333 lines (309 sloc) | 7.1 KB

```
/*
 Johnny Li
 CAP3027
 Project 3: 3D Rendering
 */
//Using the ControlP5 library.
import controlP5.*;
ControlP5 cp5;

//Global Variable
PVector vec = new PVector(0, 0, 0);
PVector cameraPosition = new PVector(0, 0, 0);
boolean wheeled = false;
boolean pressed = false;
int temp = 0;
int scaled = 0;
//Camera object
cam object=null;
//Testing
int xx=0;

public void setup() {
  cp5 = new ControlP5(this);

  //Given size for 3D rendering
  size(1600, 1000, P3D);
  //Projection Matrix
  perspective(-radians(50.0f), width/(float)height, 0.1, 1000);
  translate(width * 0.5, height * 0.5, 0);
```

```
  //Create Object Camera
  object=new cam();
}
public void draw() {
  //Reset
  colorMode(RGB, 255);
  background(155);
  //Build Grid
  grid();
  //Build Monster
  mon();
  //Build Cubes
  cubes();
  //Build Fans
  fans();
  //Update camera
  object.Update();
}

//Build grid for the objects
//Origin (0, 0, 0)
//min and max of -100 and 100 on X and Z axes, lines every 10 units.
public void grid() {
  pushMatrix();

  //White line every 10 units
  for (int i=0; i<21; i++) {
    stroke(255);
    line(-100, 0, -100+10*i, 100, 0, -100+10*i);
    line(-100+10*i, 0, -100, -100+10*i, 0, 100);
  }

  //Red line x
  stroke(255, 0, 0);
  line(-100, 0, 0, 100, 0, 0);
  //Blue line z
  stroke(0, 0, 255);
  line(0, 0, -100, 0, 0, 100);
  popMatrix();
}

//Check if spacebar is pressed
public void keyPressed() {
  if (key == 32) {
    pressed=true;
    //Testing
    print("space pressed");
  }
  //Test key
```

```
  if (key == 81) {
    xx++;
    print(xx+",");
  }
}

//Check if mouse wheel is moved
public void mouseWheel(MouseEvent event) {
  wheeled = true;
  scaled = event.getCount();
}

//Camera class
public class cam {
  //Global Variable
  float x;
  float y;
  float z;
  float theta;
  float phi;
  float scaleFactor = 0;
  int radius = 200;
  float zooming = 1;

  //Constructor
  public void cam() {
  }

  public void Update() {
    //Update Target and Zoom
    CycleTarget();
    //Map mouse position
    theta = radians(map(mouseY, 0, width-1, 0, 360));
    phi = radians(map(mouseX, 0, height-1, 1, 179));
    //Camera Positions
    cameraPosition.x = vec.x + radius*cos(phi)*sin(theta);
    cameraPosition.y = vec.y + radius*cos(theta);
    cameraPosition.z = vec.z + radius*sin(theta)*sin(phi);

    camera(cameraPosition.x*abs(Zoom(scaled)), cameraPosition.y*abs(Zoom(scaled)), cameraPosition.
    vec.x, vec.y, vec.z, // Where is the camera looking?
    0, 1, 0); // Camera Up vector (0, 1, 0 often, but not always, works)
  }

  //Load the position of the images
  public void AddLookAtTarget(PVector vec) {
    //Reset vector
    vec.mult(0);
    //Reset counter to beginning
```

```
    if (temp == 5) {
      temp=1;
    }
    //Look at Taget
    switch(temp) {
    case 1:
      vec.add(-100, 0, 0);
      break;
    case 2:
      vec.add(-50, 0, 0);
      break;
    case 3:
      vec.add(0, 0, 0);
      break;
    case 4:
      vec.add(75, 0, 0);
      break;
    }
    //Testing
    print(vec);
  };

  //Move to next traget
  public void CycleTarget() {
    if (pressed) {
      AddLookAtTarget(vec);
      //Item counter
      temp++;
      //Reset pressing to remove bouncing.
      pressed=false;
    }
  };

  //Zoom to target
  public float Zoom(float scaled) {
    if (wheeled) {
      //Scale image
      zooming += scaled*0.2;
      //Reset pressing to remove bouncing.
      wheeled=false;
      //Testing
      print("wheel moved");
      return zooming;
    } else{
      return zooming;
    }
  };
}

```

```
177 //Monster object
178 public void mon(){
179   //Monster generator
180   PShape monster=loadShape("monster.obj");
181   //Full Size
182   pushMatrix();
183   monster.setFill(true);
184   monster.setFill(color(0,0,0,0));  //Wireframe
185   monster.setStroke(true);
186   monster.setStroke(color(0));
187   monster.setStrokeWeight(1.5f);
188   monster.scale(1,1,-1);  //Flip across x axis
189   monster.translate(75, 0, 0);    //Positioning
190   shape(monster);
191   popMatrix();
192   //Half Size
193   pushMatrix();
194   monster.setStroke(true);
195   monster.setStroke(color(190,230,60));  //Yellow color
196   monster.setFill(true);
197   monster.setFill(color(190,230,60));
198   monster.translate(-75, 0, 0);    //Positioning
199   monster.scale(0.5);
200   shape(monster);
201   popMatrix();
202 }
203
204 //Cubes generator
205 public void cubes(){
206   //Cube position
207   translate(-100,0,0);
208
209   //Small cube
210   pushMatrix();
211   translate(-10,0,0);
212   cubeshape();
213   popMatrix();
214
215   //Medium cube
216   pushMatrix();
217   scale(5,5,5);
218   cubeshape();
219   popMatrix();
220   //Large cube
221   pushMatrix();
222   translate(10,0,0);
223   scale(10,20,10);
224   cubeshape();
225   popMatrix();
```

```
226     //Return to origin
227     translate(100,0,0);
228   }
229
230   //Build cube shape
231   public void cubeshape(){
232     //Cube shape
233     beginShape(TRIANGLE);
234     noStroke();
235     //FRONT
236     fill(255,255,0);  //Yellow
237     vertex(-0.5,-0.5,-0.5);
238     vertex(-0.5,0.5,-0.5);
239     vertex(0.5,-0.5,-0.5);
240     fill(0,255,0);  //Green
241     vertex(-0.5,0.5,-0.5);
242     vertex(0.5,-0.5,-0.5);
243     vertex(0.5,0.5,-0.5);
244     //BACK
245     fill(160,60,179);  //Purple
246     vertex(-0.5,-0.5,0.5);
247     vertex(-0.5,0.5,0.5);
248     vertex(0.5,0.5,0.5);
249     fill(0,128,200);  //Blue
250     vertex(-0.5,-0.5,0.5);
251     vertex(0.5,0.5,0.5);
252     vertex(0.5,-0.5,0.5);
253     //TOP
254     fill(160,209,182);  //Teal
255     vertex(0.5,0.5,0.5);
256     vertex(-0.5,0.5,0.5);
257     vertex(0.5,0.5,-0.5);
258     fill(247,107,0);  //Orange
259     vertex(-0.5,0.5,0.5);
260     vertex(0.5,0.5,-0.5);
261     vertex(-0.5,0.5,-0.5);
262     //Left
263     fill(254,60,39);  //Red
264     vertex(0.5,0.5,0.5);
265     vertex(0.5,-0.5,0.5);
266     vertex(0.5,0.5,-0.5);
267     fill(75,128,199);  //Light Blue
268     vertex(0.5,-0.5,0.5);
269     vertex(0.5,0.5,-0.5);
270     vertex(0.5,-0.5,-0.5);
271     //Bottom
272     fill(254,0,0);  //Red
273     vertex(-0.5,-0.5,0.5);
274     vertex(0.5,-0.5,0.5);
```

```
    vertex(0.5,-0.5,-0.5);
    fill(250,2,251);  //Pink
    vertex(-0.5,-0.5,0.5);
    vertex(-0.5,-0.5,-0.5);
    vertex(0.5,-0.5,-0.5);
    //Right
    fill(102,102,222);  //Blue
    vertex(-0.5,0.5,0.5);
    vertex(-0.5,-0.5,0.5);
    vertex(-0.5,-0.5,-0.5);
    fill(180,180,150);  //Grey
    vertex(-0.5,0.5,0.5);
    vertex(-0.5,-0.5,-0.5);
    vertex(-0.5,0.5,-0.5);
    endShape();
  }

  //Fans generator
  public void fans(){
    //Cube position
    translate(-50,0,0);
    scale(-1,1,1);
    //Lots of triangle
    pushMatrix();
    translate(10,0,0);
    fanshape(20);
    popMatrix();

    //Few of triangle
    pushMatrix();
    translate(-10,0,0);
    fanshape(6);
    popMatrix();
  }

  //Build fan shape
  public void fanshape(int segment){
    int col = 0;
    pushMatrix();
    //HSB color here for color
    colorMode(HSB, 360, 100, 100);
    beginShape(TRIANGLE_FAN);
    stroke(2);
    int i=10;
    //Connectine Point
    fill(col, 100, 100);  //Coloring Segment
    vertex(0,i,0);
    //Segment Vertex
    for (float angle=0; angle<=360; angle += 360/segment) {
```

```
324       float vx = i + cos(radians(angle))*i;
325       float vy = i + sin(radians(angle))*i;
326 
327       col += 360/segment;
328       fill(col, 100, 100);  //Coloring Segment
329       vertex(vx, vy);
330     }
331     popMatrix();
332     endShape();
333   }
```