```
import processing.sound.*;

//Declare variables for the 'terrain'
int cols,rows;//columns and rows
int scl = 20;//scale
int w = 3000;//width
int h = 2000;//height

// Declare the processing sound variables
SoundFile sample;
Amplitude rms;

//Declare the start for terrain
float flying = 0;
float [][] terrain;

// Declare a scaling factor
float scale = 5.0;

// Declare a smooth factor
float smoothFactor = 0.25;

// Used for smoothing
float sum;

void setup() {
  size(displayWidth, displayHeight, P3D);

  cols = w / scl;
  rows = h / scl;
  terrain = new float[cols][rows];

  //Load and play a soundfile and loop it
  sample = new SoundFile(this, "beat.mp3");
  sample.loop();

  // Create and patch the rms tracker
  rms = new Amplitude(this);
  rms.input(sample);
}

void draw() {
  flying -=0.1;
   float yoff = flying;
  for (int y = 0; y < rows; y++)
  {
    float xoff =0;
   for (int x = 0; x < cols; x++){
     terrain[x][y] = map(noise(xoff,yoff), 0, 1, -100, 100);
     xoff += 0.2;
  }
  yoff +=0.2;
}
 // Set background color, stroke and nofill
 background(0);
 stroke(255);
 noFill();

 translate(width/2, height/2+50);//place on screen
 rotateX(PI/3);
 translate(-w/2, -h/2);
 for (int y = 0; y < rows-1; y++)
   {
      beginShape(TRIANGLE_STRIP);
      for (int x = 0; x < cols; x++){
        vertex(x*scl, y*scl, terrain[x][y]);
        vertex(x*scl, (y+1)*scl, terrain[x][y+1]);
   }
   endShape();


 // Smooth the rms data by smoothing factor
 sum += (rms.analyze() - sum) * smoothFactor;

 // rms.analyze() return a value between 0 and 1. It's
 // scaled to height/2 and then multiplied by a scale factor
 float rmsScaled = sum * (height/2) * scale;

 // Draw an ellipse at a size based on the audio analysis
 ellipse(1400, -1300, rmsScaled, rmsScaled);
 ellipse(1100, -1050, rmsScaled, rmsScaled);
 ellipse(1700, -1600, rmsScaled, rmsScaled);
   }

   if (keyPressed)
   {
     exit();
   }
   if (mousePressed)
   {
   exit();
 }
}
```