CMPT 360: Lab Assignment #5
Minesweeper

Brady Coles

last updated: January 3, 2018

Contents

1 Course Goals

This assignment fulfills the following goals:

- a group II language (Delphi)
- implemented on the Windows platform

2 Problem Description

The given problem is to create a playable single player game. The described program is a recreation of Minesweeper for the windows platform, including a fully functional graphical user interface.

3 Program Documentation

3.1 Description

The minesweeper game is made up of a grid of tiles, where some portion of the tiles are mines. The player's goal is to reveal all tiles except the mines, and when a tile is revealed, it displays the number of mines in the eight tiles it touches. Using these numbers, the player can determine where the mines are and are not.

In this program, the user can start a new game anytime, resetting the game board and randomizing the location of the mines. The board is 9x9, with ten mines, the standard 'easy' board in Microsoft minesweeper. Any tiles not yet revealed can be flagged, and flagged tiles can not be revealed without first removing the flag. When a tile touching zero mines is revealed, all tiles it touches is also revealed.

When a mine is revealed, the game ends in a loss; when all tiles but the mines are revealed, the game ends in a win. A message displaying the nature of the end of the game is shown when the game ends. During a game, this message displays the number of mines on the board.

A cell is revealed by left clicking on the tile. Left clicking a revealed or flagged tile does nothing. The flag on a tile is toggled by right clicking on the tile. Right clicking on a revealed tile does nothing. Flagging mines has no effect on winning or losing, it is simply a tool and safeguard for the user to use to improve play.

3.2 Documentation

Certain program units can be utilized in other programs.

The unit `GridButton` contains the class `TGridButton` which inherits from the class `Vcl.StdCtrls.TButton`. Two properties are added, `X` and `Y`, which can be used to determine the location of a button within a grid.

The main program is not extensible, nor is its unit usable in other programs, unless it is used as a separate window.

The main program is defined in two parts, the program definition and the GUI (form) definition. The form definition is auto generated by the Delphi editor, RAD studio, and defines the objects in the window at program startup. The game grid is not generated until runtime.

The record type `CellContent` holds data for an individual cell, namely three boolean fields, `isMine` is true if the cell is a mine, `isVisible` is true if the cell has been revealed, and `isFlagged` is true if the cell has been flagged by the user.

The array types `GridOfCells` and `GridOfButtons` are for holding the data and button objects of a game board, respectively.

The constants are:

- `TOP_BUFFER` Pixels above the game board.
- `SIDE_BUFFER` Pixels to each side of the game board.
- `BOTTOM_BUFFER` Pixels below the game board.
- `CELL_SIZE` The pixel length of a side of a cell.

4 Program Listing

4.1 Main Program

```
// Author: Brady Coles
// Lab Assignment # 5
// Minesweeper clone
unit Minesweeper;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, GridButton, Vcl.ExtCtrls, Math;

type
  CellContent = record
    isMine, isVisible, isFlagged : Boolean;
  end;
  Whole = 1..MaxInt;
  GridOfCells = array[1..9, 1..9] of CellContent;
  GridOfButtons = array[1..9, 1..9] of TGridButton;
  TForm1 = class(TForm)
    GridButton1 : TGridButton;
    StatusLabel: TLabel;
    Label1: TLabel;
    procedure NewGameClick(Sender: TObject);
    procedure GridButtonClick(Sender: TObject; Button: TMouseButton; Shift: TShiftState;
            PixelX, PixelY: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

const
  TOP_BUFFER = 60;
  SIDE_BUFFER = 25;
  BOTTOM_BUFFER = 60;
  CELL_SIZE = 25;
var
  Form1: TForm1;
  Cells: GridOfCells;
  Buttons: GridOfButtons;
  BoardWidth, BoardHeight, NumberOfMines : Whole;

implementation

{£R *.dfm} // Include form definition

// Returns true if coordinates refer to a valid cell, false otherwise.
function IsValidCell(x, y: Integer): Boolean;
begin
  Result := (x >= 1) and (x <= BoardWidth) and (y >= 1) and (y <= BoardHeight);
```

```
end;

// Returns number of mines around a cell. Amount included cell (if cell is a mine).
function CellMineCount(x, y: Whole): Integer;
var
  i, j : Whole;
begin
  Result := 0;
  if IsValidCell(x, y) then
    for i := (x - 1) to (x + 1) do
      for j := (y - 1) to (y + 1) do
        if IsValidCell(i, j)
          then if Cells[i][j].isMine then Result := Result + 1;
end;

// Reveals the cell, giving the surrounding mine count if not a mine, or
// the '*' symbol if the cell is a mine. Also disables the button. If the cell
// has zero mines around it, reveals all surrounding cells as well (recursively).
procedure Reveal(x, y: Whole);
var
  i, j, count : Integer;
begin
  if IsValidCell(x, y) and not Cells[x][y].isVisible then
  begin
    if Cells[x][y].isMine then
    begin
      Buttons[x][y].Caption := '*';
      Buttons[x][y].Enabled := False;
      Cells[x][y].isVisible := True;
    end
    else
    begin
      count := CellMineCount(x, y);
      Buttons[x][y].Caption := count.ToString;
      Buttons[x][y].Enabled := False;
      // Before recursive call to avoid infinite recursion.
      Cells[x][y].isVisible := True;
      if count = 0 then
        // Reveal surrounding cells.
        for i := (x - 1) to (x + 1) do
          for j := (y - 1) to (y + 1) do
            if IsValidCell(i, j)
              then Reveal(i, j);
    end;
  end;
end;

// Toggles the flag on a game cell. If the cell has not been revealed, flags
// the cell if not flagged, removes flag if flagged. Flagged cells cannot be
// revealed.
procedure Flag(x, y : Integer);
begin
  if IsValidCell(x,y) and (not Cells[x][y].isVisible) then
  begin
```

```
    if Cells[x][y].isFlagged then
    begin
      Cells[x][y].isFlagged := False;
      Buttons[x][y].Caption := '';
    end
    else
    begin
      Cells[x][y].isFlagged := True;
      Buttons[x][y].Caption := 'F';
    end;
  end;
end;


// Initialized data for game board. Prepared data for window initialization.
// Randomly chooses cells to be mines.
procedure InitializeBoard(width, height, mines : Whole);
var
   rand, x, y, i, j : Whole;
begin
  BoardWidth := width;
  BoardHeight := height;
  NumberOfMines := mines;
  // Initialize data, needed for resets between games.
  for i := 1 to BoardWidth do
    for j := 1 to BoardHeight do
    begin
      Cells[i][j].isMine := False;
      Cells[i][j].isVisible := False;
      Cells[i][j].isFlagged := False;
    end;
  // Select mines
  for i := 1 to NumberOfMines do
    begin
    repeat
      rand := RandomRange(1, BoardWidth * BoardHeight);
      x := rand mod BoardWidth;
      y := rand div BoardHeight;
    until not Cells[x][y].isMine;
    Cells[x][y].isMine := True;
  end;
end;

// Set up form for a game. Makes window proper size for game board, creates
// or resets game cells.
procedure InitializeWindow;
var
  B : TGridButton;
  i, j : Whole;
begin
  // Set up window
  Form1.Width := BoardWidth * CELL_SIZE + 2 * SIDE_BUFFER
    + Form1.Margins.Left + Form1.Margins.Right + 10;
  Form1.Height := BoardHeight * CELL_SIZE + TOP_BUFFER + BOTTOM_BUFFER
```

```
    + Form1.Margins.Top + Form1.Margins.Bottom;
  Form1.StatusLabel.Caption := format('%u Mines', [NumberOfMines]);
  // Set up cells
  for i := 1 to BoardWidth do
    for j := 1 to BoardHeight do
    begin
      if (Buttons[i][j] = nil) then Buttons[i][j] := TGridButton.Create(Form1);
      Buttons[i][j].X := i;
      Buttons[i][j].Y := j;
      Buttons[i][j].Height := CELL_SIZE;
      Buttons[i][j].Width := CELL_SIZE;
      Buttons[i][j].Left := SIDE_BUFFER + (i - 1) * CELL_SIZE;
      Buttons[i][j].Top := TOP_BUFFER + (j - 1) * CELL_SIZE;
      Buttons[i][j].Caption := '';
      Buttons[i][j].Parent := Form1;
      Buttons[i][j].OnMouseDown := Form1.GridButtonClick;
      Buttons[i][j].Enabled := True;
    end;
end;

// Checks if the game has been won. Returns true if only unclicked cells
// are mines.
function CheckWinState : Boolean;
var
i, j : Integer;
begin
  for i := 1 to BoardWidth do
    for j := 1 to BoardHeight do
      if (not Cells[i][j].isVisible) and (not Cells[i][j].isMine) then
      begin
        Result := False;
        exit;
      end;
  Result := True;
end;

// Ends the game by revealing all cells and changing label to win or loss message
procedure EndGame(isWin : Boolean);
var
  i, j : Whole;
begin
  for i := 1 to BoardWidth do
    for j := 1 to BoardHeight do
      Reveal(i, j);
  if isWin then Form1.StatusLabel.Caption := 'You Win!'
  else Form1.StatusLabel.Caption := 'You Lose!'
end;

// Handle starting a new game. Creates a 9x9 game with 10 mines.
procedure TForm1.NewGameClick(Sender: TObject);
begin
  InitializeBoard(9, 9, 10);
  InitializeWindow;
end;
```

```

// Handle a click on a game cell.
// Checks mouse button, right is flag, left is reveal.
// Handles ending the game on a win or loss.
procedure TForm1.GridButtonClick(Sender: TObject; Button: TMouseButton; Shift: TShiftState;
        PixelX, PixelY: Integer);
var
  x, y : Whole;
  count : Integer;
begin
  // Ensure that the event was on a game cell
  if Sender is TGridButton then
  begin
    x := TGridButton(Sender).X;
    y := TGridButton(Sender).Y;
    if not IsValidCell(x, y) then exit;
    case Button of
      // Reveal. Note that already revealed cells will be disabled, so all
      // events should be on unrevealed cells. If cell is revealed, Reveal
      // procedure will ignore it.
      mbLeft:
      begin
        if Cells[x][y].isFlagged then exit;
        if Cells[x][y].isMine then EndGame(False) // Game over, player clicked mine
        else
        // Reveal cell, if last cell, win game.
        begin
          Reveal(x, y);
          if CheckWinState then EndGame(True);
        end;
      end;
      // Flag or unflag cell, depending on current state.
      mbRight: Flag(x, y)
    end;
  end;
end;

end.
```

4.2 Form Definition

```
object Form1: TForm1
  Left = 0
  Top = 0
  Caption = 'Minesweeper'
  ClientHeight = 58
  ClientWidth = 275
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object StatusLabel: TLabel
    Left = 147
    Top = 29
    Width = 70
    Height = 13
  end
  object Label1: TLabel
    Left = 24
    Top = 5
    Width = 152
    Height = 13
    Caption = 'Brady Coles - CMPT 360 Lab #5'
  end
  object GridButton1: TGridButton
    Left = 24
    Top = 24
    Width = 91
    Height = 25
    Caption = 'Start New Game'
    TabOrder = 0
    OnClick = NewGameClick
    X = 0
    Y = 0
  end
end
```

4.3 Button Component

```
unit GridButton;

interface

uses
  System.SysUtils, System.Classes, Vcl.Controls, Vcl.StdCtrls;

type
  TGridButton = class(TButton)
  private
    FX, FY : Integer;
  protected
    { Protected declarations }
  public
    { Public declarations }
  published
    property X: Integer read FX write FX;
    property Y: Integer read FY write FY;
  end;

procedure Register;

implementation

```

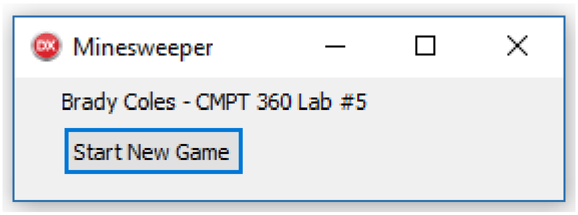
```
procedure Register;
begin
  RegisterComponents('Samples', [TGridButton]);
end;

end.
```

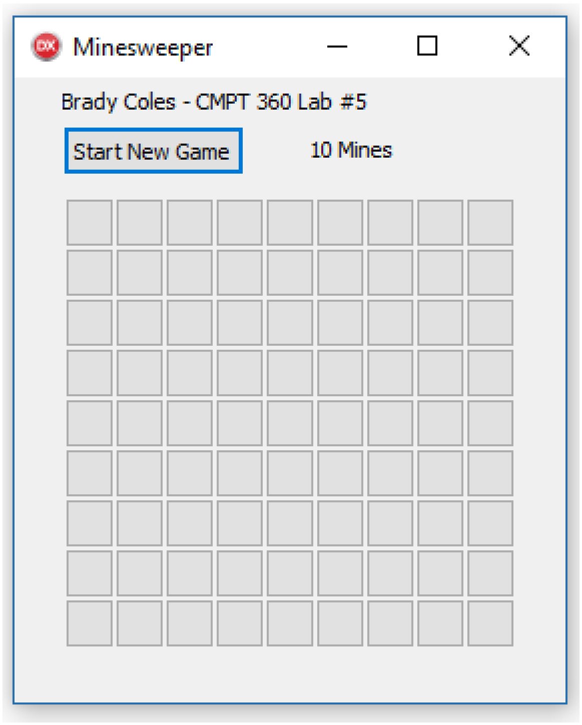
5 Sample Operation

Here are some screenshots of the program at different states.

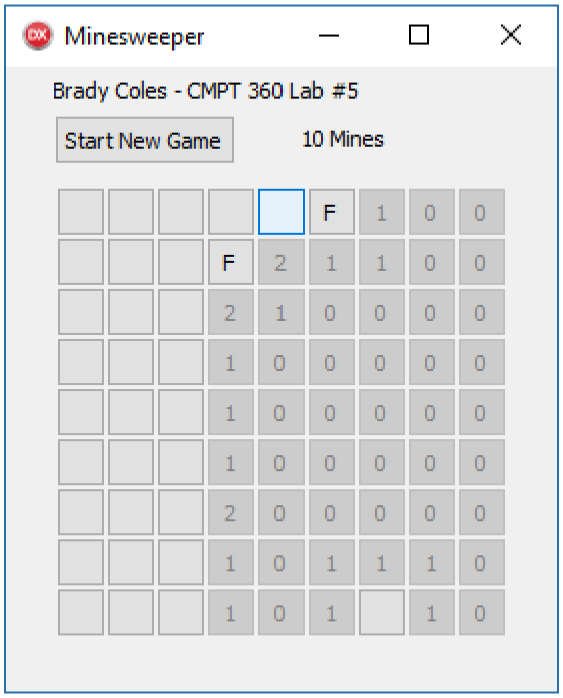
After Startup The window has no game board, just a label and a start game button.



New Game The window resizes to fit the game board, and displays the number of mines.



Game in Progress An in progress game with hidden, revealed, and flagged cells.



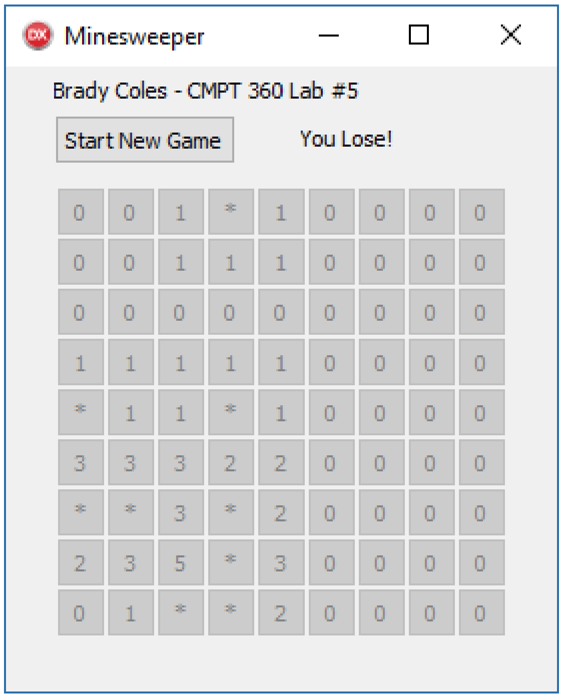
Minesweeper

Brady Coles - CMPT 360 Lab #5

Start New Game   10 Mines

```
          F 1 0 0
      F 2 1 1 0 0
      2 1 0 0 0 0
      1 0 0 0 0 0
      1 0 0 0 0 0
      1 0 0 0 0 0
      2 0 0 0 0 0
      1 0 1 1 1 0
      1 0 1   1 0
```

Game Over: Loss A lost game, revealing all tiles, and displaying the message 'You Lose!'



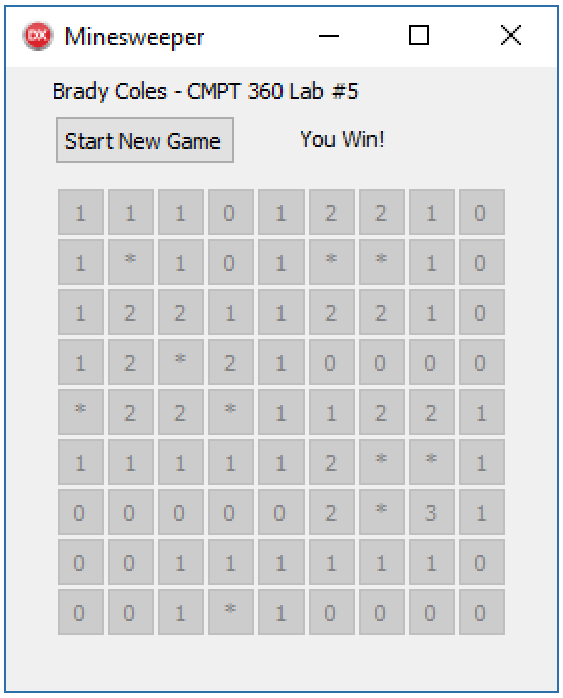
Minesweeper

Brady Coles - CMPT 360 Lab #5

Start New Game   You Lose!

```
0 0 1 * 1 0 0 0 0
0 0 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0
1 1 1 1 1 0 0 0 0
* 1 1 * 1 0 0 0 0
3 3 3 2 2 0 0 0 0
* * 3 * 2 0 0 0 0
2 3 5 * 3 0 0 0 0
0 1 * * 2 0 0 0 0
```

Game Over: Win A won game, revealing all tiles, and displaying the message 'You Win!'



Minesweeper

Brady Coles - CMPT 360 Lab #5

You Win!

1	1	1	0	1	2	2	1	0
1	*	1	0	1	*	*	1	0
1	2	2	1	1	2	2	1	0
1	2	*	2	1	0	0	0	0
*	2	2	*	1	1	2	2	1
1	1	1	1	1	2	*	*	1
0	0	0	0	0	2	*	3	1
0	0	1	1	1	1	1	1	0
0	0	1	*	1	0	0	0	0