Coin Flipping

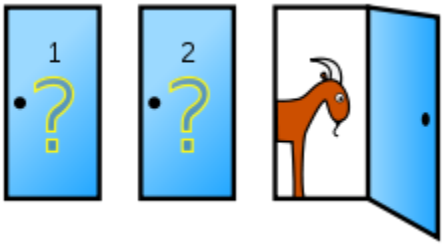
- Use the random operation to write a program that flips a coin 10 times and shows how many were heads and how many were tails. Divide one by the other to see how close the ratio is 1. Run the program several times to see how the ratio changes. What happens to the ratio if you flip 100 times? 1000 times?
- If you like, add graphics to show the result of each coin flip (but you'll still have to keep the flipping fast or it will be too boring to wait for the program to flip 100 times).

Two Dice

- Use the random operation to make a program that rolls two dice and adds up the total (a number from 2 to 12). Keep track of how many times you get each number. You can keep track using 11 variables, or you can use a list (ask for help if you want to know how). Make your program do 10 rolls, then try 100 rolls and 1000 rolls.
- Make the program draw a bar chart showing how many times each number came up. What patterns do you see? Can you explain it?
- If you like, add graphics to show the result of each dice roll.

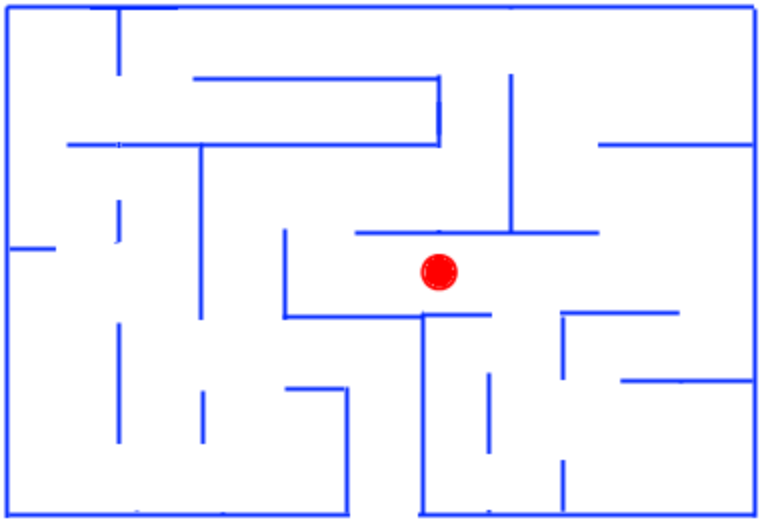
Blackjack

- Use the random operation to make a program that deals two random cards from a deck by picking a random suit (spades, hearts, diamonds, or clubs) and a random rank (2, 3, 4, 5, 6, 7, 8, 9 10, J, Q, K, or A). (Can you make sure the two cards aren't exactly the same?) Check if the total is 21 (ace counts as 11, and face cards count as 10, so you'd need an ace and a 10/J/Q/K). Have your program deal lots of times and keep track of how many two-card hands are 21. Make an estimate for the chance of being dealt a 21 with two cards
- If you like, add graphics to show the two cards.



Monty Hall Puzzle

- There are three doors. There is a good prize behind one of them, and bad prizes behind other two.
- You choose a door (but nobody opens it yet). Then, Monty opens one of the other two doors and (very important!) he always picks a door that reveals one of the bad prizes – he never opens the good door.
- Now you have a choice: you can choose to stick with your first choice of door, or you can switch to the other door (the one of the remaining two that Monty didn't open). The question is: **Is it better to keep your first choice, switch, or it doesn't matter?**
- Example: Suppose in the picture above you are told there is a car behind one of the three doors. You guess door #2. Monty says that he'll open one of the other doors, and the door he'll open has a bad prize. He opens door #3 and indeed it has a bad prize (assuming you don't want a goat). The question is: Should you change your guess to door #1, or leave it at door #2, or does it not matter?
- Write a computer simulation to find out!



Random Maze Escape

- Draw a simple maze on the background using a certain color. Make a sprite pick a direction: up, down, left, or right. The sprite moves 10 pixels at a time in that direction. If it hits a wall, it backs up 10 and chooses another random direction and keeps repeating. Can the sprite get out of your maze?

Balls and Urn

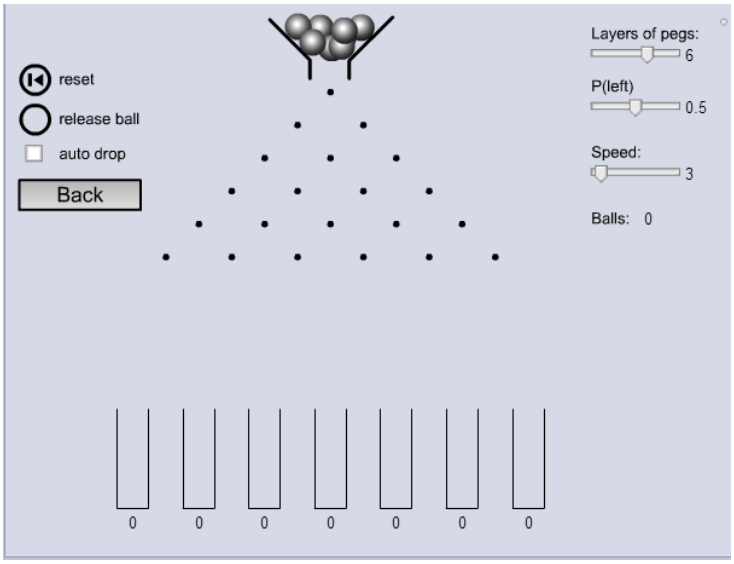
You are given two urns (containers like big vases) and 5 black balls and 5 white balls. Somebody else will come in and pick one of the urs at random and a ball drawn at random from it. If the ball they pick is white, you win. If the ball they pick is black, you lose. How would you distribute the balls so as to maximize the chances of winning?

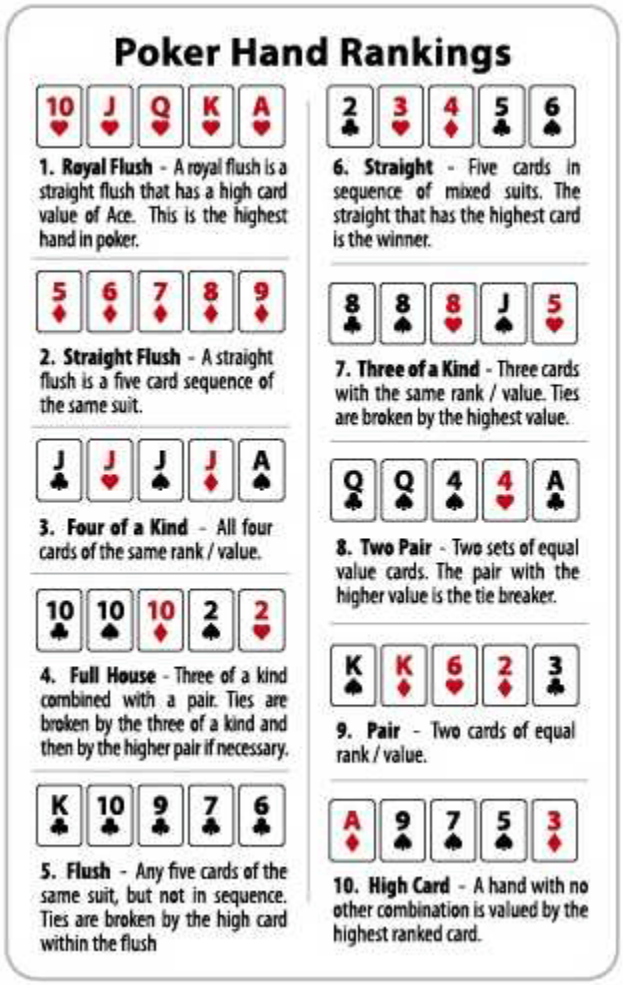
- Write a program that tries a distribution (e.g., urn1 has 2 black and 3white, urn2 has 2 white and 3 black) and figures out the probability of the person picking a white by simulating lots of times. Try various distributions and see which one works best.

Fill Squares

- Make a program that fills a 10x10 grid with colored squares. You can do this using a square sprite whose color you change randomly (it doesn't matter what the colors are for this experiment; you can use all the same color or use different colors to make a nice picture like the one above). Repeatedly pick a random x value from 0 to 9 and a y random value from 0 to 9 and draw a square at that grid location. Keep count of how many squares you plot. Watch the program and stop it by pressing a key when all squares in the grid have been filled. How many tries does it take to fill the screen? Do it lots of times and see how many tries it takes on average.

Pachinko Machine





Poker Hands

- Make a program that uses random to deal a 5-card poker hand. Check if your hand has at least a pair. Keep track of how many times you get at least a pair. Give an estimate of the probability of getting dealt at least a pair.
- You can add other poker hand types and see what the chances of getting those are – see the chart above.
- If you like, add graphics to show the five cards.