
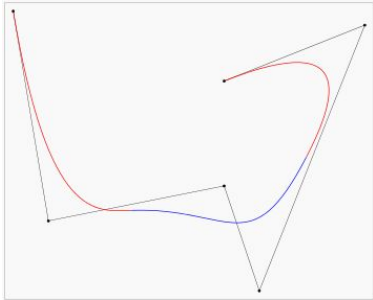


Pytania z krzywej Beziea

<p>Zad 1. Ile punktów kontrolnych ma krzywa Beziea drugiego stopnia? ODP: D 3 punkty kontrolne $W=1$ $PK=n+1$</p>	<p>Zad 11. Ile punktów kontrolnych ma krzywa Beziea trzeciego stopnia? ODP: D Żadna z wymienionych Ma 4 punkty kontrolne</p>
<p>Zad 12. Krzywa Beziea trzeciego stopnia o punktach kontrolnych p_0, p_1, p_2, p_3 jest styczna od odcinka ODP: a) p_0p_1 b) p_0p_2 c) p_0p_3 d) p_1p_2 e) do żadnego z wymienionych</p> <p>Rozwiązanie:</p>  <p>Krzywa Béziea trzeciego stopnia</p> <p>Jak widać P_0P_1 jest styczna z odcinkiem w punkcie P_0, jest jeszcze druga styczna P_2P_3 w punkcie P_3, ale nie jest wymieniona w odpowiedziach.</p>	<p>Zad 13. Za pomocą wymiernej krzywej Beziea ODP: a) nie można modelować krzywych stożkowych b) można modelować krzywe stożkowe tylko umieszczając część punktów kontrolnych w nieskończoności c) można modelować krzywe stożkowe nie umieszczając punktów kontrolnych w nieskończoności</p> <p>Rozwiązanie: Modelować można krzywe stożkowe w wymiernej. W teorii Denisjuka: Punkty kontrolne mogą być umieszczone w nieskończoności.</p> <p>Mogą, czyli nie muszą.</p>
<p>Zad Krzywa B-sklejana (B-spline) trzeciego stopnia o punktach kontrolnych p_0, p_1, p_2, p_3 jest styczna do odcinka a) p_0p_1 b) p_0p_2 c) p_0p_3 d) p_1p_2 e) do żadnego z wymienionych</p> 	<p>Zad 15 Powierzchnia NURBS</p> <p>a) zawsze zawiera się w otoczce wypukłej swoich punktów kontrolnych b) zawiera się w otoczce wypukłej swoich punktów kontrolnych tylko w przypadku gdy nie ma punktów kontrolnych w nieskończoności c) zawiera się w otoczce wypukłej swoich punktów kontrolnych tylko w przypadku gdy nie ma wielokrotnych punktów kontrolnych d) nie koniecznie zawiera się w otoczce wypukłej swoich punktów kontrolnych</p>

krzywe drugiego stopnia (trzy punkty kontrolne, np. fonty TrueType) lub trzeciego (cztery punkty kontrolne, np. fonty Type1, METAFONT, SVG, cała gama różnych pakietów graficznych)

Pytania z Aliasingu i Antialiasingu

<p>Zad2. Zjawisko Aliasinga występuje przy teksturowaniu <i>ODP: D aliasing nie występuje przy teksturowaniu</i> Ponieważ Aliasing występuje po Teksturowaniu, czyli przy Mapowaniu</p>	<p>Zad 8. Mipmapping pozwala na <i>ODP: Zwalcza zjawisko aliasinga w przypadku gdy rozdzielczość tekstury jest większa od rozdzielczości ekranu</i></p>
	<p>Zad 9. Nadpróbkowanie (supersampling) pozwala na <i>ODP: Zwalcza zjawisko aliasinga w przypadku gdy rozdzielczość tekstury jest większa od rozdzielczości ekranu</i></p>

Pytania z formatów graficznych i przekształceń kolorów

<p>Zad3.Co jest przyczyną straty danych w formacie PNG</p> <p>ODP: Nie ma czegoś takiego, PNG jest bezstratnym formatem</p>	<p>Zad4. Który z poniżej wymienionych formatów plików graficznych jest formatem grafiki wektorowej?</p> <p>ODP: SVG</p> <p>Zad 7. Grafika wektorowa jest to taki sposób prezentacji obrazów w którym</p> <p>ODP: obraz składa się z obiektów takich jak odcinek i tworzenie obrazu polega na rysowaniu tych obiektów.</p> <p>ODP. D teoria rysowania grafiki wektorowej, zbiór linii które są rysowane i cały ten zbiór tworzy obraz</p>												
<p>Zad8. Który z poniżej wymienionych modeli syntezy barw, jest modelem technicznym wykorzystywanym przy generacji obrazu na monitorach i projektorach</p> <p>ODP: RGB</p>	<p>Zad. 10 Barwie (1,0,0) w Modelu RGB odpowiada w modelu CMY Barwa</p> <p>ODP: (0,1,1)</p> <p>Rozwiązanie Negacja kolorów</p>												
<table><tr><td>RGB</td><td>CMYK</td><td>Nazwa</td></tr><tr><td>0 , 0 , 0</td><td>= 0,0,0,1</td><td>CZARNY</td></tr><tr><td>0, 0, 255</td><td>= 1,1,0</td><td>NIEBIESKI</td></tr><tr><td>0, 255, 0</td><td>= 1,0,1</td><td>ZIELONY</td></tr></table>	RGB	CMYK	Nazwa	0 , 0 , 0	= 0,0,0,1	CZARNY	0, 0, 255	= 1,1,0	NIEBIESKI	0, 255, 0	= 1,0,1	ZIELONY	<p>Zad9. Żółty (trzeba znać podstawowe kolory Red,Green,Blue i wszystkie kombinacje) Żółty to (255,255,0) w CMYK</p>
RGB	CMYK	Nazwa											
0 , 0 , 0	= 0,0,0,1	CZARNY											
0, 0, 255	= 1,1,0	NIEBIESKI											
0, 255, 0	= 1,0,1	ZIELONY											

0, 255, 255 = 1,0,0 CYAN 255, 0, 0 = 0,1,1 CZERWONY 255, 0, 255 = 0,1,0 MAGENTA 255, 255, 0 = 0,0,1 ŻÓŁTY 255, 255, 255 = 0, 0, 0, 0 BIAŁY	to ($C=1-R, M=1-G, Y=1-B$) gdzie 255 = 100% czyli 1 skala 0-1 ($C=1-1, M=1-1, Y=1-0$) = 0,0,1 ODP B (0,0,1)
ZAD 16 Korekcja Gamma służy Odp B . Do usuwania zniekształceń wprowadzonych przez monitor	Zad10 W języku POSTSCRIPT 1 - 2 zapisuje się jako ODP: 1 2 - ODP. Tu nie ma co tłumaczyć, reszta jest źle i chuj, wszystkie komendy wydaje się na końcu A B COMMAND 10 5 moveto

Pytania z postrzegania barw

Zad5 Receptory pręcikowe w oku ludzkim ODP: reagują już przy niskim poziomie oświetlenia i nie pozwalają na rozróżnianie barw	Rozwiązanie: Pręciki pozwalają na widzenie skotopowe (w czarno bieli).Czopki pozwalają na widzenie widzenie fotopowe (z barwami).
---	--

Pytania z geometrii maszynowej, obrót, skalowanie i przesunięcie równoległe

Zad. Obrót o 90* dookoła Osi Ox jest określony macierzą: Sin90 = 1 Cos90 = 0 Macierz: 1 0 0 0 0 -1 0 1 0 $R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$	Wzór: $R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$ $R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$ $R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ Inne wartości które warto znać: sin45= 0,70 cos45=0,70 sin30= 0,5 cos30=0,87 sin60= 0,87 cos60=0,5
--	---

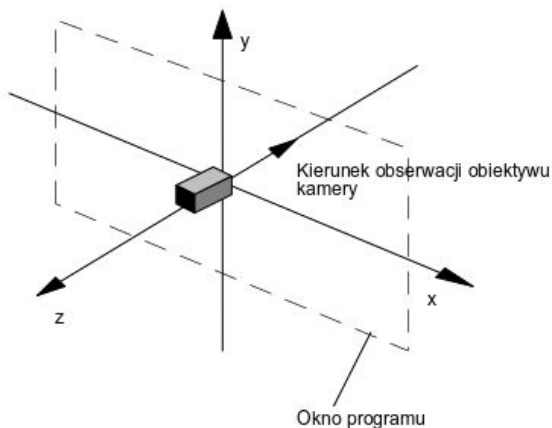
Pytania związane z Oświetleniem

<p>Zad 11 Niech materiał będzie miał współczynnik odbicia światła 0,78. Kąt odbicia</p> <p>a) Nie zmienia się</p> <p>Opowiedź: Współczynnik odbicia to stosunek wielkości strumienia danego rodzaju promieniowania odbitego od ciała do analogicznego strumienia padającego</p>	<p>Zad 18 W modelu Phong'a natężenie światła odbijanego zwierciadłanie zależy od:</p> <ul style="list-style-type: none"> a) wektora normalnego do powierzchni odbijania b) "kierunka :D " do obserwatora c) kierunku do źródła światła d) wszystkich wymienionych wektorów e) tylko 2 z wymienionych wektorów f) tylko jednego z wymienionych wektorów g) nie zależy od wymienionych wektorów <p>ODP : Zależy od kąta między promieniem odbitym a kierunkiem do obserwatora . im mniejszy kąt tym mocniejszy</p>
<p>Zad 4. W modelu Phong'a natężenie światła odbijanego rozpozno w punkcie zależy od</p> <ul style="list-style-type: none"> a) Wektora normalnego do powierzchni odbijania b) Kierunka do obserwatora c) Obydwóch wektorów d) Nie zależny od wymienionych czynników 	<p>Zad 5. Która z dwóch powierzchni (przy identycznych pozostałych warunkach) będzie miała połysk o większej średnicy</p> <ul style="list-style-type: none"> a) o wskaźniku odbijania 60 b) o wskaźniku odbijania 50 c) będą miały identyczne połyski
<p>Zad 16. Natężenie światła załamane rozporoszone w rozszerzonym modelu Phong'a zależy od</p> <ul style="list-style-type: none"> a) Wektora normalnego do powierzchni załamania b) Kierunka do obserwatora c) Obydwóch wektorów d) Nie zależy od wymienionych wektorów 	<p>Zad 19. Przy załamaniu promienia na granicy powietrze->woda kąt załamania</p> <ul style="list-style-type: none"> a) powiększa się b) zmniejsza się c) nie zmniejsza się <p>ODP: Jeżeli światło przechodzi z ośrodka o mniejszym współczynniku załamania światła do ośrodka o współczynniku większym (np. powietrze-woda), to kąt załamania jest mniejszy od kąta padania.</p>

Pytania z OpenGL

Zad 1. OpenGL ma układ współrzędnych taki, że oś y jest skierowana (względem monitoru)

- a) w dół
- b) w górę**
- c) w lewo
- d) w prawo
- e) w kierunku widza
- f) w kierunku od widza



Zad 17. Do eliminacji zasłoniętych powierzchni OpenGL używa się:

- a) Algorytmu malarza
- b) Zmodyfikowanego algorytmu malarza
- c) Algorytmu Bufora głębokości**
- d) OpenGL nie ma wbudowanego algorytmu eliminacji zasłoniętych powierzchni.

ODP: C

```
glutInitDisplayMode(GLUT_DOUBLE |
GLUT_RGB | GLUT_DEPTH);
// podwójny bufor ekranu (rysowanie na
aktualnie NIE wyświetlanym)
// kolory w trybie RGB (właściwie RGBA)
// bufor głębokości związany z oknem
(do eliminacji zasłoniętych powierzchni)
```

Zad. Następujący fragment kodu

```
glVertex3f(0,0,1);
glVertex3f(1,0,0);
glVertex3f(0,1,1);
```

- a) trójkąt
- b) trzy punkty
- c) trzy odcinki
- d) dwa odcinki
- e) nie można udzielić odpowiedzi**

Tak na logikę, to jest utworzenie trzech wektorów, żeby je wyświetlić trzeba by było wysłać je do VBO.

Pytania z Resteryzacji odcinka

Zad 6. Który z poniższych punktów zostanie wyświetlony przy rasteryzacji odcinka [(1,-1),(10,5)] algorytmem Bresenhama:

- a) (4,0)
- b) (4,1)
- c) (4,2)
- d) (4,3)

```
def AlgorytmBresenhama(i1,j1,i2,j2):
    m = 2*(j2-j1)
    b = 0
    print "%d , %d" %(i1,j1)
    j = j1
    P = i2-i1
    for i in range(i1+1,i2):
        b=b+m
        if b>P:
            j=j+1
            b= b- 2 * P
        print(i,j)
```

Zad 19 Dany jest romb ABCD. Wierzchołki rombu mają współrzędne tekstuowe odpowiednio A(1/8,1/8) B(3/4,1/4) C(3/4,3/4) D(1/8,1/2) Jakie współrzędne tekstuowe będzie miał środek boku AB po resteryzacji

Oddповідź (x,y) : x = 7/16 y = 3/16

```
def AlgorytmBresenhama(i1,j1,i2,j2):
    m = (j2-j1)/(i2-i1)
    print "%d , %d" %(i1,j1)
    y = j1
    P = i2-i1
    for i in range(i1+1,i2):
        y=y+m
        j=round(y)
        print(i,j)
```

$$x = \frac{\lambda_2 x_1 + \lambda_1 x_2}{\lambda_1 + \lambda_2}$$

Podział odcinka na dwie części w proporcji lambda1 i lambda2, w tym wypadku (1:1), dla współrzędnej y - analogicznie

Pytania z Rzutowania

Zad 14 Przy rzutowaniu prostopadłym równe odcinki będą mieć tę samą długość na ekranie

- a) zawsze
- b) jeżeli są równoległe
- c) jeżeli są równoległe i równoodległe od kamery

Zad 18. Zeby powiększyć głębie ostrości w metodzie śledzenia promieni stosuje się

- a) nadpróbkiowanie
- b) techniki oparte na metodzie energetycznej (radiosity)
- c) metoda śledzenia promieni nie ma ograniczenia na głębie ostrości

<p>d) jeżeli są równoodległe od kamery e) nigdy</p> <p>Te które są równoległe do kamery będą miały ten sam rozmiar, te które prostopadle padają do kamery będą poprostu 1 pikselem bo nie będziesz widział ich, w B brakuje od kamery, bo równoległe mogą być do siebie a nie do kamery więc będą zmienione czyli C</p>	
---	--

Pytania z algorytmu obcinania i okienkowania

<p>Zad 20 Dane jest okno $-1 \leq x \leq 1$, $-1 \leq y \leq 1$. Jaki kod będzie miał wierzchołek (0,0) w algorytmie Sutherlanda-Cohana.</p> <p>Odpowiedz : 0000</p>	
--	--

Nie wiem które kategorie

<p>Zad. Jakie nieporządane zjawisko może się pojawić przy standardowym renderowaniu trójkąta, wierzchołki którego mają kolorY: biały, czarny, biały?</p> <p>a) Cieniowanie kolorów będzie zależne od obrotu trójkąta b) Cieniowanie kolorów może mieć nieciągłości c) Żadne z</p>	<p>Zad 6 Wierzchołek A trójkąta ABC ma współrzędne barycentryczne</p> <p>a) (1,0,0) b) (0,0,1) c) (0,1,0) d) (0,0,0)</p>
--	---

wymienionych d) Obydwa wymienione	
Zad. Zbiór punktów odpowiadających kombinacjom wypukłym punktów A,B,C c R^3 zgadza się z a) pustym zbiorem b) trójkątem ABC c) płaszczyzną ABC d) przestrzenią R^3	Zad13 Wektor prostopadły do strefy o środku w początku układu współrzędnych i promieniu $\sqrt{3}$ w punkcie (-1,1,1) to a) (-1,1,1) b) (1,-1,1) c) (1,1,-1) d) (-1,1,-1) e) (-1,-1,1) f) (1,-1,-1) ? g) (-1,-1,-1) h) (1,1,1) • Sfera kuli Π o równaniu ogólnym $(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 = R^2$ w punkcie $(x,y,z) \in \Pi$ $\vec{n} = [\lambda(x-x_0), \lambda(y-y_0), \lambda(z-z_0)]$; $\lambda \in \mathbb{R} \setminus \{0\}$ wzór na normalną sfery
	<u>Zad 12 Dla wektorów i,j,k bazy kartezjańskiej i - i jest równe</u> <u>Odpowiedz : (0,0,0)</u> $i = (1,0,0)$ $j = (0,1,0)$ $k = (0,0,1)$ $i - i = (1-1, 0-1, 0-1) = (0, 0,0)$
Zad 15 Punkt (1,1) na płaszczyźnie jest przedstawiany we współrzędnych jednorodnych jako a)(1:1:0) b)(1:1:1) c)(1:1:2) d) każdy z wymienionych e) żaden z wymienionych ODP B	Jeśli współrzędne punktu (X, Y, W) podzielimy przez W (przy założeniu $W \neq 0$), to punkt ten będzie miał współrzędne $(X/W, Y/W, 1) = (x, y, 1)$. Czyli punkt ten odpowiada punktowi (x, y) z płaszczyzny R^2 .

Najważniejsze elementy z opracowania

1. Percepcja wizualna i modele barw

1.1 Modele barw popularne (zawrzeć trzeba tu pytania związane z okiem ludzkim)

1.1.1 RGB

Każdy go zna, jedynie trzeba wiedzieć że jest to model syntezy addytywnej, czyli najniższe wartości oznaczają barwę czarną, zaś najwyższe białą

Zakres 0-255

32(24) bitów = 8R + 8G + 8B + 8alpha

16 bitów = 5R + 5G + 5B + 1alpha

8bitów = 3R + 3G + 2B

1.1.2 CLUT (jakieś głupoty)

Barwy indeksowane.

1.1.3 WEB-SAFE COLORS

Tu trzeba wiedzieć tylko tyle że jest to paleta kolorów które pomimo ustawień na karcie graficznej, wszędzie będą wyświetlane identycznie. Kolorów tych chyba wszystkich znać nie trzeba. Przykładem takiego koloru jest np CZARNY :D

A dokładnie:

Każda liczba złożona z par 00, 33, 66, 99, CC oraz FF odpowiada barwie bezpiecznej np #990033 albo #336600 itp.

1.1.4 CMYK

Farmy drukarskie C-CYJAN, M-MAGENTA, Y-Zółty, K-Czarny

Zakres 0%-100%

Konwersja z RGB

$C = 1 - R$

$M = 1 - G$

$Y = 1 - B$

1.1.5 Oko ludzkie

Pręciki - oświetlenie słabe

Czopki - oświetlenie intensywne, trzy rodzaje

Percepcja barwy

Nośnikiem percepcji wizualnej jest światło

Wrażenie wzrokowe wywołuje cały skład widma wpadającego do oka

Sposób percepcji zależy od:

- właściwości źródła światła
- właściwości ośrodka i odległości między źródłem a obiektem
- zdolności fizycznych obiektu do odbijania i/lub pochłaniania światła o określonej długości fal
- właściwości otaczających obiektów
- stanu oka i systemu wzrokowego

1.2 Telewizyjne

1.2.1 YUV Nie wiem czy jest sens

1.2.2 YUQ Nie wiem czy jest sens

1.2.3 YCbCr Nie wiem czy jest sens

1.3 Modele stożkowe

1.3.1 HSV

HUE - Odcień (0° - 360°) (paleta koloru czerwony żółty zielony niebieski fioletowy i pochodne)

Saturation - nasycenie (0-1) (promień podstawy im większe nasycenie tym mocniejszy odczujemy kolor)

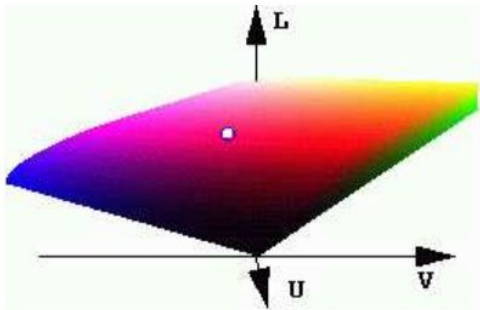
Value (Brightness) - jasność (moc światła białego)

1.3.2 HSL

To samo co HSV z jedynie różnicą taka że Value zamienia się w Lightness i przy 100% Jasności, zamiast uzyskać jakąś barwę bardzo jasną, uzyskujemy po prostu kolor biały

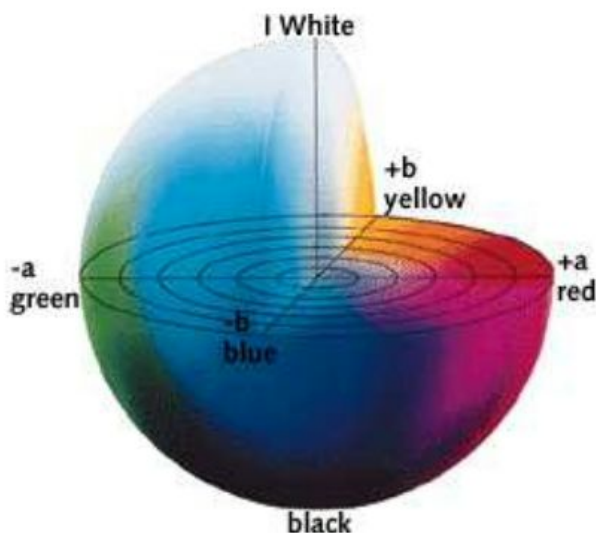
1.4 Modele percepcyjnie równomierne

1.4.1 CIE-LUV - jeden z modeli [przestrzeni barw](#), zmierzający do zlinearyzowania [percepcji](#) różnic kolorów przez oko ludzkie (przedstawienie różnic kolorów za pomocą układu liniowego).



Linearyzacja - polega na przybliżeniu modelu [układu nieliniowego](#) za pomocą modelu [układu liniowego](#).

1.4.2 CIE-LAB - [przestrzeń barw](#), stanowi matematyczną transformację przestrzeni [CIEXYZ](#). Zatem CIELab z założenia miała być [równomierną przestrzenią barw](#)



1.5 Przekształcanie modeli

Znać algorytmów nie trzeba, wystarczy nauczyć się jaki kolor ma jaki kod

1.5.1 RGB-HSL(HSV)

RGB	HSL	Nazwa
0, 0, 0	= 0 0% 0%	CZARNY
0, 0, 255	= 240 100% 50%	NIEBIESKI
0, 255, 0	= 120 100% 50%	ZIELONY
0, 255, 255	= 180 100% 50%	CYAN
255, 0, 0	= 0 100% 50%	CZERWONY
255, 0, 255	= 300 100% 50%	MAGENTA
255, 255, 0	= 60 100% 50%	ZÓŁTY
255, 255, 255	= 0 0% 100%	BIAŁY

1.5.2 RGB - CMY

RGB	CMYK	Nazwa
0, 0, 0	= 0,0,0,1	CZARNY
0, 0, 255	= 1,1,0	NIEBIESKI
0, 255, 0	= 1,0,1	ZIELONY
0, 255, 255	= 1,0,0	CYAN
255, 0, 0	= 0,1,1	CZERWONY
255, 0, 255	= 0,1,0	MAGENTA
255, 255, 0	= 0,0,1	ZÓŁTY
255, 255, 255	= 0, 0, 0, 0	BIAŁY

1.5.3 RGB-YUV Nie wiem czy jest sens

1.5.3 RGB-YCbCr Nie wiem czy jest sens

2. Formaty kompresji plików graficznych i kompresji obrazów

2.1 SVG

Skalowalne wektory czyli linie, oparty na XMLu, wspiera animacje

2.1.1 Filtry

Obsługuje filtry graficzne, które mogą wpływać na grafikę. **Umożliwia to dodanie do obrazu różnych efektów jak np. Rozmycie Gaussa czy Gradient.**

2.2 PostScript

Język stosowany przy maszynach wycinających laserowo (poligraficzne urządzenia), komendy wydaje się podając najpierw współrzędne potem komenda (odwrotna notacja, najpierw argumenty potem komenda) np, **100 200 moveto, 20 5 div**, więc w pytaniu z testu będzie

1 2 -

2.2.1 EPS

Format plików, będący podzbiorem PostScript, służy do przechowywania pojedynczych stron grafiki wektorowej w postaci umożliwiającej osadzenie ich w innych dokumentach. Wydaje mi się że ma to na celu umożliwić np osadzenie grafiki w innym języku np w C wyświetlić na ekranie (podgląd)

2.3 TIFF

Tego nie będzie mało używany, format grafiki rastrowej, ma kompresję stratną i bezstratną.

2.4 PNG

Rastrowy format plików graficznych oraz system bezstratnej kompresji danych graficznych. Obsługuje przezroczystość przez kanał alfa

2.4.1 Korekcja gamma

Korekcja gammy - poprawa czytelności obrazu, dla wartości poniżej 1 następuje przyciemnienie obrazu, zaś dla wartości powyżej 1 następuje rozjaśnienie obrazu. Usuwa zniekształcenia wprowadzone przez urządzenia (np. [monitor](#), [skaner](#)) poprzez redukcję nadmiernego kontrastu obrazu wejściowego.

2.4.2 Adam7

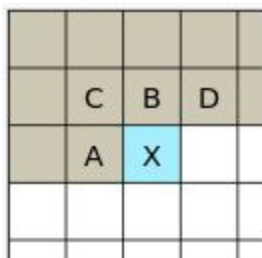
Adam7 to [algorytm przeplotu](#) stosowany w obrazach **PNG**. Przeplatany obraz PNG podzielony jest na 7 pod obrazów, z których każdy tworzony jest dzięki replice wzoru o wymiarach 8x8 na całym obrazie docelowym. Podobrazy są następnie zapisywane w pliku PNG w kolejności rosnącej. Posługuje się siedmioma przebiegami i działa na dwóch wymiarach, umożliwia to szybszy podgląd niż w 4-przebiegowym GIFie

2.4.3 Kwantyzacja (zamiast sprawdzania poprawności) bo to się pojawiło w pytaniach

– działanie polegające na zmniejszeniu liczby kolorów w [obrazie rastrowym](#) przy zachowaniu możliwie najwierniejszego ich odwzorowania. Ma na celu czytelną prezentację obrazów zawierających dużą liczbę kolorów na urządzeniach graficznych bądź w [formatach plików](#) zdolnych prezentować niewielką liczbę kolorów

2.4.4 Filtracja

Aplikowanie prostych filtrów, które upraszczają dane obrazu przed kompresją, by zwiększyć wydajność. Jest wykonywana przed właściwą kompresją. 5 typów



NONE - brak filtracji,

SUB- optymalizuje kompresję obrazów z poziomymi wzorcami lub przejściami tonalnymi, czyli przyrównuje do lewej, mamy pixel czarny(X) a z lewej czerwony(A) to je ujednolica by nie było kontrastu

UP - optymalizuje kompresję obrazów z równymi wzorami pionowymi, bierze pixel z góry(B) i przyrównuje go z naszym pixelem czarnym(X)

Average - poprawia zakłócenia poprzez wyrównanie wartości przyległych pixeli czyli jak jest pixel czarny(X), a przyległe czerwone (A) (B) , no to środkowy będzie miał jakiś tam brązowy, a otaczające będą ciemnoczerwone

Paeth - poprawa kolorów pixeli przyległych tyle że mamy pixel (X) i pixele (A) (B) (C) i optymalizujemy w taki sposób $\text{kolor} = A + B - C$

2.5 JPEG

- kompresja stratna
- używany na stronach WWW

Obraz jest konwertowany z kanałów czerwony-zielony-niebieski (RGB) na jasność (luminancji) i 2 kanały barwy (chrominancje).

Każdy z kanałów (jasność i 2 kanały barw) jest dzielony na bloki (8x8). Na takich blokach wykonywana jest DTC. Zamiast wartości pikseli posiadamy teraz informacje o średniej wartości pikseli i częstotliwości ich zmiany.

Potem dane te poddawane są kwantyzacji (z liczb zmiennoprzecinkowych następuje zamiana na całkowite).

Współczynniki DTC uporządkowane są tak, aby zera leżały obok (zygzakowato) siebie dzięki czemu dobrze się kompresują.

Wykorzystujemy algorytm Huffmana. Dzięki algorytmowi możemy ustalić stopień kompresji

Dyskretna transformacja cosinusowa -Zaletą stosowania transformaty DCT w kompresji jest to, że większość współczynników jest zwykle bliska 0 – po kwantyzacji wyzerują się, co redukuje liczbę bitów potrzebną do reprezentacji sygnału bez wnoszenia dużego błędu.

3. Rasteryzacja

3.1 Rasteryzacja odcinka (były 2 pytania ze środkiem odcinka i punktem na linii)

Tu trzeba nauczyć się algorytmu innego sposobu nie widze, Algorytm Bresenhama. Dla liczb całkowitych:

```
def AlgorytmBresenhama(i1,j1,i2,j2):  
    m = 2*(j2-j1)  
    b = 0  
    print "%d , %d" %(i1,j1)  
    j = j1  
    P = i2-i1  
    for i in range(i1+1,i2):  
        b=b+m  
        if b>P:  
            j=j+1  
            b= b- 2 * P  
        print(i,j)
```

było też pytanie z ułamkami czyli liczbami rzeczywistymi, w tym wypadku trzeba je wyeliminować:

```
def AlgorytmBresenhama(i1,j1,i2,j2):  
    m = (j2-j1)/(i2-i1)  
    print "%d , %d" %(i1,j1)  
    y = j1  
    P = i2-i1  
    for i in range(i1+1,i2):  
        y=y+m  
        j=round(y)  
        print(i,j)
```

3.2 Rasteryzacja okręgu (może być bo łatwy wzór)

Aby narysować okrąg wystarczy poddać rasteryzacji tylko połowę ćwiartki okręgu. Reszta powstaje przez symetrię. Zaczyna w wierzchołku (0,R). Później algorytm będzie poszukiwał następnego piksela do zamalowania.

Algorithm 4 Algorytm rasteryzacji okręgu

Wejście: Okrąg o promieniu R , którego środek znajduje się w $(0, 0)$

Wynik: Wyświetlenie okręgu

```
i := 0
j := R
f := 5 - 4R
zamaluj(i, j)
while i ≤ j do
  if f > 0 then
    f := f + 8i - 8j + 20
    j := j - 1
  else
    f := f + 8i + 12
  end if
  i := i + 1
  zamaluj(i, j)
end while
```

3.3 Rasteryzacja elipsy (za dużo liczenia nie będzie na egzaminie, no)

3.4 Wypełnianie (nie wydaje mi się żeby się pojawiło)

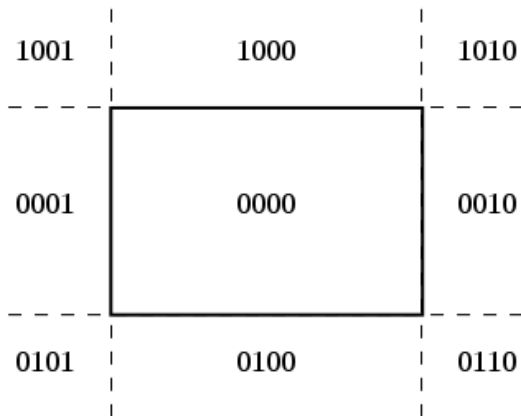
4. Algorytm obcinania i okienkowania

4.1 Odcinanie prostych i odcinków

4.1.1 Algorytm Sutherlanda-Cohena (było pytanie więc tylko na tym się skupiamy)

Prostokąt obcinający jest wyznaczony przez cztery proste równoległe do osi: x_{\min} , x_{\max} , y_{\min} , y_{\max} . Dzielą one płaszczyznę na 9 obszarów, którym przypisuje się 4-bitowe kody, każdy bit opisuje położenie punktu względem danej prostej.

- bit nr 0 ma wartość jeden, gdy $x < x_{\min}$ (punkt znajduje się po lewej prostokąta);
- bit nr 1 ma wartość jeden, gdy $x > x_{\max}$ (po prawej);
- bit nr 2 ma wartość jeden, gdy $y < y_{\min}$ (poniżej);
- bit nr 3 ma wartość jeden, gdy $y > y_{\max}$ (powyżej).



4.1.2 Algorytm Lianga-Barsky'ego (olał bym to)

4.2 Algorytm Sutherland-Hodgmana (olał bym to)

4.3 Algorytm Weilera-Athertona (olał bym to)

5. Geometria Maszynowa

5.1 Macierz przekształcenia afinicznego

5.1.1 Obrót (było pytanie)

$$\text{Wzór: } \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Przykłady dla popularnych wartości 90° 30° 45° i 60°

Sin90 = 1 więc macierz będzie miała wartości: $\begin{vmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix}$
Cos90 = 0

sin30= 0,5 **cos30**=0,87 macierz: $\begin{vmatrix} 0,87 & -0,5 & 0 \\ 0,5 & 0,87 & 0 \\ 0 & 0 & 1 \end{vmatrix}$

sin45= 0,70 **cos45**=0,70 macierz: $\begin{vmatrix} 0,70 & -0,70 & 0 \\ 0,70 & 0,70 & 0 \\ 0 & 0 & 1 \end{vmatrix}$

sin60= 0,87 **cos60**=0,5 macierz: $\begin{vmatrix} 0,5 & -0,87 & 0 \\ 0,87 & 0,5 & 0 \\ 0 & 0 & 1 \end{vmatrix}$

5.1.2 Skalowanie (może być)

Polega na zmianie wymiarów obiektu o współczynnik lambda

$$\begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \lambda_1 * x \\ \lambda_2 * y \\ 1 \end{pmatrix}$$

Przykład:

Załóżmy, że chcemy poddać trójkąt skalowaniu $\lambda_1 = 0,5$, $\lambda_2 = 1,5$. Należy wtedy po kolei obliczyć współrzędne każdego wierzchołka. Jeżeli jednym z wierzchołków jest punkt $A = (4, 4)$, to otrzymamy:

$$\begin{pmatrix} 0,5 & 0 & 0 \\ 0 & 1,5 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 \\ 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 0,5 * 4 \\ 1,5 * 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 6 \\ 1 \end{pmatrix}.$$

Po przekształceniu $A = (2, 6)$. Analogicznie można obliczyć położenie pozostałych wierzchołków.

Tyle w temacie

5.1.3 Przesunięcie równoległe (może być)

$$\begin{pmatrix} 1 & 0 & u_1 \\ 0 & 1 & u_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + u_1 \\ y + u_2 \\ 1 \end{pmatrix}$$

Załóżmy, że chcemy przesunąć równoległe trójkąt o 3 na osi X i o 4 na osi Y. Należy wtedy po kolei obliczyć współrzędne każdego wierzchołka. Jeżeli jednym z wierzchołków jest punkt $A = (4, 4)$, to otrzymamy:

$$\begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 \\ 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 + 3 \\ 4 + 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \\ 1 \end{pmatrix}.$$

Po przekształceniu $A = (7, 8)$. Analogicznie można obliczyć położenie pozostałych wierzchołków.

6. Rzutowanie


6.1 Algorytm malarza - algorytm służący do wyznaczania powierzchni widocznych. Polega na rysowaniu obiektów w kolejności od najdalszego do najbliższego, podobnie jak malarz przedstawia bliższe obiekty malując je na namalowanych wcześniej, bardziej odległych. Fragmenty są następnie rysowane w tej kolejności.

Zasada działania algorytmu malarza jest następująca:



1. Posortuj ściany w kolejności od najdalszej do najbliższej obserwatora,
2. Namaluj je w tej kolejności.


Może występować zakleszczenie, wtedy algorytm nie gwarantuje, poprawności narysowania odpowiedniego obrazu

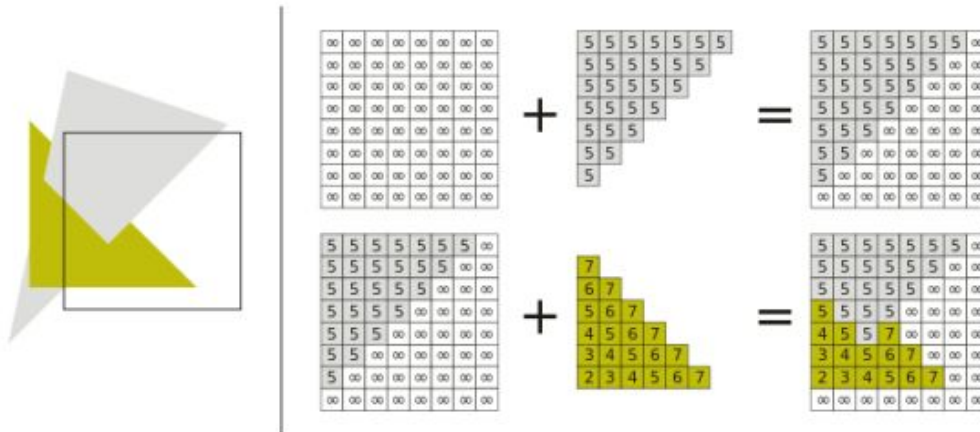
6.2 Algorytm bufora głębokości

Algorytm z buforem głębokości (tzw. -buforem), polega na użyciu dużej ilości pamięci. W algorytmie tym mamy prostokątną tablicę liczb, o wymiarach równych wymiarom szerokości i wysokości obrazu w pikselach. Liczby te są rzeczywiste, ale najczęściej, dla przyspieszenia obliczeń są one reprezentowane w postaci stałopozycyjnej, zwykle 16-, 24- lub 32-bitowej.

Algorytm:

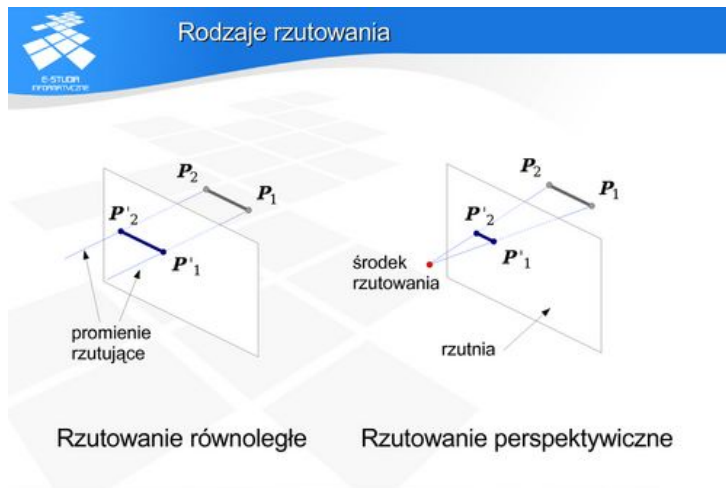
1. Przypisz wszystkim elementom -bufora wartość, która reprezentuje nieskończoność, albo największą dopuszczalną odległość od obserwatora.
2. Kolejno dla każdej ściany, narysuj jej rzut (tj. dokonaj jego rasteryzacji za pomocą algorytmu przeglądania liniami poziomymi). Dla każdego piksela, który należy do rzutu ściany, należy przy tym obliczyć współrzędną  (tj.

głębokość) punktu w układzie obserwatora. Kolor danej ściany przypisujemy pikselowi tylko wtedy, gdy obliczona współrzędna  jest mniejsza niż aktualna zawartość bufora głębokości. Jednocześnie z przypisaniem koloru należy uaktualnić zawartość bufora.



Rysunek 5: Inicjalizacja i nadpisywanie bufora nowymi wartościami

6.3 Rodzaje



6.3.1. Rzutowanie równoległe

Rzutowanie nazywamy równoległym gdy promienie rzutujące są równoległymi. Jest to przekształcenie, w którym ignorujemy wartość współrzędnej z : $(x,y,z) \rightarrow (x,y)$.

6.3.2. Rzutowanie perspektywiczne

Macierz rzutowania równoległego

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Oznaczenia:

r, l (right, left) - granice współrzędnej określającej szerokość ($l \leq x \leq r$)

b, t (bottom, top) - granice współrzędnej określającej wysokość ($b \leq y \leq t$)

n, f (near, far) - granice współrzędnej określającej głębokość ($n \leq -z \leq f$)

6.3.2 Rzutowanie perspektywiczne

Rzutowanie nazywamy perspektywnym, gdy promienie rzutujące tworzą pęk prostych. Rzutowanie perspektywiczne można opisać w następujący sposób:

$$(x, y, z) \rightarrow \left(-\frac{dx}{z}, -\frac{dy}{z}, A + \frac{B}{z}\right).$$

Macierz rzutowania perspektywicznego

$$\begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & -A & -B \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Funkcja głębokości

$$\text{Niech } A = \frac{f+n}{f-n}, B = \frac{2fn}{f-n}.$$

$$\text{Wówczas } g\acute{ł} \acute{e} b\acute{o} k\acute{o} \acute{s} \acute{c}(z) = A + \frac{B}{z}.$$

6.4 Zastosowanie

6.4.1 Cień

Jest to przekształcenie $(x, y, z) \rightarrow (\frac{x}{1-y/y_0}, 0, \frac{z}{1-y/y_0})$. Macierz przekształcenia

$$\text{dla tworzenia cienia: } \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{1}{y_0} & 0 & 1 \end{pmatrix}.$$

6.4.2 Z-fighting -

Jest to graficzna anomalia spowodowana przez dwie powierzchnie o takich samych wartościach w buforze głębokości. Wizualnie manifestuje się to naprzemiennym wyświetlaniem fragmentów powierzchni tych prymitywów, aby uniknąć z-fightingu stosuje się bufor głębokości o wyższej rozdzielczości, lub po prostu przesuwa się geometrię tak, aby powierzchnie przestały ze sobą kolidować.

7. Oświetlenie

7.1 Model odbicia światła Phong - model odbicia światła . Zakłada że źródłem światła jest punkt, a światło ma trzy składowe: RGB(alfa).Światło docierające do obserwatora jest sumą :

- światła odbijanego zwierciadlanie
- światła rozproszonego
- światła otoczenia
- światła emitowanego powierzchnią

7.1.1 Odbicie rozproszone - światło które rozprasza się na obiekcie i ma kolor taki jaki jest przypisany do obiektu.Światło rozprasza się równomiernie we wszystkich kierunkach równomiernie.Natężenie światła rozproszonego obliczamy zgodnie z modelem Lamberta(trudny wzór).Zależy jest od natężenia światła przed rozproszeniem, współczynnika odbicia światła, kąt padania światła

7.1.2 Odbicie zwierciadlane (było pytanie) - światło odbija się od obiektu i nie zmienia swojej barwy. Odbite światło jest emitowane wąską wiązką w jednym kierunku. Natężenie światła zależy od współczynnika odbicia zwierciadlanego , natężenia światła padającego , różnicy między kątem rzeczywistym obserwacji , a kątem doskonałego odbicia, współczynnika określającego własności powierzchni

7.1.3 Światło otoczenia

7.1.3 Światło otoczenia - światło które wynika z własności otoczenia. Iloczyn współczynnika otoczenia i natężenia światła padającego

7.1.4 Światło emitowania powierzchnią - światło emitowane ,gdy obiekt świeci (stałe)

7.2 Cieniowanie -przetwarzanie obrazu polegające na naniesieniu cienia na dany obiekt

7.2.1 Cieniowanie Gourauda . Gouraud miał pewne założenie do algorytmu ~~
Płaszczyzna cieniowana jest podzielona na trójkąty (inne wielokąty są triangulowane)

Ma to zastosowanie do :

a)Obliczania wektorów normalnych do wierzchołków wielokątów i wyznaczenie kąta padania światła w danym miejscu bryły(pozwala na obliczenie jasności danego wierzchołka.

b)Cieniowania wzdłuż krawędzi wielokątów metodą interpolacji liniowej

c)Cieniowanie wzdłuż kolejnych "wierszy wielokątów)

7.2.2 Cieniowanie Phong.Phong miał pewne założenie do algorytmu ~~

Płaszczyzna cieniowana jest podzielona na trójkąty (inne wielokąty są triangulowane)

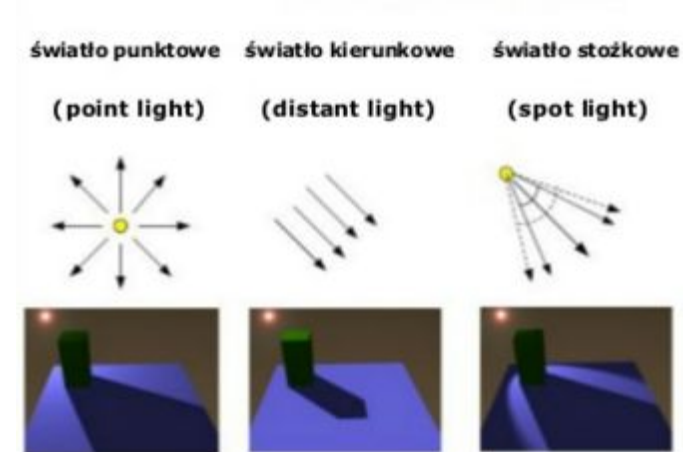
Algorytm:

1.Wyznaczamy wektor normalny w wierzchołku

2.Wyznaczamy interpolowany wektor normalny dla każdego punktu powierzchni odpowiadającego pikselowi

3. Wyznaczamy barwę piksela , na podstawie interpolowanego wektora normalnego korzystając z wybranego modelu odbicia światła

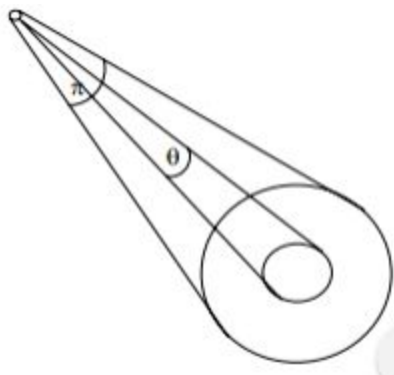
7.3 Źródła



7.3.1 Światło punktowe - Światło punktowe emitowane jest z określonego punktu i rozchodzi się we wszystkich kierunkach z jednakowym natężeniem. Posiada zdefiniowany kolor, intensywność, pozycję, zasięg działania i współczynniki, określające sposób zanikania światła wraz z oddalaniem się od źródła.

7.3.2 Światło kierunkowe posiada jedynie kierunek, kolor, intensywność. Promienie emitowane z tego typu źródła rozchodzą się do kierunku padania światła. Dla tego typu światła nie można określić zasięgu czy sposobu zaniku.

7.3.3 Światło spot(stożkowe) posiada określony kolor, intensywność, pozycję, kierunek w przestrzeni, kąty między krawędziami tworzącymi stożki, oraz parametr określający sposób zanikania światła między granicą stożka wewnętrznego, a stożka zewnętrznego. Światło rzucane składa się z dwóch stożków: wewnętrzny ~ jaśniejszy, emitujący światło właściwe i zewnętrzny, określający obszar w którym światło zanika



8. Teksturowanie - technika stosowana w grafice trójwymiarowej, której celem jest przedstawienie szczegółów powierzchni obiektów przestrzennych za pomocą obrazów bitmapowych (tekstur) lub funkcji matematycznych (tekstur proceduralnych).

8.1 Idea teksturowania - Tekstura zawiera:

- a) informacje o kolorach, które mają zastąpić obliczone kolory powierzchni.
- b) informacje o kolorach, blasku, przezroczystości, które mają zmienić charakterystyki powierzchni po obliczeniach oświetlenia i cieniowania.
- c) parametry, mające wpływy na obliczenie oświetlenia (współczynnik odbicia, przemieszczenie wektora normalnego itp.)

8.2 Czym jest tekstura -

Tekstura jest to:

- a) zdjęcie, obrazek skanowany, utworzony edytorem graficznym.
- b) obrazek zaprogramowany (skompilowany/generowany na bieżąco).
- c) obrazek generowany podczas mapowania - Teksturowanie $[0,1] \times [0,1] \rightarrow model$

8.3 Mapowanie tekstury - Odwzorowanie współrzędnych dwuwymiarowej na współrzędne obiektu trójwymiarowego (mapowanie tekstury).

8.3.1 Mapowanie cylindryczne - mapowanie cylindryczne stosuje się do obiektów cylindrycznych oraz stożki

- $p(\theta, y) = (r \sin \theta, y, r \cos \theta), 0 \leq \theta < 360, -h/2 \leq y \leq h/2$
- $s = \frac{\theta}{360}, t = \frac{y+h/2}{h}$

8.3.2 Mapowanie sferyczne - stosuje się do obiektów sferycznych, włączając częściowe kule.

$$P(\theta, \varphi) = (r \sin \theta \cos \varphi, r \sin \theta \sin \varphi, r \cos \theta)$$

- $s = \frac{\theta}{360}, t = \frac{\varphi}{180} + \frac{1}{2}$
- $s = \frac{\theta}{360}, t = \frac{\sin \varphi}{2} + \frac{1}{2}$

8.3.3 Mapowanie torusa

$$P(\theta, \varphi) = ((R + r \cos \varphi) \sin \theta, (R + r \cos \varphi) \cos \theta, r \sin \varphi)$$
$$s = \frac{\theta}{360}, t = \frac{\varphi}{360}$$

8.4 Aliasing - polega na zniekształceniu obrazu w wyniku zbyt małej częstotliwości jego próbkowania w procesie rasteryzacji (działanie polegające na jak najwierniejszym przedstawieniu płaskiej figury geometrycznej na dysponującym skończoną rozdzielczością). Rasteryzacja ta zachodzi najczęściej podczas wyświetlania obrazu na ekranie, który obecnie najczęściej jest ekranem rastrowym, ale może dotyczyć także procesu zmiany modelu opisu obrazu z wektorowego na rastrowy.

8.4.1 Idea -

- a) rozdzielczości tekstury jest mniejsza od rozdzielczości ekranu.
- b) rozdzielczości tekstury jest mniejsza od rozdzielczości ekranu.
- c) miganie, interferacja, plamy.
- d) obiekty ruszające się.

8.4.2 **Mipmapping** - technika teksturowania bitmapami, która pozwala uniknąć artefaktów i uzyskać lepszą jakość obrazów.

mipmapping:

- a) zastosowanie skalowanych tekstur,
- b) interpolacja najbliższych tekstur,
- c) zwiększenie prędkości.
- d) zwiększenie pamięci o 33%,
- e) jest implementowany sprzętowo.

8.4.3 **Supersampling** - inny rodzaj antyaliasingu (nadpróbkowanie). Jest to rozwiązanie polegające na użyciu tzw. *Brute force* do rozwiązywania problemu aliasingu. W tej technice obraz jest renderowany w rozdzielczości odpowiadającej wielokrotności rozdzielczości docelowej, a uzyskany tym sposobem dużo większy obraz jest dyskretyzowany do właściwej, niższej rozdzielczości.

8.5 **Mapowanie wypukłości - (bump mapping)** technika teksturowania, która symuluje niewielkie wypukłości powierzchni, bez integracji w geometrię obiektu trójwymiarowego

- zmiana wektora normalnego
- przed obliczeniem oświetlenia

8.6 **Mapowanie środowiska** - dany jest mały zwierciadlany obiekt (kula, sześcian). oblicza się (robi się zdjęcie) mapa tekstury jako obraz otoczenia widoczny od środka obiektu.

9. Krzywe Beziera

9.1 **Parametryzacja** - Parametryzacją krzywej na płaszczyźnie nazywamy odzorowanie $p:[a,b] \rightarrow \mathbb{R}^2$ gdzie $[a,b]$ jest pewnym odcinkiem zawartym w \mathbb{R}

$$p(t) = (x(t), y(t)), \text{ gdzie } x, y \text{ s\k{a} ci\k{a}głymi funkcjami } [a, b] \rightarrow \mathbb{R}$$

9.2 Wielomiany Bernsteina i definicja krzywej Beziera

Wielomiany Bernsteina n -tego stopnia definiujemy następująco:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad \text{gdzie } i = 0, \dots, n$$

Widać, że i -ty wielomian jest i -tym składnikiem rozwinięcia wzorem Newtona wyrażenia:

Krzywą Béziera n -tego stopnia definiujemy jako krzywą o parametryzacji: $p(t) = \sum_{i=0}^n p_i B_i^n(t)$ gdzie: p_0, p_1, \dots, p_n są wybranymi punktami tworzącymi łamaną kontrolną. Intuicyjnie można to rozumieć tak: każdy punkt łamanej kontrolnej ma swój wielomian Bernsteina, który dla danego t decyduje o tym jak duży wpływ (jak bardzo przyciąga) ma na położenie punktu krzywej $p(t)$ dany punkt kontrolny.

9.3 Własności wielomianów Bernsteina i ich konsekwencje dla krzywych Bezierra

1. $B_i^n(t) = (1-t) B_i^{n-1}(t) + t B_{i-1}^{n-1}(t)$ - wzór rekurencyjny na obliczenie wielomianu n -tego stopnia przy pomocy wielomianów $n-1$ stopnia.

Konsekwencja: algorytm deCasteljau dający możliwość dużo wydajniejszego obliczenia punktu krzywej dla danego t .

2. $\sum_{i=0}^n B_i^n(t) = 1$ - rozkład jedynki.

Konsekwencja: niezmienniczość afiniczną krzywej Béziera. Oznacza to, że możemy łamaną kontrolną potraktować dowolnym przekształceniem afinicznym (np obrót, przesunięcie, skalowanie) a otrzymana w ten sposób łamana będzie reprezentować krzywą Béziera o takim samym kształcie, będącą obrazem wyjściowej krzywej w danym przekształceniu (czyt. obroconą, przesuniętą, przeskalowaną).

3. $\forall t \in [0,1] B_i^n(t) \geq 0$ - nieujemność.

Konsekwencja: w połączeniu z 1. daje własność otoczki wypukłej.

4. $\forall t \in (0,1) B_i^n(t) > 0$ - dodatniość.

Konsekwencja: na położenie każdego punktu krzywej (poza pierwszym i ostatnim) mają wpływ wszystkie punkty łamanej kontrolnej.

5. $B_0^n(0) = 1, B_i^n(0) = 0$ dla każdego $i=1, \dots, n$.

Konsekwencja: krzywa zawsze przechodzi przez pierwszy punkt kontrolny

6. $B_n^n(1) = 1, B_i^n(1) = 0$ dla każdego $i=1, \dots, n$.

Konsekwencja: krzywa zawsze przechodzi przez ostatni punkt kontrolny.

7. $B_i^n(t) = B_{n-i}^n(1-t)$ - symetryczność.

Konsekwencja: jeżeli łamana kontrolna jest symetryczna to krzywa którą reprezentuje również jest symetryczna.

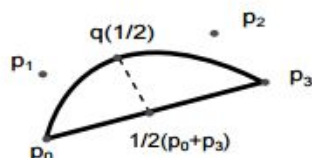
8. $B_i^n(t) = 0$ gdy $t = \frac{i}{n}$ - to znaczy, że i -ty wielomian Bernsteina osiąga ekstremum (dokładnie maksimum) dla $t = i/n$

Konsekwencja: punkt kontrolny p_i ma największy wpływ na położenie punktu $p(\frac{i}{n})$ krzywej.

9.4 Algorytm de Casteljau (nie pojawił się na teście więc nie wiem)

9.5 Krzywe Beziera jako ciąg odcinków

Aby renderować krzywą Béziera dzieli się ją za pomocą algorytmu de Casteljau tak wiele razy aż otrzymane kawałki będą dość płaskie aby móc je narysować za pomocą odcinków prostych. Aby ocenić czy dany kawałek jest wystarczająco płaski oblicza się odległość punktu $q(1/2)$ od punktu będącego środkiem odcinka łączącego początek i koniec krzywej jak pokazano na rysunku. I porównuje z arbitralnie ustaloną liczbą ε . Jeżeli odległość jest mniejsza dany kawałek krzywej zastępuje się odcinkiem łączącym jej pierwszy i ostatni punkt. Jeżeli nie rozbija się ją na dwie krzywe, o wspólnym punkcie $q(1/2)$ zgodnie z 9.4.4.



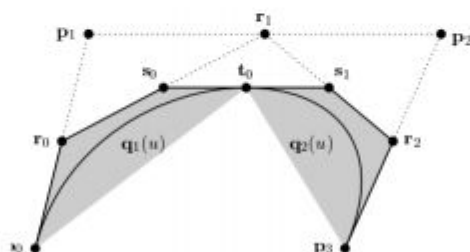
Rysunek 11: Rasteryzacja krzywej Béziera

9.6 Właściwości konsekwencji otoczki wypukłej

Otoczka wypukła zbioru- jest to najmniejszy zbiór wypukły, który zawiera ten zbiór.

Łącząc ze sobą kolejne punkty kontrolne krzywej Béziera otrzymamy pewien wielokąt. Z 2 i 4 własności wielomianów Bernsteina można pokazać, że wszystkie punkty krzywej leżą wewnątrz otoczki wypukłej tego wielokąta.

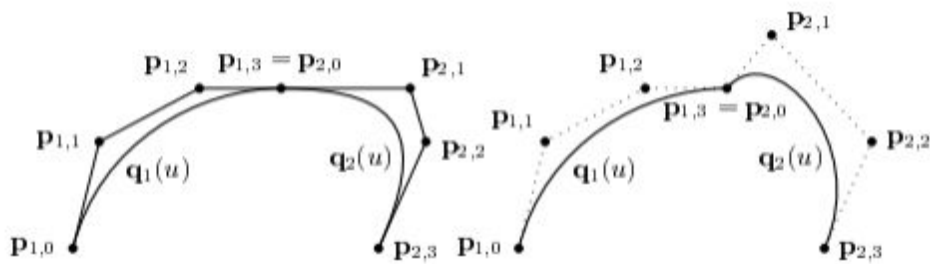
Ilustruje to rysunek na którym pokazano dwie krzywe q_1 oraz q_2 i ich otoczki wypukłe w których się zawierają.



Rysunek 12: Otoczka wypukła

9.7 Sklejanie krzywych

Aby połączyć ze sobą dwie krzywe Bezier wystarczy aby ostatni punkt kontrolny pierwszej pokrywał się z pierwszym drugiej (na rys. $p_{1,3} = p_{2,0}$). Zazwyczaj jednak interesuje nas aby połączenie było gładkie. Aby to osiągnąć wystarczy, aby przedostatni punkt pierwszej krzywej, punkt złączenia oraz drugi punkt krzywej leżały na jednej linii (na rys. $p_{1,2}, p_{1,3} = p_{2,0}, p_{2,1}$). Mocniejszym warunkiem na gładkość jest aby pochodne obu krzywych w punkcie połączenia były równe tzn. $q_1(1)' = q_2(0)'$.



Rysunek 13: Sklejanie krzywych Beziea

9.8 Podwyższanie stopnia

Mamy krzywą Beziea st. n o punktach kontrolnych: p_0, p_1, \dots, p_n i parametryzacji $p(t)$. Chcemy łamaną kontrolną reprezentującą tą samą krzywą ale mającą o jeden punkt kontrolny więcej.

Algorithm 13 Algorytm podwyższania stopnia

Wejście: p_0, p_1, \dots, p_n - punkty kontrolne

Wyjście: $\hat{p}_0, \hat{p}_1, \dots, \hat{p}_{n+1}$ - punkty kontrolne krzywej stopnia $n+1$

```

for  $i:=0$  to  $n+1$  do
   $\hat{p}_i = \frac{n+1-i}{n+1}p_i + \frac{i}{n+1}p_{i-1}$ 
end for

```

10. OpenGL

10.1 Obiekt tablicy wierzchołków

Dane obejmujące m.in współrzędne wierzchołków prymitywów (punkty, listy linii, serie linii, listy trójkątów i serie trójkątów) i dane kolorów wierzchołków przechowywane są w jednej lub kilku tablicach i renderowane bezpośrednio.

10.2 Immediate mode

Program przekazuje ciąg poleceń do GPU. Styl programowania aplikacji bibliotek graficznych, gdzie zachodzi bezpośredni rendering graficzny obiektu na ekranie. Nie wyklucza podwójnego buferowania.

10.3 Obiekt bufora

Przechowuje tablice niesformatowaną pamięci przydzielonej w kontekście OpenGL (GPU). Mogą być używane do przechowywania danych wierzchołków, pikseli pobranych z obrazów lub bufor ramki, i wiele innych rzeczy.

10.4 Shadery

Krótki program komputerowy, który w grafice trójwymiarowej opisuje właściwości pikseli oraz wierzchołków.

Vertex Shader - cieniowanie wierzchołkowe. jego zadaniem jest transformacja położenia wierzchołka wirtualnej przestrzeni 3D na współrzędne 2D na ekranie.

Geometry Shader - Cieniowanie geometryczne.