GRAPHIC 3D

Creating 3D graphics in Processing requires determining which 3D engine will be used to display the drawing. The **size()** function, called at the beginning of the program, is used for this purpose..

```
size(width, height, motor)
```

The size() function defines the size of a two-dimensional window in pixels, in which a three-dimensional drawing will be displayed. Processing enables the use of several different modes of displaying graphics given as a parameter of the „**motor**". Two motors can be selected for the 3D mode:

P3D or **OPENGL**

Example:

```
void setup()
{
size(400,400,P3D);  // window size and display mode
noLoop();
}

void draw()
{
box(200,200,200);   // drawing a cuboid
}
```

CUBOID

The cuboid is created using the **box()** function.

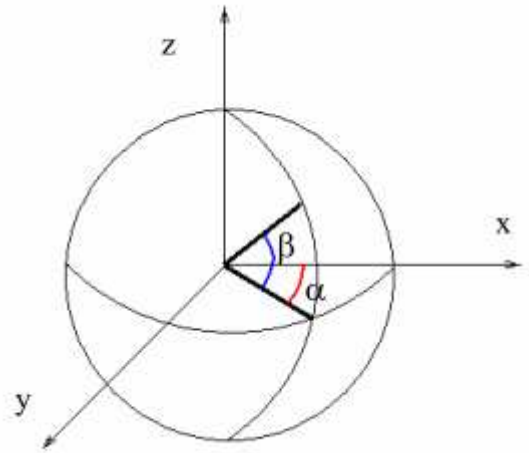
```
box(width, height, depth);
```

SPHERE

A sphere with a radius r and a centre at the beginning of the coordinate system can be described as follows parametric equations:

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

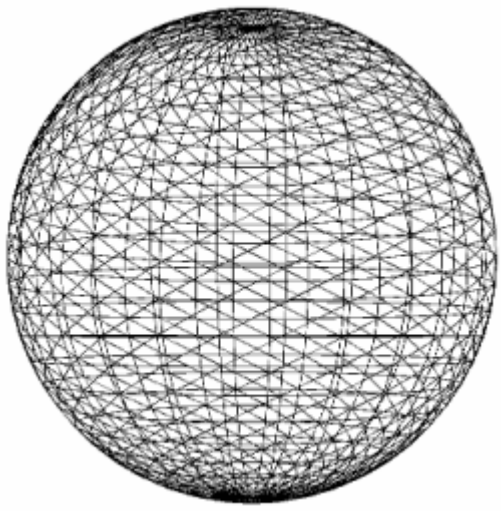
where the angle $\alpha \in \langle 0; 2\pi \rangle$ is the longitude, and the angle $\beta \in \langle -\pi/2; \pi/2 \rangle$ is the latitude.



The sphere drawing can be obtained by using the **sphere()** procedure, remembering to move the **translate()** image, because the beginning of the coordinate system is in the upper left corner of the screen and using the **[pushMatrix()]** matrix that saves the current coordinate system and at the end of the **[popMatrix()]** restoring the previous coordinate system after the transformation - in our case, the translation is the shift. Using the **pushMatrix()** function causes the beginning and directions of the axes of the new coordinate system to be determined by the transformation functions (e.g. translation). Subsequent transformations and drawing of objects are performed in the new coordinate system. After calling the **popMatrix()** function, all previous coordinate system settings are recovered.

```
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();
}
```



Three-dimensional effect can be achieved by adding lighting. You can use e.g. **spotlight()** function. The following example shows you how you can illuminate the ball. The effect is achieved by pressing and holding down the mouse button.

```
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);
}
```

Parametrs:

- 255, 0, 0 – red color
- width/2, height/2, 400 – setting in the window
- 0, 0, -1 – direction of illumination
- PI/4 – lighting angle
- 2 – light concentration

Two standard three-dimensional solids were made available in Processing: a cuboid and a sphere. The remaining solids, or in fact polyhedrons, need to be defined separately as vertices in space connected with each other. The vertices are defined using the **vertex()** function.

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

A solid definition block is a sequence of functions defining vertices limited by the following instructions

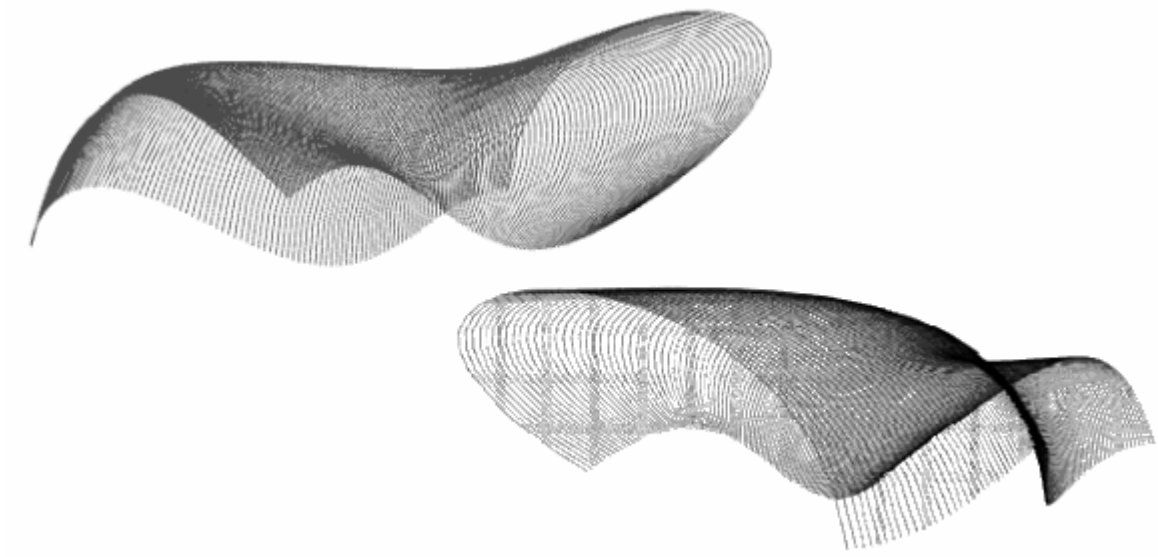
beginShape() and **endShape().**

Example (a pyramid with a quadrilateral base)

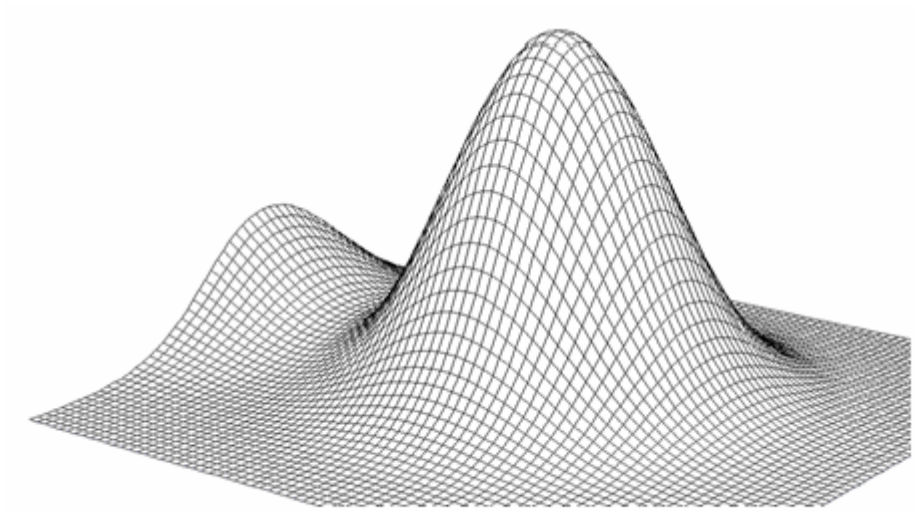
```
beginShape();
  vertex(-100, -100, -100); vertex( 100, -100, -100); vertex(0, 0, 100);
  vertex( 100, -100, -100); vertex( 100,  100, -100); vertex(0, 0, 100);
  vertex( 100, 100, -100);  vertex(-100, 100, -100);  vertex(0, 0, 100);
  vertex(-100, 100, -100);  vertex(-100, -100, -100); vertex(0,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

Using the **`bezier()`** function in the 3D version, we can obtain interesting surface drawings.
Using the mouse we can rotate them in two planes.

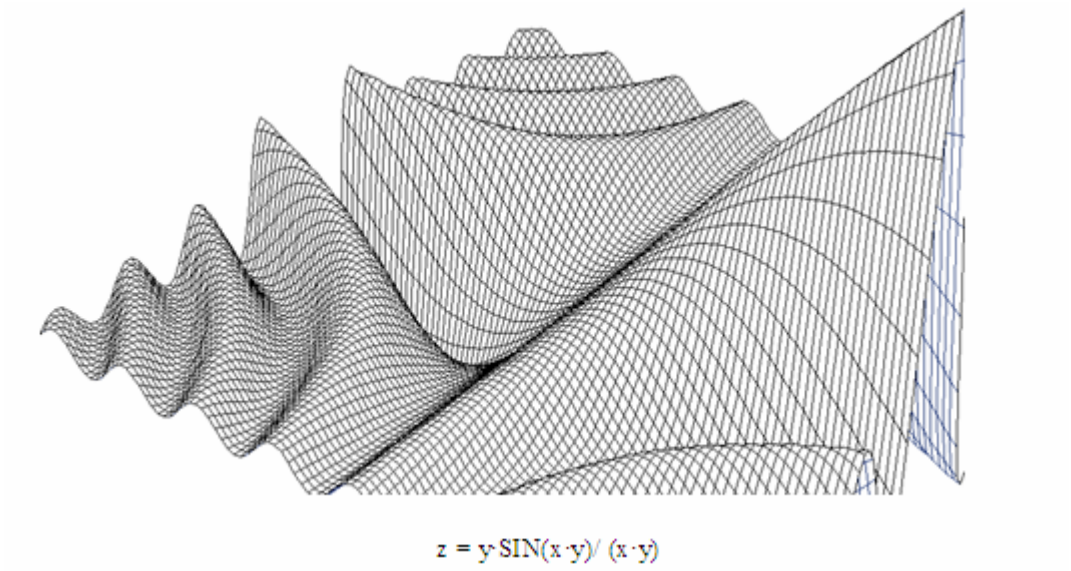
```
float px [] = new float[4];
float py [] = new float[4];
void setup()
{
  size(600,400,P3D);
}
void draw()
{
  background(60);
  translate(300,200);
  rotateX(mouseY*PI/300);
  rotateZ(mouseX*PI/300);
  noFill();
  stroke(250,160);
for (int i=-60;i<60;i++)
{
  px[0] = -100;  py[0] = -50+sin(i*PI/60.0)*25;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



Finally, examples of graphics in the form of function charts:



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



z = y·SIN(x·y)/ (x·y)