

```
1  #include <iostream>
2  #include <unistd.h>
3  #include "image.h"
4  #include "gcontext.h"
5
6  int main()
7  {
8      GraphicsContext *gc = new X11Context(800, 600, GraphicsContext::BLACK);
9      // Create new image
10     Image image;
11     // Add line to image
12     image.addLine(10, 10, 30, 30, GraphicsContext::BLUE);
13     image.draw(gc);
14     // Add triangle to image
15     image.addTriangle(300, 300, 200, 200, 250, 100, GraphicsContext::GREEN);
16     image.draw(gc);
17     sleep(2);
18     // Create copy of image
19     Image imageCopy(image);
20     // Add line to the copy
21     imageCopy.addLine(30, 30, 100, 110, GraphicsContext::RED);
22     imageCopy.draw(gc);
23     sleep(2);
24     // Make a copy of the copy
25     Image imageCopy2 = imageCopy;
26     // Add triangle to new copy
27     imageCopy2.addTriangle(100, 100, 150, 250, 200, 300, GraphicsContext::YELLOW);
28     imageCopy2.draw(gc);
29     sleep(2);
30     // Add line to new copy
31     imageCopy2.addLine(400, 300, 400, 400, GraphicsContext::WHITE);
32     // Erase original image and image copy
33     image.erase();
34     imageCopy.erase();
35     // Clear all the drawings
36     gc->clear();
37     // Redraw the images in the newest copy
38     imageCopy2.draw(gc);
39     sleep(2);
40     // Erase the copy
41     imageCopy2.erase();
42     // Clear the drawings
43     gc->clear();
44     sleep(1);
45
46     delete gc;
47     return 0;
48 }
```

```
1  #ifndef shape_h
2  #define shape_h
3
4  #include <iostream>
5  #include "xllcontext.h"
6  #include "gcontext.h"
7  using namespace std;
8
9  class Shape
10 {
11 public:
12     virtual ~Shape(){};
13     virtual void draw(GraphicsContext *) = 0;
14     virtual Shape *clone() = 0;
15
16 protected:
17     unsigned int color;
18 };
19
20 #endif
```

```

1  #include <iostream>
2  #include <vector>
3  #include "triangle.h"
4  #include "line.h"
5  #include "shape.h"
6  #include "xllcontext.h"
7  #include "drawbase.h"
8  #include "gcontext.h"
9  #include "matrix.h"
10 #include "image.h"
11 using namespace std;
12
13 // Constructor
14 Image::Image()
15 {
16 }
17
18 // Copy Constructor
19 Image::Image(const Image &from)
20 {
21     for (int i = 0; i < from.shapes.size(); i++)
22     {
23         shapes.push_back(from.shapes[i]->clone());
24     }
25 }
26
27 // Destructor
28 Image::~Image()
29 {
30     erase();
31 }
32
33 void Image::operator=(const Image &rhs)
34 {
35     erase();
36     for (int i = 0; i < rhs.shapes.size(); i++)
37     {
38         shapes.push_back(rhs.shapes[i]->clone());
39     }
40 }
41
42 // Add a line to the shapes container
43 void Image::addLine(int x0, int y0, int x1, int y1, unsigned int color)
44 {
45     shapes.push_back(new Line(x0, y0, x1, y1, color));
46 }
47
48 // Add a triangle to the shapes container
49 void Image::addTriangle(int x0, int y0, int x1, int y1, int x2, int y2, unsigned int color)
50 {
51     shapes.push_back(new Triangle(x0, y0, x1, y1, x2, y2, color));
52 }
53
54 // Draw all lines/triangles in the shapes container
55 void Image::draw(GraphicsContext *gc)
56 {
57     for (int i = 0; i < shapes.size(); i++)
58     {
59         shapes[i]->draw(gc);
60     }
61 }
62
63 // Erase all shapes and return all dynamic memory
64 void Image::erase()
65 {
66     for (int i = 0; i < shapes.size(); i++)
67     {
68         delete shapes[i];
69     }
70     shapes.clear();
71 }

```

```
1  #ifndef image_h
2  #define image_h
3
4  #include <iostream>
5  #include <vector>
6  #include "shape.h"
7  #include "matrix.h"
8  #include "line.h"
9  #include "triangle.h"
10 using namespace std;
11
12 class Image
13 {
14 public:
15     Image( );
16     Image(const Image &from);
17     ~Image();
18     void operator=(const Image &rhs);
19     void addLine(int x0, int y0, int x1, int y1, unsigned int color);
20     void addTriangle(int x0, int y0, int x1, int y1, int x2, int y2, unsigned int color);
21     void draw(GraphicsContext *gc);
22     void erase();
23
24 private:
25     vector<Shape *> shapes;
26     GraphicsContext *gc;
27 };
28
29
30 #endif
```

```
1  #include <iostream>
2  #include "line.h"
3  #include "shape.h"
4  #include "xllcontext.h"
5  #include "drawbase.h"
6  #include "gcontext.h"
7  #include "matrix.h"
8  using namespace std;
9
10 // Line constructor
11 Line::Line(int x0, int y0, int x1, int y1, unsigned int color)
12 {
13     this->coord0[0][0] = x0;
14     this->coord0[1][0] = y0;
15     this->coord0[2][0] = 0;
16     this->coord0[3][0] = 1;
17
18     this->coord1[0][0] = x1;
19     this->coord1[1][0] = y1;
20     this->coord1[2][0] = 0;
21     this->coord1[3][0] = 1;
22
23     this->color = color;
24 }
25
26 // Clone a line
27 Shape *Line::clone()
28 {
29     return new Line(*this);
30 }
31
32 // Draw the line
33 void Line::draw(GraphicsContext *gc)
34 {
35     gc->setColor(color);
36     gc->drawLine(coord0[0][0], coord0[1][0], coord1[0][0], coord1[1][0]);
37 }
```

```
1  #ifndef line_h
2  #define line_h
3
4  #include <iostream>
5  #include "shape.h"
6  #include "matrix.h"
7  using namespace std;
8
9  class Line : public Shape
10 {
11 public:
12     Line(int x0, int y0, int x1, int y1, unsigned int color);
13     Shape *clone();
14     void draw(GraphicsContext *gc);
15
16 private:
17     Matrix coord0 = Matrix(4, 1);
18     Matrix coord1 = Matrix(4, 1);
19 };
20
21 #endif
```

```
1  #include <iostream>
2  #include "triangle.h"
3  #include "shape.h"
4  #include "xllcontext.h"
5  #include "drawbase.h"
6  #include "gcontext.h"
7  #include "matrix.h"
8  using namespace std;
9
10 // Triangle constructor
11 Triangle::Triangle(int x0, int y0, int x1, int y1, int x2, int y2, unsigned int color)
12 {
13     this->coord0[0][0] = x0;
14     this->coord0[1][0] = y0;
15     this->coord0[2][0] = 0;
16     this->coord0[3][0] = 1;
17
18     this->coord1[0][0] = x1;
19     this->coord1[1][0] = y1;
20     this->coord1[2][0] = 0;
21     this->coord1[3][0] = 1;
22
23     this->coord2[0][0] = x2;
24     this->coord2[1][0] = y2;
25     this->coord2[2][0] = 0;
26     this->coord2[3][0] = 1;
27
28     this->color = color;
29 }
30
31 // Clone a triangle
32 Shape *Triangle::clone()
33 {
34     return new Triangle(*this);
35 }
36
37 // Draw the triangle
38 void Triangle::draw(GraphicsContext *gc)
39 {
40     gc->setColor(color);
41     gc->drawLine(coord0[0][0], coord0[1][0], coord1[0][0], coord1[1][0]);
42     gc->drawLine(coord0[0][0], coord0[1][0], coord2[0][0], coord2[1][0]);
43     gc->drawLine(coord1[0][0], coord1[1][0], coord2[0][0], coord2[1][0]);
44 }
```

```
1  #ifndef triangle_h
2  #define triangle_h
3
4  #include <iostream>
5  #include "shape.h"
6  #include "matrix.h"
7  using namespace std;
8
9  class Triangle : public Shape
10 {
11 public:
12     Triangle(int x0, int y0, int x1, int y1, int x2, int y2, unsigned int color);
13     Shape *clone();
14     void draw(GraphicsContext *gc);
15
16 private:
17     Matrix coord0 = Matrix(4, 1);
18     Matrix coord1 = Matrix(4, 1);
19     Matrix coord2 = Matrix(4, 1);
20 };
21
22
23 #endif
```


1	Table of Contents							
2	1 main.cpp	sheets	1 to	1 (1)	pages	1-	1	49 lines
3	2 shape.h	sheets	2 to	2 (1)	pages	2-	2	21 lines
4	3 image.cpp	sheets	3 to	3 (1)	pages	3-	3	72 lines
5	4 image.h	sheets	4 to	4 (1)	pages	4-	4	31 lines
6	5 line.cpp	sheets	5 to	5 (1)	pages	5-	5	38 lines
7	6 line.h	sheets	6 to	6 (1)	pages	6-	6	22 lines
8	7 triangle.cpp	sheets	7 to	7 (1)	pages	7-	7	45 lines
9	8 triangle.h	sheets	8 to	8 (1)	pages	8-	8	24 lines