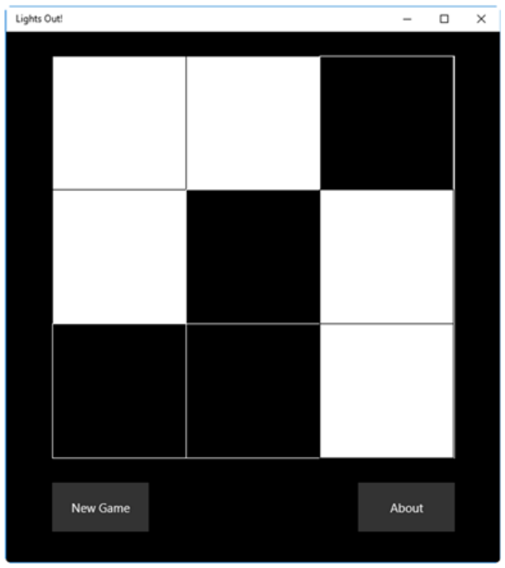
Lights Out Universal App

GUI Programming

10 Points



Convert your WPF Lights Out application into a Windows Universal application. The project should use a Main page for displaying the game board and an About page that identifies the author(s) of the app.

Use the RelativePanel to layout your Main page. Use a Canvas to display the game board that is composed of Rectangle objects. Display a New Game button that starts a new game and an About button that shows the About page.

Use the LightsOutGame class supplied below for your app's model. A LightsOutGame object should be used as a class-level variable to start a new game, make a move, determine which values are on and off, and determine how wide the grid is. Because the LightsOutGame class is the app's model, it does not contain any code for the UI.

Create a custom icon for the Windows toolbar (Square44x44Logo.targetsize-24_altform-unplated.png) and a splash screen logo (SplashScreen.scale-200.png) for your app.

Create an About page that is similar to the About dialog boxes from your WPF applications. It should have an image and display short instructions and the programmer's name. The Back button in the title bar should allow the user to navigate back to the Main page.

You may do pair programming if you'd like; just put both authors' names in the About page.

Zip up your entire project and submit it to Easel before it is due.

Much of the application logic will be identical with WPF except for a few places:

1. To set the title of window's title bar, Set the "Display name" in the Package.appxmanifest file.

2. Use a Canvas with Rectangles to display the game board. You can create the Rectangles dynamically like we did in the WPF app, or you can create them in XAML. To set the rectangles to the proper color programmatically, use a SolidColorBrush. Example:

```
SolidColorBrush black = new SolidColorBrush(Windows.UI.Colors.Black);
```

3. To determine which Rectangle the user clicked, add a Tapped handler for each Rectangle. Use the Rectangle's Tag to determine which rectangle was clicked:

```
 Rectangle rect = sender as Rectangle;
var move = (Point)rect.Tag;
```

```
game.Move((int)move.X, (int)move.Y);
```

4. To display a modal dialog box when the user wins, use the following code:

```
MessageDialog msgDialog = new MessageDialog("Congratulations!  You've won!", "Lights
Out!");

 // Add an OK button
 msgDialog.Commands.Add(new UICommand("OK"));

 // Show the message box and wait aynchrously for a button press
IUICommand command = await msgDialog.ShowAsync();
```

5. To display an image in your About page, add the image to your Assets folder. Use `<Image Source="Assets/myimage.jpg">` to display the image.

6. The LightsOutGame class defaults to using a 3x3 game grid when instantiated. If the GridSize property is changed, the grid array is resized, and a new game is started. You do not need to change the GridSize property since changing the board size is not a requirement.

```
public class LightsOutGame
{
    private int gridSize;
    private bool[,] grid;           // Store the on/off state of the grid
    private Random rand;

    public const int MaxGridSize = 10;
    public const int MinGridSize = 3;

    public int GridSize
    {
        get
        {
            return gridSize;
        }
        set
        {
            if (value >= MinGridSize && value <= MaxGridSize)
            {
                gridSize = value;
                grid = new bool[gridSize, gridSize];
                NewGame();
            }
        }
    }

    public LightsOutGame()
    {
        rand = new Random();
        GridSize = MinGridSize;
    }

    public bool GetGridValue(int row, int col)
    {
        return grid[row, col];
    }

    public void NewGame()
    {
        for (int r = 0; r < gridSize; r++)
        {
            for (int c = 0; c < gridSize; c++)
            {
                grid[r, c] = rand.Next(2) == 1;
            }
        }
    }

    public void Move(int row, int col)
    {
        if (row < 0 || row >= gridSize || col < 0 || col >= gridSize)
        {
            throw new ArgumentException("Row or column is outside the legal range of 0 to "
                + (gridSize - 1));
        }

        // Invert selected box and all surrounding boxes
        for (int i = row - 1; i <= row + 1; i++)
        {
            for (int j = col - 1; j <= col + 1; j++)
            {
                if (i >= 0 && i < gridSize && j >= 0 && j < gridSize)
                {
                    grid[i, j] = !grid[i, j];
                }
            }
        }
    }

    public bool IsGameOver()
    {
        for (int r = 0; r < gridSize; r++)
        {
            for (int c = 0; c < gridSize; c++)
            {
                if (grid[r, c])
                {
                    return false;
                }
            }
        }

        // All values must be false (off)
        return true;
    }
}
```