Лабораторная работа №8:
Создание игры сапер

Цель:

Целью данной работы является закрепление навыков, полученных в ходе выполнения предыдущих лабораторных работ.

Задание:

Разработать и реализовать программное приложение, состоящее из двух модулей:

1. Генератор уровней – набор функций, для генерации поля заданного размера и проверки допустимости действий пользователя
   - Размеры поля являются переменными параметрами, и задаются пользователем.
   - Логически, поле представлено в виде двумерного массива.
   - Каждое действие пользователя приводит к изменению состояния поля.
   - После каждого действия, совершаемого над полем, должны проверяться условия победы и поражения.
2. Графический интерфейс – WPF форма и связанные с ней функции для отображения игрового поля и обработки действий пользователя.
3. Таблица рекордов – WPF форма, связанная с базой данных содержащей имя игрока и счет.
4. Реализовать на выбор: систему звуков или элементы с анимацией.

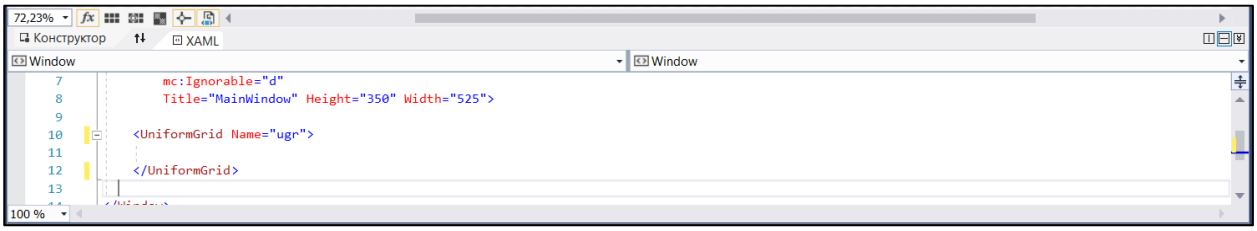
Справочная информация:

Интерфейс игры, состоит из выбора размеров поля, таймера, кнопки “Старт” и сетки “минное поле”. Для реализации большинства элементов интерфейса могут быть использованы стандартные компоненты WPF, освоенные в процессе выполнения лабораторных работ по курсу “Программирование”.

Для реализации игрового поля, предлагается использовать визуальный компонент UniformGrid.

UniformGrid.

Для того, чтобы добавить компонент UniformGrid, после создания формы, перейдите к редактору XAML и замените компонент Grid на UniformGrid. Не забудьте указать имя компонента.



После этого, вы можете получить доступ к этому компоненту из программного кода, и задать необходимые параметры, используя имя этого компонента:

```
//указывается количество строк и столбцов в сетке
ugr.Rows = 5;
ugr.Columns = 5;
//указываются размеры сетки (число ячеек * (размер кнопки в ячейки + толщина её границ))
ugr.Width = 5 * (50+4);
ugr.Height = 5 * (50+4);
//толщина границ сетки
ugr.Margin = new Thickness(5, 5, 5, 5);
```

В качестве элементов игрового поля, предлагается использовать кнопки (Button). Создать кнопку и добавить её в сетку можно следующим образом

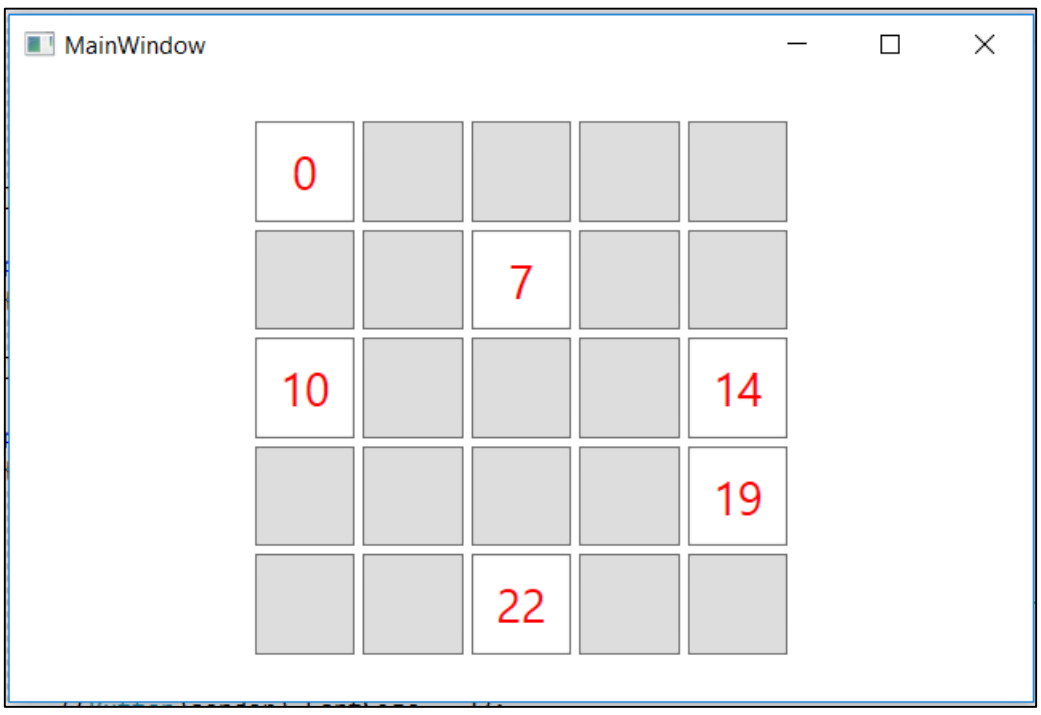
```
for (int i = 0; i < 5*5; i++)
{
    //создание кнопки
    Button btn = new Button();
    //запись номера кнопки
    btn.Tag = i;
    //установка размеров кнопки
    btn.Width = 50;
    btn.Height = 50;
    //текст на кнопке
    btn.Content = " ";
    //толщина границ кнопки
    btn.Margin = new Thickness(2);
    //при нажатии кнопки, будет вызываться метод Btn_Click
    btn.Click += Btn_Click;
    //добавление кнопки в сетку
    ugr.Children.Add(btn);
}
```

В результате работы этого цикла, в сетку будет добавлено 25 кнопок, расположенных в 5 строк по 5 кнопок в каждой.

При этом обработчик события нажатия (Click) будет выглядеть следующим образом:

```
private void Btn_Click(object sender, RoutedEventArgs e)
{
    //получение значения лежащего в Tag
    int n = (int)((Button)sender).Tag;
    //установка фона нажатой кнопки, цвета и размера шрифта
    ((Button)sender).Background = Brushes.White;
    ((Button)sender).Foreground = Brushes.Red;
    ((Button)sender).FontSize = 23;
    //запись в нажатую кнопку её номера
    ((Button)sender).Content = n.ToString();
}
```

Результат работы:



В случае, если вы хотите выполнять различные действия, при клике на кнопку средней или правой кнопками мыши, можно добавить обработчик события нажатия кнопки мыши:

```
btn.PreviewMouseDown += Btn_MouseDown;
```

Обработчик события будет выглядеть следующим образом:

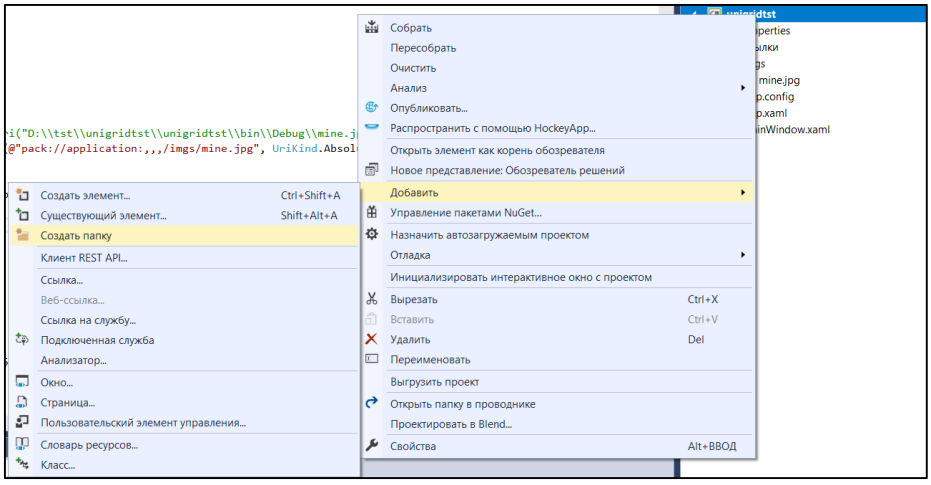
```
private void Btn_MouseDown(object sender, MouseButtonEventArgs e)
{
    if (e.LeftButton == MouseButtonState.Pressed)
    {
        MessageBox.Show("Нажата левая кнопка мыши!");
    }
    if (e.MiddleButton == MouseButtonState.Pressed)
    {
        MessageBox.Show("Нажата средняя кнопка мыши!");
    }
    if (e.RightButton == MouseButtonState.Pressed)
    {
        MessageBox.Show("Нажата правая кнопка мыши!");
    }
}
```

Image и StackPanel.

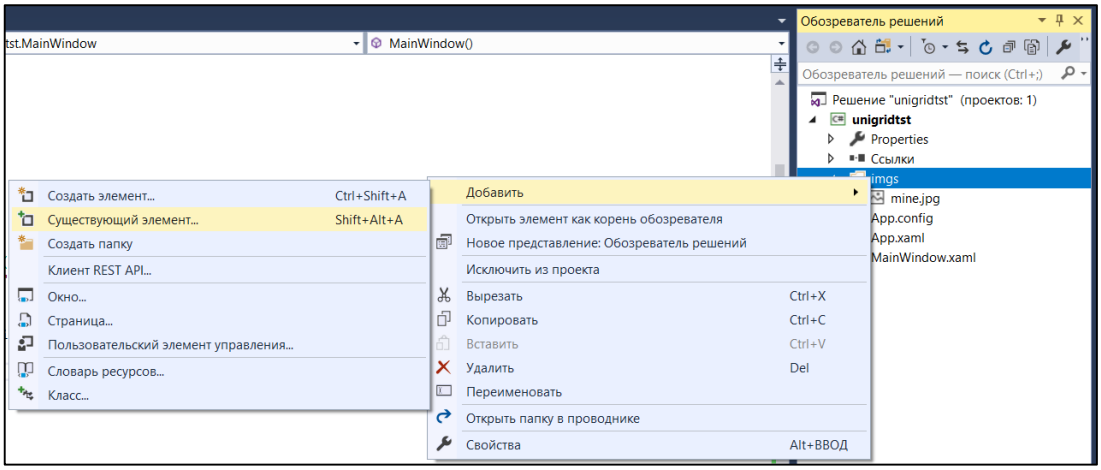
Для того, чтобы вывести на поверхность кнопки изображение, можно использовать контейнер Image и компонент StackPanel.

Для того, чтобы загрузить изображения, которые вам могут понадобиться в процессе выполнения лабораторной работы, нужно добавить их к проекту.

Для этого, создайте папку:



Добавьте в папку существующие элементы (изображения):



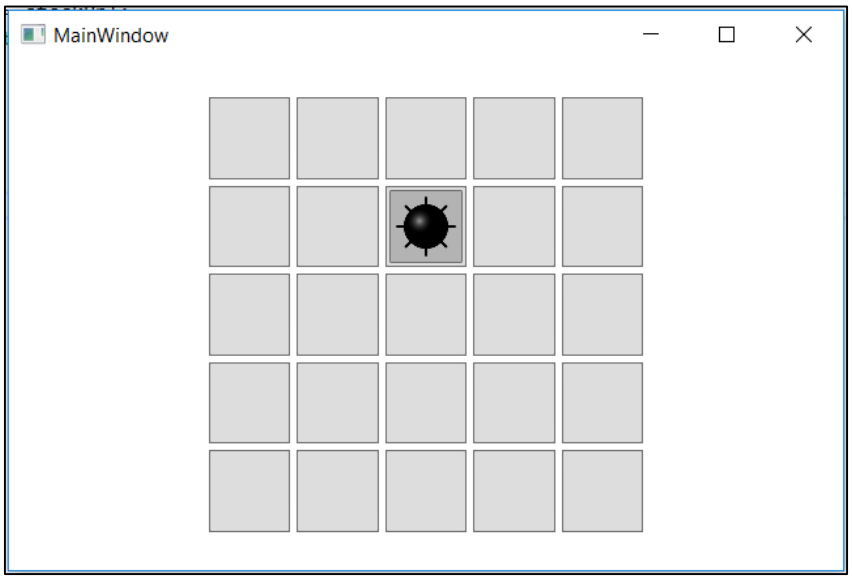
После этого, вы можете загрузить их в коде:

```
//создание и инициализация глобальной переменной для хранения изображения мины
BitmapImage mine  = new BitmapImage(new Uri(@"pack://application:,,,/imgs/mine.jpg",
UriKind.Absolute));
```

А затем, создать компонент для отображения и записать его в нажатую кнопку в обработчике `Btn_Click` или `Btn_MouseDown`:

```
private void Btn_Click(object sender, RoutedEventArgs e)
{
    //создание контейнера для хранения изображения
    Image img = new Image();
    //запись картинки с миной в контейнер
    img.Source = mine;
    //создание компонента для отображения изображения
    StackPanel stackPnl = new StackPanel();
    //установка толщины границ компонента
    stackPnl.Margin = new Thickness(1);
    //добавление контейнера с картинкой в компонент
    stackPnl.Children.Add(img);
    //запись компонента в кнопку
    ((Button)sender).Content = stackPnl;
}
```

Результат:



Программная реализация

Игровое поле может быть представлено в виде массива целых чисел:

```
int[,] field = new int[M, N];
```

Где M и N число строк и столбцов в поле. Элементы массива могут принимать следующие значения:

0 – пустая клетка

9 – клетка содержит мину

1..8 – число мин в соседних клетках

Функция распределения мин будет иметь следующий прототип:

```
void plantMines(int mines)
{
    //реализация функции
}
```

Где mines – число мин, которые необходимо разместить на поле.

Алгоритм работы функции будет выглядеть следующим образом:

1. Выбрать в пределах поля случайную ячейку.
2. Проверить, что в этой ячейке ещё нет мины (если есть, вернуться к шагу 1)
3. Проверить, что рядом с этой ячейкой есть хотя бы одна пустая (если нет, вернуться к шагу 1)
4. Разместить в выбранной ячейке мину. (записать в массив 9)

Функция подсчёта мин в соседних ячейках будет иметь следующую структуру:

1. Для каждой ячейки поля. (перебор координат ячеек по i и по j)
2. Если в ячейке нет мины. (значение массива не равно 9)
3. Объявить переменную счётчик равную 0.
   - Перебрать 8 соседних ячеек.
   - Если в ячейке встречается мина (значение массива 9), увеличить счётчик.
4. Записать в ячейку значение счётчика.

Если предположить, что координаты проверяемой ячейки i и j, то 8 её соседей будут иметь координаты (i-1, j-1), (i, j-1), (i+1, j-1), (i-1, j), (i+1, j), (i-1, j+1), (i, j+1), (i-1, j+1). Обратите внимание, что для ячеек, находящихся на границе поля, соседей будет меньше 8ми.

Если всё было сделано правильно, то после реализации и исполнения описанных функций, будет сгенерировано поле для игры в “Сапёр”.

Дополнительно, может быть реализована рекурсивная функция, открывающая области пустых клеток, при попадании в одну из них.

Список литературы:

Класс UniformGrid:

https://msdn.microsoft.com/ru-ru/library/system.windows.controls.primitives.uniformgrid(v=vs.110).aspx

Класс Button:

https://msdn.microsoft.com/ru-ru/library/system.windows.controls.button(v=vs.110).aspx

Класс StackPanel:

https://msdn.microsoft.com/ru-ru/library/system.windows.controls.stackpanel(v=vs.110).aspx