Rock-Paper-Scissors

Multiple versions

Nested If / Else If / Else

Random Numbers

Topics Covered

- Project Definition
- Multiple versions of the program
- Determine the logic of who wins
- Multiple If statements vs using Nested If statements
- Random numbers
Description of the Game

The players count aloud to three, or speak the name of the game (e.g. "Rock Paper Scissors!" or "Ro Sham Bo!"), each time either raising one hand in a fist and swinging it down on the count or holding it behind. On the third count throwing their gesture ("Rock Paper or Scissors").

- **Rock**, represented by a clenched fist
- **Paper**, represented by an open hand, with the fingers extended and touching, in order to represent a sheet of paper
- **Scissors**, represented by two fingers extended and separated

The objective is to select a gesture which defeats that of the opponent. Gestures are resolved as follows:

- Rock breaks, or crushes scissors: rock defeats scissors.
- Scissors cut paper: scissors defeats paper.
- If both players choose the same, the game is tied and the players throw again.

From Wikipedia

---

Project Definition

Write a program to score the rock-paper-scissor game for two players. Each player clicks an image representing Rock, Paper or Scissors.

The program announces the winner and the basis for determining the winner: Paper covers rock, Rock breaks scissors, Scissors cut paper, or a Tie game.

Enhance the program by letting the computer select a random choice for Rock, Paper or Scissors.
Multiple Versions

- Decide on the **User Interface**
- Separate the user interface from the program logic

Different User Interfaces

1. Command Line Interface (CLI) – It works but it is not very attractive compared to a graphical interface.

2. Graphical User Interface (GUI) – Better than the CLI, but not much. The user still needs to type letters and click a button to see who won the game.

3. Nice images for the player to click. Everything is gray and boring. The player still needs to click a button to see who won. The message identifying the winner is small and hard to read.
Different User Interfaces

Color adds to the visibility of the game. The dark green background is similar to many of the card games on Windows. The winner is identified as soon as a hand is clicked for each player.

A button is added to let the computer choose a random hand. Now you can play against the computer or have the computer play against itself. You can also cheat on the game by seeing what the computer picks before you make your choice.

What else could be done? Get rid of the "New Game" button by having the player click anywhere on the screen. Get rid of the "Exit" button by having the player click the "X" at the top of the screen.

Separating Interface from Logic

Although the user interface has changed from one version of the program to another, the logic for choosing the winner is the same.
Determine the Logic of Who Wins

- Truth Tables
- Decision Charts
- Multiple If Statements

Truth-Table

A Truth-Table is used to simplify the design and to make sure that all conditions are tested.

<table>
<thead>
<tr>
<th></th>
<th>Rock</th>
<th>Player2 Paper</th>
<th>Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player1 Rock</td>
<td>Tie</td>
<td>Player2 wins Paper covers Rock</td>
<td>Player1 wins Rock breaks Scissors</td>
</tr>
<tr>
<td>Player1 Paper</td>
<td>Player1 wins Paper covers Rock</td>
<td>Tie</td>
<td>Player2 wins Scissors cut Paper</td>
</tr>
<tr>
<td>Player1 Scissors</td>
<td>Player2 wins Rock breaks Scissors</td>
<td>Player1 wins Scissors cut Paper</td>
<td>Tie</td>
</tr>
</tbody>
</table>
List All Possible Combinations
Option 1 – Test Each Individually

There are nine conditions to test

<table>
<thead>
<tr>
<th>Player 1</th>
<th>Player 2</th>
<th>Who wins?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td>Rock</td>
<td>Tie – Nobody wins</td>
</tr>
<tr>
<td>Rock</td>
<td>Paper</td>
<td>Player 2 – Paper covers rock</td>
</tr>
<tr>
<td>Rock</td>
<td>Scissors</td>
<td>Player 1 – Rock breaks scissors</td>
</tr>
<tr>
<td>Paper</td>
<td>Rock</td>
<td>Player 1 – Paper covers rock</td>
</tr>
<tr>
<td>Paper</td>
<td>Paper</td>
<td>Tie – Nobody wins</td>
</tr>
<tr>
<td>Paper</td>
<td>Scissors</td>
<td>Player 2 – Scissors cut paper</td>
</tr>
<tr>
<td>Scissors</td>
<td>Rock</td>
<td>Player 2 – Rock breaks scissors</td>
</tr>
<tr>
<td>Scissors</td>
<td>Paper</td>
<td>Player 1 – Scissors cut paper</td>
</tr>
<tr>
<td>Scissors</td>
<td>Scissors</td>
<td>Tie – Nobody wins</td>
</tr>
</tbody>
</table>

Test each of the nine conditions individually, or ...

List All Possible Combinations
Option 2 – Test the nine conditions as three groups of three

<table>
<thead>
<tr>
<th>Player 1</th>
<th>Player 2</th>
<th>Who wins?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td>Paper</td>
<td>Player 2 – Paper covers rock</td>
</tr>
<tr>
<td>Rock</td>
<td>Scissors</td>
<td>Player 1 – Rock breaks scissors</td>
</tr>
<tr>
<td>Paper</td>
<td>Rock</td>
<td>Player 1 – Paper covers rock</td>
</tr>
<tr>
<td>Paper</td>
<td>Paper</td>
<td>Tie – Nobody wins</td>
</tr>
<tr>
<td>Paper</td>
<td>Scissors</td>
<td>Player 2 – Scissors cut paper</td>
</tr>
<tr>
<td>Scissors</td>
<td>Rock</td>
<td>Player 2 – Rock breaks scissors</td>
</tr>
<tr>
<td>Scissors</td>
<td>Paper</td>
<td>Player 1 – Scissors cut paper</td>
</tr>
<tr>
<td>Scissors</td>
<td>Scissors</td>
<td>Tie – Nobody wins</td>
</tr>
</tbody>
</table>
Design the Form

- The figure on the next slide lists most of the controls from the ToolBox and their properties. Controls for Player2 are similar to Player1.
- Get the images from Moodle.
- Make sure the GroupBox is selected when placing pictures on the form.
Check the Names for All Controls

Click the little down-arrow on the line below Properties on the Properties window to get a list of all controls that are on the form and the name for each control.

These are the names used for the rest of the discussion. Use these names if you want your project to match the example code shown on the next several slides.

Get the Images from Moodle

The lab assignment has six images of the different hand formations. Save these images on your desktop or another location on your disk.

As you place a PictureBox on the form, select the image property and import the picture into your program. Refer to the PlayersOnTeam lab assignment to review how to get images into your program.
Code the Program

- Title Block and Global Variables
- Event Handlers for the Picture click events
- Subroutine to test for the winner
  - using nine individual tests, or
  - using three groups of three tests
- The "New Game" and "Exit" buttons

Title Block and Global Variables

`' RockPaperScissors.vb: The program implements the classic game of Rock-Paper-Scissors
' Version: 1.0
' Date 2/4/2015
' Programmer: Dan McElroy
' Class: CIS073 Visual Basic Programming
'
' Inputs: Images for Rock, Paper and Scissors for each player
'        Buttons to let the computer pick for each player
'        Buttons for "New Game" and "Exit"
' Outputs: Message indicates a "Tie" or a "Winner" and the reason why

Public Class Form1

    ' Define the global variables (global means outside the subroutines)
    Dim Player1 As String = String.Empty
    Dim Player2 As String = String.Empty
    Dim rand As New Random

Here comes a little bit of object oriented programming. **New Random** creates a new object from the **Random** class definition. The object being created in this program is named **rand**. It will be used to generate a random number when the player clicks one on the buttons to have the computer choose Rock, Paper or Scissors.
PictureBox Click Event Handlers

Double-click player1's ROCK picture while in the Design view to bring up the code window with an empty event handler for the picture's **Click Event Handler**. Buttons are not the only control that can have a click event handler.

Enter the code for the event handler, then do the same for the other pictures.

```
Private Sub picRock1_Click(sender As Object, e As EventArgs) Handles picRock1.Click

End Sub
```

**Player1 "Rock" Event Handler**

```
Private Sub picRock1_Click(sender As Object, e As EventArgs) Handles picRock1.Click
    Player1 = "Rock"
    picPaper1.Enabled = False
    picScissors1.Enabled = False
    TestForWinner()

End Sub
```

Inside the picRock1.Click event handler, set the **Enabled** property = False for the other two pictures belonging to Player1. Also set the "Let the Computer Pick" button **Enabled** property = False. This will prevent them from being clicked until the "New Game" button is clicked which re-enables all controls.

The **TestForWinner()** subroutine will see if a choice has been made by both Player1 and Player2. If so, then it will determine the winner.

It is your job to determine the correct code for Player1's other two pictures and all three pictures for Player2. In the design mode, double-click each of the pictures and enter the code for their event handlers. Copy and Paste is cool, but go line by line to make sure everything has been changed correctly.
Test for Winner – part 1

The original version of the program had a separate button that would execute the code to test for a winner. When I wanted the program to automatically determine the winner, I moved the code into a separate subroutine named `TestForWinner()` and called the subroutine each time one of the PictureBoxes of a hand was clicked. Example code for player1 "Rock" picture:

```vba
Private Sub picRock1_Click(sender As Object, e As EventArgs) Handles picRock1.Click
    Player1 = "Rock"
    picPaper1.Enabled = False
    picScissors1.Enabled = False
    TestForWinner()
End Sub
```

Test for Winner – part 2

The code for `TestForWinner()` uses a nested `If`. The outer `If` statement just checks to see if a selection has been made for both players.

```vba
Private Sub TestForWinner()
    ' A Selection must be made for both players
    If Player1 <> String.Empty And Player2 <> String.Empty Then
        ' All of the If statements to test for winner go here.
        ' You can either use nine individual If / ElseIf statements or three groups of three If / ElseIf statements to test all nine conditions.
        End If
        End Sub
```

In Visual Basic, The `<>` logical operator means **Not Equal To**. The `If` statement is testing to see if something is in both **Player1** and **Player2**.
Sample ways of coding the program
1. The most important thing is to make sure that all conditions are tested
2. Simplify the code if possible, to make it easier for you and the next person to understand
3. Either is acceptable for the lab assignment

Test for Winner – part 3, pick only one

Nine Individual Tests

```
' Here is sample code to test if Player1 = "Rock"
If Player1 = "Rock" And Player2 = "Rock" Then
    lbWinner.Text = "Tie game"
ElseIf Player1 = "Rock" And Player2 = "Paper" Then
    lbWinner.Text = "Player 2 wins. Paper covers rock"
ElseIf Player1 = "Rock" And Player2 = "Scissors" Then
    lbWinner.Text = "Player 1 wins. Rock breaks scissors"
' This sequence needs to be repeated six more times to test for Player1 = "Paper" and Player1 = "Scissors".

' End IF
```

Three Groups of Three Tests

```
' Here is the code to test if Player1 = "Rock"
If Player1 = "Rock" Then
    lbWinner.Text = "Player 2 wins. Paper covers rock"
ElseIf Player2 = "Paper" Then
    lbWinner.Text = "Player 1 wins. Rock breaks scissors"
ElseIf Player2 = "Scissors" Then
    lbWinner.Text = "Player 1 wins. Rock breaks scissors"
End If
```

Each of these ways to test for a winner is described in more detail
Test for Winner – Nine Individual Tests (part 1 of 2)

Nine Individual Tests

Private Sub TestForWinner()
' A Selection must be made for both players
If Player1 <> String.Empty And Player2 <> String.Empty Then
    ' Here is sample code to test if Player1 = "Rock"
    If Player1 = "Rock" And Player2 = "Rock" Then
        lblWinner.Text = "Tie Game"
    ElseIf Player1 = "Rock" And Player2 = "Paper" Then
        lblWinner.Text = "Player 2 wins. Paper covers rock"
    ElseIf Player1 = "Rock" And Player2 = "Scissors" Then
        lblWinner.Text = "Player 1 wins. Rock breaks scissors"
    End If
End If
End Sub

Test for Winner – Nine Individual Tests (part 2 of 2)

Look Closely at the If Statement

If Player1 = "Rock" And Player2 = "Rock" Then

True or False?  True or False?

Each side of the And must evaluate to True or False.
A statement like:

If Player1 And Player2 = "Rock" Then

may be good English, but it is not a valid statement because Player1 is a string data type not a True/False value.
Test for Winner – Three Groups of Three

Three Groups of Three Tests

Private Sub TestForWinner()
' A Selection must be made for both players
If Player1 <> "String.Empty And Player2 <> "String.Empty Then
    ' Here is the code to test if Player1 = "Rock"
    If Player1 = "Rock" Then
        If Player2 = "Rock" Then
            lblWinner.Text = "Tie Game"
        ElseIf Player2 = "Paper" Then
            lblWinner.Text = "Player 2 wins. Paper covers rock"
        ElseIf Player2 = "Scissors" Then
            lblWinner.Text = "Player 1 wins. Rock breaks scissors"
        End If
    ElseIf Player1 = "Paper" Then
        ' This sequence needs to be repeated to test
        ' for Player1 = "Paper" and Player1 = "Scissors". The sequence
        ' of IF/ElseIf statements must be terminated with the End If
        End If
End If
End If
End Sub

The New Game and Exit Buttons

Private Sub btnNewGame_Click(sender As Object, e As EventArgs) Handles btnNewGame.Click
    ' new game. Clear the label that shows who won, re-enable buttons
    Player1 = "String.Empty"
    Player2 = "String.Empty"
    lblWinner.Text = "String.Empty"
    picRock1.Enabled = True
    picPaper1.Enabled = True
    picScissors1.Enabled = True
    picRock2.Enabled = True
    picPaper2.Enabled = True
    picScissors2.Enabled = True
End Sub

Private Sub btnExit_Click(sender As Object, e As EventArgs) Handles btnExit.Click
    Me.Close() ' end the program
End Sub
Debug and Test

- Clean up any syntax errors that were detected by the compiler
- Test all nine conditions

Enhance the Program
Let the Computer Pick

AFTER you have verified that the program works properly for all nine conditions of Rock, Paper and Scissors, you can add two more buttons to let the computer choose a random value.
Add Two More Buttons

Random Pick for Player1

Private Sub btnRandomPlayer1_Click(sender As Object, e As EventArgs) Handles btnRandomPlayer1.Click
    Dim ComputerChoice = rand.Next(3) ' returns 0, 1 or 2
    If ComputerChoice = 0 Then
        picRock1_Click(sender, e) ' activate Player 1's Rock picture click event subroutine
    ElseIf ComputerChoice = 1 Then
        picPaper1_Click(sender, e) ' activate Player 1's Paper picture click event subroutine
    Else
        picScissors1_Click(sender, e) ' activate Player 1's Scissors picture click event subroutine
    End If
End Sub

In this example, the **rand.Next** function is called with one numeric value inside the parentheses. **rand.Next(3)** can return three different values: 0, 1 or 2. The **If** and **ElseIf** and **Else** statements are used to turn the 0, 1 or 2 into selecting rock, paper or scissors. Only one piece of code will be executed in the **If / ElseIf / Else** block.
Simulate a Picture Being Clicked

The computer’s random choice is implemented by simulating one of the picture's click events. For example if `rand.Next(3)` returns a value of 1 then the program calls the `picPaper1_Click` subroutine. The values in `sender` and `e` contain information about which control (button or picture) was originally clicked.

```vbnet
Private Sub btnRandomPlayer1_Click(sender As Object, e As EventArgs) Handles btnRandomPlayer1.Click
    Dim ComputerChoice = rand.Next(3) ' returns 0, 1 or 2
    If ComputerChoice = 0 Then
        picRock1_Click(sender, e) ' activate Player 1's Rock picture click event subroutine
    Else
        picPaper1_Click(sender, e) ' activate Player 1's Paper picture click event subroutine
    End If
End Sub
```

Also add the `btnRandomPlayer1.Enabled = False` to each event handler for Player1 Paper and Scissors pictures.

---

Do the Same for Player2

Double-click the second "Let the Computer Pick" button and enter the code for the random choice for Player2.

```vbnet
Private Sub btnRandomPlayer2_Click(sender As Object, e As EventArgs) Handles btnRandomPlayer2.Click
    Dim ComputerChoice = rand.Next(3) ' returns 0, 1 or 2
    If ComputerChoice = 0 Then
        picRock2_Click(sender, e) ' activate Player 2's Rock picture click event subroutine
    Else
        picPaper2_Click(sender, e) ' activate Player 2's Paper picture click event subroutine
    End If
End Sub
```

Also add the `btnRandomPlayer2.Enabled = False` to each event handler for Player2 Paper and Scissors pictures.
Update the **New Game** Subroutine

Update the **New Game** button event handler to re-enable the "Let the Computer Pick" buttons

```vbnet
Private Sub btnNewGame_Click(sender As Object, e As EventArgs) Handles btnNewGame.Click
    ' new game. Clear the label that shows who won, re-enable buttons
    Player1 = String.Empty    ' clear the player selections
    Player2 = String.Empty
    lblWinner.Text = String.Empty    ' clear the label
    picRock1.Enabled = True    ' re-enable all the pictures
    picPaper1.Enabled = True
    picScissors1.Enabled = True
    picRock2.Enabled = True
    picPaper2.Enabled = True
    picScissors2.Enabled = True
    btnRandomPlayer1.Enabled = True    ' re-enable the buttons
    btnRandomPlayer2.Enabled = True
End Sub
```

---

**Most Common Error**

If a PictureBox was double-clicked during the design mode before its name property was set, the subroutine may be named **Private Sub** `PictureBox1_Click (sender ...` instead of **Private Sub** `picRock1_Click (sender ...` If this happened, just rename the subroutine but don't change anything else on the line.

```vbnet
Private Sub btnRandomPlayer1_Click(sender As Object, e As EventArgs) Handles btnRandomPlayer1.Click
    Dim ComputerChoice As Integer = rand.Next(3)    ' returns 0, 1 or 2
    If ComputerChoice = 0 Then
        picRock1_Click(sender, e)    ' activate Player 1's Rock picture click event subroutine
    Else If ComputerChoice = 1 Then
        picPaper1_Click(sender, e)    ' activate Player 1's Paper picture click event subroutine
    Else
        picScissors1_Click(sender, e)    ' activate Player 1's Scissors picture click event subroutine
    End If
End Sub
```

```vbnet
Private Sub picRock1_Click(sender As Object, e As EventArgs) Handles picRock1.Click
    Player1 = "Rock"    ' Save Player1's choice for this game
    picPaper1.Enabled = False    ' Disable the PAPER picture for Player1
    picScissors1.Enabled = False    ' Disable the SCISSORS picture for Player1
    btnRandomPlayer1.Enabled = False    ' Disable the "LET THE COMPUTER PICK"
    TestForWinner()    ' See if both Player1 and Player2 have chosen
End Sub
```
Re-test the Enhanced Program

- Clean up any syntax errors that were detected by the compiler
- Test combinations of clicking pictures and buttons

```
<table>
<thead>
<tr>
<th></th>
<th>Rock</th>
<th>Paper</th>
<th>Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player1</td>
<td>Tie</td>
<td>Player2 wins Paper covers Rock</td>
<td>Player1 wins Rock breaks Scissors</td>
</tr>
<tr>
<td>Paper</td>
<td>Player1 wins Paper covers Rock</td>
<td>Tie</td>
<td>Player2 wins Scissors cut Paper</td>
</tr>
<tr>
<td>Scissors</td>
<td>Player2 wins</td>
<td>Player1 wins Scissors cut Paper</td>
<td>Tie</td>
</tr>
</tbody>
</table>
```

Credits

Diagram by Enzoklopp Wikipedia

Photos by Fluff, modified by Certion Wikipedia