Who Wants to be a Millionaire

Terminal App

Daniel Hinton

Features

- Simulates the experience of being a contestant on the game show Who Wants to be a Millionaire
- Includes 130 multiple-choice questions in 15 levels of difficulty
- Money tree true to the show includes safe levels at $1,000 and $10,000
- Lifelines - Ask the Audience, Phone a Friend and 50/50 all available in the game
- Virtual host - A virtual Eddie McGuire guides you through your game, with random quotes pulled straight from the man himself after each question you get correct
- Ascii art - Used in the load screen, cheque (if you win money during the game) and if you win the million dollars

Playthrough

Code

Question Bank

Each question is an array in the form: Question, possible answers, correct answer, ask the audience result and phone a friend result

```
q1 = ["In the 1960s, The Righteous Brothers had a number one hit with 'You've Lost That' What",
"Lovin' Feeling", "Caring Sensation", "Tender Heart", "Credit Card Again", "A", 84, 7, 6, 3, phoneafriend1("A")]

q2 = ["In a stadium or arena, the seating area farthest and highest from the stage is known as the what 'section'",
"Nosebleed", "Headache", "Eyestrain", "Crook Neck", "A", 88, 6, 4, 2, phoneafriend2("A")]

q3 = ["In basic mathematics, a whole non-negative number is known as a what number",
"Natural", "Biodynamic", "Organic", "Gluten-free", "A", 76, 13, 11, 0, phoneafriend1("A")]

q4 = ["First airing in 2005 was the long running Australian TV series 'McLeod's' what",
"Daughters", "Uncles", "Sons", "Neighbours", "A", 95, 1, 1, 3, phoneafriend2("A")]

q5 = ["'What's up Doc?' is a famous catchphrase of which character",
"Elmer Fudd", "Bugs Bunny", "Daffy Duck", "Mr Hyde", "B", 14, 66, 18, 2, phoneafriend1("B")]

q6 = ["Born in Philidelphia in 1861, William Wrigley Jr is best known for selling what",
"Baseball Bats", "Sleeping Bags", "Chewing Gum", "Snakes", "C", 9, 4, 85, 2, phoneafriend2("B")]

q7 = ["To be highly proficient at something is to 'have it down to a' what",
"Political science", "Foreign language", "English major", "Fine art", "D", 3, 1, 7, 89, phoneafriend1("B")]

q8 = ["In the classic nursery rhyme 'This Little Piggy', the third little piggy did what",
"Stayed home", "Had roast beef", "Went to market", "Had none", "B", 22, 37, 19, 22, phoneafriend3("C")]

q9 = ["Which of these is a common term that describes a light snowfall",
"Sweeping", "Dusting", "Mopping", "Vacuuming", "B", 7, 79, 11, 3, phoneafriend1("B")]

q10 = ["The largest portion of something is commonly known as 'the what share'",
"Shark's", "Pig's", "Tiger's", "Lion's", "D", 7, 11, 9, 73, phoneafriend1("D")]

q11 = ["Which of these is the correct spelling for a word meaning a deep shaft, particularly one for drawing water",
"Bore", "Baw", "Boar", "Boor", "A", 68, 7, 19, 6, phoneafriend3("A")]

q12 = ["What colour was the ball used in the first day-night test cricket match",
"Yellow", "Red", "White", "Pink", "D", 5, 13, 35, 47, phoneafriend2("D")]
```

Code

Generating a question stack

- Questions are placed into their appropriate difficulties
- The question stack generator function takes a random question from each difficulty and creates a list of questions for each new game

```
# Placing the questions into their difficuly categories:

$questions100 = [q1, q2, q3, q4, q5]
$questions200 = [q6, q7, q8, q9, q10]
$questions300 = [q11, q12, q13, q14, q15]
$questions500 = [q16, q17, q18, q19, q20]
$questions1000 = [q21, q22, q23, q24, q25]
$questions1500 = [q26, q27, q28, q29, q30]
$questions2500 = [q31, q32, q33, q34, q35]
$questions4k = [q36, q37, q38, q39, q40]
$questions6k = [q41, q42, q43, q44, q45]
$questions10k = [q46, q47, q48, q49, q50]
$questions20k = [q51, q52, q53]
$questions50k = [q54, q55, q56]
$questions100k = [q57, q58, q59]
$questions250k = [q60, q61, q62]
$questions1million = [q63, q64, q65]

# Generating a new, random question stack for each game:

module_function
def questionstackgenerator
  questionstack = []

  questionlist = [$questions100, $questions200, $questions300, $questions500,
    $questions1000, $questions1500, $questions2500, $questions4k, $questions6k,
    $questions10k, $questions20k, $questions50k, $questions100k, $questions250k, $questions1million]

  for question in questionlist
    questionstack << question.sample
  end

  return questionstack
end
```

Code

Displaying questions

The app essentially just loops though each question in the question stack. If the user gets the answer wrong or walks away, the loop breaks.

```ruby
def initialize(questionstack, questioncounter)
  system('clear')
  puts "Ok, let's play Who Wants to be a Millionaire!"
  sleep(2)
  for question in questionstack
    questionfunc(question, questioncounter)
    questioncounter += 1
  end

end

# This function runs for each question in the question stack
def questionfunc(question, questioncounter)
  system('clear')

  # Question displayed on the screen
  puts "Question #{questioncounter} is for #{$moneytree[questioncounter]}"
  sleep(1)
  Functions::moneytree(questioncounter)
  sleep(2)
  puts "\n" + question[0] + "?"
  sleep(4)
  print "A: ".colorize(:red)
  puts question[1]
  sleep(2)
  print "B: ".colorize(:red)
  puts question[2]
  sleep(2)
  print "C: ".colorize(:red)
  puts question[3]
  sleep(2)
  print "D: ".colorize(:red)
  puts question[4]
  sleep(1)
  puts "\n1: Lifeline"
  puts "2: Walk Away"
```

Code

User response to question

The user has the choice of:

- Answering the question
- Walking away
- Using a lifeline

```ruby
# The user gives their answer here
loop do
  print "\nANSWER: "
  answer = gets.chomp.capitalize

  if questioncounter >= 8 && (answer == "A" || answer == "B" || answer == "C" || answer == "D")
    answer = Functions::answerchecker(answer)
  end

  if answer == question[5] # Correct answer
    Functions::millionwin(answer) if questioncounter == 15
    puts "\n#{answer} is Locked in..."
    sleep(2)
    puts "\n" + EddieLines::eddieCorrectAnswer + "!"
    sleep(2)
    puts "\nYou've won #{$moneytree[questioncounter]}"
    sleep(2)
    puts "\n" + EddieLines::eddieQuips + "!"
    sleep(2)
    puts "\nPress ENTER for the next question"
    gets
    break
  elsif answer == "1" # User wants to use a lifeline
    Functions::lifelines(question)
  elsif answer == "2" # User opts to walk away
    puts "\nWalk Away"
    sleep(1)
    prize = $moneytree[questioncounter-1]
    puts "\nCongratulations, you walk away with #{prize}!"
    sleep(2)
    username = $username
    Functions::cheque(username, prize)
    sleep(2)
    puts "\nThanks for playing Who Wants to be a Millionaire!"
    exit
  elsif answer == "A" || answer == "B" || answer == "C" || answer == "D" #Incorrect answer
    puts "\n#{answer} is Locked in..."
    sleep(2)
```

Code

User response to question

The user has the choice of:

- Answering the question
- Walking away
- Using a lifeline

```
    if questioncounter < 5
      prize = "0"
    elsif questioncounter < 10
      prize = "1,000"
    else
      prize = "10,000"
    end
    array1 = ["\nI'm sorry, that's the wrong answer!",
    "The correct answer was #{question[5]}",
    "You leave with $#{prize}",
    "Thanks for playing!"]
    Functions::sleeplines(array1)
    username = $username
    Functions::cheque(username, prize) unless questioncounter < 5
    exit
  else
    puts "Error: Invalid response"
  end
end
```

Code

Lifelines

- The user can only use a lifeline if they haven’t already used it in that game
- If they have used all three lifelines already, they get the error message at the bottom
- All lifelines use the gem tty-spinner for visual effect

```ruby
# User decides which lifeline to use when opting to use a lifeline
module_function
def lifelines(question)
  if $asktheaudience == false && $phoneafriend == false && $fiftyfifty == false
    puts "\nI'm sorry, you're out of lifelines"
    return nil
  else
    puts "\nOkay, you're going to use a lifeline."
    sleep (1)
    puts "\nThese are the lifelines you still have available:"
    puts "1. Ask the Audience" if $asktheaudience
    puts "2. Phone a Friend" if $phoneafriend
    puts "3. 50/50" if $fiftyfifty
    puts "4. Return to question"
  end

  loop do
    lifeline = gets.chomp

    case lifeline
    when "1"
      asktheaudience(question)
      break
    when "2"
      phoneafriend(question)
      break
    when "3"
      fiftyfifty(question)
      break
    when "4"
      break
    else
      puts "\nI'm sorry, That's not a valid answer. Please enter a valid input"
    end
  end
```

Code

Ask the Audience

- Displays the poll results coded in the question
- Uses the tty-pie gem to display the results as a pie chart

```ruby
# Ask the Audience lifeline
module_function
def asktheaudience(question)
  if $asktheaudience == true
    puts "\nOk, audience, buzzers at the ready, vote now!"
    spinner = TTY::Spinner.new(format: :pulse_2)
    spinner.auto_spin
    sleep(3)
    spinner.stop
    data = [
      {name: 'A', value: question[6]*100, color: :bright_yellow, fill: '*'},
      {name: 'B', value: question[7]*100, color: :bright_green, fill: 'x'},
      {name: 'C', value: question[8]*100, color: :bright_magenta, fill: '@'},
      {name: 'D', value: question[9]*100, color: :bright_cyan, fill: '+'}
    ]
    pie_chart = TTY::Pie.new(data: data, radius: 5)
    puts "Results:"
    puts pie_chart
    sleep(1)
    $asktheaudience = false
  else
    puts "I'm sorry, you've already used Ask the Audience"
  end
end
```

Code

Phone a Friend

Eight different phone a friend results possible - from 100% certainty to no idea

```ruby
#Phone a Friend lifeline
module_function
def phoneafriend(question)
  if $phoneafriend == true
    puts "\nOk, you're going to phone a friend. Dialling now..."
    spinner = TTY::Spinner.new(format: :classic)
    spinner.auto_spin
    sleep(3)
    spinner.stop
    puts "Hello?"
    sleep(1)
    puts question[10]
    sleep(1)
    $phoneafriend = false
  else
    puts "\nI'm sorry, you've already used Phone A Friend"
  end
end
```

Code

Phone a Friend

Eight different phone a friend results possible - from 100% certainty to no idea

```ruby
# There are the different phone a friend functions
# Each question has one of these hard coded in if the user wants to phone a friend for that question

module_function
def phoneafriend1(answer)
  "Ahh, that's easy, the answer is #{answer}"
end

module_function
def phoneafriend2(answer)
  "Yep, I'm 90% sure the answer's #{answer}"
end

module_function
def phoneafriend3(answer)
  "Pretty confident. I think it's #{answer}"
end

module_function
def phoneafriend4(answer)
  "Not 100% sure but I have an inklink it's #{answer}"
end

module_function
def phoneafriend5(answer)
  "I think it's #{answer}. But it could also be D"
end

module_function
def phoneafriend6(answer)
  "Boy, really really not sure. If I had to guess I'd say #{answer}"
end

module_function
def phoneafriend7
  "Boy, really really not sure. If i had to guess I'd say C"
end
```

Code

50/50

- 50/50 is the simplest of the lifelines
- It leaves A and B if the answer is A or B and leaves C and D if the answer is C or D

```ruby
# 50/50 lifeline
module_function
def fiftyfifty(question)
  if $fiftyfifty == true
    puts "\nOkay, 50/50. Computer, take away two wrong answers leaving the right answer and one remaining wrong answer"
    spinner = TTY::Spinner.new(format: :dots)
    spinner.auto_spin
    sleep(3)
    spinner.stop
    if question[5] == "A" || question[5] == "B"
      puts "\nThe remaining answers are A and B"
    else
      puts "\nThe remaining answers are C and D"
    end
    sleep(1)
    $fiftyfifty = false
  else
    puts "\nI'm sorry, you've already used 50/50"
  end
end
```

Code

End of game

- At the end of each game, the user is thanked for playing and presented an ascii art cheque if they have won any money
- The game then exits back to the Terminal

```
array1 = ["\nI'm sorry, that's the wrong answer!",
"The correct answer was #{question[5]}",
"You leave with $#{prize}",
"Thanks for playing!"]
Functions::sleeplines(array1)
username = $username
Functions::cheque(username, prize) unless questioncounter < 5
exit
```

Code

Ascii Art

Ascii art is used for the load screen, a cheque displayed when you win money at the end of a game, and if you win the million dollars

```
module AsciiArt

  class AsciiArt

    def loadscreen
      puts "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM00000000000000000008MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMZZOZD000:~DI8D8~D8DI+~8808008MMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMM0ZZZ:70Z,:80D8+~D80:888I?~8808~~=880DMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMM7$Z$,.8$0ZIZ,~8I800000000??88800:88~~~888MMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMM$$777I..8.70Z80ZZZZ0000000000000008D88~88~88888MMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMM$7$.$7777,$$$$$$~~MNMNMMM$MMMMMMM=~00000$=~8888~088MMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMN77.?$$7II7777~?M8MMZ=Z$Z7$$$Z0080DMMMMZ~0000000:=7=88NMMMMMMMMMMMMMM
MMMMMMMMMMM77?7I.:.8III~NMMM=????+?MDMMMNMI?$IOZ$$0MMN~0008~D=I8888MMMMMMMMMMMMM
MMMMMMMMM8$77=.II$7II~MMZ+++I?M8MNMM$7MMM07M$$$ZM$7I?IIZN=Z00088=~+88DMMMMMMMMMMM
MMMMMMMM$$$..?$?7II:MM?+++M$$M7M$NMIN8NNMI0$7N7MMM7NM7+?+MM~008~=Z:I888MMMMMMMMMM
MMMMMMMZZ.$70.7II=M7=++=?MMM7M7M7INNNNMNN$$8N7NMNN7MMI$=++=IN=808=888888MMMMMMMM
MMMMMMZZ.0.77777DM+===MMNMMNN$IMNMINNDONMNDNZMNNNONNNIMMM=++=MN800088888MMMMMMM
MMMMMZOZ.D8?$7~M0===DMMMMID$7I7INOND8N?N?DNNNNNNN7NIIIMMMM0=++8N~00=I=I788MMMMMM
MMMMZZ?7.,+$$~MM~~?M$$MN7$$INNNIIINM7NNDNNMDNIDNNN?IMMNNNIINZ++MM=000+78888MMMMM
MMMD0ZZZZ0Z$+MM~~+MMMM7?$NNNDM$877IDZND?DNZNN?NZ777NMND?ZMMMMZ=+$M~008888888MMMM
MMM000ZZZZZ~MM:~$M?NNNN7N=NINN$D$$7NZNDMDD7NZZZI$NN?N+N$$7NMIM7==MN=00008888MMMM
MM00000ZZZZMM$~MMMII7NNN$NN0=NDZMNZ$8DMDDM7$ZDN0DN$$ND7NNZZMMMMM=+NM+00088888MMM
MM08+00ZZ0MM$:8IMMNN?I$D7DNDDMNMDIDZZD$D$I7DD$DNMDDDD87DN?Z$NMM?M==MD80080+$DMMM
M87=+=000~I$~IMMD?$DNIIINNN$DDDD$D$N00DM8I7NDDZNNDNINNNNNNNI$IONM===+=008II788MM
M00=80000MMIZNMMMNNINNI$ZDNNNDI8N8?808Z87I7DID$8+DNIND??M7IIIIMMIO~~MM80088888MM
M0000000~M7~Z$7NNNNNN7M$ZNZD0780000D$I00II+8700ZDIIDD0$77?DNN7ZM+M=0=N~0088888NM
80000000~0~~MZ$NN?NNNNMNZZ?8DD?87000ID?IID+I007DZDD0$$7INNNNM?N8MMN=0$?00888888M
7$$$$000ZZ0$MZZ87ZZ7INDD0ZZNDI87$ZOZ$I?7I$?ZODDIDI$Z$IDMMZ+DMMMN$MM=$MD8DD88DN8M
0...000$...7M=.77,.7NND$8.$I7IIIM.M$:..D..~?7..I07.8DDZ,.8MZ7.77.,$$?.,7..$$ZZ8M
0..,.8Z....7M~.7N,.7MNNDN.$?D88$M.M7.$8880.?7.,,77.?7=,,:.M?I.$7..0+$..7.:??,88M
0..$....I..7M~.7N,.7IMNNI.$D7DDIM.M$.78887.N7.7M...NNI.=+.,7I.$7. 0..MM7..MI:88M
8..70.:IM..7$~.7+,.0000DI.$0088IM.MI:. $ .~=7.II$..I$.MMMI.$7.$7..M?,.M7..77778M
8088D8IMMD88ND0DMNDDDDDDN$DDDDDZ8DN0?$MNMM7Z?M0880MDZMDDDMMMNM7$MMM~NMMMMMMMMI8M
88888800~0Z+MMMM87MNNND77$$$7NNN?00IDI7?I7ND00780M?$$7ZNNNNNNIMM$ZN~~N+08888888M
N8888880~M$$+MZMN+NNNI77$$+DD?7000$NI7?708787000Z$DD7NZ$NNNNNNMM?$$~~N~8888888NM
M88888888MM++Z7MN7?II7NNZMN7?DN78D+87II8I0$8IDD7NDI8DM0$7NN?NNMMMZZMN88888?88MM
M88I7I888~Z=++MNM877NNN?NNN8?NND~D=I7I8?DZZ=7D78DD$MDNNII?N8I$MMM=:ZI=888I7I78MM
MMD$IN8888MM++Z0MMN$$7MN08NNN7ID8D7D7IND8DZZ$DDD7+DDDNINZI?NNMM$N~ZMN88888788MMM
```

Code

Virtual Host

- The virtual host has some lines hard coded at the start of the game like prompting the user for their name
- There is also two arrays of lines the host will pick from randomly each time the user gets a question correct

```
module EddieLines

  module_function
  def eddieQuips
    ["Great work",
    "You're going beautifully",
    "Great stuff",
    "That's the way",
    "Well played",
    "Well done",
    "Let's keep going",
    "Fantastic",
    "That's the stuff",
    "Yes",
    "There you go",
    "Good on you",
    "Keep it up",
    "Bravo",
    "You're getting going now",
    "Woohoo",
    "Too good",
    "You must know your stuff"].sample
  end

  module_function
  def eddieCorrectAnswer
    ["That's right",
    "Correct",
    "That's the right answer"].sample
  end

end
```