**Randomness and Probability  
Extra notes…**

1. Random events are discrete: those that have gone before have no effect on those that are still to come.
   1. Gambler’s Fallacy or Monte Carlo Fallacy:  
      The belief that if you keep losing, sooner or later you have to win. In fact the chance never changes.
2. What is the chance of rolling a 6 on 2d6?
   1. It’s not 2 in 6 (33.33%) as people often think.
      1. Easier to work out chance of it not happening: (5/6) x (5/6) = 0.69444
      2. Chance of it happening is 1 – 0.69444 = 0.30555 (30.55%)
3. Birthday Problem: How many people do you need in a room before there is a 50% chance that 2 of them were born on the same day?
   1. Answer: 23
   2. It’s easier to calculate the chance of it not happening:
      1. (365/365) x (364/365) x (363/365) x (362/365) x (361/365)… x (343/365)
      2. = 0.492703 (chance that there is no match)
      3. Chance of match = 1 – 0.492703 or 0.507297 (50.72%)
4. Monty Hall Problem: <https://en.wikipedia.org/wiki/Monty_Hall_problem>
   1. When you originally chose your door, the chance of winning was 1 in 3.
   2. This remains unchanged after the host reveals behind a door.
   3. However now that you know the opened door contains a goat, the chance that the other door has the car must cover the other 2 in 3 not covered by your choice.
5. How long would it take to generate every combination of a 15x10 pixel image, assume each pixel is grayscale, able to show the range of 0-255.  
   <http://petapixel.com/2013/02/07/exhibition-uses-a-computer-to-generate-every-possible-photograph/>
   1. 256^150
   2. A long time!
6. While we’re on silly number problems: imagine you have a piece of paper which is 0.1mm thick. You can fold it in half to double that thickness. Imagine you could do so any number of times and you have unlimited paper to work with.
   1. How many folds would it take for the paper to reach the moon? (384,400 km)
      1. Answer: 42 (total of 439,804.651 km)

**Dice Probability**

The most common form of probability in games is to emulate dice rolls to determine random loot drops, random monster spawns and other similar aspects of gameplay.

**Dice Notation**

Invented by role playing games, dice are commonly represented by a “d” followed by the number of sides the dice has. If there is more than one dice, they are listed before the “d”.

e.g:

* 1 six-sided die: d6 or 1d6
* 3 six-sided dice: 3d6
* Two dice with 20 sides: 2d20
* A die with 100 sides (a random percentage): d100, 1d100, or d%
* Roll two dice with 8 sides and add three to the result: 2d8+3

**Dice generator**

<http://anydice.com/>

Some things to try:

* Graph the results of 2d6
  + output 2d6
* What are the chances of at least one 6 being rolled on 6 dice with 6 sides each?
  + output [count {6} in 6d6]

Random number generator website: uses atmospheric noise to make real random numbers:

<https://www.random.org/>

**Non-transitive Dice**

Dice don’t have to be uniform, consider what you could do with unusual dice types:

<http://singingbanana.com/dice/article.htm>

**Generating Random Numbers**

**The C Way**

#include <cstdlib> //include for rand()

#include <time.h> //include for time.

srand(time(nullptr)); //seed the random number generator with time.

int value = rand()%6 + 1; //Generate a random number between 1 and 6.

**The C++ Way**

#include <random>

std::default\_random\_engine generator; //create the random number generator

std::uniform\_int\_distribution<int> distribution(1,6); //scale the values to between 1 and 6

int value = distribution(generator); //Generate a random number

**Don’t use purely random numbers for game mechanics!**

Basing loot drops and other mechanics off purely random numbers can be frustrating to the player, no one wants to spend days farming the same monster over and over for the item that only drops 0.001% of the time.

It’s better to bias or weight the results in the player’s favor so that the longer they try to do something, the greater the chance they will succeed. Here’s one way to do this:

**Shuffle Bag**

Instead of thinking about dice rolls, imagine you have the numbers on counters which you place into a bag. Every time you need a number, you pull one at random from the bag and discard it. Now that number is no longer in the bag and won’t be drawn again. This way the player is guaranteed to get every number eventually. When you’ve gone through all the numbers, you shuffle them all back in and start the process over.

You can make this seem more random by increasing how many of each number are in the bag, for example if you have 3 of every number, it’s possible to get the same number 3 times in a row, but you’ll still get every number eventually.