Rock Paper Scissors Project Report

Name: Joe Sample
SID: 12345687

Introduction

Rock, Paper, Scissors is a game that involves two players, each deciding on a hand gesture that is revealed simultaneously. In this project, the task was to create a computer game that enabled a human player to play a game of Rock, Paper, Scissors with a computer.

Design

The flow of the game can be described using the flowchart shown in Figure 1. This flowchart was implemented in the main function in the file *rps.c* found in Appendix A. Here, it should be noted that it is important for the computer to choose its gesture first. If the human gesture was obtained first, it would be unfair because the computer would be able to choose their gesture based on the human gesture.

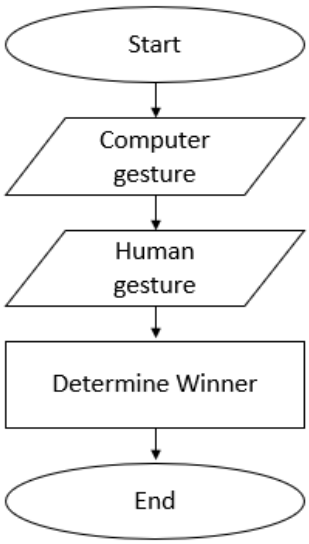

Figure 1 Flowchart describing the flow of the game.

For each of the components in the flowchart shown in Fig. 1, I have converted them into a separate function. These functions are implemented in the file *game_elements.c*.

To make it easier, I have abstracted the gestures rock, paper, scissors to the numbers 0, 1 and 2. This allows me to use the random number generator to produce the computer's gesture.

In the following, I will describe the implementation of each function and how they were tested.

Function: get_computer_gesture()

For this function, I used a random number generator to obtain a number between 0 and 2. To test this code, I ran it 50 times to check that it would produce 0, 1 and 2 at random and that no numbers outside this range was produced. The code I used for testing and output of my tests can be found in Appendix B.

Function: get_human_gesture()

The flowchart describing the code can be found in Fig. 2. For this function, I used the get_int() function from cs50.h to prompt and obtain the human's player gesture. I then used an if-statement to test the input to check that it is valid. If the input was outside the valid range, I let the human player know that the gesture was invalid and prompt again for a gesture. Otherwise, the function would end.

Unfortunately, I couldn't get the code working and kept getting errors. Fig. 3 shows a screen capture of the errors. If I had more time, I think I would need to understand how to write a while loop correctly.

I also didn't get a chance to test this code but if I had the chance, I would have manually tested the following cases to see if everything was working properly:

(1) A number within the range
(2) A negative number
(3) A number greater than 2

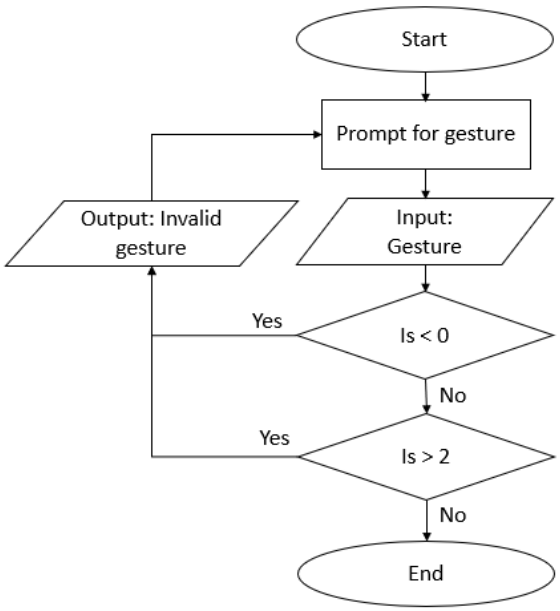

Figure 2 Flowchart describing the function get_human_gesture().

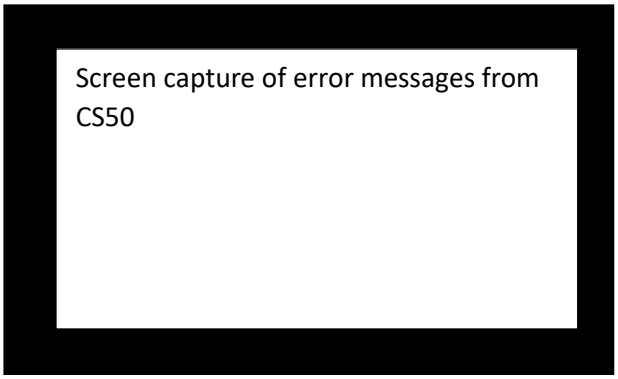

Figure 3 Screen capture of outputs from testing get_human_gesture().

Function: determine_winner ()

The determine winner function is described by the flowchart shown in Fig. 4. This function ret

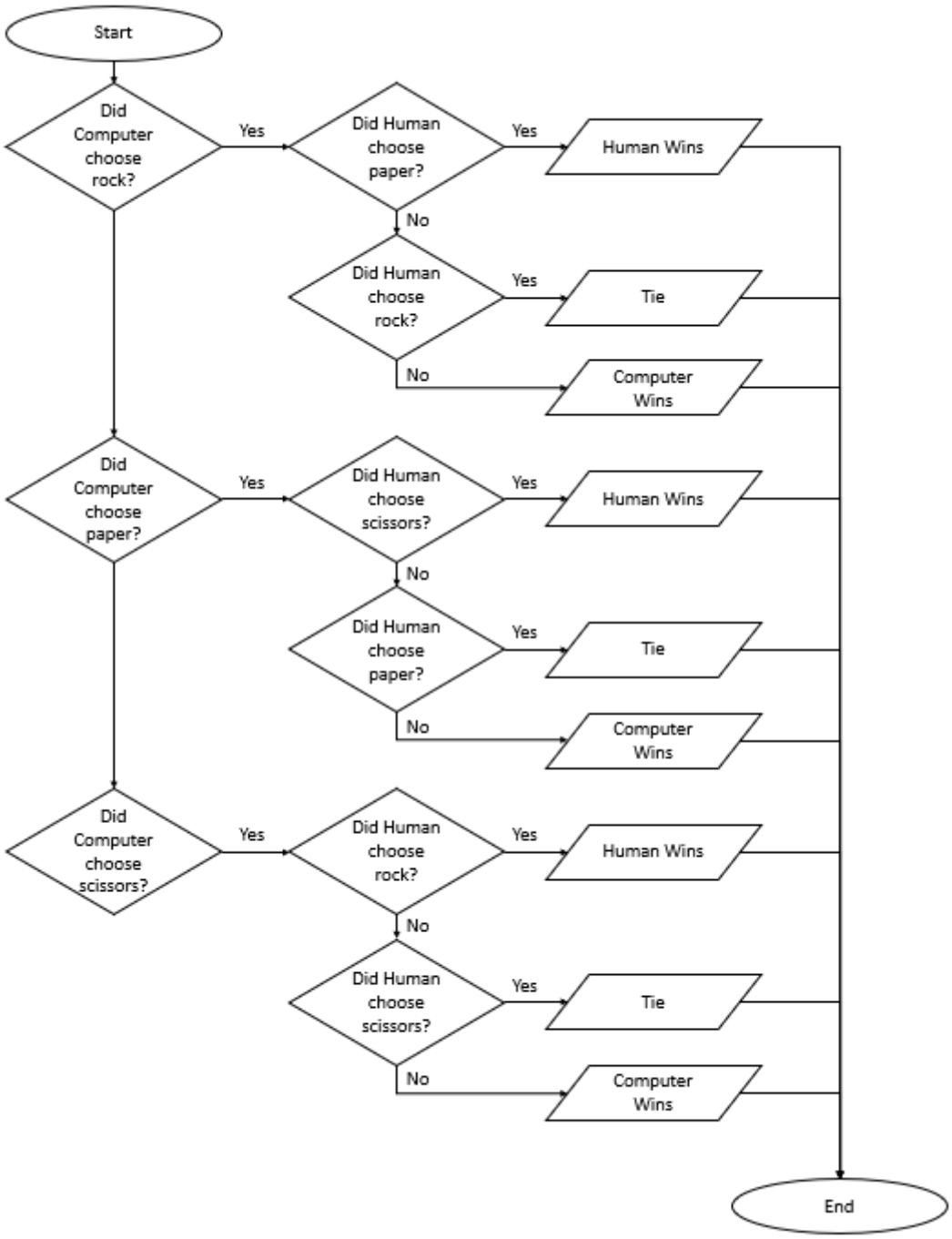

Figure 4 Flowchart for determine_winner() function.

Discussion and future work

I had some difficulty with implementing the checking of the human player's gesture. I think I would have been able to get it done, if I had more time to figure out how to write loops properly.

Also, I didn't get a chance to implement how the game should behave if there was a tie. If I had more time, I would change the main() function to implement the flowchart shown in Fig. 5, which describes an algorithm for handling a tie.

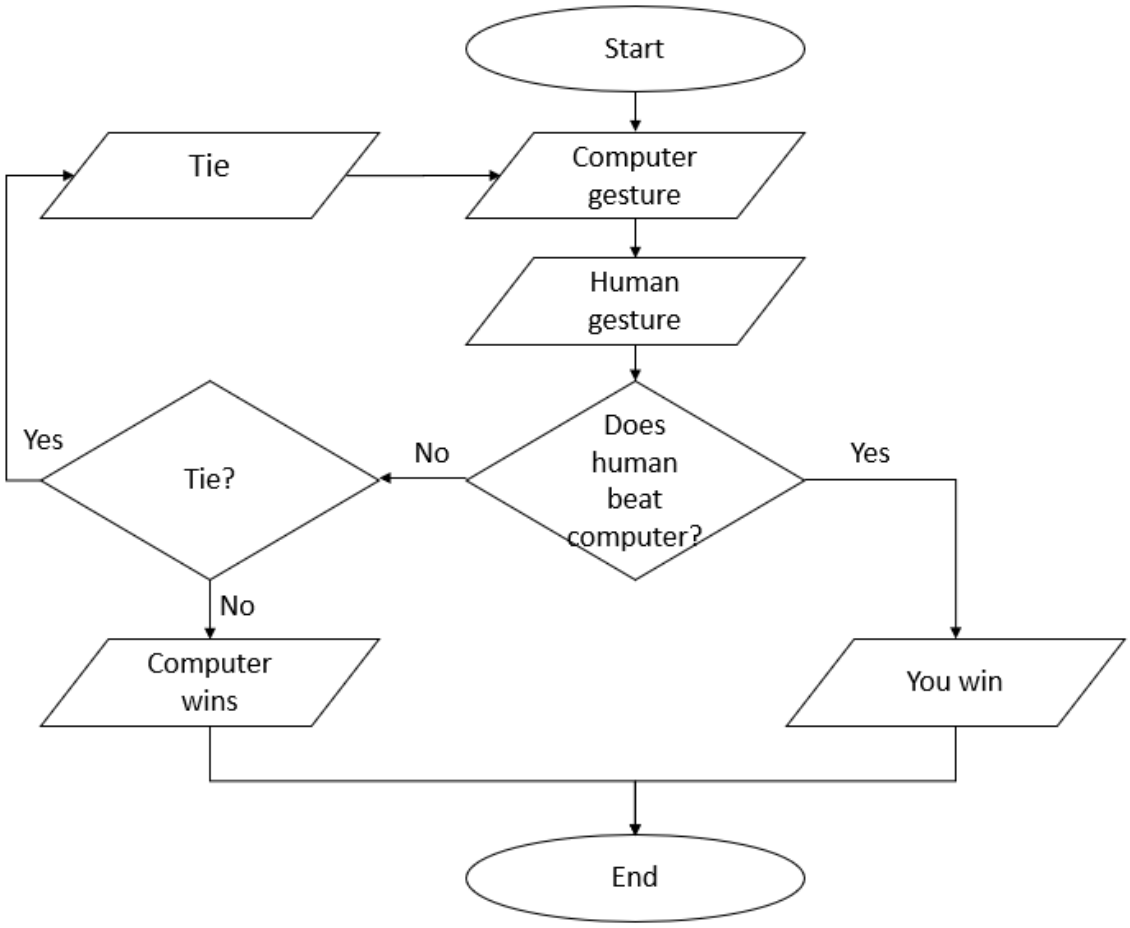

Figure 5 Flowchart for handling tie in program so that it would play again

Appendix A

rps.c

```
/**
   This program implements the game Rock Paper Scissors.
**/
#include “game_elements.c"

// this is the function that gets executed when the program starts
int main() {
   // randomly pick computer gesture
  int computer = get_computer_gesture();


   // ask human for their gesture
   Int human = get_human_gesture();

   // determine winner (calls determine_winner function described above)
   string winner = determine_winner(computer,human);

   // show outcome of game
   printf("The computer chose %d and you chose %d. %s\n",computer,human,winner);

   return 0;
}
```

game_elements.c

```
int get_computer_gesture() {
   srand(time(NULL));
   int computer = (rand() / ((double) RAND_MAX + 1)) * 3;
}

int get_human_gesture() {
   while (human < 0 && human > 2) {
      printf("Please enter your gesture\n");
      int human = get_int("Enter 0 for rock, 1 for paper, 2 for scissors: ");
   }
}

// this function is used to determine the winner
string determine_winner(computer,human) {
   string winner = NULL;
   if (computer == 0) {                  // computer chose rock
      if (human == 1) {                  // human chose paper - win
         winner = "You win!";
      } else if (human == 0) {           // human chose rock - tie
         winner = "The game is a tie.";
      } else {                     // human chose scissors - lose
         winner = "Computer wins!";
      }
   } else if (computer == 1) {           // computer chose paper
      if (human == 2) {                  // human chose scissors - win
         winner = "You win!";
      } else if (human == 1) {           // human chose paper - tie
         winner = "The game is a tie.";
```

```
      } else {                    // human chose rock - lose
         winner = "Computer wins!";
      }
   } else if (computer == 2) {    // computer chose scissors
      if (human == 0) {           // human chose rock - win
         winner = "You win!";
      } else if (human == 2) {    // human chose scissors - tie
         winner = "The game is a tie.";
      } else {                    // human chose paper - lose
         winner = "Computer wins!";
      }
   }
   return winner;
}
```

Appendix B

Please imagine there is some code here for testing and a screen capture of the output of testing.