Ruby

Jovana Banovic

Ruby

Ruby is a programming language from Japan. Why i choose Ruby is because of its simplicity . Even though it is a simple language it is powerful as well.

1. Intro

Problem:

Create a TIcTacToe Game :

- Create a Board
- Determine all the Win possibilities
- Determine who is going to play
- As the game progressed recreate the board
- Final messages of who won

```ruby
class Board

	def initialize
		@board= Array.new(3) { Array.new(3," ") }
	end

	def defineframe
		puts "1 | 2 | 3 |",
			"4 | 5 | 6 |",
			"7 | 8 | 9 |"
		print "\n"
	end
```

First Function Initialize and Board Frame

This TicTacToe game is a game between you and a friend

2. Functions

After having a good idea of how the board will look like i proceeded to make a function if a tie occurs,

This was a simple function.

However all the possible wins was the most challenging function since there are multiple ways a player can win

game - Tic ... | CSCI 3055U... | project-exa... | Assignment... | project-exa... | Branches · ... | ruby - Goog...

GitHub, Inc. (US) https://github.com/banojov/FinalProject/blob/master/board.rb

```ruby
class Board

  def initialize
    @board = Array.new(3) { Array.new(3, " ") }
  end

  #how the bord will look like
  def defineboard
    puts "1 | 2 | 3",
         "4 | 5 | 6",
         "7 | 8 | 9"
    print "\n"
  end

  #the function if a tie happens
```

The board function was later on changed to look more like the following piece of code

```ruby
#print board after each of the players choose their "X" or "O" move
def print_Board
  (0...3).each do |row|
    print "       "
    (0...3).each do |col|
      print @board[row][col]
      print " | " unless col == 2
    end
    print "\n"
    print  unless row == 2
  end
  print "\n"
end
# all the possible wins
def find_Winner
  #diagonal wins
   if ( @board[0][2] == @board[1][1] && @board[1][1] == @board[2][0] ) ||
     ( @board[0][0] == @board[1][1] && @board[1][1] == @board[2][2] )
    return @board[1][1] unless @board[1][1] == " "
  end
  #wins on the sides
      (0...3).each do |i|
    if @board[i][0] == @board[i][1] && @board[i][1] == @board[i][2]
      return @board[i][0] unless @board[i][0] == " "

    elsif @board[1][i] == @board[2][i] && @board[0][i] == @board[1][i]
      return @board[0][i] unless @board[0][i] == " "
    end
  end
  # the game is tied
```

All the possible win instances

Source: theguardian.com

```ruby
#let the game begin
board = Board.new
currentplayer = "X"

puts "\n" * 100
board.defineboard

while board.find_Winner == "U"

  puts " #{currentplayer}'s turn. Choose a Number!"

  moves = gets.chomp.to_i - 1
  row = moves / 3
  col = moves % 3

  if board.remove(currentplayer, row, col)
    if currentplayer == "X"
      currentplayer = "O"
    else
      currentplayer = "X"
    end
  else
    puts "Invalid move, please select again\n\n"
  end

  board.print_Board
end

winner = board.find_Winner

  if winner == "C"
    puts "   PLAYER C WON G A M E"
```

Implementation of board

Definition of X and O

Source: theguardian.com

Terminal - jovana@jovana-E5450: ~/Desktop/FinalProject

File Edit View Terminal Tabs Help

```
1
       X |   |
         |   |
         |   |

 O's turn. Choose a Number!
2
       X | O |
         |   |
         |   |

 X's turn. Choose a Number!
3
       X | O | X
         |   |
         |   |

 O's turn. Choose a Number!
4
       X | O | X
       O |   |
         |   |

 X's turn. Choose a Number!
5
       X | O | X
       O | X |
         |   |

 O's turn. Choose a Number!
6
       X | O | X
       O | X | O
         |   |

 X's turn. Choose a Number!
7
       X | O | X
       O | X | O
       X |   |

     X ' S   W I N

       X | O | X
       O | X | O
       X |   |

jovana@jovana-E5450:~/Desktop/FinalProject$
```

Source: theguardian.com

Code in action

```ruby
def merge_sort(lists)
  return lists if lists.size <= 1

 middle = lists.size / 2
 left = lists[0, middle]
 right= lists[middle,lists.size]
 merge(merge_sort(left),merge_sort(right))
end

 def merge(left,right)
    sorted=[]
    if left.first <= right.first
        sorted << left.shift
    else
        sorted <<right.shift
    end
end
sorted.concat(left).concat(right)
merge_sort [1,2,3,4,5,6,7,8]
end
```

Ruby Merge Sort

```
public void MergeSort(int lowerIndex,int highgerIndex){

        if(lowerIndex<highgerIndex){

        int middle=lowerIndex+(highgerIndex-lowerIndex)/2;
        //sorts the left side of the array

        MergeSort(lowerIndex,middle);
        //right side

        MergeSort(middle+1,highgerIndex);
        //merge both sides

        mergeParts(lowerIndex,middle,highgerIndex);
        }
        }

        public void mergeParts(int lowerIndex, int highgerIndex, int
middle ){

        for(int i=lowerIndex;i<=highgerIndex;i++){
        tempMergArr[i]=array[i];
        }

        int i=lowerIndex;
        int j=middle+1;
        int k=lowerIndex;

        while(i<=middle && j <= highgerIndex){

        if(tempMergArr[i] <= tempMergArr[j]){

        array[k]=tempMergArr[i];
        i++;

        }else{

        array[k]=tempMergArr[j];
        j++;
        }
        k++;

        }

        if(i<= middle){

        array[k]=tempMergArr[i];
        k++;
        }
        }
}
```

Java Merge Sort

4. Closing

- We cannot mention Ruby without mentioning Ruby on Rails
- https://www.youtube.com/watch?v=OaDhY_y8WTo&feature=youtu.be
- A good intro to ruby on rails
- Ruby on Rails, is a server-side web application framework written in Ruby under the MIT License. Rails is a model–view–controller framework that provides default structures for a database, a web service, and web pages.

https://www.tutorialspoint.com/ruby-on-rails/rails-examples.htm