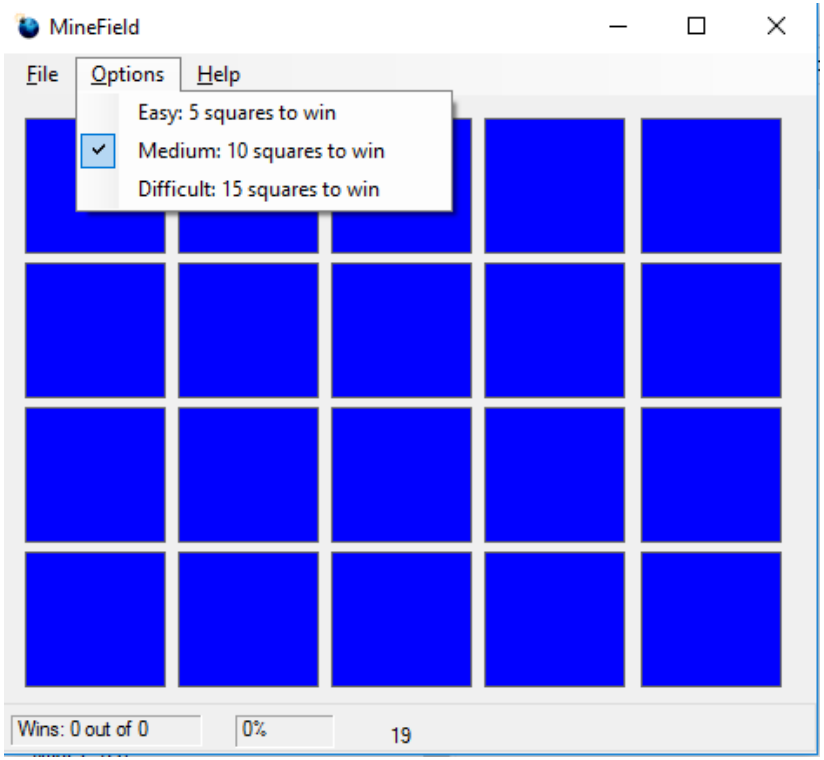
MineField

Contents

Time required: 180 minutes

Comment each line of code as shown.

Create a Form



1. Create a new Visual C# Form project called MineField.
2. Create the following form design.
3. Add a **MenuStrip**.
    a. The **File** menu has an **Exit** command.
    b. The **Options** menu should have CheckMarks, and Medium should be enabled by default. Right Click the menu item to choose **Checked**.
    c. Each menu item connects to the appropriate code which you will be supplied later in this project.
4. The Help menu has an **About** command which displays directions for the game and information about the author. This could be a messagebox or a form.
5. Right Click on the project name, **MineField**, Click **Properties**, go to **Resources**, Click the **Add resource** down triangle, **Add existing file**, browse to m**inefield.ico**. Click OK.
    a. Go to **Application**, **Icon and manifest**, click the dropdown by Icon: Choose **Resources\minefield.ico**. Close the Application Properties.
    b. Add **minefield.ico** to the **Icon** property of the main form.
6. The blue squares are labels. They are placed in a TableLayoutPanel.
    a. Resize your form to your preferred size.
    b. Add a **TableLayoutPanel** to your form.
    c. Click **Edit Rows and Columns**.

d. Add 3 more columns for a total of 5 columns

e. Select all the columns, and set the **Size Type** to **Percent 20.00**.

f. Click the drop-down menu to choose **Rows**.

g. Add 2 more rows for a total of 4 rows.

h. Select all the rows, and set the **Size Type** to **Percent 25.00**. Click OK

i. Resize the **TableLayoutPanel**, taking care to keep the table square.

j. Add a label to the first cell. Resize the label to fit. Change the **BackColor** to **Blue**. Delete the Text property.

k. Copy this label, and paste. The label will automatically go into each of the cells from left to right, and then to the next row.

l. Leave the label names label1, label2, etc.

7. The label with the number 19 on this design is a label used for troubleshooting and testing the program. The label displays the random mine number to test the program. When your program works, you can remove the label and the code that ran the label.

Create Form Level/Global Variables

Create global variables at the form level.

1. The Label lblLabel variable tracks which label we are working with or have clicked. (We did this in Tic Tac Toe 2)
2. An Integer variable to track the mine location amongst the squares.
3. An Integer variable to track how many times we have clicked a square.
4. An Integer variable to track how many squares we click to win. (This should be set to 10 to start the game.)
5. An Integer variable to track wins.
6. An Integer variable to track games played.

7. Integer constants to set the lower and upper bounds of random numbers for mine generation.
8. A Random object to create random numbers.

MineLabel_Click Event

Create the MineLabel_Click Event as shown below.

```
private void MineLabel_Click(object sender, EventArgs e)
{
    intClickCounter += 1;       // Increase the click counter by 1 for each mine we click to track when we win
    lblLabel = (Label)sender;   // Use the label variable to hold the label that was just clicked

    if (intClickCounter == intWIN) // We clicked the WIN number of times, time to end this round
    {
        // We clicked the mine and lost
        if (lblLabel.TabIndex == intMine)
        {
            // Blink the mine on and off
            BlinkMine();
            intGames += 1;
            lblWins.Text = ("Wins: " + intWins + " out of " + intGames);
            lblHitRate.Text = HitRateCalculation.HitRate(intWins, intGames);

            // Ask the player if they want to play again
            DialogResult ResponseDialogResult = MessageBox.Show("You lost. Play again?", "Play again?", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (ResponseDialogResult == DialogResult.Yes)
            {
                lblLabel.Text = "";
                ResetGame();
            }
            else
            {
                this.Close();   // Exit the game
            }
        }
        // We win
        else
        {
            lblLabel.BackColor = Color.White;   // Indicate the label has been clicked
            intGames += 1;  // Increment the number of rounds played
            intWins += 1;   // Increment the number of games won
            lblWins.Text = ("Wins: " + intWins + " out of " + intGames);        // Display game stats
            lblHitRate.Text = HitRateCalculation.HitRate(intWins, intGames);    // Display percentage of wins

            // Ask the player if they want to play again
            DialogResult ResponseDialogResult = MessageBox.Show("You won! Play again?", "Play again?", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (ResponseDialogResult == DialogResult.Yes)
            {
                ResetGame();    // Reset the round
            }
            else
            {
                this.Close();   // Exit the game
            }
        }
    }
```

```
    // Go here until we reach intWIN
    else
    {
        // User clicked the mine and lost
        if (lblLabel.TabIndex == intMine)
        {
            BlinkMine();    // Blink the mine on and off
            intGames += 1;  // Increment the number of rounds played
            lblWins.Text = ("Wins: " + intWins + " out of " + intGames);        // Display game stats
            lblHitRate.Text = HitRateCalculation.HitRate(intWins, intGames);    // Display percentage of wins

            // Ask the player if they want to play again
            DialogResult ResponseDialogResult = MessageBox.Show("You lost. Play again?", "Play again?", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (ResponseDialogResult == DialogResult.Yes)
            {
                lblLabel.Text = ""; // Clear the label
                ResetGame();        // Reset this round
            }
            else
            {
                this.Close();   // End the game
            }

        }
        // User missed the mine and continues trying
        else
        {
            lblLabel.BackColor = Color.White;   // Change the color of the mine to white to indicate that it has been clicked
            lblLabel.Enabled = false;           // Disable the control, it can't be clicked again
        }
    }
}
```

NewGame

```
// Reset everything for a new game
private void NewGame()
{
    // Reset all counters and labels
    intClickCounter = 0;
    intWins = 0;
    intGames = 0;
    lblWins.Text = "Wins: 0 out of 0";
    lblHitRate.Text = "0%";
    BlinkSquares();         // Blink the square to different colors to indicate that the game is being reset

    // Generate a random integer between 1 & 20 and assign it to the intMine variable
    // Basically, create a new random mine
    intMine = randomMine.Next(MIN, MAX);
    label21.Text = intMine.ToString();  // Display the mine number for program testing, comment out when done
}
```

ResetGame()

```
// Reset the mine location and the game pieces
private void ResetGame()
{
    intClickCounter = 0;    // Reset the click counter to 0
    BlinkSquares();         // Blink the square to different colors to indicate that the game is being reset

    // Generate a random integer between 1 & 20 and assign it to the intMine variable
    // Basically, create a new random mine
    intMine = randomMine.Next(MIN, MAX);
    label21.Text = intMine.ToString();  // Display the mine number for program testing, comment out when done
}
```

BlinkMine

```
// Blink the selected mine white and red to indicate losing
private void BlinkMine()
{
    int intSleep = 75;  // Variable for how many milliseconds to pause the program to allow the form to redraw
    // Change color of mine back and forth 6 times
    for (int i = 1; i < 6; i++)
    {
        lblLabel.BackColor = Color.White;           // Change mine to White
        this.Refresh();                             // Redraw the form to display change in mine color
        System.Threading.Thread.Sleep(intSleep);    // Pause program for indicated milliseconds to display new label color
        lblLabel.BackColor = Color.Red;             // Change mine to Red
        this.Refresh();                             // Redraw the form to display change in mine color
        System.Threading.Thread.Sleep(intSleep);    // Pause the program for indicated milliseconds to display new color
    }
    lblLabel.Text = "*Mine*";                       // Change text of mine to indicate hit
}
```

BlinkSquares

```
// Blink all the mines and reset color to White
private void BlinkSquares()
{
    int intSleep = 75;  // Variable for how many milliseconds to pause the program to allow the form to redraw

    // Go through all the controls/labels in the boardTableLayoutPanel
    foreach (Label lbl in boardTableLayoutPanel.Controls.OfType<Label>())
    {
        lbl.Enabled = true;                         // Enable labels
        lbl.BackColor = Color.White;                // Change label color
        this.Refresh();                             // Redraw the form to display change in mine color
        System.Threading.Thread.Sleep(intSleep);    // Pause program for indicated milliseconds to display new label color
        lbl.BackColor = Color.Blue;                 // Change label color back to original color
        this.Refresh();                             // Redraw the form to display change in mine color
        System.Threading.Thread.Sleep(intSleep);    // Pause program for indicated milliseconds to display new label color
    }
}
```

Form Load Event

Generate a random mine number when the form loads.

Set the Difficulty of the Game

Double click the associated menu item to create the following events.

```
// Set the difficulty of the game
private void tsmiEasy5SquaresToWin_Click(object sender, EventArgs e)
{
    intWIN = 5;     // Set the win variable to 5 mines
    // Set the appropriate check for the menu to indicate which is selected
    tsmiEasy5SquaresToWin.Checked = true;
    tsmiMedium10SquaresToWin.Checked = false;
    tsmiDifficult15SquaresToWin.Checked = false;
    NewGame();  // Start a new game
}

// Set the difficulty of the game
private void tsmiMedium10SquaresToWin_Click(object sender, EventArgs e)
{
    intWIN = 10;     // Set the win variable to 10 mines
    // Set the appropriate check for the menu to indicate which is selected
    tsmiEasy5SquaresToWin.Checked = false;
    tsmiMedium10SquaresToWin.Checked = true;
    tsmiDifficult15SquaresToWin.Checked = false;
    NewGame();  // Start a new game
}

// Set the difficulty of the game
private void tsmiDifficult15SquaresToWin_Click(object sender, EventArgs e)
{
    intWIN = 15;     // Set the win variable to 15 mines
    // Set the appropriate check for the menu to indicate which is selected
    tsmiEasy5SquaresToWin.Checked = false;
    tsmiMedium10SquaresToWin.Checked = false;
    tsmiDifficult15SquaresToWin.Checked = true;
    NewGame();  // Start a new game
}
```

Add Events to Labels

1. Select all the labels.

2. Go to Properties, Select the **lightning bolt (Events)**.

3. Click the drop-down list next to Click, and choose **MineLabel_Click**.

Additional Challenges (Extra Credit)

Mention the extra credit piece you added when you submit the assignment.

- Add a menu item that starts a new game and clears the statistics.
- Turn Minefield into a two-player game.
- Add a menu item that increase the number of square and/or mines.
- Add more difficulty levels, you must hit more mines to win.
- Add a sound when the user clicks a square, and an explosion when they click a mine.

Run the Minefield program

1. Press **F5** to start debugging the program. In a moment the program should run. Test it at all three levels to make sure it works.
2. Zip up the project folder and submit to Blackboard.