Scott Hawkins
MBA 614
Final Project Summary
4/13/10

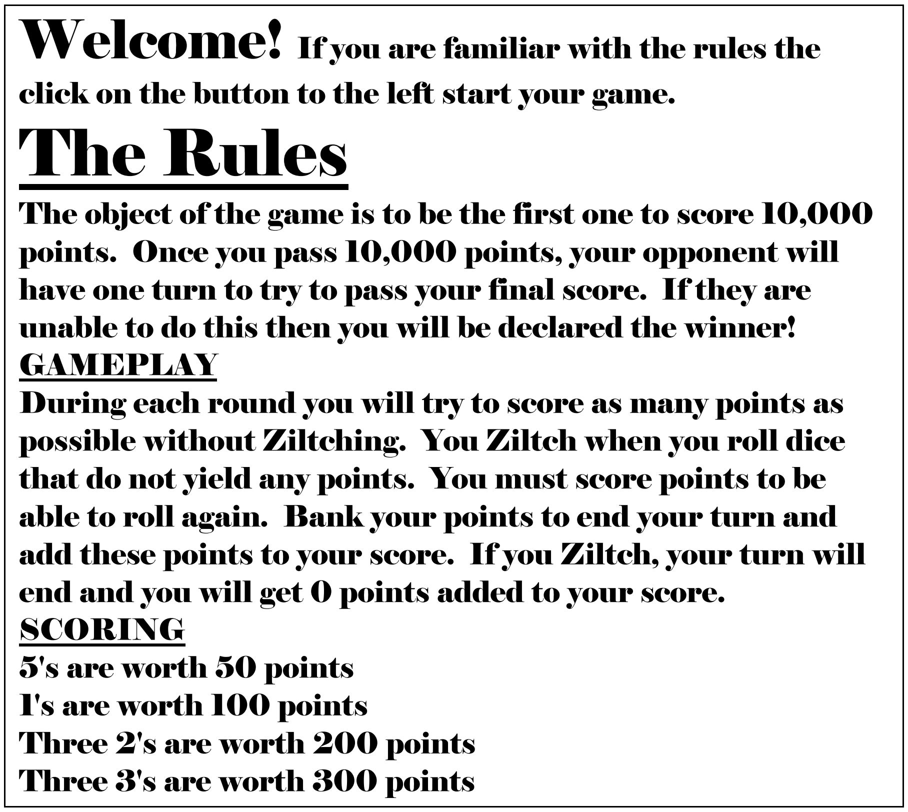
Zilch! - Playing with VBA

Introduction

For my project I choose to be a little unorthodox and create a game rather than a business solution. I chose to recreate the game Zilch which I have played various versions of throughout the years. I initially chose the game with the idea of adding AI to the program in the form of a computer opponent. However, after many grueling hours, I decided to stick with the basic functions and just make sure that the game worked without bugs.

Ziltch! is a game of chance, where you decide how many points to risk every time you roll the dice. Players who play too conservatively will not reach 10,000 points as fast as their more aggressive opponents. However, those players who play too aggressively will Ziltch often and be left behind. While there is a lot of luck involved, the game is enjoyable because of the conservative/aggressive dynamic.

Here are the official rules taken from the game:



Welcome! **If you are familiar with the rules the click on the button to the left start your game.**

The Rules

The object of the game is to be the first one to score 10,000 points. Once you pass 10,000 points, your opponent will have one turn to try to pass your final score. If they are unable to do this then you will be declared the winner!

GAMEPLAY

During each round you will try to score as many points as possible without Ziltching. You Ziltch when you roll dice that do not yield any points. You must score points to be able to roll again. Bank your points to end your turn and add these points to your score. If you Ziltch, your turn will end and you will get 0 points added to your score.

SCORING

5's are worth 50 points
1's are worth 100 points
Three 2's are worth 200 points
Three 3's are worth 300 points

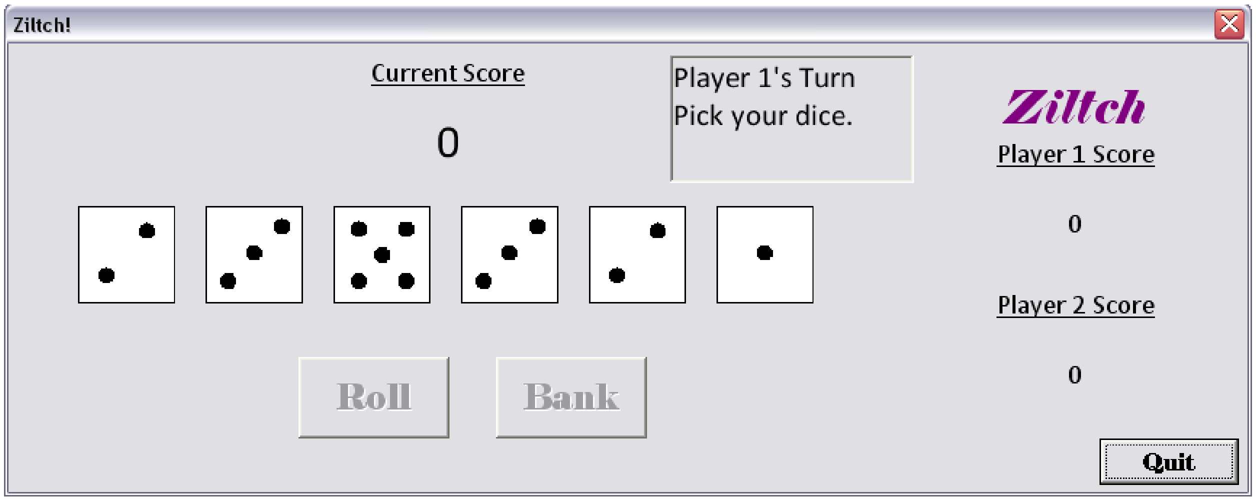
Three 4's are worth 400 points
Three 5's are worth 500 points
Three 6's are worth 600 points
Three 1's are worth 1000 points
4-of-a-kind is worth double the 3-of-a-kind value
(example: Four 2's are worth 400 points)
5-of-a-kind is worth double the 4-of-a-kind value
(example: Five 2's are worth 800 points)
6-of-a-kind is worth double the 5-of-a-kind value
(example: Six 2's are worth 1600 points)
Three pair is worth 1500 points
A run is worth 1500 Points
Having six dice with no score is worth 500 points

Rolling the Dice

Getting the dice to roll was probably the easiest part of the programming process. Using paint I created 18 different dice: 6 white, 6 purple and 6 grey. I then created an array of picture boxes and placed them on my form (see form below.)

The rolling of the dice happened by selecting a random number 1-6 and then changing the image on the corresponding picture box. This was very simple code, but does require the pictures to be in the same folder as the excel spreadsheet.

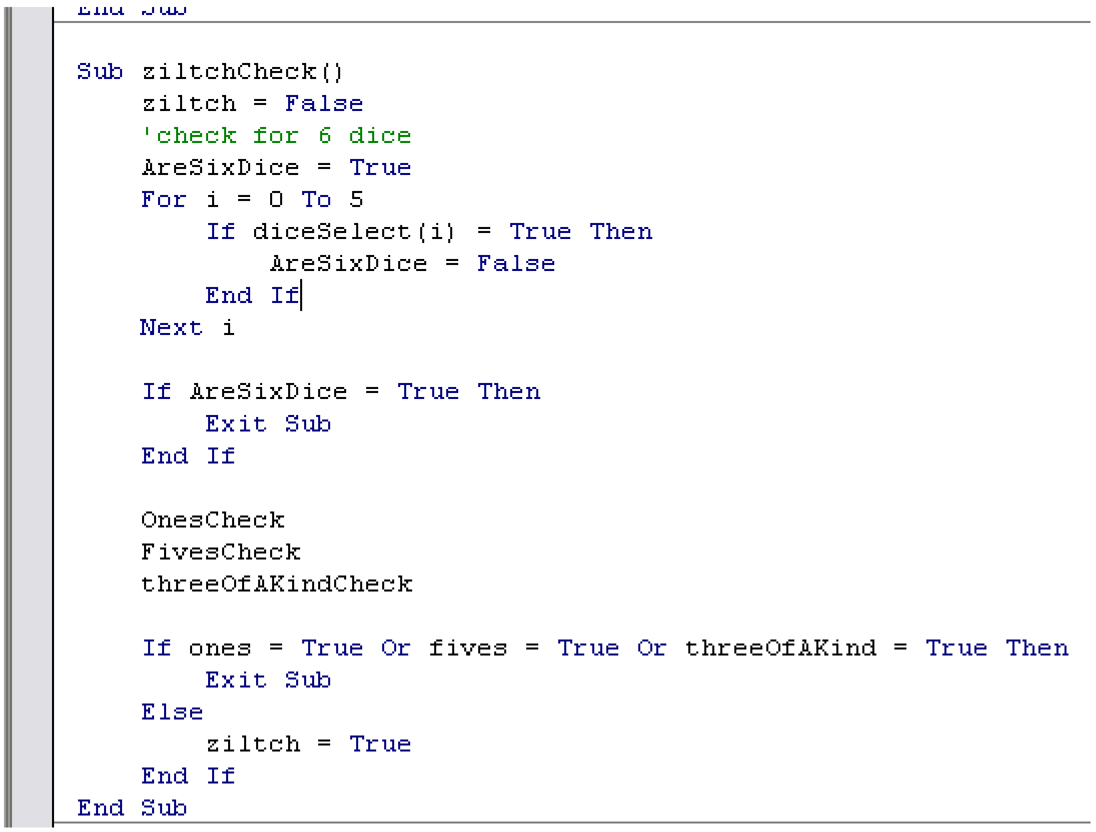
A better solution would have been to create 18 separate picture boxes and make them visible as needed. After finishing the project I deemed this too time consuming of a change to make and therefore left it as is.



Player 1's Turn
Pick your dice.

Recognizing a Ziltch

My first programming task was to create a way for the program to recognize when scoring was not possible and therefore end the turn of said player. This was a fairly straightforward set of sub routines. If the player has six dice left then a Ziltch is impossible. This made it easier to screen only for ones, fives and three or more of a kind. A simple subroutine for each could quickly determine if a player Ziltches.



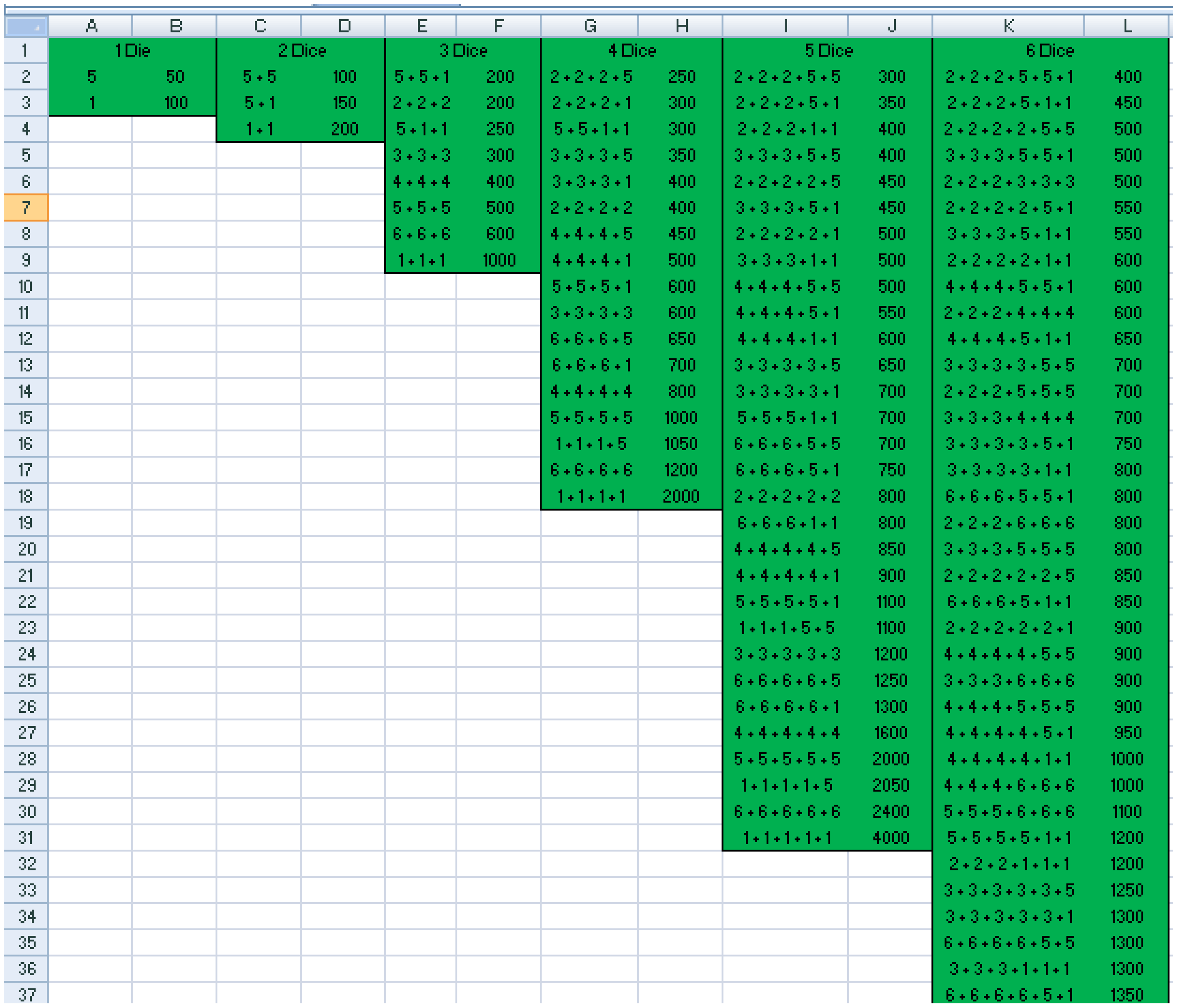
```
End Sub

Sub ziltchCheck()
    ziltch = False
    'check for 6 dice
    AreSixDice = True
    For i = 0 To 5
        If diceSelect(i) = True Then
            AreSixDice = False
        End If
    Next i

    If AreSixDice = True Then
        Exit Sub
    End If

    OnesCheck
    FivesCheck
    threeOfAKindCheck

    If ones = True Or fives = True Or threeOfAKind = True Then
        Exit Sub
    Else
        ziltch = True
    End If
End Sub
```

Scoring

By far the most difficult part of the code was in calculating the scoring. This is where I spent the bulk of my time writing code. Upon reflection, I’m sure that I could have found a way to make cleaner code that was less repetitive. However, I am very confident that it works and works well. Sometimes my ‘cleaner’ solutions have a lot more bugs.

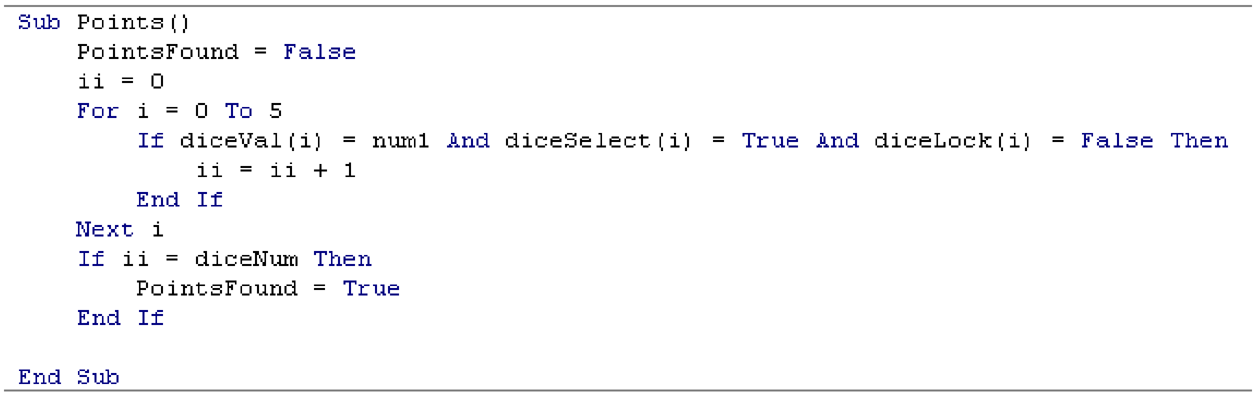
The most difficult part of the whole process was in the sheer number of combinations. Below I have included a section of my scoring chart that shows most, but not all of the combinations. The problem is that in Ziltch, you can score different items in one roll. For example, I could get three 4’s and score 400 points, while also selecting a 5 and a 1, which would add 50 and 100 points respectively. I had to make sure my code would score the 400 without ignoring the 150 points.

Just to see what I was up against I started listing possible combinations. It turns out that there are 55 ways to score with 6 dice, 30 ways with 5 dice, 17 with 4 dice, 8 with 3 dice, 3 with two dice and 2 with 1 die. That’s a total of 115 different scoring possibilities!



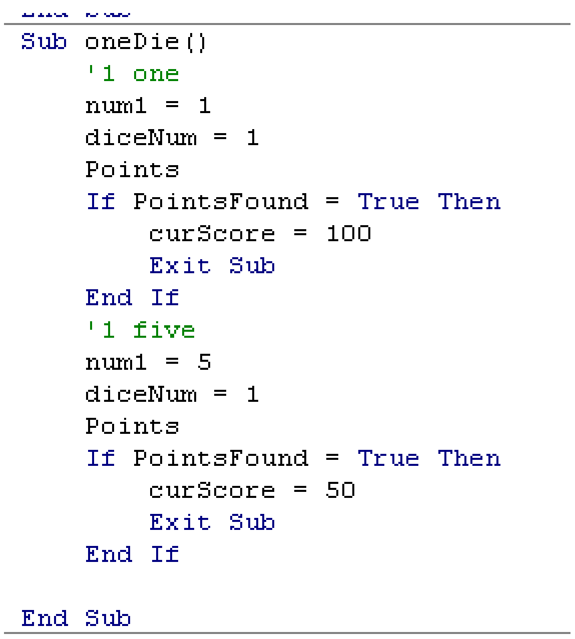
	A	B	C	D	E	F	G	H	I	J	K	L
1	1 Die		2 Dice		3 Dice		4 Dice		5 Dice		6 Dice	
2	5	50	5+5	100	5+5+1	200	2+2+2+5	250	2+2+2+5+5	300	2+2+2+5+5+1	400
3	1	100	5+1	150	2+2+2	200	2+2+2+1	300	2+2+2+5+1	350	2+2+2+5+1+1	450
4			1+1	200	5+1+1	250	5+5+1+1	300	2+2+2+1+1	400	2+2+2+2+5+5	500
5					3+3+3	300	3+3+3+5	350	3+3+3+5+5	400	3+3+3+5+5+1	500
6					4+4+4	400	3+3+3+1	400	2+2+2+2+5	450	2+2+2+3+3+3	500
7					5+5+5	500	2+2+2+2	400	3+3+3+5+1	450	2+2+2+2+5+1	550
8					6+6+6	600	4+4+4+5	450	2+2+2+2+1	500	3+3+3+5+1+1	550
9					1+1+1	1000	4+4+4+1	500	3+3+3+1+1	500	2+2+2+2+1+1	600
10							5+5+5+1	600	4+4+4+5+5	500	4+4+4+5+5+1	600
11							3+3+3+3	600	4+4+4+5+1	550	2+2+2+4+4+4	600
12							6+6+6+5	650	4+4+4+1+1	600	4+4+4+5+1+1	650
13							6+6+6+1	700	3+3+3+3+5	650	3+3+3+3+5+5	700
14							4+4+4+4	800	3+3+3+3+1	700	2+2+2+5+5+5	700
15							5+5+5+5	1000	5+5+5+1+1	700	3+3+3+4+4+4	700
16							1+1+1+5	1050	6+6+6+5+5	700	3+3+3+3+5+1	750
17							6+6+6+6	1200	6+6+6+5+1	750	3+3+3+3+1+1	800
18							1+1+1+1	2000	2+2+2+2+2	800	6+6+6+5+5+1	800
19									6+6+6+1+1	800	2+2+2+6+6+6	800
20									4+4+4+4+5	850	3+3+3+5+5+5	800
21									4+4+4+4+1	900	2+2+2+2+2+5	850
22									5+5+5+5+1	1100	6+6+6+5+1+1	850
23									1+1+1+5+5	1100	2+2+2+2+2+1	900
24									3+3+3+3+3	1200	4+4+4+4+5+5	900
25									6+6+6+6+5	1250	3+3+3+6+6+6	900
26									6+6+6+6+1	1300	4+4+4+5+5+5	900
27									4+4+4+4+4	1600	4+4+4+4+5+1	950
28									5+5+5+5+5	2000	4+4+4+4+1+1	1000
29									1+1+1+1+5	2050	4+4+4+6+6+6	1000
30									6+6+6+6+6	2400	5+5+5+6+6+6	1100
31									1+1+1+1+1	4000	5+5+5+5+1+1	1200
32											2+2+2+1+1+1	1200
33											3+3+3+3+3+5	1250
34											3+3+3+3+3+1	1300
35											6+6+6+6+5+5	1300
36											3+3+3+1+1+1	1300
37											6+6+6+6+5+1	1350

I essential created a sub that would do all the work for me and return a true if it found the number of requested dice.



```
Sub Points()
    PointsFound = False
    ii = 0
    For i = 0 To 5
        If diceVal(i) = num1 And diceSelect(i) = True And diceLock(i) = False Then
            ii = ii + 1
        End If
    Next i
    If ii = diceNum Then
        PointsFound = True
    End If

End Sub
```

This little sub did a lot of work as it was repeated an untold number of times as the program runs. After creating this I simply had to load 115 different scenarios to input values into the sub to find if there was a match. The creation of these 115 sections of code was the more tedious portion of the programming and takes up the bulk of the code. Here is what it looks like for when there is only one die selected:



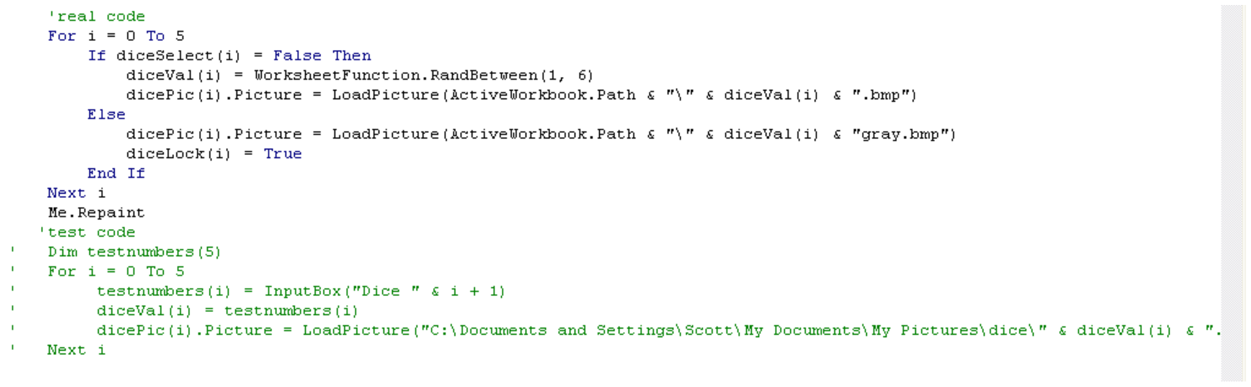
```
Sub oneDie()
    '1 one
    num1 = 1
    diceNum = 1
    Points
    If PointsFound = True Then
        curScore = 100
        Exit Sub
    End If
    '1 five
    num1 = 5
    diceNum = 1
    Points
    If PointsFound = True Then
        curScore = 50
        Exit Sub
    End If

End Sub
```

Very simply, it asks if num1 (the dice value) can be found diceNum (number of times found) number of times. If it comes back true then it accepts a value for the current score and stops searching for new possibilities.

Looking for more than one value simply required a nested if statement to check if the second value existed the required number of times.

Debugging

Once I had the game play and the winning ironed out, it was time to debug the program and make sure that it worked properly. I used a couple of tools to be able to test with ease. This was important since I had to debug over 115 different scoring possibilities.



```
     'real code
     For i = 0 To 5
         If diceSelect(i) = False Then
             diceVal(i) = WorksheetFunction.RandBetween(1, 6)
             dicePic(i).Picture = LoadPicture(ActiveWorkbook.Path & "\" & diceVal(i) & ".bmp")
         Else
             dicePic(i).Picture = LoadPicture(ActiveWorkbook.Path & "\" & diceVal(i) & "gray.bmp")
             diceLock(i) = True
         End If
     Next i
     Me.Repaint
    'test code
'    Dim testnumbers(5)
'    For i = 0 To 5
'         testnumbers(i) = InputBox("Dice " & i + 1)
'         diceVal(i) = testnumbers(i)
'         dicePic(i).Picture = LoadPicture("C:\Documents and Settings\Scott\My Documents\My Pictures\dice\" & diceVal(i) & ".
'    Next i
```

The above code when activated called an input box for me to enter the dice values. This let me quickly enter the combinations I need to test. The process of running through 115 possible combinations went fairly quickly using this method. After testing I found around 7 different combinations where I had made simple typing errors when copying the code.

Another trick I used was some test code to adjust the score to 9000 per player. This allowed me to test on the winning code without having to play the game all the way through. Believe it or not, but having the correct player win was the most buggy part of the whole program and had to be adjusted several times.

Conclusion

This was a very fun project to work on. It turned out to be a lot more difficult than I originally anticipated, however I am pleased with the results. If I want to play with this program in the future, it is set up so that I can easily make it more customizable and add players names, a high score list, computer players, etc. It is very conceivable that I will return to this project and add a few new layers to improve my VBA skills.

Let me conclude then with a programming joke: Three men were driving together in a car, when they approached a steep hill. As the car went down the hill, the driver lost control and the car went off into the ditch. The first man, who was a mechanical engineer, suggested they pop the hood and see where the car had malfunctioned. The second man, who was an electrical engineer, suggested they check the breaker box to check for electrical failure. The third man, who was a computer programmer, disagreed with his colleagues. He thought they should simply push the car back up the hill and see if it crashed again on the way down!