CMSC427 Fall 2017 Fan Yang

The Frenet frame is calculate by the following steps:

First, find the tangent vector T by computing the first derivatives of the x, y, z functions. The tangent vector does not have to be normalized for now.

$T = \langle -\sin(t), 5\cos(10t), \cos(t) \rangle$

Second, find the normal vector N′ by computing the second derivatives of the x, y, z functions. Similar to the tangent, the normal does not have to be normalized for now. Moreover, this normal is not orthogonal to the tangent. But they two define the tangent-normal plane, so we can use them to calculate the binomial vector.

$N' = \langle -\cos(t), -50\sin(10t), -\sin(t) \rangle$

Third, calculate the binomial by taking the cross product of tangent and normal. The binomial is perpendicular to both T and N. It can also be unnormalized for now.

$B = T \times N'$

Forth, re-compute the normal vector by taking the cross product of B and T. This step fixes the issue that N is not orthogonal to T. At this point, we have achieved a set of three orthogonal vectors T, N, B.

$N = B \times T$

Finally, normalized the three vectors. The resulting orthogonal unit-length vectors define the desired Frenet frame.

```
T = T.normalize();
N = N.normalize();
B = B.normalize();
```

I have also changed the way the program rotates the 3D shape. Instead of using mouse position to define the angle of rotation, I use the accumulated change of the mouse position to do the job. As the mouse moves, the changes of the x & y coordinates of the mouse are accumulated and assigned to two variables which replace "mouseX" and "mouseY" in the rotation methods. This modification makes the rotation smoother and avoid abrupt jumps when the mouse enters the canvas region.

```
// CMSC427 Lab 4 Fall 2017
// HelixFly2.pde
// Draw a parametric "crown" in 3D
// Use Frenet frame to align boxes with curve
// R. Eastman & Fan Yang

void setup() {
  size(1000,600,P3D);
  colorMode(HSB,360,100,100);
}

void draw() {
  background(0,0,0);
  translate(width/2,height/2);
  // rotate the 3D shape using the accumulated change of the mouse position.
  rotateY(rx*PI/width);
  rotateX(ry*PI/height);
  for(float t = 0; t < 2*PI; t+=PI/120){
    float x = 200*cos(t);
    float y = 100*sin(10*t);
    float z = 200*sin(t);
    fill(1.3*(x+100),100,100,255);
    // We push the matrix for each box so translations don’t accumulate
    pushMatrix();
    translate(x,y,z);
    applyFrenet(t); // Correct the method and then uncomment this
    scale(1,1,2);
    box(10);
    popMatrix();
  }
}

// Accumulate the change of mouse position as the mouse moves.
// The accumulated changes in x and y directions are assigned to rx and ry, respectively
int mx, my, rx = 0, ry = 0;
void mouseMoved(MouseEvent e) {
  //when the mouse enters the canvas, initialize mx & my
  if(e.getX()<10 || e.getX()>width-10 || e.getY()<10 || e.getY()>height-10){
    mx = e.getX();
    my = e.getY();
    return;
  }

  // Accumulate the change of mouse position as the mouse moves.
  int dx = e.getX() - mx;
  int dy = e.getY() - my;
  if(dx*dx + dy*dy > 10) {
    rx += dx;
    ry += dy;
    mx = e.getX();
    my = e.getY();
  }
}

// Method to create a local coordinate system
// and then generate a rotation matrix
PVector T = new PVector(), N = new PVector(), B = new PVector();
void applyFrenet(float t) {
  // Tangent vector = <-sint, 5cos(10t), cost>
  T.set(-sin(t), 5*cos(10*t), cos(t)).normalize();

  // Curvature vector = <-cost, -50sin(10t), -sint>. This curvature vector is not orthogonal to the tantgent.
  N.set(-cos(t), -50*sin(10*t), -sin(t));

  // Binormal
  B = T.cross(N).normalize();

  // re-compute the curvature vector using N = B x T so that T and N become perpendicular to each other
  N = B.cross(T);

  // ascert that the three vectors are of unit lengths and are perpendicular to each other
  assertEquals(T.mag(), 1.0);
  assertEquals(N.mag(), 1.0);
  assertEquals(B.mag(), 1.0);
  assertEquals(T.dot(N), 0.0);
  assertEquals(N.dot(B), 0.0);
  assertEquals(B.dot(T), 0.0);

  applyMatrix(B.x, N.x, T.x, 0,
              B.y, N.y, T.y, 0,
                        B.z, N.z, T.z, 0,
                        0,  0,  0,  1);
}

void assertEquals(float a, float b){
  if(abs(a-b) > 0.000001){
    println("not equal!");
    System.exit(-1);
  }
}
```