Samuel Chew
Riley Harrison
Patrick Huarng
Susan Onesky
Armand Parajon
Peter Yetti
Nathan Zimmerman
02/12/2017
CS162 400 W2017

Group Project Plan

Identify Problem

Simulate a multi-round game of rock paper scissors between a user and a computer. Track both user and computer score. Allow user to select their tool and strengths for all the tools. Choose a tool for the computer based on the user's past choices. Allow the user to control the game using a menu function.

Identify Inputs

Tool strengths, menu options, tool choices.

Identify Outputs

Win, loss, or tie. Total score.

Computer Choice Algorithm

The computer's choice in tool will be random for the first 3 rounds played. After the first 3 rounds, the computer will select their tool using a char vector recording past human tool choices. The index of the vector will be selected randomly. The choice for the computer will then be the opposite of the value at the selected vector index. An opposite value is simply the tool that beats the selected tool. If the user selects a certain tool more than the others, this method ensures that the computer choice will be weighted towards choosing the winning option vs that choice.

Fight Algorithm

The fight algorithm if called on the computer's tool will take as a parameter the user's tool choice. The function will create temporary strength variables for the function tool and the tool entered as a parameter. Conditional statements will then check the parameter tool type against the function tool type. The temporary strength variables will then be modified per game rules. If the tools are the same, the strength variables will not be modified. The two temporary strength variables are then compared. The higher value results in a win, the lower in a loss, and even strengths in a tie. The function returns a char representing a win, loss, or tie.

Program Design

The Main file will be known as play_game.cpp per project specs. The play_game.cpp file will start by instantiating an RPSGame class object using the classes default constructor. The default constructor will ask the user if they want to change the strength of any of the tools. The constructor will

then set the tool strengths to those values. A while loop will then call the RPSGame object's playRound function until the function returns false.

In the playRound function a menu function will be called. The menu will have four options. The first being to select rock, the second to select paper, the third to select scissors, and the last to select exit. A conditional statement will check if the exit value was chosen. If the exit value was not chosen, then a Tool object will be dynamically created using the RPSGame Tool* humanTool data member. The constructor for the tool will contain type and strength as parameters. The computer choice algorithm will then be executed to assign a type value for the computer. The RPSGame Tool* computerTool data member will then be used to create a new object for the computer. The RPSGame class vector data member will record the human's choice as a new element. The object's constructor will contain the result of the computer choice algorithm as well as the appropriate strength value data member from RPSGame.

The Tool class virtual fight algorithm will then be called on either the computer or the human tool. The fight algorithm will return a char representing a win, loss, or tie. Either the human_wins or the computer_wins variable will then be incremented depending on the return value for the fight function. A print statement will display the results to the user. The playRound function will then delete memory associated with the dynamically created objects. The function will by default return true. If the exit value was chosen it will return false. After the while loop in the play_game.cpp a final score will be printed to the screen

Pseudocode

algorithm: play_game.cpp

Test: If exit character is selected, program exits.

```
Instantiate RPSGame object
Set gameExit variable to false
while gameExit returns false
        gameExit is equal to RPSGame playRound
print message saying the game has ended
print final scores
```

Note: For the fight algorithm, the general structure is represented below. For the rock and scissor version of the function the tool names will need to be switched based on the standard rules of rock paper and scissors.

algorithm: Paper::fight

Test: User choice accurately reflects a win, loss, or tie.

```
Create temp paper strength variable and set to strength data member
Create temp parameter strength and set to strength lookup value
Create outcome variable
If object parameter is of type rock
        Temp paper strength is equal to double that of parameter strength
        Compare the two temp strengths
                Outcome is the higher of the two temp strengths
```

If object parameter is of type scissors
    Temp paper strength is equal to half that of parameter strength
    Compare the two temp strengths
        Outcome is the higher of the two temp strengths
If object parameter is of type paper
    Outcome is a tie
Return outcome

Algorithm: nextAIMove
Test: User input choices are reflected in the AI choices as the game goes on past round 4. Wins, losses, and ties are reflected accurately.

If the current round is less than 4
    Pick a random tool
    Return random tool
Else
    Set index choice to a random number from 0 to array length
    Set move equal to value of vector variable at set index
If the move value equals scissors
    Set the move value to rock
If the move value equals rock
    Set the move value to paper
If the move value equals paper
    Set the move value to scissors
Return the move value

Algorithm: playRound
Test: User choice of tool is accurately reflected in the total win, loss, and tie scores.

If user chooses menu option for exit
    Return false
else
    if user chooses rock
        Dynamically create a rock object
        set rock strength to RPSGame rock strength
    else if user chooses paper
        dynamically create a paper object
        set paper strength to RPSGame paper strength
    else if user chooses scissors
        dynamically create a scissor object
        set scissor strength to RPSGame scissor strength
    add user choice to knowledgebase vector
    set a char variable value to result of nextAIMove function
    if char variable is equal to rock

Dynamically create a rock object
set rock strength to RPSGame rock strength
else if char variable is equal to paper
dynamically create a paper object
set paper strength to RPSGame paper strength
else if char variable is equal to scissors
dynamically create a scissor object
set scissor strength to RPSGame scissor strength
call the fight function on the human tool object using the computer tool object as a parameter
if the function returns W
increment human wins data member
else if the function returns L
increment computer wins data member
else
increment the ties variable
print who won and the current score to the screen
return true

Class Hierarchy Diagram