

```

1  #include "mydrawing.h"
2  #include "gcontext.h"
3
4  // Constructor
5  MyDrawing::MyDrawing()
6  {
7      cout << "COLORS:" << endl;
8      cout << "1: White" << endl;
9      cout << "2: Black" << endl;
10     cout << "3: Red" << endl;
11     cout << "4: Yellow" << endl;
12     cout << "5: Blue" << endl;
13     cout << "6: Green" << endl;
14     cout << "Press T to draw a triangle." << endl;
15     cout << "Press L to draw a line." << endl;
16     cout << "To undo previous shape, press backspace." << endl;
17     numClicks = 0; // Track the number of clicks
18     mode = 0; // Default mode is line
19     color = GraphicsContext::GREEN; // Default color is green
20 }
21 void MyDrawing::paint(GraphicsContext *gc)
22 {
23     im.draw(gc);
24 }
25 void MyDrawing::mouseButtonDown(GraphicsContext *gc, unsigned int button, int x, int y)
26 {
27     if (mode == 0) // Line
28     {
29         if (numClicks == 0) // 1st click
30         {
31             x0 = x;
32             y0 = y;
33             numClicks++;
34         }
35         else // 2nd click. Draw line
36         {
37             gc->drawLine(x0, y0, x, y);
38             im.addLine(x0, y0, x, y, color);
39             numClicks = 0;
40         }
41     }
42     else if (mode == 1) // Triangle
43     {
44         if (numClicks == 0) // 1st click
45         {
46             x0 = x;
47             y0 = y;
48             numClicks++;
49         }
50         else if (numClicks == 1) // 2nd click
51         {
52             x1 = x;
53             y1 = y;
54             numClicks++;
55         }
56         else // 3rd click. Draw triangle
57         {
58             gc->drawLine(x0, y0, x1, y1);
59             gc->drawLine(x0, y0, x, y);
60             gc->drawLine(x1, y1, x, y);
61             im.addTriangle(x0, y0, x1, y1, x, y, color);
62             numClicks = 0;
63         }
64     }
65 }
66 void MyDrawing::undoShape(GraphicsContext *gc)
67 {
68     gc->clear();
69     im = im.undoShape(im);
70     paint(gc);
71 }
72 void MyDrawing::keyDown(GraphicsContext *gc, unsigned int keycode)
73 {
74     switch (keycode)
75     {
76     case 0x31:
77         gc->setColor(GraphicsContext::WHITE);
78         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 }
```

```
1  #ifndef MYDRAWING_H
2  #define MYDRAWING_H
3  #include "drawbase.h"
4  #include "image.h"
5
6  // forward reference
7  class GraphicsContext;
8  class MyDrawing : public DrawingBase
9  {
10 public:
11     MyDrawing();
12     // we will override just these
13     virtual void paint(GraphicsContext *gc);
14     virtual void mouseButtonDown(GraphicsContext *gc, unsigned int button, int x, int y);
15     virtual void keyDown(GraphicsContext *gc, unsigned int keycode);
16
17 private:
18     Image im;
19     Image copyIm;
20     // We will only support one "remembered" line
21     int x0;
22     int y0;
23     int x1;
24     int y1;
25     int numClicks;
26     int mode; // 0 == line, 1 == triangle
27     unsigned int color;
28     void undoShape(GraphicsContext *gc);
29 };
30 #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 }
72
73 Image Image::undoShape(Image im)
74 {
75     im.shapes.pop_back();
76     return im;
77 }

```

```
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     Image undoShape(Image im);
24
25 private:
26     vector<Shape *> shapes;
27     GraphicsContext *gc;
28 };
29
30
31 #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
```

```
1  #include "x11context.h"
2  #include <unistd.h>
3  #include <iostream>
4  #include "mydrawing.h"
5  int main(void)
6  {
7      GraphicsContext *gc = new X11Context(800, 600, GraphicsContext::BLACK);
8      gc->setColor(GraphicsContext::GREEN);
9      // make a drawing
10     MyDrawing md;
11     // start event loop - this function will return when X is clicked
12     // on window
13     gc->runLoop(&md);
14     delete gc;
15     return 0;
16 }
```

1	<b>Table of Contents</b>							
2	1 mydrawing.cpp.....	sheets	1 to	2 ( 2)	pages	1-	2	111 lines
3	2 mydrawing.h.....	sheets	3 to	3 ( 1)	pages	3-	3	31 lines
4	3 image.cpp.....	sheets	4 to	4 ( 1)	pages	4-	4	78 lines
5	4 image.h.....	sheets	5 to	5 ( 1)	pages	5-	5	32 lines
6	5 drawbase.h.....	sheets	6 to	6 ( 1)	pages	6-	6	22 lines
7	6 main.cpp.....	sheets	7 to	7 ( 1)	pages	7-	7	17 lines