```
    1: #include "sierpinski.hpp"
    2: #include <cmath>
    3: #include <iostream>
    4:
    5: std::vector <sf::ConvexShape> triangle_vector;
    6: int count = 0;
    7:
    8: Sierpinski::Sierpinski(int size, int depth)
    9: {
   10:     float sierpinski_height = size * sqrt(3)/2;
   11: //    sierpinski_depth = depth;
   12:  //   sierpinski_size = size;
   13:
   14:
   15:     sf::Vector2f p1, p2, p3;
   16:     p1.x = size/2;
   17:     p1.y = 0;
   18:     p2.x = 0;
   19:     p2.y = sierpinski_height;
   20:     p3.x = size;
   21:     p3.y = sierpinski_height;
   22:
   23:     //Set Initial Triangle
   24:     sf::ConvexShape initial_triangle;
   25:     initial_triangle.setPointCount(3);
   26:     initial_triangle.setPoint(0, p1);
   27:     initial_triangle.setPoint(1, p2);
   28:     initial_triangle.setPoint(2, p3);
   29:     initial_triangle.setFillColor(sf::Color::Yellow);
   30:
   31:     triangle_vector.push_back(initial_triangle);
   32:
   33:     if (depth > 0)
   34:     {
   35:         triangle_vector.pop_back();
   36:         Sierpinski(p1.x, p1.y, p2.x, p2.y, p3.x, p3.y, depth, size, sierpins
ki_height);
   37:     }
   38:
   39:
   40:
   41: }
   42:
   43:
   44: Sierpinski::Sierpinski(float x1, float y1, float x2, float y2, float x3, flo
at y3, int depth,
   45:                        float size, float height)
   46: {
   47:     /*if (depth not reached)
   48:     {
   49:         child (top, midleft (w/4) (h/2), midright (3w/4)(h/2))
   50:         child (midleft, left, middle (top w/2) (h))
   51:         child (midright, middle, right)
   52:     }
   53:      else
   54:      {
   55:         build triangle with current data
   56:      }
   57:      */
   58:
   59:
```

```
   60:
   61:      if (depth > 0)
   62:      {
   63:         depth--;
   64:
   65:         Sierpinski(x1, y1, size/4, height/2, ((3 * size)/4), height/2, depth
, size/4, height/2);
   66:         Sierpinski(size/4, height/2, x2, y2, x1, height, depth, size/4, heig
ht/2);
   67:         Sierpinski(((3 * size)/4), height/2, x1, height, x3, y3, depth, size
/4, height/2);
   68:
   69:      }
   70:      else
   71:      {
   72:          sf::Vector2f p1, p2, p3;
   73:          p1.x = x1;
   74:          p1.y = y1;
   75:          p2.x = x2;
   76:          p2.y = y2;
   77:          p3.x = x3;
   78:          p3.y = y3;
   79:
   80:         sf::ConvexShape triangle;
   81:         triangle.setPointCount(3);
   82:         triangle.setPoint(0, p1);
   83:         triangle.setPoint(1, p2);
   84:         triangle.setPoint(2, p3);
   85:         triangle.setFillColor(sf::Color::Yellow);
   86:          triangle.setOutlineColor(sf::Color::Red);
   87:          triangle.setOutlineThickness(2);
   88:
   89:         triangle_vector.push_back(triangle);
   90:          count++;
   91:      }
   92: }
   93:
   94:
   95: Sierpinski::~Sierpinski()
   96: {
   97:
   98: }
   99:
  100:
  101: void Sierpinski::draw(sf::RenderTarget &target, sf::RenderStates states) con
st
  102: {
  103:     for(unsigned int i = 0; i < triangle_vector.size(); i++)
  104:     {
  105:         target.draw(triangle_vector.at(i), states);
  106:     }
  107: }
  108:
  109:
  110:
  111:
```