```
#include "mydrawing.h"
#include "gcontext.h"

// Constructor
MyDrawing::MyDrawing()
{
    cout << "COLORS:" << endl;
    cout << "1: White" << endl;
    cout << "2: Black" << endl;
    cout << "3: Red" << endl;
    cout << "4: Yellow" << endl;
    cout << "5: Blue" << endl;
    cout << "6: Green" << endl;
    cout << "Press T to draw a triangle." << endl;
    cout << "Press L to draw a line." << endl;
    cout << "To undo previous shape, press backspace." << endl;
    numClicks = 0; // Track the number of clicks
    mode = 0;      // Default mode is line
    color = GraphicsContext::GREEN; // Default color is green
}
void MyDrawing::paint(GraphicsContext *gc)
{
    im.draw(gc);
}
void MyDrawing::mouseButtonDown(GraphicsContext *gc, unsigned int button, int x, int y)
{
    if (mode == 0) // Line
    {
        if (numClicks == 0) // 1st click
        {
            x0 = x;
            y0 = y;
            numClicks++;
        }
        else // 2nd click. Draw line
        {
            gc->drawLine(x0, y0, x, y);
            im.addLine(x0, y0, x, y, color);
            numClicks = 0;
        }
    }
    else if (mode == 1) // Triangle
    {
        if (numClicks == 0) // 1st click
        {
            x0 = x;
            y0 = y;
            numClicks++;
        }
        else if (numClicks == 1) // 2nd click
        {
            x1 = x;
            y1 = y;
            numClicks++;
        }
        else // 3rd click. Draw triangle
        {
            gc->drawLine(x0, y0, x1, y1);
            gc->drawLine(x0, y0, x, y);
            gc->drawLine(x1, y1, x, y);
            im.addTriangle(x0, y0, x1, y1, x, y, color);
            numClicks = 0;
        }
    }
}
void MyDrawing::undoShape(GraphicsContext *gc)
{
    gc->clear();
    im = im.undoShape(im);
    paint(gc);
}
void MyDrawing::keyDown(GraphicsContext *gc, unsigned int keycode)
{
    switch (keycode)
    {
    case 0x31:
        gc->setColor(GraphicsContext::WHITE);
        color = GraphicsContext::WHITE;
```

```
79              break;
80          case 0x32:
81              gc->setColor(GraphicsContext::BLACK);
82              color = GraphicsContext::BLACK;
83              break;
84          case 0x33:
85              gc->setColor(GraphicsContext::RED);
86              color = GraphicsContext::RED;
87              break;
88          case 0x34:
89              gc->setColor(GraphicsContext::YELLOW);
90              color = GraphicsContext::YELLOW;
91              break;
92          case 0x35:
93              gc->setColor(GraphicsContext::BLUE);
94              color = GraphicsContext::BLUE;
95              break;
96          case 0x36:
97              gc->setColor(GraphicsContext::GREEN);
98              color = GraphicsContext::GREEN;
99              break;
100         case 0x6C: // L key
101             mode = 0; // Line mode
102             break;
103         case 0x74: // T key
104             mode = 1; // Triangle mode
105             break;
106         case 0xFF08: // Backspace key
107             undoShape(gc);
108             break;
109         }
110 }
```

```
#ifndef MYDRAWING_H
#define MYDRAWING_H
#include "drawbase.h"
#include "image.h"

// forward reference
class GraphicsContext;
class MyDrawing : public DrawingBase
{
public:
    MyDrawing();
    // we will override just these
    virtual void paint(GraphicsContext *gc);
    virtual void mouseButtonDown(GraphicsContext *gc, unsigned int button, int x, int y);
    virtual void keyDown(GraphicsContext *gc, unsigned int keycode);

private:
    Image im;
    Image copyIm;
    // We will only support one "remembered" line
    int x0;
    int y0;
    int x1;
    int y1;
    int numClicks;
    int mode;       // 0 == line, 1 == triangle
    unsigned int color;
    void undoShape(GraphicsContext *gc);
};
#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();
}

Image Image::undoShape(Image im)
{
    im.shapes.pop_back();
    return im;
}
```

```
#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();
    Image undoShape(Image im);

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


#endif
```

```
1  #ifndef DRAWBASE_H
2  #define DRAWBASE_H
3
4  // forward reference
5  class GraphicsContext;
6
7  class DrawingBase
8  {
9  public:
10     // prevent warnings
11     virtual ~DrawingBase() {}
12     virtual void paint(GraphicsContext *gc) {}
13     virtual void keyDown(GraphicsContext *gc, unsigned int keycode) {}
14     virtual void keyUp(GraphicsContext *gc, unsigned int keycode) {}
15     virtual void mouseButtonDown(GraphicsContext *gc,
16                                  unsigned int button, int x, int y) {}
17     virtual void mouseButtonUp(GraphicsContext *gc,
18                                unsigned int button, int x, int y) {}
19     virtual void mouseMove(GraphicsContext *gc, int x, int y) {}
20 };
21 #endif
```

```cpp
#include "x11context.h"
#include <unistd.h>
#include <iostream>
#include "mydrawing.h"
int main(void)
{
    GraphicsContext *gc = new X11Context(800, 600, GraphicsContext::BLACK);
    gc->setColor(GraphicsContext::GREEN);
    // make a drawing
    MyDrawing md;
    // start event loop - this function will return when X is clicked
    // on window
    gc->runLoop(&md);
    delete gc;
    return 0;
}
```

Table of Contents