Wprowadzenie do grafiki maszynowej(LAB-3)

Paweł Idzikowski, nr.albumu: 141080

4 listopada 2018

Spis treści

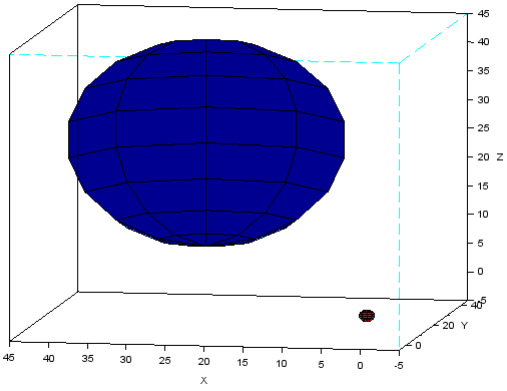
1 Zadanie 1

Treść zadania:
Stwórz dwie sfery na jednym wykresie, jedną większą od drugiej +- 20x, przesunięte względem siebie. (trochę jak księżyc i planeta)

Rozwiązanie:

```
// Zadanie 1

u = linspace(-%pi/2, %pi/2, 10);
v = linspace(0, 2*%pi, 10);

// sfera1
x=cos(u)'*cos(v);
y=cos(u)'*sin(v);
z = sin(u)'*ones(v);
plot3d2(x,y,z);
e=gce();
e.color_mode=5;

// sfera2
x=x*18;
y=y*18;
z=z*18;
plot3d2(x+25,y+25,z+25, alpha=100, theta=80);
e=gce();
e.color_mode=9;
```

1.1 Wniosek do zadania



Rysunek 1: Efekt skompilowania skryptu

Wnioski:
Dwie sfery rysowane w jednym oknie uzyskałem posługując się poleceniem *plot*3*d*2 dwukrotnie. Aby uniknąć problemu nachodzenia na siebie obu sfer, przesunąłem jedną(tą większą) o $\angle 25$ stopni w każdej z dostępnych osi. Żeby uzyskać z kolei

żądany efekt jednej sfery większej od drugiej o +/- 20x, wymnożyłem każdą zmienną x, y i z o 18, przypisując do nich nową wartość przed narysowaniem kolejnej sfery(sfera2). Sfery zostały opatrzone kolorami za pomocą funkcji robiącej uchwyt(get current entity) i dostępnej właściwości dla tego obiektu czyli color_mode.

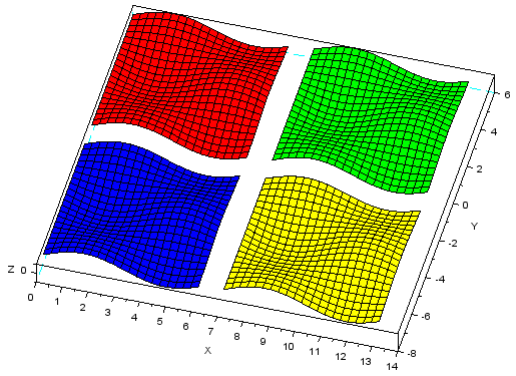
2 Zadanie 2

Treść zadania:
Stwórz „Logo Windowsa” z 4 przesuniętych pofalowanych płaszczyzn.

Rozwiązanie:

```
// Zad2 - logo windows

t=[0:0.3:2*%pi]';
z=sin(t)*cos(t');

[xx, yy, zz] = genfac3d(t,t,z);

// pytanie jak dodac kolorki inaczej :(?

plot3d([xx xx+7 xx xx+7],[yy yy yy-7 yy-7],
list([zz zz zz zz],[5*ones(1,400) 3*ones(1,400)
 2*ones(1,400) 7*ones(1,400)]), alpha=25, theta=285);
// z powodu ze jest to obszerna linia kodu
// zostala podzielona na 3 linie
```

2.1 Wniosek do zadania



Rysunek 2: Logo Windows w scilab

Wnioski:
Zadanie z logiem Windows było pierwszym zadaniem za które się zabrałem. Przy dostosowaniu odległości między poszczególnymi płaszczyznami trzeba było po prostu potestować jaka wartość będzie odpowiednia. Ja zdecydowałem się na wartość 7 i tak każdy z kawałków przesunąłem, manipulując odpowiednio współrzędnymi. Trudniejszą zagadką okazało się pokolorowanie tych płaszczyzn. Próbowałem znaleźć inny sposób pokolorowania płaszczyzn nadając go w sposób np. color=(”red”) jednak bezskutecznie, argument style w plot3d nie jest dostępny... Zdecydowałem zatem, że pozostane przy sposobie z objęciem osi **z**

wszystkich czterech płaszczyzn listą. Tu także musiałem odkryć metodą prób i błędów jakie wartości stanowią jakie kolory i czasami zdarzał się problem, że przy wymnożeniu jednej macierzy jedynek, druga macierz także zmieniała kolor na inny niż był poprzednio. Na szczęście udało się otrzymać "windowsowe" barwy, mnożąc macierze kolejno przez 5, 3, 2 i 7 - dobra kombinacja. Warto zwrócić uwagę, że kolor nadaliśmy tylko w osi Z zatem jeżeli spróbujemy "przekręcić" nasze płaszczyzny to otrzymamy ich domyślny jasno-niebieski kolor. Aby płaszczyzny i ich kolory były ułożone tak jak to jest w logo Windowsa oraz, żeby nie musieć ustawiać dobrej perspektywy za każdym razem gdy uruchomie skrypt, skorzystałem z argumentów alpha i theta.

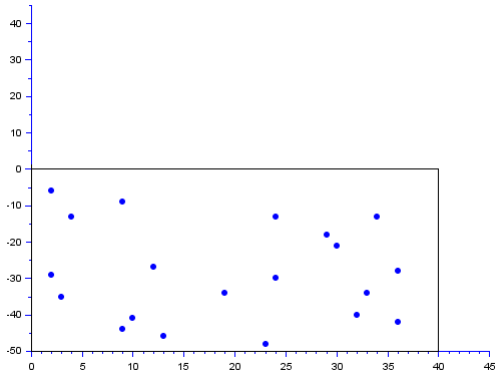
3 Zadanie 3

Treść zadania:
Stwórz program liczący pole figury metodą „rzutek” (Monte Carlo). Oczywiście był jeszcze podpunkt aby stworzyć podskrypt lub drugi skrypt animujący pierwszych 20 rzutów. Rozwiązanie umieściłem na nowej stronie z racji, że jest obszerne a zależało mi na tym, aby zmieścić kod właśnie na jednej stronie.

Rozwiązanie:

```
// Skrypt - Monte Carlo

// Liczba rzutow
amunicja=1000;

// Predkosc animacji
v=1;

// Przechowuje liczbe trafien
trafienia=0;

// ustalamy podzialke
plot2d(0,0,-1,"010","*",[0,0,45,45]);
xset("color",2);

// figura
// sciaga: xrect - x y w h
// Pole tej figury to: 40*50=2000
xrect(0,0,40,50);
gce().foreground = color("black");

// historia przechowuje wspolrzedne trafien
historia = [];

// blad przechowuje trafienia w miejsca w
// ktorych znajdowala sie juz rzutka
blad=0;

// animacja pierwszych 20 rzutow
for i=1:20
        // oblicz wspolrzedne dla rzutki
        // zakres X 0;40
        // zakres Y 0;-50
        x=round(0 + (40-1).*rand());
        y=round(0 + (-50-1).*rand());

        // narysuj rzutke
        plot(x, y, '.');

        // dodajemy pare x,y do macierzy
        historia=[historia x y];

        // sprawdz czy nie ma w tym miejscu juz rzutki
        // nie bierzemy pod uwage przed chwila dodanych
        // wspolrzednych x i y, stad -2
        for m=1:1:length(historia)-2
            p=m+1;

            if(x == historia(m) && y == historia(p))
                blad=blad+1;
            end
        end

        trafienia=trafienia+1;

        sleep(v);
end

// pozostale rzuty
for i=21:amunicja

        x=round(0 + (40-1).*rand());
        y=round(0 + (-50-1).*rand());

        historia=[historia x y];

        for m=1:1:length(historia)-2
            p=m+1;

            if(x == historia(m) && y == historia(p))
                blad=blad+1;
            end
        end

        trafienia=trafienia+1;
end

// zalozmy, ze jedna rzutka stanowi pole=1, zatem liczba trafien = pole
// korygujemy wartosc pola w razie gdyby w jedno miejsce padlo wiele
// rzutek

Pole=trafienia-blad;
```

3.1 Wniosek do zadania



Rysunek 3: Animacja pierwszych 20 rzutów

Wnioski:
Jeżeli chodzi o zadanie z wykonaniem skryptu Monte Carlo - podszedłem do niego całościowo tzn. wygodniej mi było zająć się w jednym skrypcie i animacją i obliczaniem pola. W Internecie natknąłem się na różne sposoby prezentacji tej metody. Na jednej, animacja pokazywała jak przy wzroście liczby rzutek(kropek) pole i błąd zwiększają się(-Wikipedia). Z kolei w innej prezentacji, widziałem opcje taką, że w jednej większej figurze była zamieszczona mniejsza i to tej mniejszej figury pole chcieliśmy uzyskać za pomocą rzutek. Podszedłem do tego zadania w pierwszej kolejności tak jak w tej prezentacji, że mam dużą figurę i jakąś mniejszą. Rzuty mogły być wykonywane w zakresie dużej figury. Trafienia, które wpadły do małej figury były zliczane do jej pola. Zdecydowałem jednak, że wykonam zadanie w stylu jaki prezentuje to animacja na stronie Wikipedii.. Na chwilę obecną skrypt działa w sposób taki, że jest jeden prostokąt i w jego obszarze wykonywane są rzuty. Do policzenia pola obszaru zapełnionego przez rzutki wprowadzana jest korekta na podstawie błędów(rzutek, które trafiły w dokładnie te samo miejsce po raz kolejny).

Kilka kompilacji skryptu i otrzymane rezultaty:

- liczba rzutów: 1000, błąd: 222, pole: 778
- **liczba rzutów: 1000, błąd: 250, pole: 750** (źródło fotografii)
- liczba rzutów: 1000, błąd: 268, pole: 732
- liczba rzutów: 1000, błąd: 259, pole: 741
- liczba rzutów: 1000, błąd: 234, pole: 766

Pole	750
amunicja	1e+03
blad	250
historia	1x2000
i	1e+03
m	2e+03
p	2e+03
trafienia	1e+03
v	1
x	37
y	-20

(a) Po kompilacji skryptu zaprezentowanego w listingu otrzymaliśmy następujące informacje: 250 rzutek trafiło w miejsce w którym była już jakaś rzutka a pole obszaru objętego rzutkami, po korekcji błędu wynosi 750(przy liczbie 1000 rzutów). Jeżeli nie zepsułem pętli sprawdzającej czy nie było już w danym miejscu rzutki to na już 1000 rzutów możemy otrzymać całkiem duży błąd, bo będzie to +/-20%

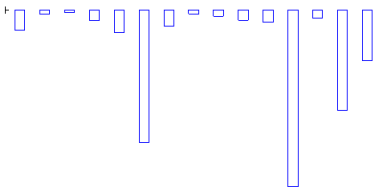
4 Zadanie 4* - dodatkowe

Treść zadania:
Zadanie specjalne – Napisz skrypt animujący bubblesorta lub quicksorta. Rozwiązanie umieściłem na nowej stronie z racji, że jest obszerne a zależało mi na tym aby zmieścić kod właśnie na jednej stronie.

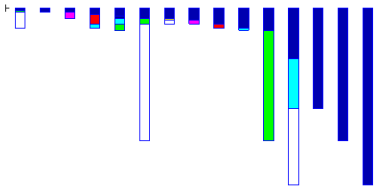
Rozwiązanie:

```
// Skrypt - sortowanie Bubblesort

// zmienna przechowuje wartosci do posortowania
matryca=[10 2 1 5 11 66 8 2 3 5 6 88 4 50 25];

// startowa pozycja zmiennej
pozycjaX=2;

// ustawienia figury
// (nie wiem niestety czemu tym razem podzialki nie wyswietla..)
plot2d(0,0,-1,"010","*",[0,0,50,50]);
xset("color",2);

// predkosc kolorowania
v = 50;

// rysujemy prostokaty o wysokosci odpowiadajacej wartosci
//(kolorem niebieskim, bez wypelnienia)
for i=1:length(matryca)
    xrect(pozycjaX, 0, 2, matryca(i));
    pozycjaX=pozycjaX + 5;
end

// domyslna wartosc zmiennej kolorek
kolorek=2;

// kontrolka petli while
niePosortowane=1;

// zmienna liczaca ile bylo potrzebnych iteracji do posortowania matrycy
ileIteracji=0;

while(niePosortowane)

        // resetujemy pozycjeX
        pozycjaX=2;

        // zmieniamy kolor iteracji
        kolorek=kolorek+1;

        // iteracja sortowania
        for i=1:length(matryca)
            j=i+1;

            // etap sortowania
            if(j <= length(matryca))
                if(matryca(i) >= matryca(j))
                    kopia=matryca(i);
                    matryca(i)=matryca(j);
                    matryca(j)=kopia;
                end
            end

            // etap rysowania
            xrect(pozycjaX, 0, 2, matryca(i));
            e=gce();
            e.background = kolorek;
            e.fill_mode = "on";
            pozycjaX=pozycjaX + 5;

            // przerwa..
            sleep(v);
        end

        // sprawdzenie stanu matrycy
        for m=1:length(matryca)
            p=m+1;

            if(p <= length(matryca))
                if(matryca(m) <= matryca(p))
                    niePosortowane=0;
                else
                    // jezeli nadal zle to potrzebna kolejna iteracja
                    niePosortowane=1;
                    break;
                end
            end
        end

        ileIteracji=ileIteracji+1;
end
```

4.1 Wniosek do zadania



(b) Rozłożenie liczb w matrycy (interpretacja graficzna za pomocą 'słupków')



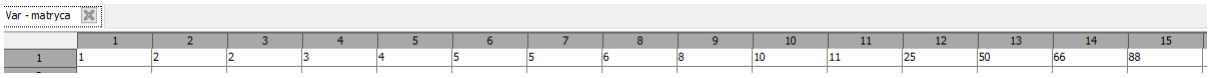
(c) Rezultat po posortowaniu macierzy o nazwie matryca(kolor ciemny-niebieski prezentuje ostateczny układ liczb w macierzy)

Wnioski:
Wykonując poprzednie zadanie wykorzystałem komendę xrect. W Internecie wpadłem na zdjęcie na którym było kilka słupków obok siebie i postanowiłem właśnie w ten sposób zrealizować animację sortowania. Przy pierwszym podejściu wykonałem sortowanie, które działało na zasadzie, że przykładowo - skrypt patrzy na pierwszy element macierzy i szuka liczby, która jest od niej mniejsza po całej macierzy jednowierszowej, jeżeli znalazł, to są one zamieniane miejscami. Dzięki rysowaniu słupków z tłem na zielono uzyskałem fajny efekt tworzenia się schodków.. jednak... w treści zadania była mowa o bubblesort albo quicksort więc oczywiście trzeba było przerobić działanie skryptu. Obecnie działa on na zasadzie jaką proponuje sortowanie bąbelkowe czyli porównujemy sąsiednie elementy i sprawdzamy czy są dobrze ustawione - jeżeli nie, robimy zamianę. Wykonujemy tyle iteracji ile jest potrzebne, aby uzyskać w pełni posortowaną macierz nazwaną "matryca". Każda iteracja obarczona jest oczywiście rysowaniem. Rysowane elementy są dodatkowo wypełniane kolorem(kolor przy każdej kolejnej iteracji jest zmieniany aby było widać efekt pracy sortowania). Wspomnijmy, że na zaproponowanym przykładzie(rysunek b) ciemny kolor niebieski jest ostateczną prezentacją posortowanej 'matrycy' i tak samo widzimy w efekcie końcowym "schodki" od najmniejszego do największego. Sortowanie graficznie jest odwrócone z racji tego, że wysokość w komendzie xrect jest "dodatnia w dół".

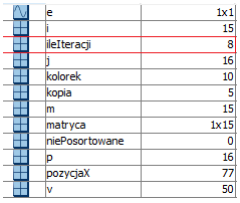
Macierz jednowierszowa przed wykonaniem sortowania prezentuje się tak:

$matryca = [10\ 2\ 1\ 5\ 11\ 66\ 8\ 2\ 3\ 5\ 6\ 88\ 4\ 50\ 25];$

Macierz jednowierszowa po posortowaniu prezentuje się następująco:



Warto zwrócić uwagę na to, że trzeba było wykonać dla takiego ciągu liczb **8 iteracji** aby uporządkować go poprawnie.



e	1x1
i	15
ileIteracji	8
j	16
kolorek	10
kopia	5
m	15
matryca	1x15
niePosortowane	0
p	16
pozycjaX	77
v	50