```
// Node struct
class Node {
  Vec3 pos;
  Vec3 vel;
  Vec3 last_pos;

  Node(Vec3 pos) {
    this.pos = pos;
    this.vel = new Vec3(0,0,0);
    this.last_pos = pos;
  }
  void handleSphereCollision() {
    float d = Sphere_Pos.distanceTo(this.pos);
    if (d < Sphere_Radius + 0.09) {
      Vec3 Sphere_normal = (this.pos).minus(Sphere_Pos);
      Sphere_normal.normalize();
      (this.pos) = Sphere_Pos.plus((Sphere_normal.times(Sphere_Radius
+ 0.09)));


    }
  }
}


int numNodes = 40;
int Num_Rows = 75;
int Num_Cols = 100;

Vec3 base_pos = new Vec3(2, 0,0);


Node nodes[] = new Node[numNodes];

Node nodes_cloth[][] = new Node[Num_Cols][Num_Rows];


float d = 0;
Vec3 Sphere_normal = new Vec3(0,0,0);
Vec3 bounce = new Vec3(0,0,0);

PImage textureImage;
PShape sphereShape;

void setup() {
  size(1000, 1000 , P3D);
  surface.setTitle("Chain of Nodes for cloth simulation!");
    aka = loadImage("aka.jpg");
      textureMode(NORMAL);
    hidden = loadImage("rain.jpg");
    moon = loadImage("moon.png");
      textureMode(NORMAL);
  scene_scale = width / 10.0f;

  textureImage = loadImage("moon.png"); // Replace with your image
file
  textureMode(NORMAL);
  // Sphere_Pos.times(scene_scale) , Sphere_Radius*scene_scale
  noStroke();
  sphereShape = createShape(SPHERE, Sphere_Radius*scene_scale); //
Create a sphere shape
  sphereShape.setTexture(textureImage); // Apply the texture to the
sphere


  nodes_cloth[0][0] = new Node(base_pos);



  for(int i = 0; i < Num_Cols; i++){

    for(int j = 0; j < Num_Rows; j++){

      Vec3 node_pos = new Vec3(base_pos.x + (i * (0.1)) , base_pos.y +
(j * (0.1)) , base_pos.z + (i * 0.1)/2);
      nodes_cloth[i][j] = new Node(node_pos);

    }

  }

}



// Link length
float link_length = 0.09;
```

```
// Gravity
Vec3 gravity = new Vec3(0, 10,0);


// Scaling factor for the scene
float scene_scale = width / 10.0f;

// Physics Parameters
int relaxation_steps = 10;
int sub_steps = 10;




void update_physics(float dt) {


  for(int i = 0; i < Num_Cols; i++){

    for(int j = 1; j < Num_Rows; j++){

      nodes_cloth[i][j].last_pos = nodes_cloth[i][j].pos;
      nodes_cloth[i][j].vel =
nodes_cloth[i][j].vel.plus(gravity.times(dt));
      nodes_cloth[i][j].pos =
nodes_cloth[i][j].pos.plus(nodes_cloth[i][j].vel.times(dt));
      // Sphere-Cloth Collision
      nodes_cloth[i][j].handleSphereCollision();


    }
  }


  for (int counter = 0; counter < relaxation_steps; counter++) {

    for(int i = 0; i < Num_Cols; i++){

      for(int j = 0; j < Num_Rows-1; j++){

      Vec3 delta_y =
nodes_cloth[i][j+1].pos.minus(nodes_cloth[i][j].pos);
      float delta_y_len = delta_y.length();
      float correction_y = delta_y_len - link_length;
      Vec3 delta_y_normalized = delta_y.normalized();
      nodes_cloth[i][j+1].pos =
nodes_cloth[i][j+1].pos.minus(delta_y_normalized.times(correction_y /
2));
      nodes_cloth[i][j].pos =
nodes_cloth[i][j].pos.plus(delta_y_normalized.times(correction_y /
2));
      nodes_cloth[i][0].pos = new Vec3(base_pos.x + (i*0.1) ,
base_pos.y , base_pos.z);

      }

    }


    for(int i = 0; i < Num_Cols-1; i++){

      for(int j = 0; j < Num_Rows; j++){

      Vec3 delta_x =
nodes_cloth[i+1][j].pos.minus(nodes_cloth[i][j].pos);
      float delta_x_len = delta_x.length();
      float correction_x = delta_x_len - link_length;
      Vec3 delta_x_normalized = delta_x.normalized();
      nodes_cloth[i+1][j].pos =
nodes_cloth[i+1][j].pos.minus(delta_x_normalized.times(correction_x /
2));
      nodes_cloth[i][j].pos =
nodes_cloth[i][j].pos.plus(delta_x_normalized.times(correction_x /
2));

      }

    }




    for(int i = 0; i < Num_Cols; i++){

      for(int j = 0; j < Num_Rows; j++){

        nodes_cloth[i][j].vel =
nodes_cloth[i][j].pos.minus(nodes_cloth[i][j].last_pos).times(1 / dt);
        //nodes_cloth[i][j].handleSphereCollision();
      }
    }




  }


}



boolean paused = false;
boolean move_up = false;
boolean move_down = false;
boolean move_right = false;
boolean move_left = false;
boolean z_in = false;
boolean z_out = false;

void keyPressed() {
  if (key == ' ') {
    paused = !paused;
  }

  if (keyCode == UP) {
    move_up = true;
  }

  if (keyCode == DOWN) {
    move_down = true;
  }

  if (keyCode == RIGHT) {
    move_right = true;
  }

  if (keyCode == LEFT) {
    move_left = true;
  }

  if (key == 'a') {
    z_in = true;
  }

  if (key == 'd') {
    z_out = true;
  }

}


void keyReleased() {
  //if (key == ' ') {
  //  paused = !paused;
  //}

  if (keyCode == UP) {
    move_up = false;
  }

  if (keyCode == DOWN) {
    move_down = false;
  }

  if (keyCode == RIGHT) {
    move_right = false;
  }

  if (keyCode == LEFT) {
    move_left = false;
  }

  if (key == 'a') {
    z_in = false;
  }

  if (key == 'd') {
    z_out = false;
  }

}




void drawSphere(Vec3 center, float radius){
  beginShape();
  pushMatrix();
  translate(Sphere_Pos.x * scene_scale, Sphere_Pos.y * scene_scale,
Sphere_Pos.z * scene_scale);
  moon = loadImage("moon.png");
  texture(moon);
  sphere(Sphere_Radius * scene_scale);
  //texture(moon); // Apply the texture to the sphere
  popMatrix();
  endShape();
}

Vec3 Sphere_Pos = new Vec3(3,4,0);
float Sphere_Radius = 0.5;


float time = 0;
PImage aka;
PImage hidden;
PImage moon;
void draw() {
  float dt = 1.0 / 20; //Dynamic dt: 1/frameRate;

  if (!paused) {
    for (int i = 0; i < sub_steps; i++) {
      time += dt / sub_steps;
      update_physics(dt / sub_steps);
    }
  }

  if(move_up == true){
    Sphere_Pos.y -= 0.05;

  }

  if(move_down == true){
    Sphere_Pos.y += 0.05;
  }

  if(move_right == true){
    Sphere_Pos.x += 0.05;
    //rotateY(0.02);
    //rotateZ(0.05);
  }
  if(move_left == true){
    Sphere_Pos.x -= 0.05;

  }

  if (z_in == true) {
    Sphere_Pos.z -= 0.05;
  }

  if (z_out == true) {
    Sphere_Pos.z += 0.05;
  }



 background(hidden);

 noStroke();
  fill(100);
  noStroke();
    //drawSphere(Sphere_Pos.times(scene_scale) ,
Sphere_Radius*scene_scale);
  beginShape();
  pushMatrix();
  //noStroke();
  translate(Sphere_Pos.x * scene_scale, Sphere_Pos.y * scene_scale,
Sphere_Pos.z * scene_scale);
  rotateY(frameCount * 0.1);
  //moon = loadImage("moon.png");
  //texture(moon);
  //sphere(Sphere_Radius * scene_scale);
  //texture(moon); // Apply the texture to the sphere
  //noStroke();
  shape(sphereShape);
  //noStroke();
  popMatrix();
  endShape();

  noStroke();
  noFill();
  for(int i = 0; i < Num_Cols-1; i++){
    for(int j = 0; j < Num_Rows-1; j++){
      beginShape(TRIANGLE_STRIP);
      texture(aka);
```

```
         // Define u and v coordinates using the map function
      float u1 = map(i, 0, Num_Cols-1, 0, 1);
      float v1 = map(j, 0, Num_Rows-1, 0, 1);
      float u2 = map(i + 1, 0, Num_Cols-1, 0, 1);
      float v2 = map(j + 1, 0, Num_Rows-1, 0, 1);
      // Connect the nodes using vertices
      //nodes_cloth[i][j].handleSphereCollision();
    // Vertex 1
    vertex(nodes_cloth[i][j].pos.x * scene_scale,
nodes_cloth[i][j].pos.y * scene_scale, nodes_cloth[i][j].pos.z *
scene_scale, u1, v1);

    // Vertex 2
    vertex(nodes_cloth[i][j + 1].pos.x * scene_scale, nodes_cloth[i][j
+ 1].pos.y * scene_scale, nodes_cloth[i][j + 1].pos.z * scene_scale,
u1, v2);

    // Vertex 3
    vertex(nodes_cloth[i + 1][j].pos.x * scene_scale, nodes_cloth[i +
1][j].pos.y * scene_scale, nodes_cloth[i + 1][j].pos.z * scene_scale,
u2, v1);

    // Vertex 4
    vertex(nodes_cloth[i + 1][j + 1].pos.x * scene_scale,
nodes_cloth[i + 1][j + 1].pos.y * scene_scale, nodes_cloth[i + 1][j +
1].pos.z * scene_scale, u2, v2);
     endShape(CLOSE);

    }

  }


}
```