

CSCI 1302
Homework 1 & 2
Fall, 2004

Rock, Paper, Scissors

Introduction

This program is intended to be a reminder of some methods using array with some additional features. The program will introduce inheritance in objects. The general classification of the problem is cellular automata.

The Game

Rock paper scissors is a game often played by children (and probably adults!) to see who wins quickly. Both parties hold out their fist, count to three, and then show a rock (with closed fist), paper (with flat hand), or scissors (with first two fingers extended). Rock breaks scissors, paper covers rock, and scissors cut paper, so the wins are shown in this table:

	Rock	Paper	Scissors
Rock	nobody	Paper	Rock
Paper	Paper	nobody	Scissors
Scissors	Rock	Scissors	nobody

Cellular Automata

In a cellular automata algorithm objects or agents move around on a grid independently, each following their own set of rules. For this program the objects will be rocks, papers, and scissors. A graphic example of the sort of system this program should emulate to be found at will be demonstrated in class . This assignment, however, does not require graphics, although it is possible that extra credit may be earned for a nice graphical solution.

The Program

The major features of your program will be:

- A two-dimensional array to form the grid on which your objects will move
- A class of objects that can move around the grid
- Three objects that inherit from the main class to represent rocks, papers, and scissors
- A main program that will start everything moving and report results

The Objects

The three objects will interact within the rules of the game any time they appear in two adjacent cells in the grid. (You may wish to restrict this to just cells in the same row for greater simplicity.) However, instead of a simple “win”, the loser of each encounter will change the object that “loses” into the other type of object. For example, if a rock and a scissors come together, the scissors will be changed to a paper because the rock will win. Each time each object will move randomly up, down, left, or right from its current position, whether it encountered another object or not. To start the game, you may have your program randomly place objects of the three types onto the grid or you may ask the user for coordinates on which to place objects.

Results

Your program will need to report results periodically as it runs. You may want to have the program print the contents of the grid at each iteration in order to check that it is working correctly. One iteration is one visit to each of the objects on the grid. You will probably want to do this by traversing the grid in a nested loop.

There should be a count of the number of papers, rocks, and scissors that is accurate at all times.

This is a program that could run indefinitely, so you must put some mechanism in place for the user to determine how many iterations will be needed.

Artifacts

The final product must consist of a hard copy of your source code, a hard copy of an example run of your program, and ***IF AND ONLY IF YOUR PROGRAM RUNS*** a disk with your executable file and your source code. Points will be deducted for turning in any disk that does not have an executable file for this problem.

Program 1 – Due August 30th

For the first part of the program you must create an object that can be used in the eventual array as a base object. Three other objects will inherit from this object, and these three will represent rocks, papers, and scissors. The objects should know the rules for their movements. Your Program 1 must show off its objects by allowing a user to add objects as desired and to tell the movements each may make.

Program 2 – Due September 10th

This will be your final version of the program.

Advice

As you work on your program it is most strongly recommended that you save intermediate versions of your program on a floppy disk to ensure that you always have a backup copy and that if your next stage of program creation does not wipe out a partially working version. Lots of credit is possible even for a program that does not do everything specified, so you should *always* have something prepared to turn in on the due date of any program, even if it is not the perfect creation you'd like it to be.