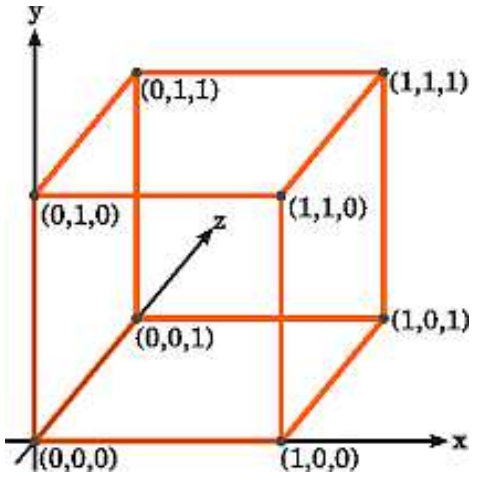
Transformacje obiektów 3D

Opracowanie:
dr inż. Grzegorz Szwoch
Politechnika Gdańska
Katedra Systemów Multimedialnych

Lokalny układ współrzędnych

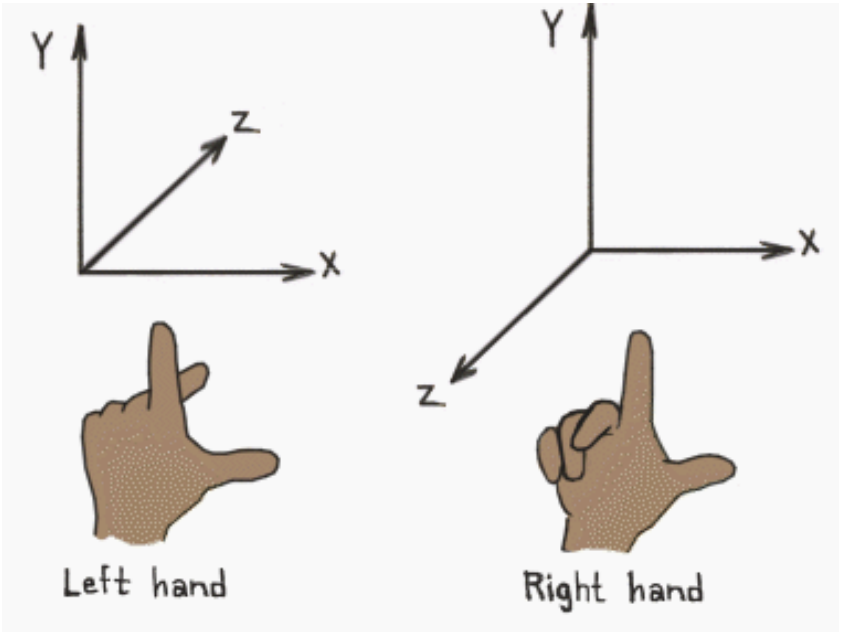
- Tworząc model obiektu, zapisujemy współrzędne wierzchołków siatki w układzie współrzędnych lokalnych (modelu, obiektu).
- Początek układu (0,0,0) i orientacja mogą być wybrane dowolnie.
- Dla wygody, początek układu powinien być w środku obrotu i skalowania modelu.



Orientacja osi z

Dygresja: układ współrzędnych może być prawoskrętny (*right-handed*) lub lewoskrętny *(left-handed*).

- Różnica – kierunek współrzędnej z
- Wybór wpływa na znak z w macierzach przekształceń.
- OpenGL przyjmuje układ prawoskrętny.
- DirectX przyjmuje układ lewoskrętny.

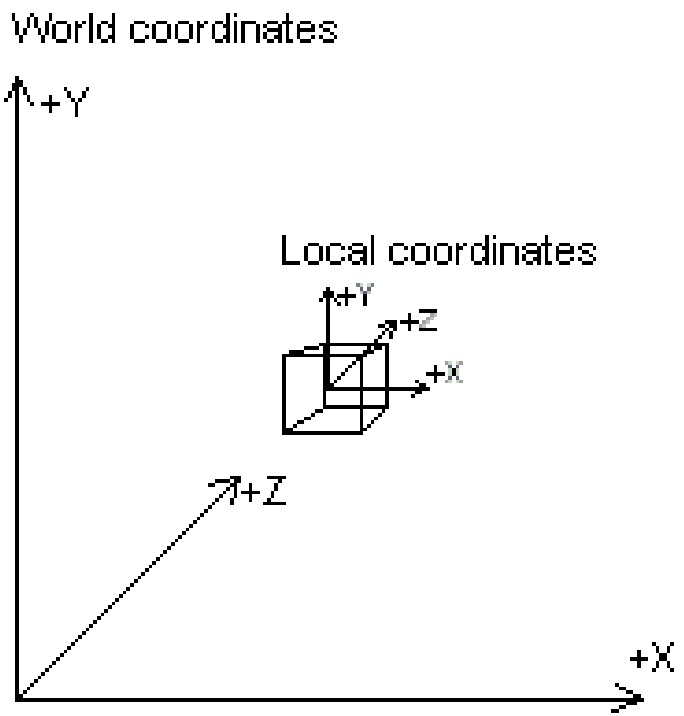


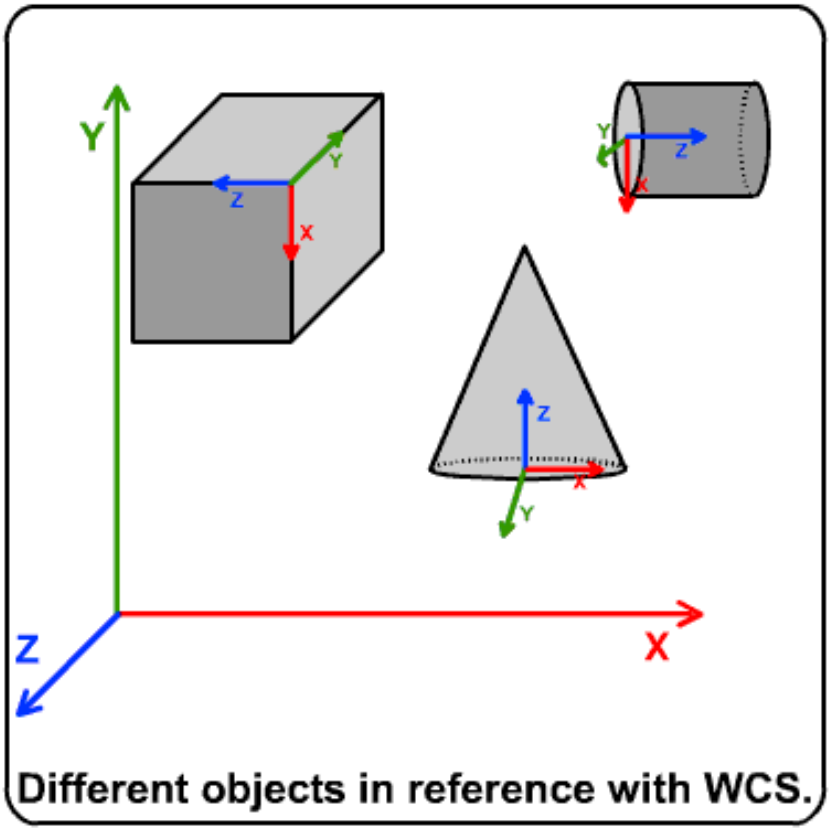
Układ współrzędnych świata

- Świat 3D (*world*) – wirtualna scena opisana w trzech wymiarach.
- Świat używa własnego układu współrzędnych.
- Każdy obiekt 3D jest umieszczany w świecie 3D w odpowiedniej pozycji, skali i pozie.
- Przekształcenie świata (*world transform*)
  - konwersja wsp. lokalnych na wsp. świata.
- Wszystkie wierzchołki jednego obiektu są przekształcane w taki sam sposób.
- Różne obiekty mają własne przekształcenia.

Układ współrzędnych świata

Współrzędne lokalne i świata





Macierz przekształcenia

- Współrzędne jednorodne (*homogenous coordinates*) punktu w układzie świata: wektor 4-elementowy: (x, y, z, w)
- Osie ukł. wsp. świata są wyznaczone przez trzy wektory jednostkowe o długości 1: (x_X, y_X, z_X), (x_Y, y_Y, z_Y), (x_Z, y_Z, z_Z)
- Przesunięcie początku układu: (t_X, t_X, t_Z)
- Macierz przekształcenia układu:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x_X & y_X & z_X & t_X \\ x_Y & y_Y & z_Y & t_Y \\ x_Z & y_Z & z_Z & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix}$$

Współrzędne jednorodne

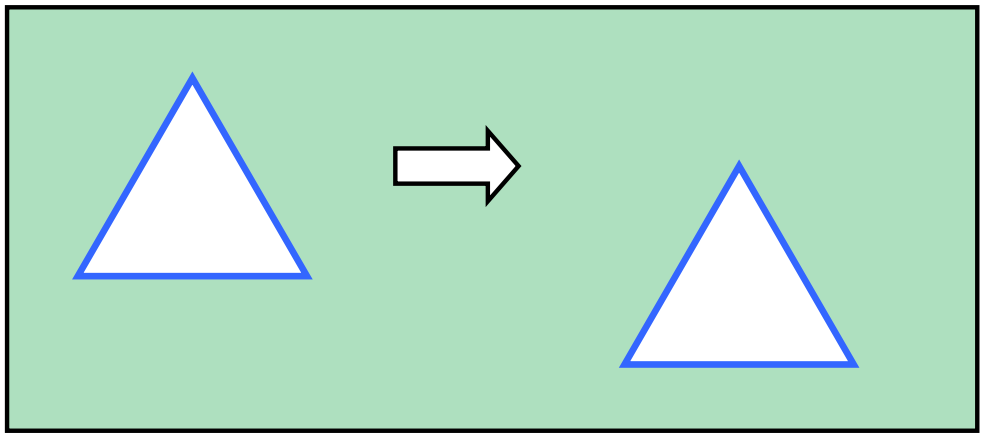
- Uproszczony przypadek:
  - „klasyczny” układ kartezjański, wyznaczony przez wektory: (1,0,0) (0,1,0) (0,0,1),
  - brak przesunięcia (wspólny początek),
  - ta sama orientacja układów.
- Przekształcenie przez macierz jednostkową:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix}$$

Translacja (przesunięcie)

Translacja (*translation*) o wektor (t_X, t_X, t_Z)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix} + \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix}$$



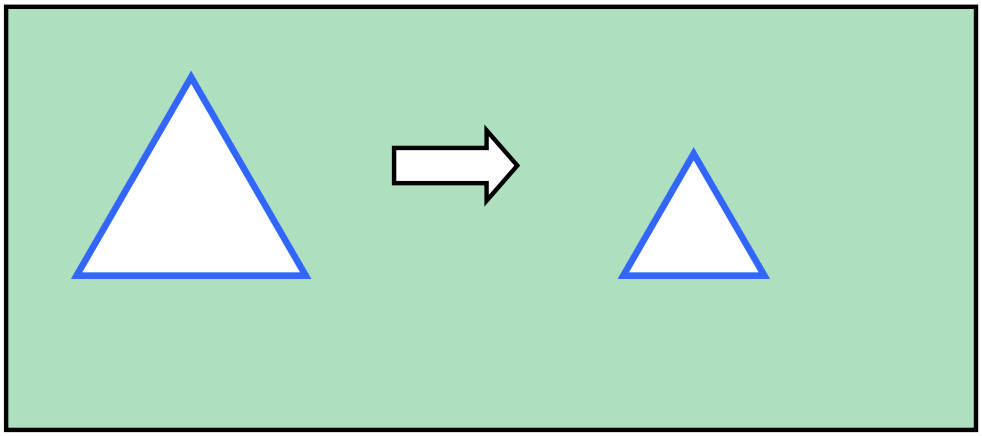
Macierz przekształcenia:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_X \\ 0 & 1 & 0 & t_Y \\ 0 & 0 & 1 & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix}$$

Skalowanie

Skalowanie (*scaling*) o współczynniki skalowania: s_X, s_X, s_Z

Skalowanie: jednorodne (wszystkie współczynniki jednakowe)
lub niejednorodne.



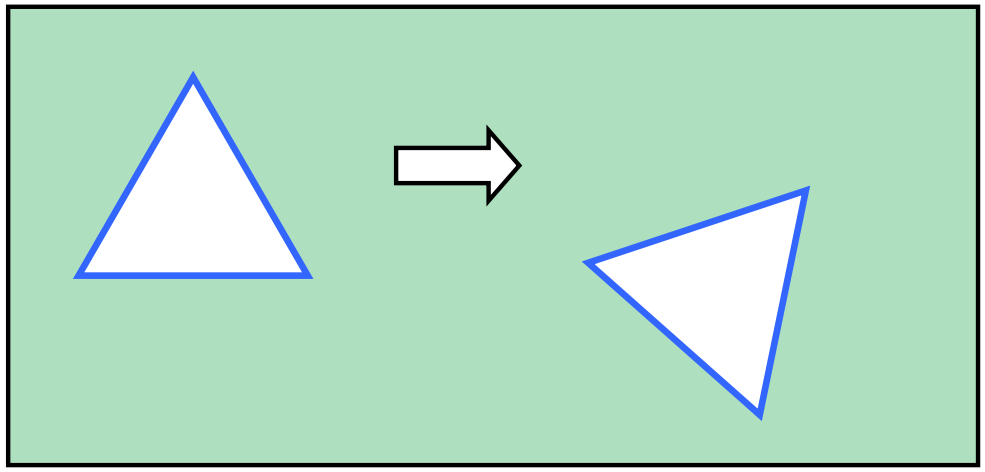
Macierz przekształcenia:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} s_X & 0 & 0 & 0 \\ 0 & s_Y & 0 & 0 \\ 0 & 0 & s_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix}$$

Obrót

Obrót (*rotation*) o kąt θ (w radianach)

- W przestrzeni 3D możliwe są obroty wokół każdej z trzech osi (X, Y, Z).
- Inaczej: obrót wokół osi X to obrót na płaszczyźnie YZ.
- Osobna macierz przekształcenia dla każdego z obrotów.



Obrót

Macierze przekształceń dla obrotu wokół każdej z osi X, Y, Z:

$$R_X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_Y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_Z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Przekształcenia afiniczne

Przekształcenia afiniczne:

- zachowują równoległość linii
  (linie, które były równoległe do siebie,
  pozostają równoległe po przekształceniu),
- nie muszą zachowywać odległości i kątów.

Wszystkie omówione przekształcenia:

- translacja, skalowanie, obrót

są afiniczne.

Przekształcenia, które nie są afiniczne, nazywa się przekształceniami deformującymi.

Środek obrotu i skalowania

UWAGA #1!

- Obrót i skalowanie są wykonywane zawsze w stosunku do początku układu wsp.!
- Dlatego na etapie modelowania, warto ustawić początek układu lokalnego w środku obrotu/skalowania.
- W przeciwnym razie, musimy najpierw „wyrównać” układy za pomocą dodatkowej translacji.

Składanie przekształceń

- Zwykle wykonujemy na wierzchołku kilka przekształceń.
- Obliczamy jedną macierz przekształcenia jako iloczyn (macierzowy) składowych macierzy.
- Obliczoną macierz stosujemy osobno do każdego wierzchołka obiektu.
- Zalecana kolejność:
  - skalowanie
  - obroty (dowolna kolejność)
  - przesunięcie – zawsze na końcu!

Składanie przekształceń

UWAGA #2!

- Kolejność macierzy przy mnożeniu: „od prawej do lewej”.
- Np. dla operacji (kolejno): S, R_Y, T:

$$\mathbf{p}_1 = \mathbf{S} \cdot \mathbf{p}_0$$

$$\mathbf{p}_2 = \mathbf{R}_Y \cdot \mathbf{p}_1 = \mathbf{R}_Y \cdot \mathbf{S} \cdot \mathbf{p}_0$$

$$\mathbf{p}_3 = \mathbf{T} \cdot \mathbf{p}_2 = (\mathbf{T} \cdot \mathbf{R}_Y \cdot \mathbf{S}) \cdot \mathbf{p}_0$$

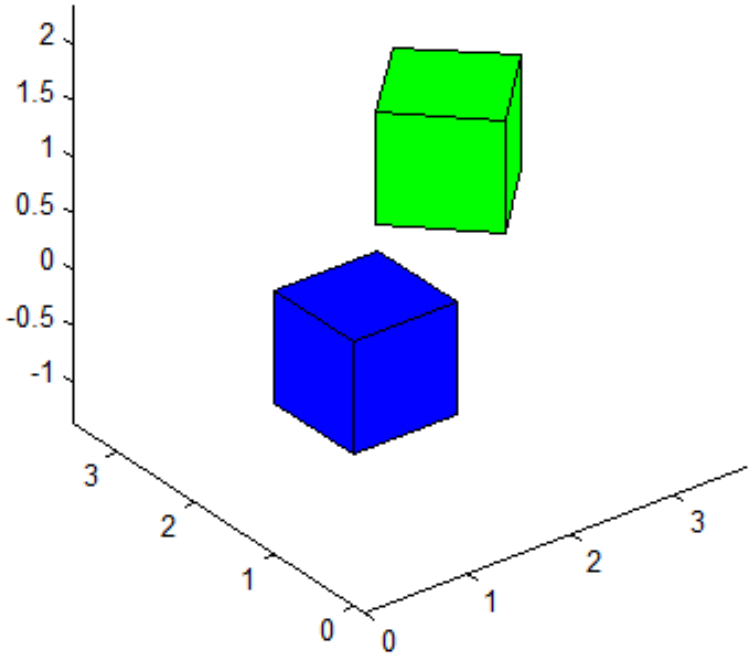
$$\mathbf{p}_3 = \mathbf{M}_\mathbf{W} \cdot \mathbf{p}_0 \qquad \mathbf{M}_\mathbf{W} = \mathbf{T} \cdot \mathbf{R}_Y \cdot \mathbf{S}$$

Składanie przekształceń

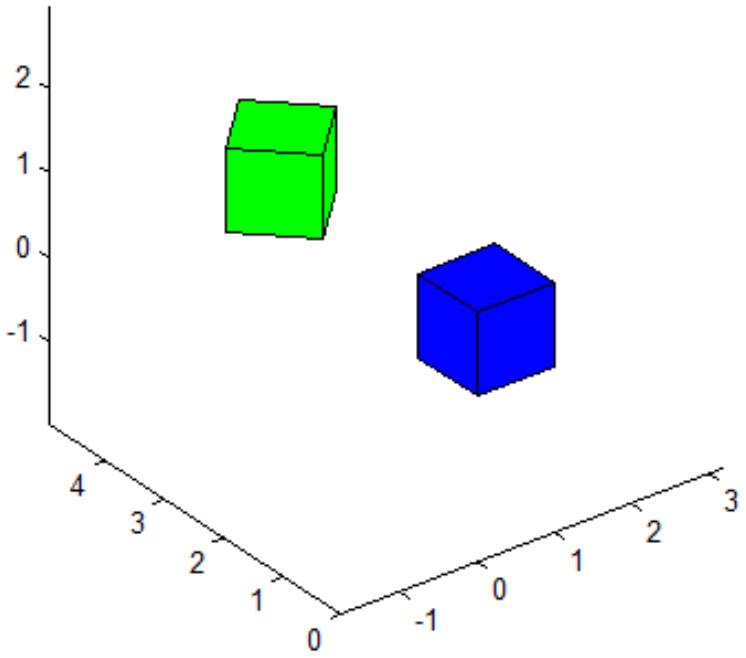
UWAGA #3!

- Kolejność operacji ma znaczenie.

Obrót
i przesunięcie:

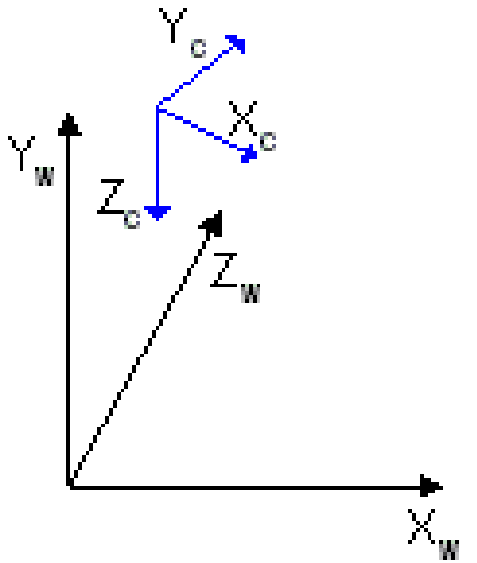


Przesunięcie
i obrót:



Układ współrzędnych widoku

- Możemy oglądać świat 3D z różnego miejsca i pod różnym kątem.
- Kamera (*camera*), także: obserwator, oko, widz – punkt w ukł. wsp. świata, z którego obserwowany jest świat.
- Układ współrzędnych widoku (*view*):
  - kamera jest w jego początku,
  - orientacja osi wyznacza kierunek, w jakim „patrzy” kamera względem układu świata



Układ współrzędnych widoku

Przykład opisu ukł. widoku – NVU

- początek w ukł. świata: (c_X, c_X, c_Z)
- orientacja: trzy jednostkowe wektory:
  - N: kierunek, w którym patrzy kamera (oś Z)
  - V: kierunek „w górę” (oś Y)
  - U: kierunek „w prawo” (oś X) – nie musi być podany, można go obliczyć.
- Macierz transformacji widoku:

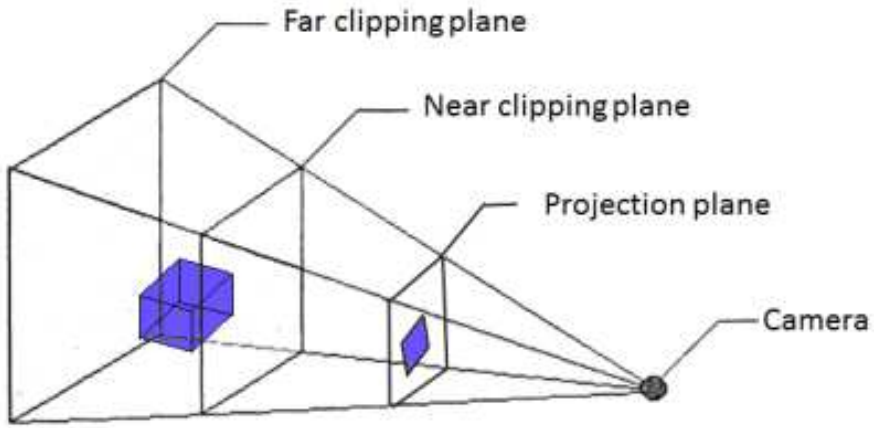
$$\mathbf{M_V} = \begin{bmatrix} U_X & U_X & U_X & 0 \\ V_Y & V_Y & V_Y & 0 \\ N_Z & N_Z & N_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & c_X \\ 0 & 1 & 0 & c_Y \\ 0 & 0 & 1 & c_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformacja widoku

- Cały świat (wszystkie współrzędne świata wszystkich wierzchołków wszystkich obiektów) jest przekształcany za pomocą tej samej macierzy transformacji widoku.
- Macierz tr. widoku będzie zmieniać się gdy przesuniemy kamerę.
- Całkowita macierz przekształcenia:
  m. transformacji widoku * m. transf. świata
  $\mathbf{p_V} = \mathbf{M_V} \cdot \mathbf{p_W} = \mathbf{M_V} \cdot \mathbf{M_W} \cdot \mathbf{p_L}$

Rzutowanie

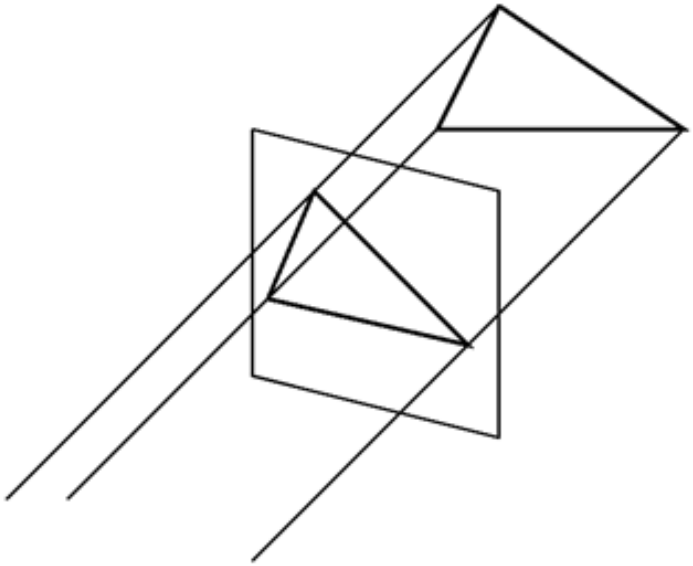
- Rzutowanie (*projection*) polega tutaj na przekształceniu współrzędnych 3D na 2D.
- Obiekty ze świata 3D są rzutowane na płaszczyznę prostopadłą do osi kamery.
- Okno *(viewport*) – fragment płaszczyzny rzutowania zawierający obiekty, które mają „zmieścić się” na obrazie. Jest „wizjerem” kamery.



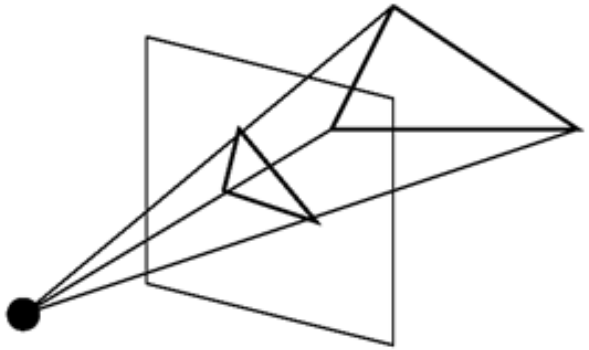
Rzutowanie

Dwa zasadnicze rodzaje rzutów:

rzut równoległy
(*ortographic projection*),



rzut perspektywiczny
(*perspective projection*)

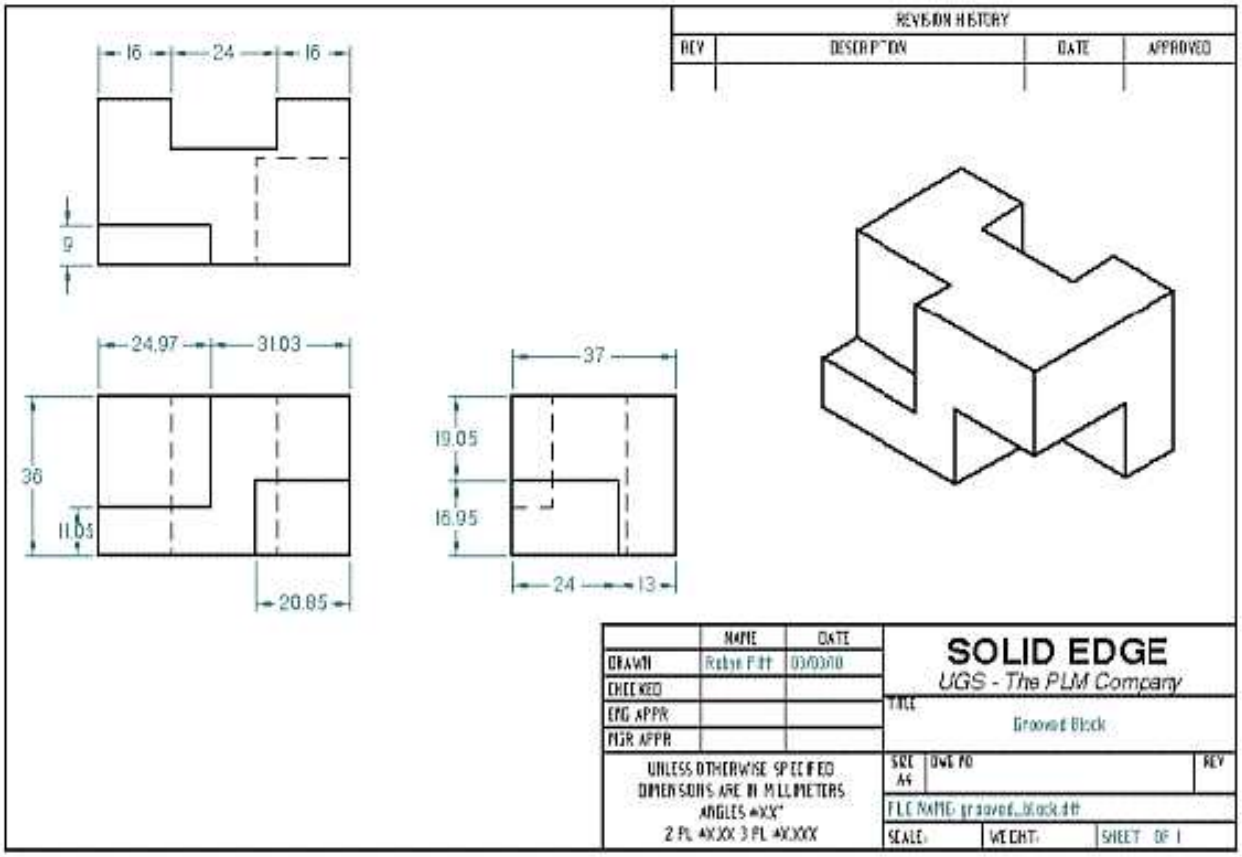


Rzut równoległy

- Rzut równoległy nie zachowuje perspektywy.
- Zachowuje kształt i rozmiar obiektów.
- Rodzaje rzutu równoległego:
  - prostokątny (przedni, górny, boczny)
  - ukośny
  - izometryczny
- Metoda stosowana w projektowaniu CAD, nie używana w grach.
- Rzut równoległy jest prostszy, wystarczy pominąć współrzędne *z*.

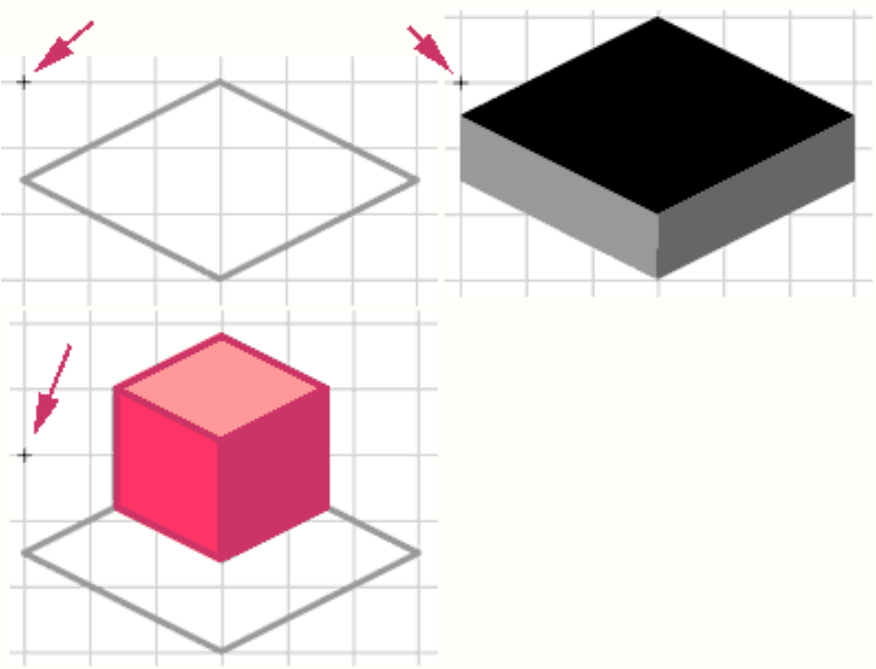
Rzut równoległy

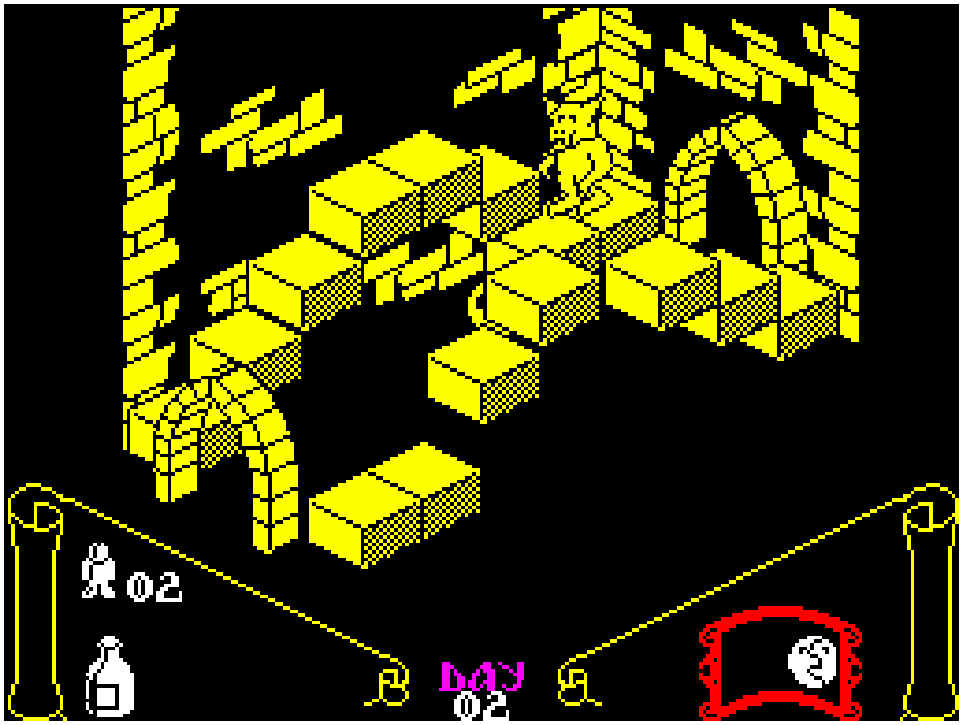
Przykład: trzy rzuty prostokątne
i rzut izometryczny



Rzut izometryczny

Rzut stosowany często w starszych grach komputerowych 2D. Iluzja trzech wymiarów - brak perspektywy.



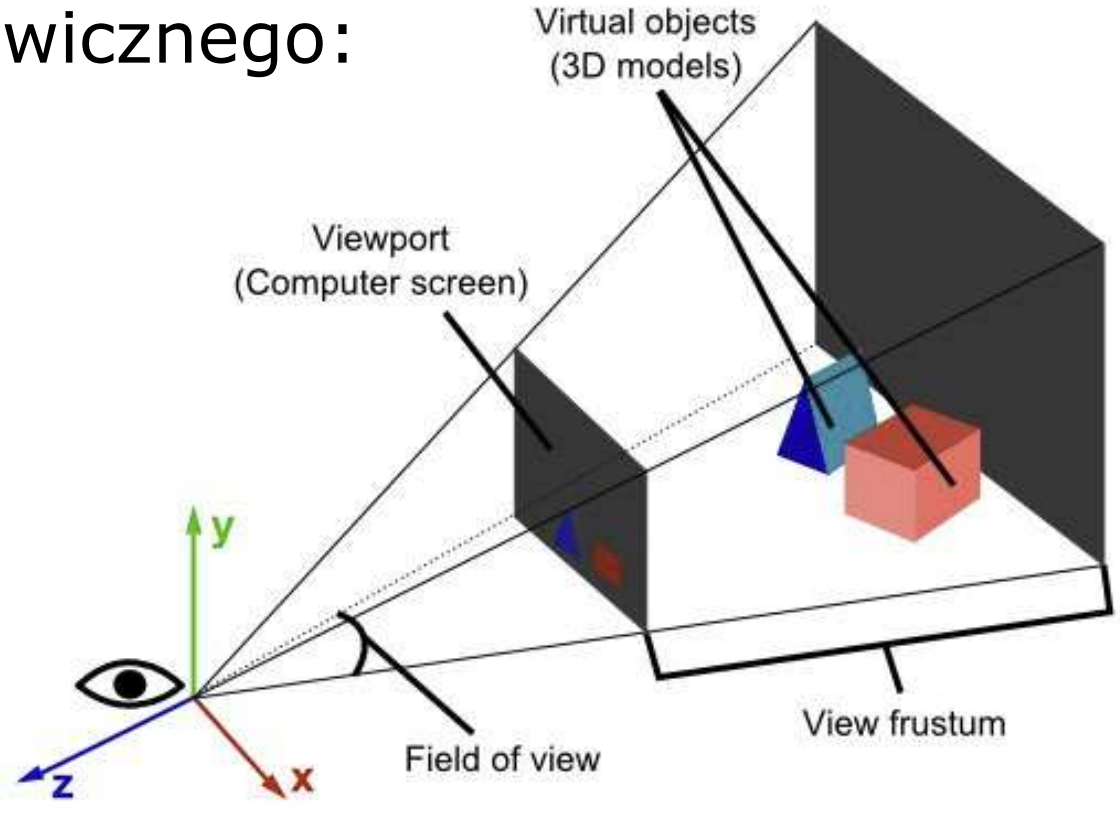


Rzut perspektywiczny

- Rzut perspektywiczny odwzorowuje sposób widzenia człowieka (skrót perspektywiczny).
- Wielkość rzutu zmniejsza się przy wzroście odległości między środkiem rzutowania (kamerą) a obiektem.
- Zniekształcane są kształty i wymiary obiektów.
- Stosowany np. w grach komputerowych.
- Trudniejsze przekształcenia niż dla rzutu równoległego.

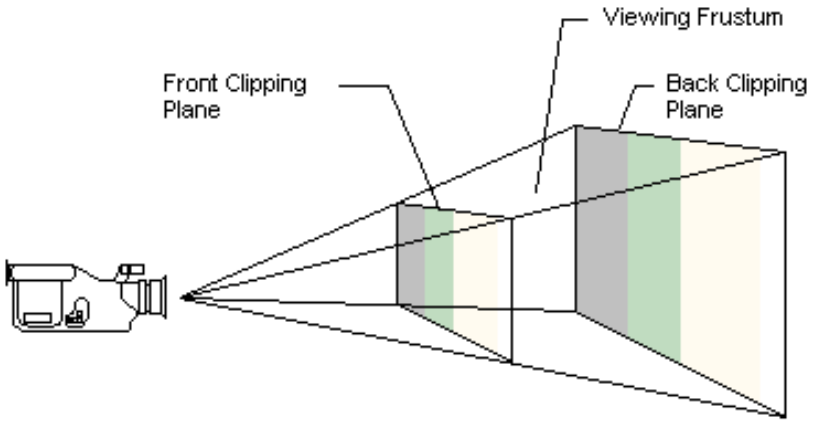
Bryła widzenia

- Bryła widzenia jest wyznaczona przez promienie prowadzone od kamery, przez krawędzie okna.
- Dla rzutu równoległego: prostopadłościan.
- Dla rzutu perspektywicznego: ostrosłup.



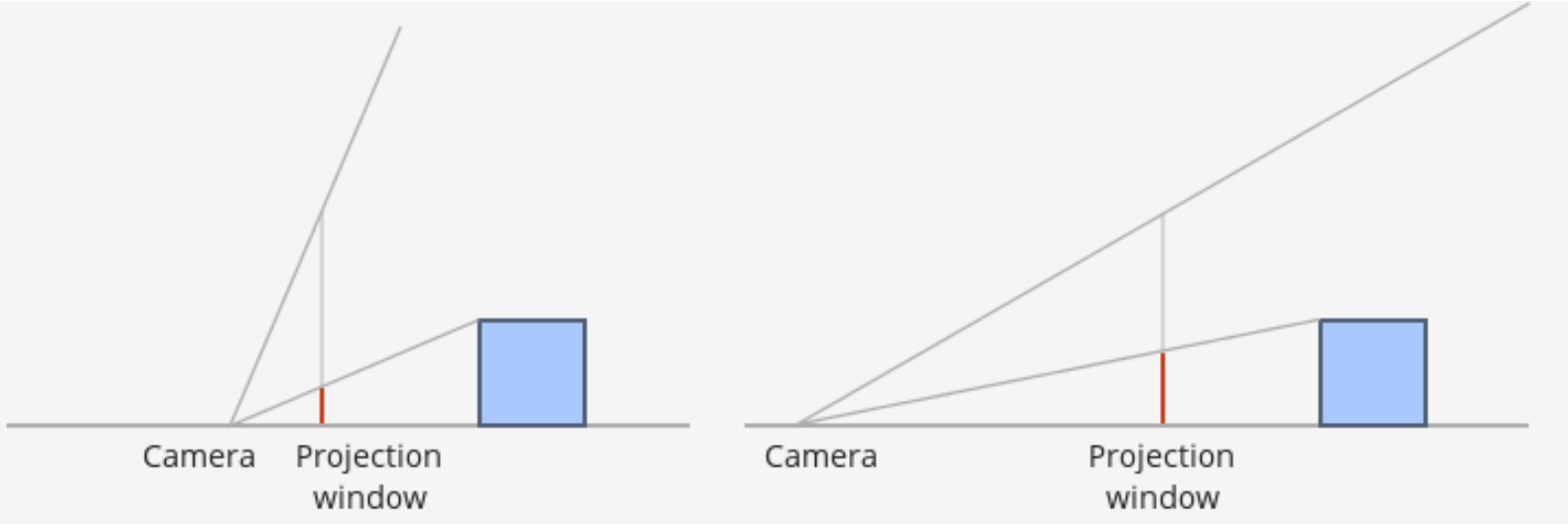
Bryła widzenia

- Bryła jest obcięta przez dwie płaszczyzny.
- Przednia płaszczyzna obcinania (*front clipping plane*) – zabezpiecza przed błędami dzielenia przez 0 i małe liczby.
- Tylna płaszczyzna obcinania (*back clipping plane*) – eliminuje rysowanie zbyt odległych obiektów („*draw distance*”)
- Powstaje ostrosłup ścięty – *frustum*.



Pole widzenia

- Pole widzenia – *field of vision* (FOV) jest kątem pionowym bryły widzenia.
- Kąt zależy od odległości okna od kamery:
  - blisko kamery – duży kąt,
  - daleko od kamery – mały kąt.
- FOV działa więc jak ogniskowa aparatu – ustawia „zoom” kamery na świat 3D.



Przekształcenie perspektywiczne

Macierz przekształcenia perspektywicznego:

- pole widzenia (FOV): α
- położenie płaszczyzn obcinania: z_N, z_F
- wsp. proporcji widoku (*aspect ratio*): $a_r = w / h$

$$\mathbf{M}_P = \begin{bmatrix} \frac{1}{a_r \cdot \tan(\alpha/2)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\alpha/2)} & 0 & 0 \\ 0 & 0 & \frac{z_N + z_F}{z_N - z_F} & \frac{2 z_N z_F}{z_N - z_F} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Dzielenie perspektywiczne

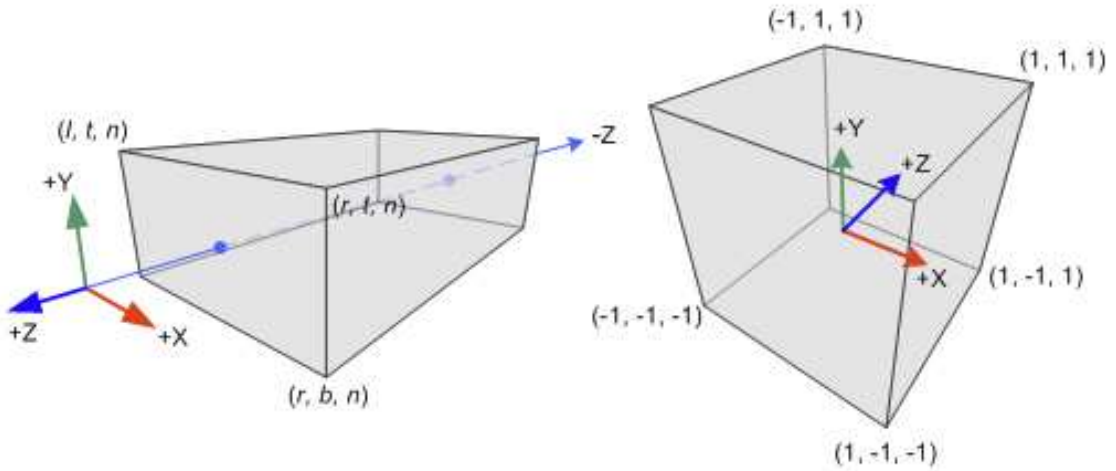
- *Perspective divide* – podzielenie rzutowanych wartości (x, y, z) przez z.
- Jest wykonywane sprzętowo przez GPU.
- Aby nie stracić wartości z, skopiowaliśmy ją do w – jest to współczynnik skali.
- Dostajemy zatem punkt:

$$p = \left(\frac{x}{z}, \frac{y}{z}, 1 \right)$$

- Po usunięciu składowej $z = 1$, dostajemy punkt na płaszczyźnie – w układzie znormalizowanych współrzędnych.

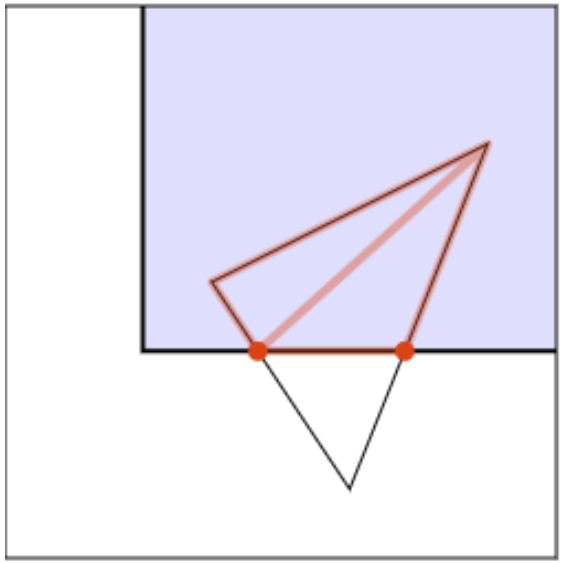
Znormalizowane współrzędne urządzenia

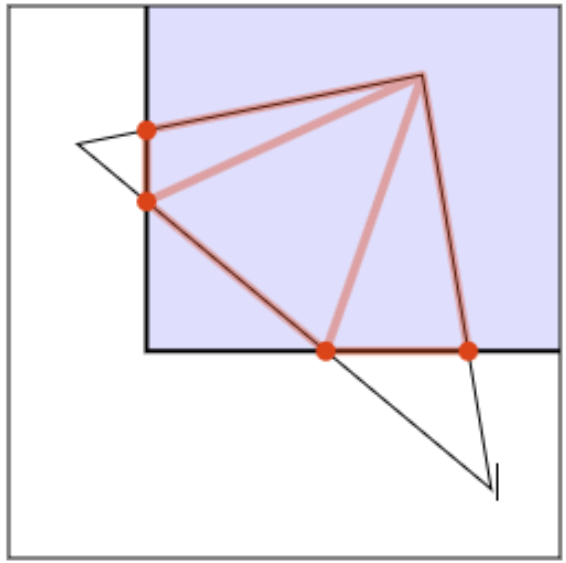
- Po tych wszystkich operacjach, dostajemy współrzędne znormalizowane
  – *normalized device coordinates* (NDC):
  - x: od -1 (lewa krawędź okna)
    do +1 (prawa krawędź)
  - y: od -1 (dolna krawędź)
    do +1 (górna krawędź)
  - w: od -1 (przednia płaszczyzna obcinania)
    do +1 (tylna płaszczyzna obcinania)



Obcinanie

- Punkty wewnątrz bryły widzenia mają współrzędne z zakresu [-1, 1].
- Tylko obiekty wewnątrz bryły widzenia „mieszczą się” w widoku (obrazie).
- Wszystkie trójkąty o wierzchołkach leżących poza bryłą widzenia są usuwane!
- Trójkąty leżące „na krawędzi” są przycinane (*clipping*).





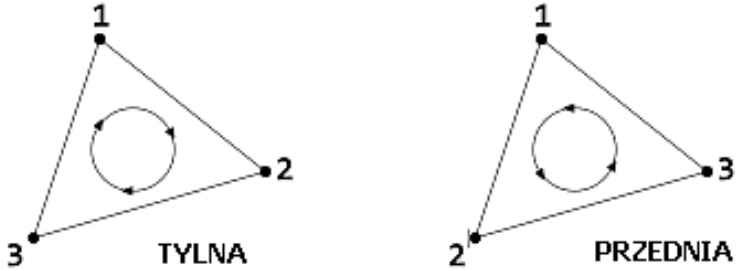
Usuwanie tylnych powierzchni

- Tylne powierzchnie obiektów (odwrócone od kamery) są niewidoczne i nie muszą być brane pod uwagę.
- Usuwanie tylnych powierzchni (*backface culling*) – operacja eliminacji trójkątów siatki obiektów, które są odwrócone od kamery.
- Operacja jest wykonywana przez system, ale należy ją aktywować.
- Zaleta: nie wykonujemy niepotrzebnych obliczeń i eliminujemy błędy.

Usuwanie tylnych powierzchni

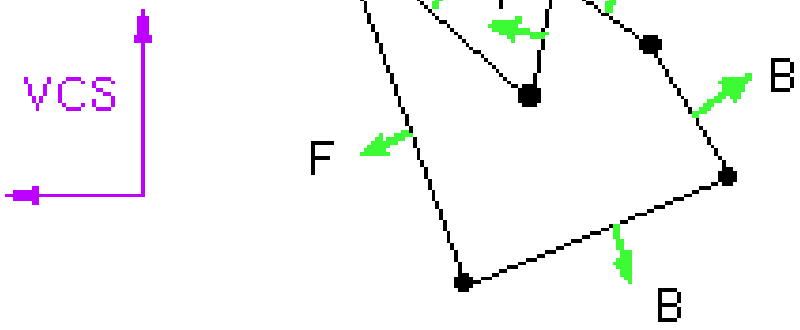
Co to znaczy „przednia powierzchnia trójkąta”?

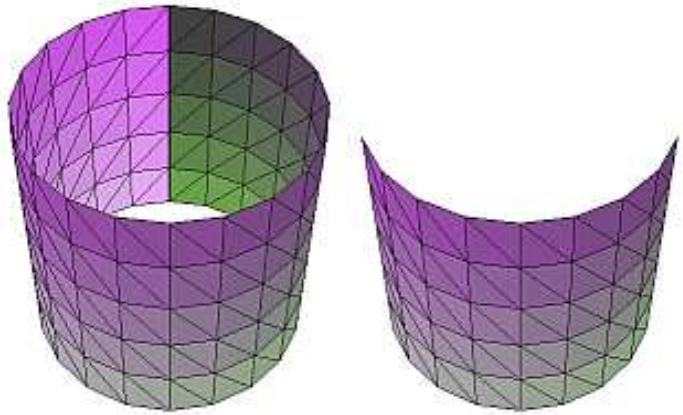
- „Nawinięcie” (*winding order*) jest to kolejność, w jakiej podawane są wierzchołki trójkąta.
- Domyślnie: ściana przednia to ta, dla której werteksy są podane w kolejności odwrotnej do ruchu wskazówek zegara.
- Można to zmienić, ale trzeba zachować tę samą konwencję dla wszystkich trójkątów.
- Wniosek: kolejność definiowania werteksów ma znaczenie!



Usuwanie tylnych powierzchni

- Wektor normalny trójkąta – wektor prostopadły do jego przedniej powierzchni.
- Obliczamy kąt między normalną trójkąta a kierunkiem osi *z* kamery.
- Jeśli kąt ≥ 90 stopni – powierzchnia jest niewidoczna, może być usunięta.



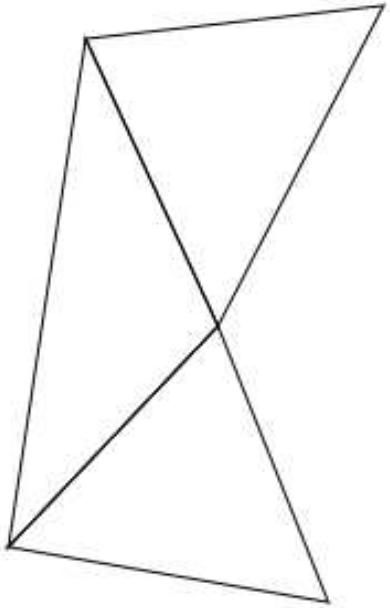


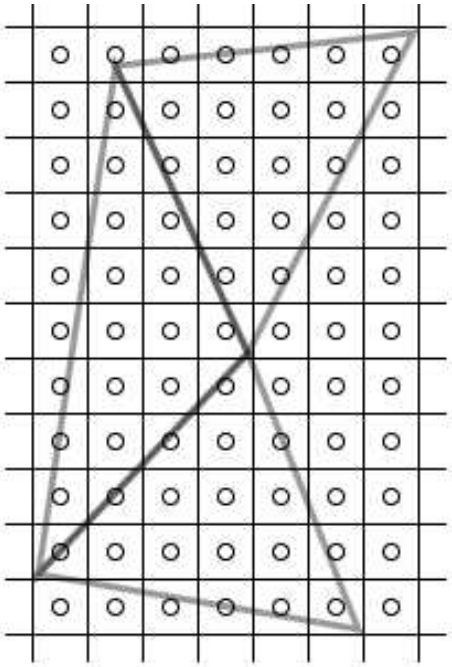
Rasteryzacja

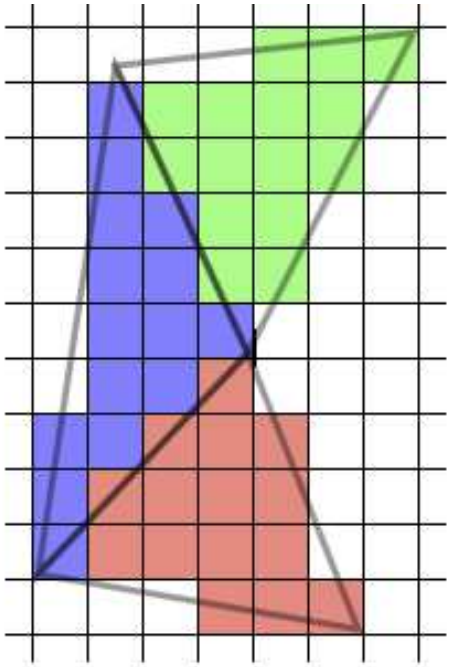
- Na widok po rzutowaniu nakładany jest raster – siatka o rozmiarach odpowiadających docelowemu obrazowi (np. 1920x1080).
- Współrzędne NDC są przeliczane do układu współrzędnych okna (w pikselach).
- Każdy trójkąt w rastrze jest skanowany rząd po rzędzie (*scanline conversion*).
- Z każdego punktu siatki rastra jest pobierany fragment (DirectX używa terminu *pixel*).
- Dla każdego fragmentu obliczana jest barwa.
- Barwa jest przenoszona na piksel obrazu.

Rasteryzacja

Przykład rasteryzacji trójkątów.







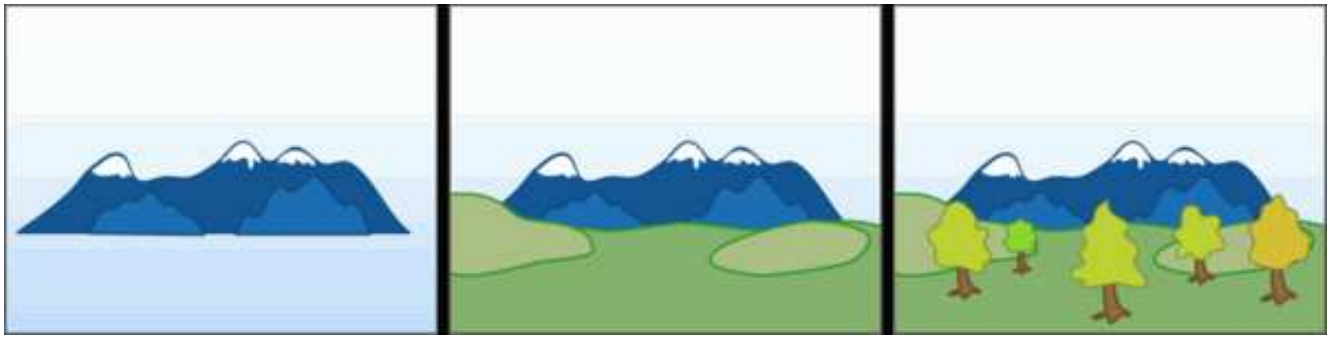
Rasteryzacja – bufor głębokości

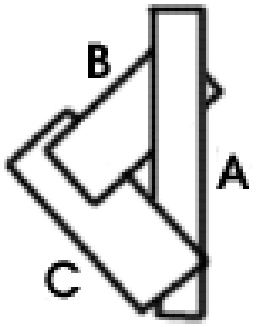
- Trójkąty na płaszczyźnie okna mogą się pokrywać, chociaż są w różnej odległości od kamery.
- Musimy pobrać fragment z trójkąta najbliższego kamerze (o najmniejszej *z*).
- Dlatego zachowaliśmy informację o głębokości w zmiennej *w*.
- Rasteryzer przegląda trójkąty po kolei. Jak znaleźć najbliższy kamerze?

Algorytm malarza

Algorytm malarza (*painter's algorithm*)
- naiwne podejście do problemu.

- Wszystkie trójkąty są sortowane według położenia na osi *z*.
- Trójkąty są „rysowane" w kolejności od najdalszego do najbliższego.
- Algorytm nie radzi sobie z przecinającymi się nawzajem trójkątami.



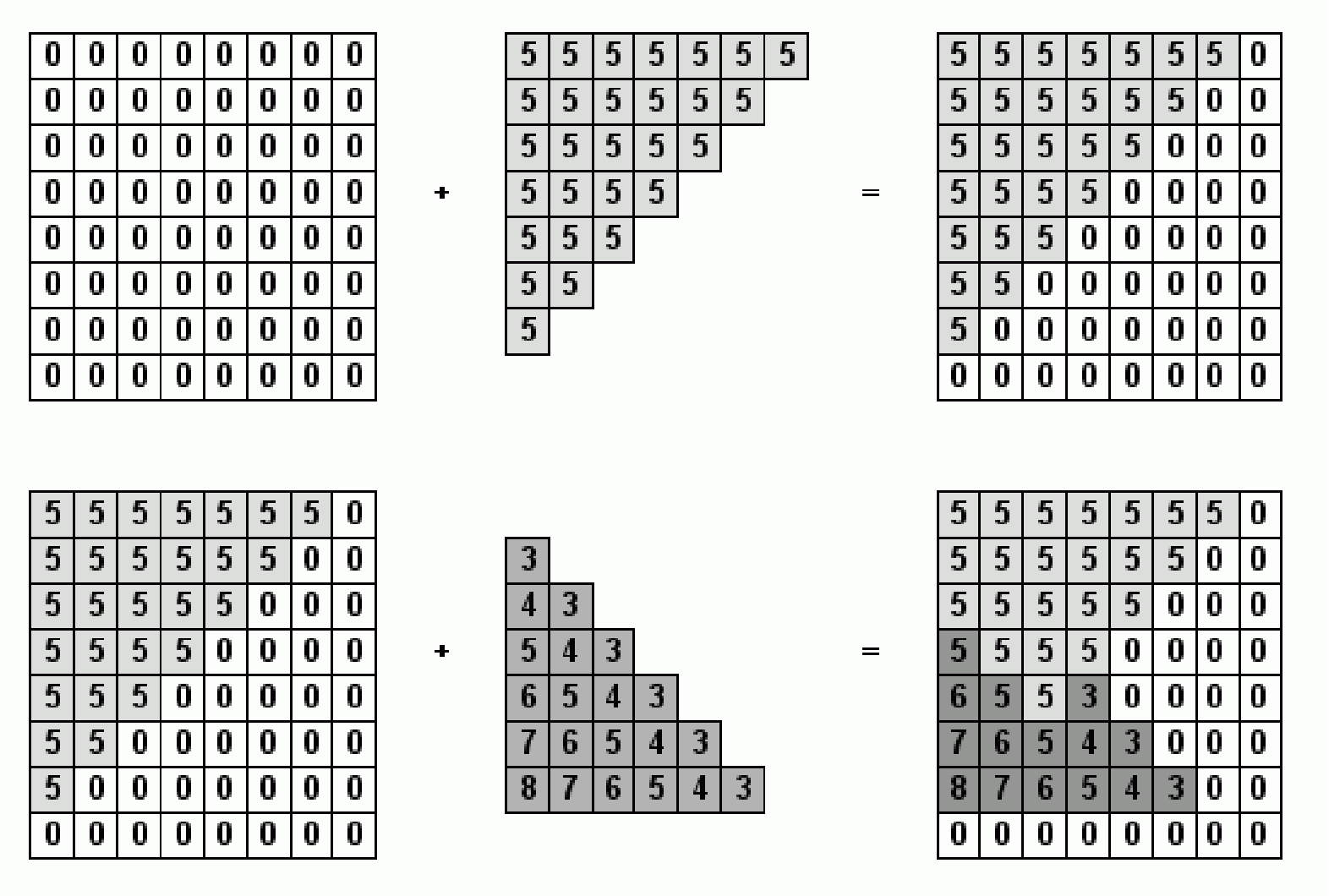


*Algorytm bufora głębokości (*z-buffer*)*

- Używamy dodatkowego obszaru pamięci – bufora głębokości (*z-buffer*). Jest on inicjalizowany wartością 1 (maksymalną).
- Dla każdego fragmentu, porównujemy jego wartość *z* (właściwie *w*) z zapisaną w buforze.
- Jeżeli bieżąca wartość jest większa – pomijamy fragment (jest on dalej niż inny, wcześniej przetworzony).
- Jeżeli jest mniejsza – przetwarzamy fragment i zapisujemy nową wartość *z* w buforze.

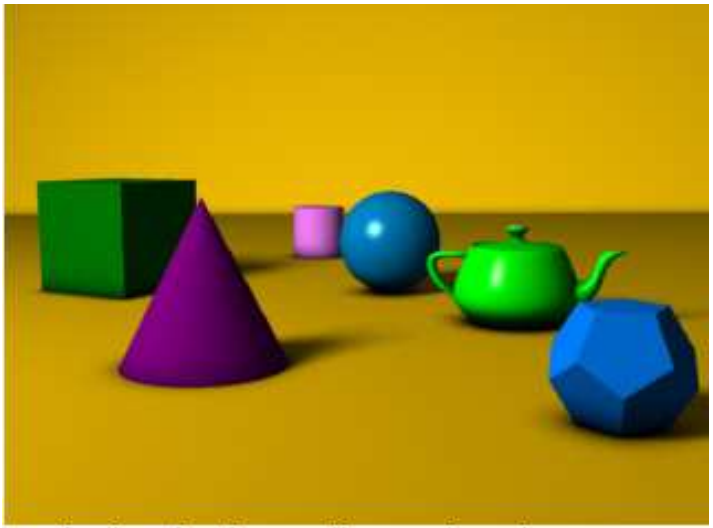
Algorytm bufora głębokości

Ilustracja działania metody (tutaj: większa wartość to bliższa odległość).



Algorytm bufora głębokości

Przykład obrazu i bufora głębokości.



A simple three dimensional scene

Z-buffer representation

Implementacja bufora *z* w kartach graficznych: sprzętowa, bufor 24-bitowy lub 32-bitowy (przy 16 bitach występują zniekształcenia).

Podsumowanie (1)

ETAP #1 – przekształcenia wierzchołków (*wykonuje: programista*)

- Dla każdego z obiektów:
  - obliczyć macierz przekształcenia świata $\mathbf{M_W}$ (obroty, skalowanie, translacja)
- Dla wszystkich obiektów (dla świata):
  - obliczyć macierz tr. widoku $\mathbf{M_V}$
  - obliczyć m. rzutu perspektywicznego $\mathbf{M_P}$
- Zastosować do każdego wierzchołka:

$$\mathbf{M} = \mathbf{M_P} \cdot \mathbf{M_V} \cdot \mathbf{M_W}$$

Podsumowanie (2)

ETAP #2 – przygotowanie do rasteryzacji (*wykonuje: system i GPU*)

- Dzielenie perspektywiczne.
- Usunięcie trójkątów poza bryłą widzenia.
- Przycięcie trójkątów na krawędzi bryły.
- Usunięcie tylnych powierzchni.

Podsumowanie (3)

ETAP #3 – rasteryzacja
(*wykonuje: system i GPU, z jednym wyjątkiem*)

- Nałożenie rastra i próbkowanie fragmentów.
- Interpolacja danych zapisanych w werteksach.
- Obliczenie barwy fragmentu (cieniowanie, teksturowanie, efekty specjalne, itp.)
  – *wykonuje programista*!
- Obsługa bufora głębokości.
- Algorytmy antyaliasingu.
- Zapisanie barwy piksela w buforze obrazu.