C++ Program Design/3e
Chapter 10 – The EzWindows API: a detailed examination
Answers to Self-Check Exercises

1. answer: As illustrated with mouse and timer events in this chapter, the basic idea is for the application to tell EzWindows what function to call when, for example, a mouse or timer event occurs. This procedure is called "registering a callback."

2. answer:

```
#include <iostream>
#include <assert.h>
#include "bitmap.h"
using namespace std;

SimpleWindow W1("Window One", 15.0, 9.0,
   Position(1.0, 1.0));

int ReportMousePosition(const Position &p) {
   cout << p.GetXDistance() << endl;
   cout << p.GetYDistance() << endl;
   return 1;
}

int ApiMain() {
   W1.Open();
   assert(W1.GetStatus() == WindowOpen);
   W1.SetMouseClickCallback(ReportMousePosition);
   return 0;
}

int ApiEnd() {
   W1.Close();
   return 0;
}
```

3. answer:

```
#include <iostream>
#include <fstream>
#include <assert.h>
#include "bitmap.h"
#include <string>
using namespace std;

SimpleWindow W1("Window One", 15.0, 9.0,
   Position(1.0, 1.0));

int ApiMain() {
   cout << "Please enter name of bitmap file: " << flush;
   string FileName;
   cin >> FileName;
   ifstream fin(FileName.c_str());
   if (! fin) {
      cerr << "Cannot open " << FileName
           << " for averaging." << endl;
      exit(1);
   }
   W1.Open();
   Position WindowCenter =W1.GetCenter();
   BitMap Photo(W1);
   Photo.Load(FileName);
   Position PhotoPosition = WindowCenter +
      Position(-.5*Photo.GetWidth(),
         -.5*Photo.GetHeight());
   Photo.SetPosition(PhotoPosition);
   Photo.Draw();
   assert(W1.GetStatus() == WindowOpen);
   return 0;
}

int ApiEnd() {
   W1.Close();
      return 0;
}
```

4. answer:

```
#include <assert.h>
#include "ezwin.h"
#include "bitmap.h"

SimpleWindow HelloWindow("Hello EzWindows", 10.0, 8.0,
   Position(5.0, 6.0));

BitMap Picture(HelloWindow);

int ReportMousePosition(const Position &p) {
   if (Picture.IsInside(p))
      HelloWindow.Message("The bitmap was selected.");
   else
      HelloWindow.Message("The bitmap was not selected.");
   return 1;
}

int ApiMain() {
   HelloWindow.Open();
   assert(HelloWindow.GetStatus() == WindowOpen);
   Position Center = HelloWindow.GetCenter();
   Position UpperLeft = Center + Position(-1.0, -1.0);
   Position LowerRight = Center + Position(1.0,  1.0);
   Picture.Load("c11.bmp");
   Position PicturePosition = Center +
      Position(-0.5 * Picture.GetWidth(),
         -0.5 * Picture.GetHeight());
   Picture.SetPosition(PicturePosition);
   Picture.Draw();
   HelloWindow.SetMouseClickCallback(ReportMousePosition);
   return 0;
}

// ApiEnd(): shutdown the window
int ApiEnd() {
   HelloWindow.Close();
   return 0;
}
```

5. answer: Here is one way:

```
#include <iostream>
#include "rect.h"

using namespace std;

SimpleWindow ColorWindow("Color Palette", 14.0, 6.0);

int ApiMain() {
   const float width = 3.0;
   const float height = 1.0;
   float XPosition = 2.0;
   float YPosition = 1.0;

   ColorWindow.Open();

   RectangleShape ColorPatch(ColorWindow,
      XPosition, YPosition, Blue, width, height);

   for (int i=1; i<=4; ++i) {
      ColorPatch.SetPosition(XPosition, YPosition);
      ColorPatch.Draw();
      XPosition += ColorPatch.GetWidth();
      YPosition += ColorPatch.GetHeight();
   }

   return 0;
}
```

6. answer:

```
#include <assert.h>
#include "bitmap.h"
#include "randint.h"

const float WindowWidth = 15.0;
const float WindowHeight = 9.0;
const float WindowX = 1.0;
const float WindowY = 1.0;

// Instantiate a window
SimpleWindow HolesWindow("I'm Makin' Holes!", WindowWidth,
   WindowHeight, Position(WindowX, WindowY));
// Instantiate a bitmap for display in window HolesWindow
BitMap HoleBmp(HolesWindow);
int HolesWindowTimerEvent() {
   RandomInt RandomX(1, (int) HolesWindow.GetWidth());
   int XCoord = RandomX.Draw();

   RandomInt RandomY(1, (int) HolesWindow.GetHeight());
   int YCoord = RandomY.Draw();

   //HoleBmp.Erase();
   if (XCoord + HoleBmp.GetWidth() > HolesWindow.GetWidth() )
      XCoord = XCoord - HoleBmp.GetWidth();
   if (YCoord + HoleBmp.GetHeight() > HolesWindow.GetHeight())
      YCoord = YCoord - HoleBmp.GetHeight();
   HoleBmp.SetPosition(Position(XCoord, YCoord));
   HoleBmp.Draw();
   return 1;
}

int ApiMain() {
   EzRandomize();
   HolesWindow.Open();
   assert(HolesWindow.GetStatus() == WindowOpen);
   HoleBmp.Load("hole.bmp");
   assert(HoleBmp.GetStatus() == BitMapOkay);
   HoleBmp.SetPosition(Position(1.0, 1.0));
   HoleBmp.Draw();
   HolesWindow.SetTimerCallback(HolesWindowTimerEvent);
   HolesWindow.StartTimer(750);
   return 0;
}

// User is shutting down the program
int ApiEnd() {
   HolesWindow.StopTimer();
   HolesWindow.Close();
   return 0;
}
```

7. answer:

```
#include <iostream>
#include <assert.h>
#include "ray.h"
using namespace std;

SimpleWindow W1("Window One", 15.0, 9.0,
      Position(1.0, 1.0));

int ReportMousePosition(const Position &p) {
   static bool started = true;
   static float startX, startY;
   if (started == true) {
      startX = p.GetXDistance();
      startY = p.GetYDistance();
      cout << "\nClick mouse to select end point of ray\n";
      started = false;
   }
   else {
      float endX = p.GetXDistance();
      float endY = p.GetYDistance();
      RaySegment R(W1,startX, startY, endX, endY);
      R.Draw();
      started = true;
      cout << "\nClick mouse to select start point of ray";
   }
   return 1;
}

int ApiMain() {
   W1.Open();
   assert(W1.GetStatus() == WindowOpen);

   cout << "Click mouse button to select start point of ray";
   W1.SetMouseClickCallback(ReportMousePosition);
   return 0;
}

int ApiEnd() {
   W1.Close();
   return 0;
}
```

8. answer:

```
#include <iostream>
#include <string>
#include <assert.h>
#include "bitmap.h"
#include "randint.h"
using namespace std;

const float WindowWidth = 15.0;
const float WindowHeight = 9.0;
const float WindowX = 1.0;
const float WindowY = 1.0;

int CountDownValue;

SimpleWindow CountDownWindow("I'm Makin' Holes!",
      WindowWidth, WindowHeight,
      Position(WindowX, WindowY));

BitMap DigitBmp(CountDownWindow);

int CountDownWindowTimerEvent() {
   BitMap Number(CountDownWindow);
   Number.Erase();
   static int n = CountDownValue;
   string zzz[10] = {"digit0.bmp", "digit1.bmp",
      "digit2.bmp", "digit3.bmp", "digit4.bmp",
      "digit5.bmp", "digit6.bmp", "digit7.bmp",
      "digit8.bmp", "digit9.bmp"};
   Position WindowCenter = CountDownWindow.GetCenter();
   Number.Load(zzz[n]);
   Position NumberPosition = WindowCenter +
      Position(-.5*Number.GetWidth(),-.5*Number.GetHeight());
   Number.SetPosition(NumberPosition);
   Number.Draw();
   --n;
   if (n==-1)
      CountDownWindow.StopTimer();
   return 1;
}

int ApiMain() {
   cout << "Enter an integer between 1 and 9: ";
   cin >> CountDownValue;
   CountDownWindow.Open();
   assert(CountDownWindow.GetStatus() == WindowOpen);
   CountDownWindow.SetTimerCallback(CountDownWindowTimerEvent);
   CountDownWindow.StartTimer(1000);
   return 0;
}

int ApiEnd() {
   CountDownWindow.Close();
   return 0;
}
```