GRAFIKA 3D

Tworzenie grafik trójwymiarowych w Processingu wymaga określenia, który silnik 3D będzie stosowany do wyświetlania rysunku. W tym celu wykorzystywana jest funkcja **size()**, wywoływana na początku programu.

```
size(szerokość, wysokość, silnik)
```

Funkcja **size()** definiuje nam rozmiar okna dwuwymiarowego w pikselach, w którym wyświetlany będzie rysunek trójwymiarowy. Processing umożliwia korzystanie z kilku różnych trybów wyświetlania grafiki podawanych jako parametr **silnik**. Dla trybu 3D możliwe do wyboru są dwa silniki:

P3D albo **OPENGL**

Przykład:

```
void setup()
{
size(400,400,P3D);  //rozmiar okna i tryb wyświetlania
noLoop();
}

void draw()
{
box(200,200,200);   //rysowanie prostopadłościanu
}
```

PROSTOPADŁOŚCIAN

Prostopadłościan tworzy się przy pomocy funkcji **box()**.

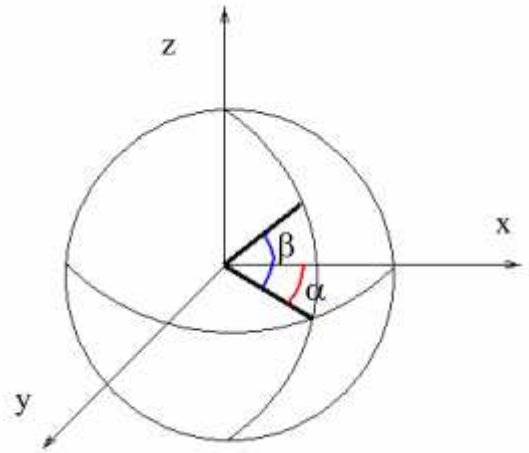
```
box(szerokość, wysokość, głębokość);
```

SFERA

Sferę o promieniu r i o środku w początku układu współrzędnych przestrzennych można opisać równaniami parametrycznymi:

$$x = r \cdot \cos\beta \cdot \cos\alpha$$
$$y = r \cdot \cos\beta \cdot \sin\alpha$$
$$z = r \cdot \sin\beta$$

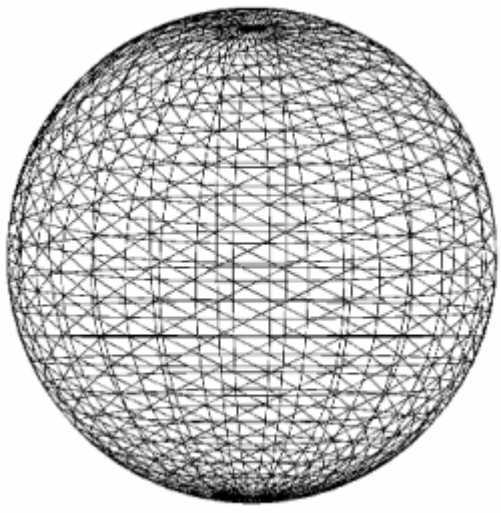
gdzie kąt $\alpha \in \langle 0; 2\pi \rangle$ to długość geograficzna, a kąt $\beta \in \langle -\pi/2; \pi/2 \rangle$ to szerokość.



Rysunek sfery można uzyskać stosując procedurę **sphere()**, pamiętając o przesunięciu obrazu **translate(),** gdyż początek układu współrzędnych znajduje się w górnym, lewym rogu ekranu i użyciu macierzy zapisującej **[pushMatrix()]** aktualny układ współrzędnych oraz na koniec przywracającej **[popMatrix()]** poprzedni układ współrzędnych po dokonaniu transformacji - w naszym przypadku przesunięcia, czyli translacji. Użycie funkcji **pushMatrix()** powoduje, że początek i kierunki osi nowego układu współrzędnych określone są przez funkcje transformujące (np. przesunięcie). Kolejne przekształcenia i rysowanie obiektów wykonywane są już w nowym układzie współrzędnych. Po wywołaniu funkcji **popMatrix()** wszystkie poprzednie ustawienia układu współrzędnych są odzyskane.

```
void setup()
{
  size(800,600,P3D);
  background(255);
  lights();
  strokeWeight(0.7);
  noLoop();
}

void draw()
{
  pushMatrix();
  translate(400, height*0.35, -400);
  noFill();
  stroke(1);
  sphere(280);
  popMatrix();
}
```



Efekt trójwymiarowości można uzyskać przez dodanie oświetlenia. Można użyć do tego np. funkcji **spotLight()**. Poniższy przykład pokazuje jak można oświetlić kulę. Efekt otrzymamy po naciśnięciu i przytrzymaniu przycisku myszy.

```
void setup()
{
  size(200, 200, P3D);
}

void draw()
{
  background(0);
  translate(100, 100, 0);
  if (mousePressed)
  spotLight(255, 0, 0, width/2, height/2, 400, 0, 0, -1, PI/4, 2);

  noStroke();
  fill(255);
  sphere(50);
}
```

Parametry:

- 255, 0, 0 – kolor czerwony
- width/2, height/2, 400 – ustawienie w oknie
- 0, 0, -1 – kierunek oświetlenia
- PI/4 – kąt oświetlenia
- 2 – stopień koncentracji światła

- przed naciśnięciem myszy

- po naciśnięciu myszy

W Processingu udostępniono dwie standardowe bryły trójwymiarowe: prostopadłościan i sfera. Pozostałe bryły, a tak naprawdę wielościany wymagają osobnego zdefiniowania jako połączonych ze sobą wierzchołków w przestrzeni. Wierzchołki określa się przy pomocy funkcji **vertex()**.

```
vertex (x, y, z);
```

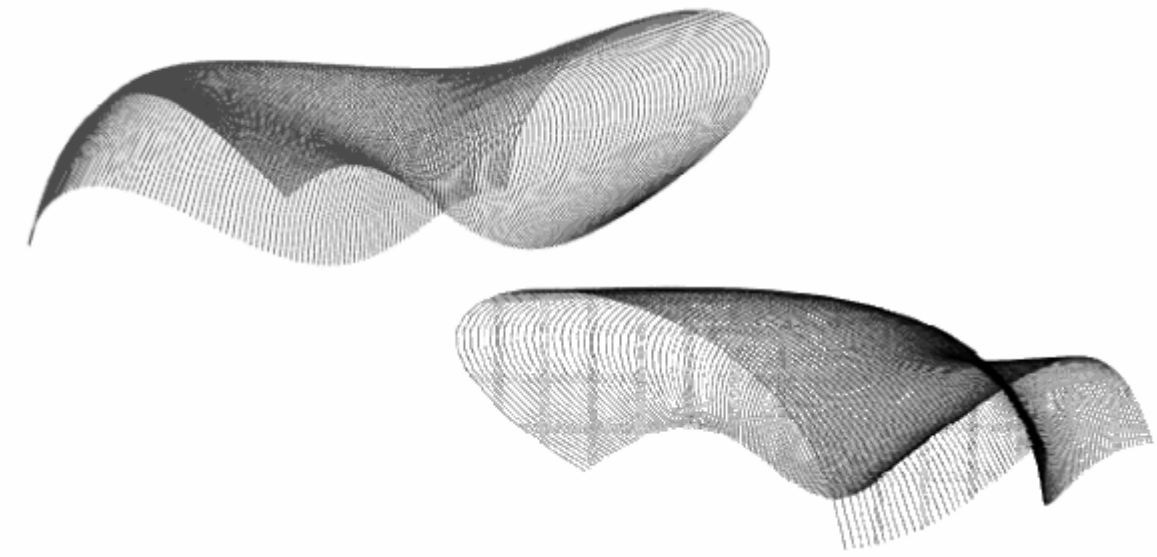
Blok definicji bryły to ciąg funkcji definiujących wierzchołki ograniczony przez instrukcje **beginShape()** i **endShape().**

Przykład (ostrosłup o podstawie czworokąta).:

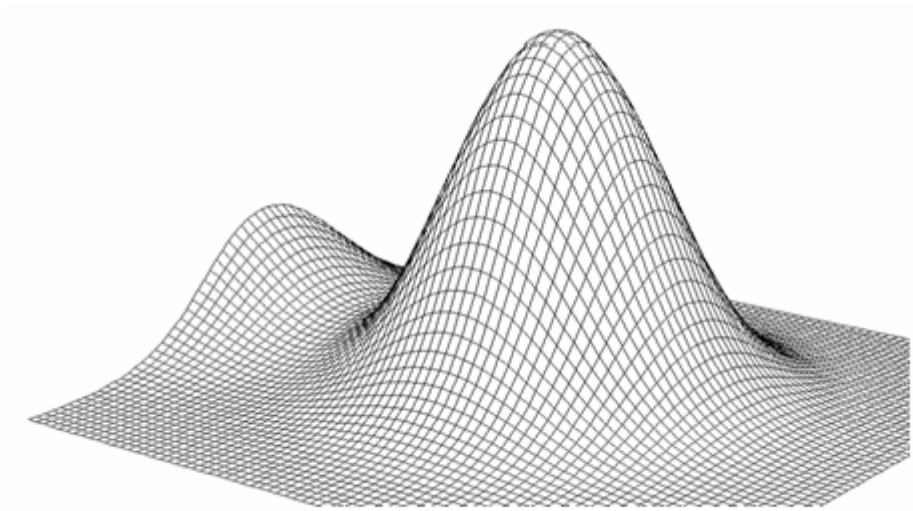
```
beginShape();
  vertex(-100, -100, -100); vertex( 100, -100, -100); vertex(0, 0, 100);
  vertex( 100, -100, -100); vertex( 100,  100, -100); vertex(0, 0, 100);
  vertex( 100, 100, -100);  vertex(-100, 100, -100);  vertex(0, 0, 100);
  vertex(-100, 100, -100);  vertex(-100, -100, -100); vertex(0, 0, 100);
endShape();
```

Stosując funkcję **bezier()** w wersji 3D możemy otrzymać interesujące rysunki powierzchni.
Używając myszy możemy wykonywać ich obroty w dwóch płaszczyznach.

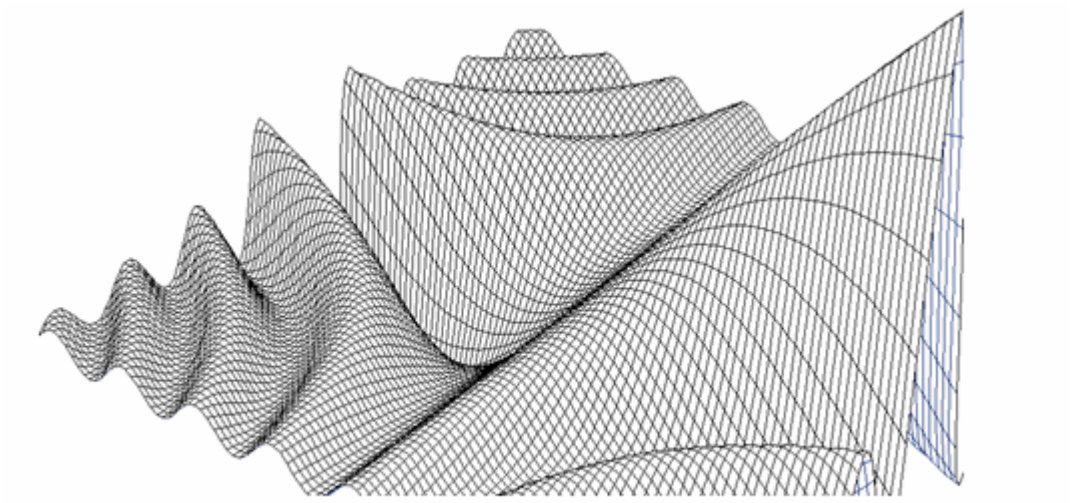
```
float px [] = new float[4];
float py [] = new float[4];
void setup()
{
  size(600,400,P3D);
}
void draw()
{
  background(60);
  translate(300,200);
  rotateX(mouseY*PI/300);
  rotateZ(mouseX*PI/300);
  noFill();
  stroke(250,160);
for (int i=-60;i<60;i++)
{
  px[0] = -100;  py[0] = -50+sin(i*PI/60.0)*25;
  px[1] =  -80+sin(i*PI/45.0)*50;   py[1] = 50;
  px[2] =   80;   py[2] = 50;
  px[3] =  100+sin(i*PI/60.0)*25;  py[3] = -50+sin(i*PI/40.0)*15;
  bezier(px[0], i*2, py[0],
  px[1 ], i*2+sin(i*PI/120.0)*150,  py[1],
  px[2],  i*2, py[2],
  px[3], i*2, py[3]);
}
}
```



Na koniec przykłady grafik w postaci wykresów funkcji:



z = EXP(- x·x - y·y) + 0.5·EXP(- (y + 3)·(y + 3) - x·x)



z = y·SIN(x·y)/ (x·y)