Lab 8: Graphics

Objective

The objective to this lab is to get experience with:

1. BlackBerry graphics APIs
2. Drawing using graphics objects
3. Develop an application using these objects

Experiment 1 Develop an application using graphics objects

In this experiment, you will learn how to write an application for BlackBerry devices using the BlackBerry Graphics objects. Graphics objects enable applications to perform the drawing methods. In this experiment, you get experience with drawing lines, arcs, text, and image as well as Bitmap and BitmapField. Figures 1 demonstrates how your application should be appeared on the screen.

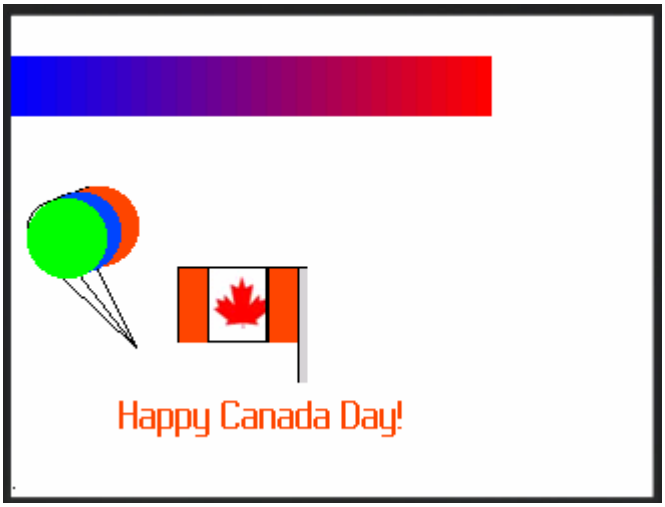


Figure 1- First screen

To start the experiment, do the following steps:

1. Create a class extended from **net.rim.device.api.ui.UiApplication** class. Name it "Lab8".
2. Write the constructor and main() methods and add the required codes to these methods.
3. Import the required classes.

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import java.io.*;
```

4. Create the first screen class and name it "FirstScreen". Figure 1 depicts how it should look like.

```
final class FirstScreen extends MainScreen
 {
        public FirstScreen() {
                super();
         }
}
```

5. FirstScreen consists of the Graphic objects. You create an instance of these objects in the constructor. For this purpose, do the following steps.
6. Get the width and height of the screen size. This is done using these two methods: getScreenWidth() retrieves total drawing width of the screen, and getScreenHeight() retrieves total drawing height of the screen.

```
final class FirstScreen extends MainScreen
{
        public FirstScreen()
        {
                super();
                int width = Graphics.getScreenWidth( );
                int height = Graphics.getScreenHeight( );

        }
}
```

7. In order to draw the shapes demonstrated in Figure 1 over the screen, you should construct an instance of the *Graphics* class. *Graphics* class is defined in the package **net.rim.device.api.ui.** You can use *Graphics* class to draw shapes over the entire screen or on a BitmapField object. BitmapField is a container to display a bitmap. In this experiment, you draw shapes over the BitmapField. In order to construct the *Graphics* object within your class, do the following g steps:
    - Construct a bitmap instance using *Bitmap* object.

      ```
      Bitmap surface = new Bitmap(width, height);
      ```

    - Construct a container for this bitmap object using *BitmapField* object

      ```
      BitmapField surfaceField = new BitmapField(surface);
      ```

    - Add the *BitmapField* instance to the screen

```
add(surfaceField);
```

- Construct a graphic context for the Bitmap instance. This is done by invoking *Graphics* constructor and passing the instance of the *Bitmap* field into this constructor.

```
Graphics g = new Graphics(surface);
```

So far, you should have the following lines in the constructor:

```
public FirstScreen()
{
        super();
        int width = Graphics.getScreenWidth( );
        int height = Graphics.getScreenHeight( );

        Bitmap surface = new Bitmap(width, height);
        BitmapField surfaceField = new BitmapField(surface);
        add(surfaceField);

        Graphics g = new Graphics(surface);
}
```

8. Now that you have constructed and obtained an instance of the *Graphics* object, you can access to its methods to draw shapes. Figure 2 depicts some of these methods.

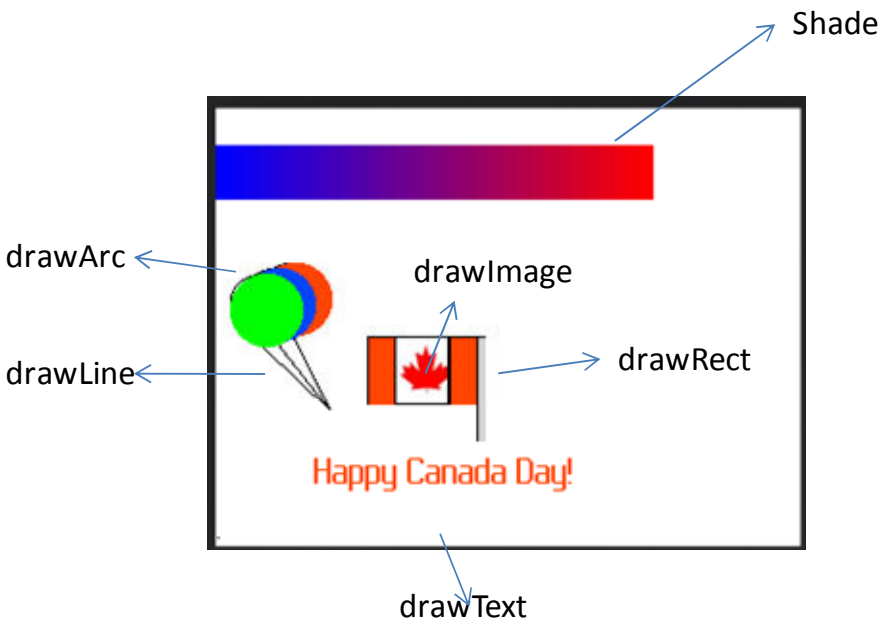


Figure 2- Graphics class's methods used to draw shapes

9. In order to draw shade on the top of the screen, use drawShadedFilledPath() method. This method draws a set of shaded filled path.

```
int[] X_PTS = { 0, 0, 240, 240 };
int[] Y_PTS = { 20, 50, 50, 20 };
int[] drawColors = { 0x0000FF, 0x0000FF, 0xFF0000, 0xFF0000 };

g.drawShadedFilledPath(X_PTS, Y_PTS, null, drawColors, null);
```

   X_PTS is an ordered list of x values for the path.
   Y_PTS is an ordered list of y values for the path.
   "drawColors" is an order list of color values supposed to be appeared in the path.

10. In order to draw the balloons on the left side of the screen, you should use drawArc(int x, int y, int width, int height, int startAngle, int arcAngle) to draw circles and drawLine(int x1, int y1, int x2, int y2) to draw the lines. Then, you should use fillArc(int x, int y, int width,int height, int startAngle, int arcAngle) to fill the balloons. Before using fillArc, set a color. This is done by setColor(int color) method.

```
//Draw circle and fill that with a color
g.drawArc(24,85, 35, 30, 0, 360);
g.setColor(0xFF4500);
g.fillArc(24,85,40,40,0,360);

//draw a black  line
g.setColor(0);
g.drawLine(24+18, 85+40,24+18+20,85+30+50);
```

11. Repeat the step 10 for the other balloons:

```
g.drawArc(15,88, 35, 30, 0, 360);
g.setColor(0x0045FF);
g.fillArc(15,88,40,40,0,360);

g.setColor(0);
g.drawLine(15+18, 88+40,24+18+20,85+30+50);

g.drawArc(8,91, 35, 30, 0, 360);
g.setColor(0x00FF00);
g.fillArc(8,91,40,40,0,360);

g.setColor(0);
g.drawLine(8+18, 91+40,24+18+20,85+30+50);
```

12. The next step is to draw the flag. It consists of several rectangles and one image displaying the maple leaf. To draw rectangles use the drawRect(int x, int y, int width, int height) method. To fill them with a color use setColor(int color) method.

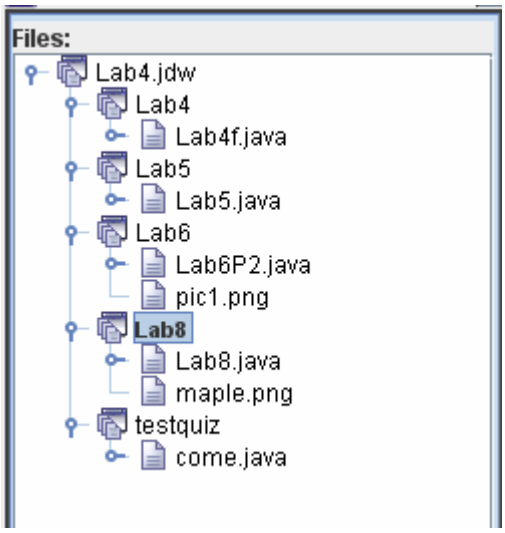
```
 g.setColor(0);
g.drawRect(30+width/6, 125, 15, 38);
g.setColor(0xFF4500);
g.fillRect(30+width/6+1, 125+1, 14,37);

 g.setColor(0);
 g.drawRect(30+width/6+15, 125, 30, 38);

 g.setColor(0);
 g.drawRect(30+width/6+15+30, 125, 15, 38);
 g.setColor(0xFF4500);
 g.fillRect(30+width/6+15+30+1, 125+1, 14,37);

 g.setColor(0);
 g.drawRect(30+width/6+60, 125, 5,58);
 g.setColor(0xD3D3D3);
 g.fillRect(30+width/6+60+1, 125+1, 4,57);
```

13. In order to draw the maple leaf image, use drawImage():
    drawImage (int x, int y, int width, int height, EncodedImage image, int frameIndex, int left, int top)
    This method draws an encoded image. In order to create an encoded image, do the following steps:
    - Save a maple leaf image to the project folder or subfolder. The size of image should be small (30*37 pixels)
    - Add the image file to the project in BlackBerry JDE. To do this, just right-click on the project name, a menu is opened. Click on "Add File to Project…" or simply press Insert keyword. Select the image file and click on Open.



    - Access the image. To do this, invoke Class.getResourceAsStream() method to retrieve the image as an input stream of bytes.

```
InputStream input =null;
try {
      input = Class.forName("Lab8").getResourceAsStream("maple.png");
      // Please note you need to create the image…around 50x50 pixels
} catch (Exception e) {
      System.out.println("Class not found");
}
```

- Encode the image.
  To do this, invoke EncodedImage.createEncodedImage(). This method creates an instance of EncodedImage using the raw image data in the array of byte.

```
byte[] data = new byte[2042];

 try {
       int code = input.read(data);
  } catch (Exception e) {
       System.out.println("Reading Error");
  }
  EncodedImage mapleImg =
          EncodedImage.createEncodedImage(data, 0, data.length);
```

- Now, you can draw the encoded image using the drawImage() method.

```
g.drawImage (30+width/6+17, 127, 27, 29, mapleImg, 0, 0, 0);
```

14. The next step is to draw the text, "Happy Canada Day!". This is done using the drawText() method.

```
g.setColor(0xFF4500);
g.drawText("Happy Canada Day!", width/6, 80+height/3+30 ,
Graphics.TOP | Graphics.HCENTER);
```

15. Demo your work to the TA. [**10 marks**]

Listing 1 – Lab8.java:

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import java.io.*;

class Lab8 extends net.rim.device.api.ui.UiApplication {


  public Lab8()
  {
    FirstScreen screen  = new FirstScreen();
```

```
    pushScreen(screen );
}

public static void main(String[] args) {
    Lab8 theApp = new Lab8();
    theApp.enterEventDispatcher();
}

final class FirstScreen extends MainScreen
{
    private static final int IMAGE_SIZE = 2430;

    public FirstScreen()
    {
        super();
        int width = Graphics.getScreenWidth( );
        int height = Graphics.getScreenHeight( );

        Bitmap surface = new Bitmap(width, height);
        BitmapField surfaceField = new BitmapField(surface);
        add(surfaceField);
        Graphics g = new Graphics(surface);

        int[] X_PTS = { 0, 0, 240, 240 };
        int[] Y_PTS = { 20, 50, 50, 20 };
        int[] drawColors = { 0x0000FF, 0x0000FF, 0xFF0000, 0xFF0000 };
        InputStream input = null;
        byte[] data = new byte[IMAGE_SIZE];

        try {
            input = Class.forName("Lab8").getResourceAsStream("maple.png");
        } catch (Exception e) {
            System.out.println("Class not found");
        }

        try {
            int code = input.read(data);
            EncodedImage mapleImg =
                    EncodedImage.createEncodedImage(data, 0, data.length);

            g.drawShadedFilledPath(X_PTS, Y_PTS, null, drawColors, null);

            g.drawArc(24,85, 35, 30, 0, 360);
            g.setColor(0xFF4500);
            g.fillArc(24,85,40,40,0,360);

            g.setColor(0);
            g.drawLine(24+18, 85+40,24+18+20,85+30+50);

            g.drawArc(15,88, 35, 30, 0, 360);
            g.setColor(0x0045FF);
```

```
                g.fillArc(15,88,40,40,0,360);

                g.setColor(0);
                g.drawLine(15+18, 88+40,24+18+20,85+30+50);

                g.drawArc(8,91, 35, 30, 0, 360);
                g.setColor(0x00FF00);
                g.fillArc(8,91,40,40,0,360);

                g.setColor(0);
                g.drawLine(8+18, 91+40,24+18+20,85+30+50);

                g.setColor(0);
                g.drawRect(30+width/6, 125, 15, 38);
                g.setColor(0xFF4500);
                g.fillRect(30+width/6+1, 125+1, 14,37);

                g.setColor(0);
                g.drawRect(30+width/6+15, 125, 30, 38);

                g.setColor(0);
                g.drawRect(30+width/6+15+30, 125, 15, 38);
                g.setColor(0xFF4500);
                g.fillRect(30+width/6+15+30+1, 125+1, 14,37);

                g.setColor(0);
                g.drawRect(30+width/6+60, 125, 5,58);
                g.setColor(0xD3D3D3);
                g.fillRect(30+width/6+60+1, 125+1, 4,57);

                g.drawImage (30+width/6+17, 127,27,29, mapleImg, 0,0,0);

                g.setColor(0xFF4500);
                g.drawText("Happy Canada Day!", width/6, 80+height/3+30 ,
                                        Graphics.TOP |   Graphics.HCENTER);

            } catch (Exception iae) {
                System.out.println("Error in handling commands");
            }
        }
    }

}
```