translate(tx, ty, tz); scale(sx, sy, sz)

rotateX(a), rotateY(a), rotateZ(a)

box(width, height, depth)

sphere(radius) -> center (0,0,0)

sphereDetail(n) -> vertices are spaced by 360°/n, default is n = 30

stroke (r, g, b); fill(r, g, b); noFill() -> wired

lights() -> basic illumination

Two cubes

```
size(800, 600, P3D);
background(0);
translate(0, height/2, -height/4);
box(60, 80, 60);
translate(width, 0, -height/2);
box(60, 80, 60);
```

Two spheres

```
size(800, 600, P3D);
background(0);
translate(0, height/2, -height/4);
sphere(100);
translate(width, 0, -height/2);
sphere(100);
```

Cube and sphere, lights, fill

```
size(800, 600, P3D);
noStroke();
background(0);
lights();
translate(0, height/2, -height/2);
```

```
fill(153);
box(100);
translate(width, 0, 0);
fill(255);
sphere(150);
```

Rotating cube

```
void setup() {
  size(800, 600, P3D);
  fill(255, 0, 0);
// noStroke();
  frameRate(30);
}
void draw() {
  background(0);
 // lights();
  translate(width/2, height/2);
  rotateX(frameCount*PI/60.0);
  rotateY(frameCount*PI/120.0);
  rotateZ(frameCount*PI/180.0);
  box(200, 200, 200);
}
```

```
void setup() {
  size(800, 600, P3D);
}
void draw() {
  background(255);
  translate(width/2, height/2, 0);
 // theta += 0.01;
```

```
// rotateX(theta);
// rotateY(theta);
 drawPyramid(150);

/*
  translate(100, 100, 20);
  drawPyramid(10);
*/
}
void drawPyramid(int t) {
  stroke(0);
  beginShape(TRIANGLES);
    fill(150, 0, 0, 127);
    vertex(-t, -t, -t);
    vertex( t, -t, -t);
    vertex( 0, 0, t);
    fill(0, 150, 0, 127);
    vertex( t, -t, -t);
    vertex( t, t, -t);
    vertex( 0, 0, t);
    fill(0, 0, 150, 127);
    vertex( t, t, -t);
    vertex(-t, t, -t);
    vertex( 0, 0, t);
    fill(150, 0, 150, 127);
    vertex(-t, t, -t);
    vertex(-t, -t, -t);
    vertex( 0, 0, t);
 endShape();
}
```

Growing rectangle (with translate on z)

```
float void setup() {
  size(200, 200, P3D);
}
void draw() {
  background(255);
  stroke(0);
  fill(175);
  translate(width/2, height/2, z);
  rectMode(CENTER);
  rect(0, 0, 8, 8);
  z += 2;
  if (z > 200) {
    z = 0;
  }
}
```

Growing rectangle (with scale)

```
float r = 0.0;
void setup() {
  size(200, 200);
}
void draw() {
  background(255);
  translate(width/2, height/2);
  scale(r);
  stroke(0);
  fill(175);
  rectMode(CENTER);
```

```
  rect(0, 0, 10, 10);
  r += 0.2;
  if (r > 20) {
    r = 0;
  }
}
```

Two rectangles scaled

```
float a = 0.0;
float s = 0.0;
void setup() {
  size(640, 360);
  noStroke();
  rectMode(CENTER);
  frameRate(30);
}
void draw() {
  background(102);
  a = a + 0.04;
  s = cos(a)*2;
  translate(width/2, height/2);
  scale(s);
  fill(51);
  rect(0, 0, 50, 50);
  translate(75, 0);
  fill(255);
  scale(s);
  rect(0, 0, 50, 50);
}
```

Rotating rectangle by Y (същото по x и по z)

```
float theta = 0;
void setup() {
  size(200, 200, P3D);
}
void draw() {
  background(255);
  stroke(0);
  fill(175);
  rectMode(CENTER);
  translate(150, 150);
  rotateY(theta);
  rect(0, 0, 60, 60);
  theta2 += 0.02;
}
```

Rotater class

```
class Rotater {
  float x, y;   // x,y location
  float theta;  // angle of rotation
  float speed;  // speed of rotation
  float w;      // size of rectangle
  Rotater(float tempX, float tempY, float tempSpeed, float tempW) {
    x = tempX;
    y = tempY;
    theta = 0;
    speed = tempSpeed;
    w = tempW;
  }
```

```
  void spin() {
    theta += speed;
  }
  void display() {
    rectMode(CENTER);
    stroke(0);
    fill(0, 100);
    pushMatrix();
    translate(x, y);
    rotate(theta);
    rect(0, 0, w, w);
    popMatrix();
  }
}

Rotater[] rotaters;
void setup() {
  size(640, 360);
  rotaters = new Rotater[20];
  for (int i = 0; i < rotaters.length; i++ ) {
    rotaters[i] = new Rotater(random(width), random(height),
                              random(-0.1, 0.1), random(48));
  }
}
void draw() {
  background(255);
  for (int i = 0; i < rotaters.length; i++ ) {
    rotaters[i].spin();
    rotaters[i].display();
  }
```

```
}
```

Simple solar system

```
float theta = 0;
void setup() {
  size(640, 360);
}
void draw() {
  background(255);
  stroke(0);
  translate(width/2, height/2);
  fill(255, 200, 50);
  ellipse(0, 0, 64, 64);
  pushMatrix();
  rotate(theta);
  translate(100, 0);
  fill(50, 200, 255);
  ellipse(0, 0, 32, 32);
  pushMatrix();
  rotate(-theta*4);
  translate(36, 0);
  fill(50, 255, 200);
  ellipse(0, 0, 12, 12);
  popMatrix();
  pushMatrix();
  rotate(theta*2);
  translate(24, 0);
  fill(50, 255, 200);
  ellipse(0, 0, 6, 6);
  popMatrix();
```

```
  popMatrix();
  theta += 0.01;
}
-------------------------------------------------------------------------
// The PShape object
PShape star;
void setup() {
  size(640, 360);
  star = createShape();
  star.beginShape();
  star.fill(102);
  star.stroke(0);
  star.strokeWeight(2);
  star.vertex(0, -50);
  star.vertex(14, -20);
  star.vertex(47, -15);
  star.vertex(23, 7);
  star.vertex(29, 40);
  star.vertex(0, 25);
  star.vertex(-29, 40);
  star.vertex(-23, 7);
  star.vertex(-47, -15);
  star.vertex(-14, -20);
  star.endShape(CLOSE);
}
void draw() {
  background(255);
  translate(mouseX, mouseY);
  shape(star);
```

```
}
```

Solar system OOP

```
class Planet {
  float theta;      // Rotation around sun
  float diameter;   // Size of planet
  float distance;   // Distance from sun
  float orbitspeed; // Orbit speed
  Planet(float distance_, float diameter_) {
    distance = distance_;
    diameter = diameter_;
    theta = 0;
    orbitspeed = random(0.01, 0.03);
  }
  void update() {
    // Increment the angle to rotate
    theta += orbitspeed;
  }
  void display() {
    pushMatrix();
    rotate(theta);
    translate(distance, 0);
    stroke(0);
    fill(175);
    ellipse(0, 0, diameter, diameter);
    popMatrix();
  }
}
```

```
Planet[] planets = new Planet[4];
void setup() {
  size(640,360);
  for (int i = 0; i < planets.length; i++ ) {
    planets[i] = new Planet(64 + i*32,24);
  }
}
void draw() {
  background(255);
  // Drawing the Sun
  pushMatrix();
  translate(width/2,height/2);
  stroke(0);
  fill(255);
  ellipse(0,0,64,64);
  // Drawing all Planets
  for (int i = 0; i < planets.length; i++ ) {
    planets[i].update();
    planets[i].display();
  }
  popMatrix();
}
```

Solar system OOP with moon

```
class Planet {
  float theta;      // Rotation around sun
  float diameter;   // Size of planet
  float distance;   // Distance from sun
  float orbitspeed; // Orbit speed
  Moon moon;  // Each Planet now has a Moon!
```

```
  Planet(float distance_, float diameter_) {
    distance = distance_;
    diameter = diameter_;
    theta = 0;
    orbitspeed = random(0.01,0.03);
    // create the Moon 24 pixels from the planet with a diameter of 5
    moon = new Moon(24,8);
  }
  void update() {
    theta += orbitspeed;      // Increment the angle to rotate
    moon.update();            // Update the moon
 }
 void display() {
    pushMatrix();
    rotate(theta);
    translate(distance,0);
    stroke(0);
    fill(175);
    ellipse(0,0,diameter,diameter);
    moon.display();
    popMatrix();
  }
}
class Moon {
  float theta;      // Rotation around sun
  float diameter;   // Size of planet
  float distance;   // Distance from sun
  float orbitspeed; // Orbit speed
  Moon(float distance_, float diameter_) {
    distance = distance_;
```

```
    diameter = diameter_;
    theta = 0;
    orbitspeed = random(-0.1,0.1);
  }
 void update() {
    theta += orbitspeed; // Increment the angle to rotate
  }
 void display() {
    pushMatrix();
    rotate(theta);              // Rotate orbit
    translate(distance,0);    // translate out distance
    stroke(0);
    fill(175);
    ellipse(0,0,diameter,diameter);
    popMatrix();
  }
}
```

Extend the solar system example into three dimensions.
