```cpp
#include <iostream>
#include <unistd.h>
#include "image.h"
#include "gcontext.h"

int main()
{
    GraphicsContext *gc = new X11Context(800, 600, GraphicsContext::BLACK);
    // Create new image
    Image image;
    // Add line to image
    image.addLine(10, 10, 30, 30, GraphicsContext::BLUE);
    image.draw(gc);
    // Add triangle to image
    image.addTriangle(300, 300, 200, 200, 250, 100, GraphicsContext::GREEN);
    image.draw(gc);
    sleep(2);
    // Create copy of image
    Image imageCopy(image);
    // Add line to the copy
    imageCopy.addLine(30, 30, 100, 110, GraphicsContext::RED);
    imageCopy.draw(gc);
    sleep(2);
    // Make a copy of the copy
    Image imageCopy2 = imageCopy;
    // Add triangle to new copy
    imageCopy2.addTriangle(100, 100, 150, 250, 200, 300, GraphicsContext::YELLOW);
    imageCopy2.draw(gc);
    sleep(2);
    // Add line to new copy
    imageCopy2.addLine(400, 300, 400, 400, GraphicsContext::WHITE);
    // Erase original image and image copy
    image.erase();
    imageCopy.erase();
    // Clear all the drawings
    gc->clear();
    // Redraw the images in the newest copy
    imageCopy2.draw(gc);
    sleep(2);
    // Erase the copy
    imageCopy2.erase();
    // Clear the drawings
    gc->clear();
    sleep(1);

    delete gc;
    return 0;
}
```

```
1  #ifndef shape_h
2  #define shape_h
3
4  #include <iostream>
5  #include "x11context.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
```

```cpp
#include <iostream>
#include <vector>
#include "triangle.h"
#include "line.h"
#include "shape.h"
#include "x11context.h"
#include "drawbase.h"
#include "gcontext.h"
#include "matrix.h"
#include "image.h"
using namespace std;

// Constructor
Image::Image()
{
}

// Copy Constructor
Image::Image(const Image &from)
{
    for (int i = 0; i < from.shapes.size(); i++)
    {
        shapes.push_back(from.shapes[i]->clone());
    }
}

// Destructor
Image::~Image()
{
    erase();
}

void Image::operator=(const Image &rhs)
{
    erase();
    for (int i = 0; i < rhs.shapes.size(); i++)
    {
        shapes.push_back(rhs.shapes[i]->clone());
    }
}

// Add a line to the shapes container
void Image::addLine(int x0, int y0, int x1, int y1, unsigned int color)
{
    shapes.push_back(new Line(x0, y0, x1, y1, color));
}

// Add a triangle to the shapes container
void Image::addTriangle(int x0, int y0, int x1, int y1, int x2, int y2, unsigned int color
)
{
    shapes.push_back(new Triangle(x0, y0, x1, y1, x2, y2, color));
}

// Draw all lines/triangles in the shapes container
void Image::draw(GraphicsContext *gc)
{
    for (int i = 0; i < shapes.size(); i++)
    {
        shapes[i]->draw(gc);
    }
}

// Erase all shapes and return all dynamic memory
void Image::erase()
{
    for (int i = 0; i < shapes.size(); i++)
    {
        delete shapes[i];
    }
    shapes.clear();
}
```

```
#ifndef image_h
#define image_h

#include <iostream>
#include <vector>
#include "shape.h"
#include "matrix.h"
#include "line.h"
#include "triangle.h"
using namespace std;

class Image
{
public:
    Image( );
    Image(const Image &from);
    ~Image();
    void operator=(const Image &rhs);
    void addLine(int x0, int y0, int x1, int y1, unsigned int color);
    void addTriangle(int x0, int y0, int x1, int y1, int x2, int y2, unsigned int color);
    void draw(GraphicsContext *gc);
    void erase();

private:
    vector<Shape *> shapes;
    GraphicsContext *gc;
};


#endif
```

```cpp
#include <iostream>
#include "line.h"
#include "shape.h"
#include "x11context.h"
#include "drawbase.h"
#include "gcontext.h"
#include "matrix.h"
using namespace std;

// Line constructor
Line::Line(int x0, int y0, int x1, int y1, unsigned int color)
{
    this->coord0[0][0] = x0;
    this->coord0[1][0] = y0;
    this->coord0[2][0] = 0;
    this->coord0[3][0] = 1;

    this->coord1[0][0] = x1;
    this->coord1[1][0] = y1;
    this->coord1[2][0] = 0;
    this->coord1[3][0] = 1;

    this->color = color;
}

// Clone a line
Shape *Line::clone()
{
    return new Line(*this);
}

// Draw the line
void Line::draw(GraphicsContext *gc)
{
    gc->setColor(color);
    gc->drawLine(coord0[0][0], coord0[1][0], coord1[0][0], coord1[1][0]);
}
```

```
#ifndef line_h
#define line_h

#include <iostream>
#include "shape.h"
#include "matrix.h"
using namespace std;

class Line : public Shape
{
public:
    Line(int x0, int y0, int x1, int y1, unsigned int color);
    Shape *clone();
    void draw(GraphicsContext *gc);

private:
    Matrix coord0 = Matrix(4, 1);
    Matrix coord1 = Matrix(4, 1);
};

#endif
```

```cpp
#include <iostream>
#include "triangle.h"
#include "shape.h"
#include "x11context.h"
#include "drawbase.h"
#include "gcontext.h"
#include "matrix.h"
using namespace std;

// Triangle constructor
Triangle::Triangle(int x0, int y0, int x1, int y1, int x2, int y2, unsigned int color)
{
    this->coord0[0][0] = x0;
    this->coord0[1][0] = y0;
    this->coord0[2][0] = 0;
    this->coord0[3][0] = 1;

    this->coord1[0][0] = x1;
    this->coord1[1][0] = y1;
    this->coord1[2][0] = 0;
    this->coord1[3][0] = 1;

    this->coord2[0][0] = x2;
    this->coord2[1][0] = y2;
    this->coord2[2][0] = 0;
    this->coord2[3][0] = 1;

    this->color = color;
}

// Clone a triangle
Shape *Triangle::clone()
{
    return new Triangle(*this);
}

// Draw the triangle
void Triangle::draw(GraphicsContext *gc)
{
    gc->setColor(color);
    gc->drawLine(coord0[0][0], coord0[1][0], coord1[0][0], coord1[1][0]);
    gc->drawLine(coord0[0][0], coord0[1][0], coord2[0][0], coord2[1][0]);
    gc->drawLine(coord1[0][0], coord1[1][0], coord2[0][0], coord2[1][0]);
}
```

```
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
```

Table of Contents