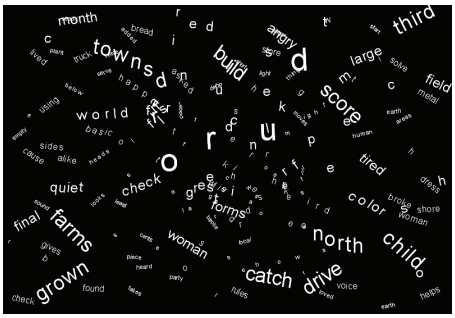
Font Physics

CODE




```
import java.awt.geom.Line2D;

static final float wordRatio = 2.4; // approx ratio of string lenth to font size
static final int fontdepth = 5; // recursion depth for word creation

static final float G = -1.0; //change gravity
static final float cursor_mass = 10000000.; //mass of cursor
float damping = 2.5;

float max_length;

Letter[] letters; //array of the letters
Line2D[] lines; // text lines for intersections

static final char[] charset = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z' };

class Letter
{
  PVector p;
  PVector o;
  PVector dp;
  char letter;
  float fontsize;
  float theta;
  float mass;
  float k;
  Letter(PVector pos,PVector dpos,char letterval,float fontsizeval,float thetaval)
  {
    p = new PVector(pos.x,pos.y);
    o = new PVector(pos.x,pos.y);
    dp = new PVector(dpos.x,dpos.y);
    letter = letterval;
    theta = thetaval;
    fontsize = fontsizeval;
    //mass = fontsize/20.;
    mass = 1.;
    k = 10.;
  }
}

void setup()
{
  //size(600,600,P2D);
  //size(screen.width,screen.height,JAVA2D);
  size(800,550,JAVA2D);
  hint(ENABLE_NATIVE_FONTS);
  reset();
}

void keyPressed()
{
  if( key == ' ' ) {
    reset();
  } else if( key == '[' ) {
    damping *= 0.8;
  } else if( key == ']' ) {
    damping *= 1.2;
  }
}
```

CODE

```
void reset()
{
  max_length = min(width,height)/4.0;
  letters = new Letter[0];
  lines = new Line2D[0];
  textFont(createFont("Arial", max_length/wordRatio, true, charset));
  //textMode(SCREEN);
  textSize(24.0);
  text( "loading", (displayWidth - textWidth("loading"))/2.0,
(displayHeight - 24.0)/2.0);
  for(int i = 0; i < fontdepth; ++i) {
    float fontsize = (max_length*pow(0.66,i)/wordRatio);
    for(int j = 0; j < 10*i + 1; ++j) {
      boolean success = false;
      for(int t = 0;t < 100; ++t) {
        boolean thisTry = makeWord(fontsize);
        if( thisTry ) {
          success = true;
          break;
        }
      }
      if( !success ) {
        println("failed to make letter");
      }
    }
  }
  background(0);
}

boolean makeWord(float fontsize)
{
  textSize(fontsize);
  PVector pos = new PVector(random(width),random(height));
  String word = words[(int)random(words.length)];
  float theta = random(PI/2.0) - PI/4.0;
  float wlength = textWidth(word);
  PVector end = new PVector(pos.x + wlength*cos(theta), pos.y +
wlength*sin(theta));
  if( end.x < 0 || width < end.x || end.y < 0 || height < end.y ) {
    return false;
  }
  Letter[] newword = layoutWord(word, pos, fontsize, theta);
  Line2D[] newlines = new Line2D[4];
  float xoff = fontsize*cos(theta-HALF_PI);
  float yoff = fontsize*sin(theta-HALF_PI);
  newlines[0] = (Line2D) new Line2D.Float( pos.x, pos.y, end.x, end.y
);
  newlines[1] = (Line2D) new Line2D.Float( end.x, end.y, end.x +xoff,
end.y +yoff );
  newlines[2] = (Line2D) new Line2D.Float( end.x +xoff, end.y +yoff,
pos.x +xoff, pos.y +yoff);
  newlines[3] = (Line2D) new Line2D.Float( pos.x +xoff, pos.y +yoff,
pos.x, pos.y );

  for(int i = 0; i < lines.length; ++i) {
    for(int j = 0; j < 4; ++j) {
      if( newlines[j].intersectsLine(lines[i]) ) {
        return false;
      }
    }
  }
  letters = (Letter[])concat(letters, newword );
  lines = (Line2D[])concat(lines,newlines);
  return true;
}

float time = 0.0;
```

CODE

```
void draw()
{
  float dtime = millis()/1000.0 - time;
  time += dtime;
  calcAccel(dtime);
  calcMove(dtime);
  resetMatrix();
  background(0);
  fill(255);
  stroke(255);
  for(int i = 0; i < letters.length; ++i) {
    pushMatrix();
    translate(letters[i].p.x,letters[i].p.y);
    rotate(letters[i].theta);
    textSize(letters[i].fontsize);
    text(letters[i].letter,0,0);
    popMatrix();
  }
}

void calcAccel(float time)
{
  if( mousePressed ) {
    PVector mouse_p = new PVector((float)mouseX,(float)mouseY);
    float Gprime = G;
    if( mouseButton == RIGHT ) {
      Gprime *= -1;
    }
    for(int i = 0; i < letters.length; ++i) {
      PVector r = new PVector(letters[i].p.x, letters[i].p.y);
      r.sub(mouse_p);
      PVector d = new PVector(letters[i].p.x, letters[i].p.y);
      d.sub(letters[i].o);
      PVector accel = new PVector(r.x,r.y);
      accel.mult( Gprime*cursor_mass/max( pow(r.mag(),3.0), 250.0 )
);
      PVector tmp_1 = new PVector(d.x,d.y);
      tmp_1.mult( -letters[i].k/letters[i].mass );
      accel.add( tmp_1 );
      PVector tmp_2 = new PVector(letters[i].dp.x,letters[i].dp.y);
      tmp_2.mult( -damping/letters[i].mass );
      accel.add( tmp_2 );
      accel.mult( time );
      //println( str(accel.x) + str(accel.y) );
      letters[i].dp.add(accel);
    }
  } else {
    for(int i = 0; i < letters.length; ++i) {
      PVector d = new PVector(letters[i].p.x, letters[i].p.y);
      d.sub(letters[i].o);
      PVector accel = new PVector(d.x,d.y);
      accel.mult( -letters[i].k/letters[i].mass );
      PVector tmp_2 = new PVector(letters[i].dp.x,letters[i].dp.y);
      tmp_2.mult( -damping/letters[i].mass );
      accel.add( tmp_2 );
      accel.mult( time );
      letters[i].dp.add(accel);
    }
  }
}

void calcMove(float time)
{
  for(int i = 0; i < letters.length; ++i) {
    PVector tmp_3 = new PVector(letters[i].dp.x,letters[i].dp.y);
    tmp_3.mult(time);
    letters[i].p.add(tmp_3);
  }
}
```

CODE

```
Letter[] layoutWord(String word, PVector pos, float fontsize, float
theta)
{
  Letter[] newletters = new Letter[5];
  PVector cur_pos = new PVector(pos.x,pos.y);
  PVector zero = new PVector(0.0,0.0);
  for(int i = 0; i < 5; ++i) {
    Letter l = new Letter( cur_pos, zero, word.charAt(i), fontsize, theta
);
    newletters[i] = l;
    textSize(fontsize);
    float w = textWidth(word.charAt(i));
    cur_pos.x += w*cos(theta);
    cur_pos.y += w*sin(theta);
  }
  return newletters;
}

String[] words = {
  "which", "there", "their", "about", "would", "these", "other",
"words", "could", "write", "first", "water", "after", "where", "right",
"think", "three", "years", "place", "sound", "great", "again", "still",
"every", "small", "found", "those", "never", "under", "might",
"while", "house", "world", "below", "asked", "going", "large",
"until", "along", "shall", "being", "often", "earth", "began", "since",
"study", "night", "light", "above", "paper", "parts", "young",
"story", "point", "times", "heard", "whole", "white", "given",
"means", "music", "miles", "thing", "today", "later", "using",
"money", "lines", "order", "group", "among", "learn", "known",
"space", "table", "early", "trees", "short", "hands", "state", "black",
"shown", "stood", "front", "voice", "kinds", "makes", "comes",
"close", "power", "lived", "vowel", "taken", "built", "heart", "ready",
"quite", "class", "bring", "round", "horse", "shows", "piece",
"green", "stand", "birds", "start", "river", "tried", "least", "field",
"whose", "girls", "leave", "added", "color", "third", "hours",
"moved", "plant", "doing", "names", "forms", "heavy", "ideas",
"cried", "check", "floor", "begin", "woman", "alone", "plane",
"spell", "watch", "carry", "wrote", "clear", "named", "books",
"child", "glass", "human", "takes", "party", "build", "seems",
"blood", "sides", "seven", "mouth", "solve", "north", "value",
"death", "maybe", "happy", "tells", "gives", "looks", "shape",
"lives", "steps", "areas", "sense", "speak", "force", "ocean",
"speed", "women", "metal", "south", "grass", "scale", "cells",
"lower", "sleep", "wrong", "pages", "ships", "needs", "rocks",
"eight", "major", "level", "total", "ahead", "reach", "stars", "store",
"sight", "terms", "catch", "works", "board", "cover", "songs",
"equal", "stone", "waves", "guess", "dance", "spoke", "break",
"cause", "radio", "weeks", "lands", "basic", "liked", "trade",
  "fresh", "final", "fight", "meant", "drive", "spent", "local", "waxes",
"knows", "train", "bread", "homes", "teeth", "coast", "thick",
"brown", "clean", "quiet", "sugar", "facts", "steel", "forth", "rules",
"notes", "units", "peace", "month", "verbs", "seeds", "helps",
"sharp", "visit", "woods", "chief", "walls", "cross", "wings", "grown",
"cases", "foods", "crops", "fruit", "stick", "wants", "stage", "sheep",
"nouns", "plain", "drink", "bones", "apart", "turns", "moves",
"touch", "angle", "based", "range", "marks", "tired", "older",
"farms", "spend", "shoes", "goods", "chair", "twice", "cents",
"empty", "alike", "style", "broke", "pairs", "count", "enjoy", "score",
"shore", "roots", "paint", "heads", "shook", "serve", "angry",
"crowd", "wheel", "quick", "dress", "share", "alive", "noise", "solid",
"cloth", "signs", "hills", "types", "drawn", "worth", "truck", "piano",
"upper", "loved", "usual", "faces", "drove", "cabin", "boats",
"towns", "proud"
};
```