# **Boggle Game**

Alex Thayn



## **Boggle Basics**

The player is presented a 4x4 matrix of letters

The object of the game is to find as many words of length three or in three minutes

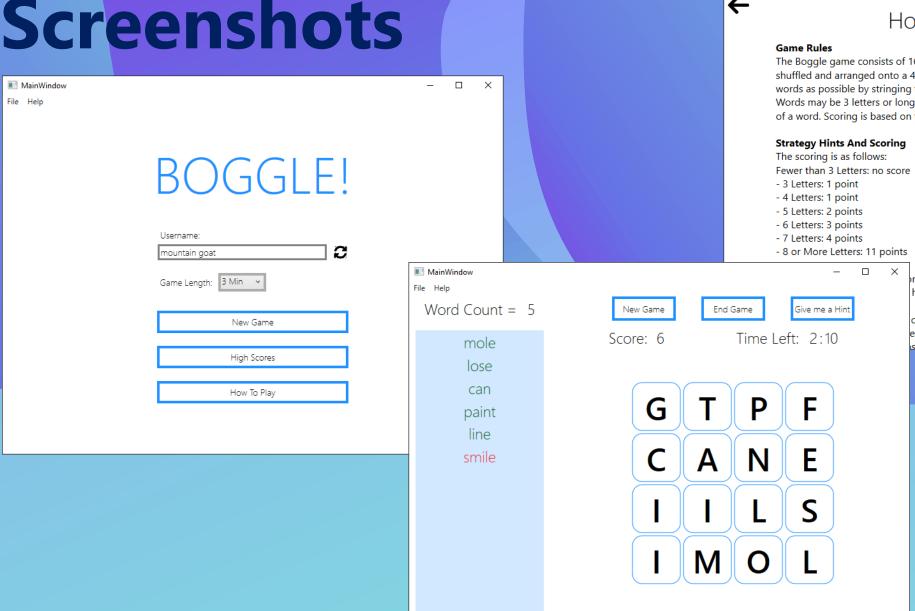
Form words created out of adjoining letters, it is valid to move across diagonally

Words are scored according to the number of letters used





# Screenshots



How To Play Boggle The Boggle game consists of 16 6-sided dice, each side containing a letter. The dice are shuffled and arranged onto a 4x4 grid where players are given 3 minutes to find as many words as possible by stringing together adjacent letters from the top face of each die. Words may be 3 letters or longer and a die cannot be used more than once in the formation of a word. Scoring is based on the length of the word.

MainWindow

Submit Guess

ords as possible is a good strategy, it is worth bearing in mind highly, so one 5-letter word is worth two 4-letter words.

occupies a single space in the grid, it counts as two letters for edit is awarded for both the singular and plural forms of a s separate words, so cat and cats each score a point. The

### **My Code – Unit Tests**

```
[SetUp]
public void SetUp()
{
    //Setup test game grid
    string[] r1 = new string[4] { "S", "E", "R", "S" };
    string[] r2 = new string[4] { "P", "A", "T", "G" };
    string[] r3 = new string[4] { "L", "I", "N", "E" };
    string[] r4 = new string[4] { "S", "E", "R", "S" };
    string[][] grid = new string[4][] { r1, r2, r3, r4 };
    BoggleGame = new BoggleGame("FakeUser", 0);
    BoggleGame.GameBoard.GameGrid = grid;
    gameVM.TheGame = BoggleGame;
    BoggleGame.ListOfPossibleAnswers = possibleGuesses;
}
```

```
[TestCase("pal ", 1)]
[TestCase("ale", 1)]
[TestCase("Hi!", 0)]
[TestCase("pig", 0)]
public void Test3LetterScoreIsCalculatedCorrectly(string word, int expectedScore)
    gameVM.UserGuess = word;
    gameVM.SubmitGuessCommand.Execute(null);
    Assert.AreEqual(expectedScore, gameVM.TheGame.GetScore());
[TestCase("sang", 1)]
[TestCase(" line", 1)]
[TestCase("lats", 1)]
[TestCase("bear ", 0)]
public void Test4LetterScoreIsCalculatedCorrectly(string word, int expectedScore)
    gameVM.UserGuess = word;
    gameVM.SubmitGuessCommand.Execute(null);
    Assert.AreEqual(expectedScore, gameVM.TheGame.GetScore());
```



### **My Code – Game Logic**



```
public void SubmitGuess(string Word)
   //Check if the user entered a duplicate guess
   foreach(PlayerGuess g in ListOfGuesses)
       if (g.Guess == Word.ToLower())
           return;
   bool isGuessOnGameGrid = ListOfPossibleAnswers.Contains(Word.ToUpper());
   ListOfGuesses.Add(new PlayerGuess() { Guess = Word.ToLower(), IsValidGuess = isGuessOnGameGrid });
   if (isGuessOnGameGrid)
       WordCount++;
       int wordLength = Word.Count(w => char.IsLetter(w));
       if (wordLength < 3)</pre>
       switch (wordLength)
           case 3: Score += 1;
                break;
           case 4: Score += 1;
                break;
           case 5: Score += 2;
                break;
           case 6: Score += 3;
                break:
           case 7: Score += 4;
           //return a score of 11 points if the length of the word is greater than 8 letters
           default: Score += 11;
                break;
```

#### **Design Decisions & Tradeoffs**

#### **Navigation**

- One page versus
- Multiple pages
- Creation of multiple view models







# **Providing Feedback to the player**

- Visual feedback
- Showing previously entered quesses
- How to handle duplicate guesses
- Asynchronous timer

# How to validate player guesses

- Find all possible words on the grid
- Check the grid for each guess entered









### **What I learned**

Azure DevOps and creating a build pipeline

The importance of TDD and creating high quality unit tests

Using a code first approach to database creation

