CSCI 1301 - Lab 20

Clément Aubert

March 22, 2018

Part 0 - Another Look at the Code From the Lecture

You can have a look at the code presented during the lecture, at http://spots.augusta.edu/caubert/teaching/2018/spring/csci1301/lab/20/2018_03_22.zip, and make sure you understand it before moving on.

Part I - Problem-Solving 1: A Guessing Game

We will revisit our guessing game from Lab 13 – Part II. Read carefully the challenge below, and think about the parts of your previous program you can re-use to solve this one.

Write a program that asks the user which level of difficulty (s)he wants to play: easy, medium or hard. Then, generate a random number: between 0 and 10 for easy, between 0 and 50 for medium, and between 0 and 100 for hard. Then, let the user try to guess the number you randomly generated:

- if the number given by the user is less than the random number generated, your program should display a message saying that the user should try with a greater value,
- if the number given by the user is greater than the random number generated, your program should display a message saying that the user should try with a lesser value,
- and if the number given by the user is the same as the random number generated, your program should display a message saying that the user won, and how many attempts it took him/her to guess correctly.

Part II - Problem-Solving 2: A Collision!

Two trains are on the same track at opposite ends, but traveling toward each other. Ask the user to enter the starting position and speed (in MPH) of each train. Display each train's position every hours until they have collided. The starting time is 0 and goes up by 1's.

You don't have to calculate the exact collision point, just stop displaying messages after the collision.

Part III - Pushing Further (Optional)

a. Go back to Part II, and calculate the exact time of the collision and display the distance the first train traveled using:

$t = (p_2 - p_1)/(s_1 + s_2)$

$d = p_1 + (s_1 \times t)$

where

- t is the time where the collision happened,
- p_i is the initial position of the ith train,
- s_i is the speed of the ith train
- d is the distance traveled by the first train before the colision.

b. Think about the "bowling" program we wrote in class. What if we were to play a game where there is a **fixed** number of turns, total: that is, a game where there are 10 frames, no matter the number of players. If we had 3 players, the order would be $p_1, p_2, p_3, p_1, p_2, p_3, p_1, p_2, p_3, p_1$, i.e., player 1 would play 4 times, and players 2 and 3 would play 3 times. Can you write a program that would display at the screen whose turn it is, in a game where the number of frames is independent of the number of players?