Autumn Ava Barnes
2023-04-07

What is PROC, and how to roll one and a half Dice

Abstract - What is this about?

In this guide/paper/resource I will be explaining what **PROC** is, how to roll one and a half dice, and a neat way to simulate being lucky. This can be useful in anything in which there is a random chance for something to happen.

Who is this for?

I am mainly making this for hobby game-developers, and as such I'll make sure to give plenty of examples, and break down the math behind it in a *hopefully* easy to understand way. I want this to be for anyone no matter your expereince. This kind of stuff can be really fun, and I wanted to share my enjoyment of the subject with you, and maybe I'll even get you to laugh.

Why is this in a *professional* paper format?

I typically hate looking up something game design related, and then stumbling upon the god-awful confusing mess that is technical lingo where I spend more time deciphering it than I do learning from it. I wanted to give it a shot on my own and see if I can make something that is educational, easy to understand, as well as fun. This is also an excuse to get better in typesetting for my school work. Enjoy!

Table of contents

yeah it is pretty short

Scroll Down!

Autumn Ava Barnes
2023-04-07

Rolling Dice

Lets say you have a random chance for an event to happen. Maybe a coin flip to see who starts first, or a dice roll to see if your knight lands their attack on the goblin perhaps.

The important part is that something happens as a result of a random number. In the wonderful world of computers, we roll virtual dice. We can generate random numbers from 1 to 6 (*like a six-sided dice*), or since it is usually easier to work with, we can generate a random number from 0 to 1.

Now why is random numbers from 0 to 1 easier to work with? It seems kinda counter-intuitive right? Well lets say that we want to have a %50 percent chance for something to happend, well with dice we would check if our dice roll is 4 or more. If it helps, you can think about it like this:

$$\text{if } (\text{d6} \geq 4) \text{ then}$$

Now this is kinda lame, the probability isn't easily recognizable just from looking at it. Also, if we are only generating integers (*meaning whole numbers, so no 7.5 or 3.8561*), then we lose some configurability because checking if the dice roll is, lets say, above 5.5, isn't really useful.

On the other hand, random numbers ranging from **any** number inbetween 0 and 1 make so much more sense. Here is what a %50 chance would look like:

$$\text{if } r < 0.5 \text{ then}$$

You can see that all we do is check to see if our random number r is below 0.5, which is the same thing as %50. Here is what %75 chance would look like:

$$\text{if } r < 0.75 \text{ then}$$

From here on out, when I say rolling dice, I am usually refering to a computer generated random number from 0 to 1.

Whats PROC?

The act of having a random chance of an event happening has a fun name too!

Procedural **R**andom **OC**currence
also known as **PROC**.

and just for the hell of it, we can also call succeding the dice roll and triggering the event **PROCCING**.

Now lets define a function for us to use, lets call it DoesPROC. It will take in a *probability* (*This is our 0.5 or %75*), do our dice roll, and will return **true** if the value is less than out *probability*.

This *probability* is also called the **coefficient**, though both terms are interchangeable in this case.

Lets implement it; my personal programming language of choice is **Lua**, Here is what that would look like :

```lua
function DoesPROC(coefficient)
  return math.random() < coefficient
end

-- Used like this:
if DoesProc(0.75) then
  --Do Event
end
```

Yeah, not too complex, and I bet you are bored at this point. But lets see something cool you can do with this.

Lets Roll Two Dice!

Yeah I know it still sounds pretty boring.

What all of this is leading up to rolling multiple dice. More specifially, lets say you get to roll 2 dice, and if any of them meet the coefficient, then it returns true.

If that doesn't make sense, you can also view this as rolling multiple dice, and taking the best one. This method is used in a few games, but I discovered it from Risk of Rain

With this idea, we can up the players probability of succeeding, without having to increase the

coefficient. You can see this as giving the player multiple chances, or as Risk of Rain uses it, controlling how lucky the player is.

Lets put this into code so you can see how it works :

```
function DoesPROC(coefficient, dice)
    local Rolls = {}
    for i=1, dice do
        Rolls[i] = math.random()
    end

    for i=1, #Rolls do
        if Rolls[i] < coefficient then
            return true
        end
    end

    return false
end
```

This code is unoptimized, there are better ways to do this that don't include keeping a table of the dice frolls, I just wanted to demonstrate it in a way that would be similar to how to would do it with real dice

Because of the fun world of probability, with a coefficient of 0.5 and rolling a single dice, we get a probability of %50, as expected. But if we roll 2 dice, then our chance of triggering our event jumps up to 0.75 percent! And again, just to clarify, all we are doing is rolling extra dice and seeing if any of them meet the requirement.

Cool right, maybe, I don't know about you but I'm stoked. Anyway, this method is kinda messy, and it still has a few limitations.

The Math AHHHH

That's right, unfortunately, there is math involved. Rather than doing this convoluted nonsense, we can turn it into a single equation.

Now why would we want an equation when the previous method works just fine? Well first of all, what if I wanted to roll **one and a half dice**? Like we can't just do that, that's just not possible with the previous method. Also, what the hell would you even do if you gave a negative number of dice to roll?! *You toss a negative dice onto the table and it just falls through?*

The second reason for this is that for a developer, it might be a bit harder to visual with the old method. If we had an equation, we could graph a nice curve that shows how an increase in dice rolls can effect the probability of Proccing.

Time for math, the equation is pretty small :

$$f(p, d) = 1 - (1 - p)^d$$

Where:
- p is our coefficient
- d is how many dice

Now don't freak our it's going to be okay. Let's break it down bit by bit. $f(p, d)$ is a function, it takes the input of p and d. p is our coefficient and d is how many dice we are rolling. This function is going to return a new coefficient, in which we can check by.

Lets start with the first step in the equation. This would be $(1 - p)$.

$(1 - p)$ will basically *flip the coefficient*. So if we had a value of 0.25, it would get flipped to 0.75. Here are some examples if it helps:

$$\begin{aligned} 1 - 0.25 &= 0.75 \\ 1 - 0.75 &= 0.25 \\ 1 - 1.0 &= 0 \\ 1 - 0.4 &= 0.6 \\ 1 - 0.5 &= 0.5 \end{aligned}$$

That's the first step done. Next would be the scary part, the exponent. What we are doing is taking our first step and raising it to the power of d (*how many dice we are rolling*).

Remember that raising a number to a power just means that we are multipling it by itself, so for three dice we are multiplying our first step by itself three times. If this is a bit overwhelming, than it's okay to black box it, meaning you dont have to

worry about the math behind it, just understand that it modifys the number.

Lets walk through it step by step if we had a coefficent of 0.25, and rolled two dice.

$(1-p)^d$	Starting Formula
$(1-0.25)^2$	Subsitute our p and d values
$(0.75)^2$	Flip the coefficent
0.5625	Raise to the power of 2

That's our second step done! Just one last one, and we already have done it once. All we do is do the subtract it from 1 again.

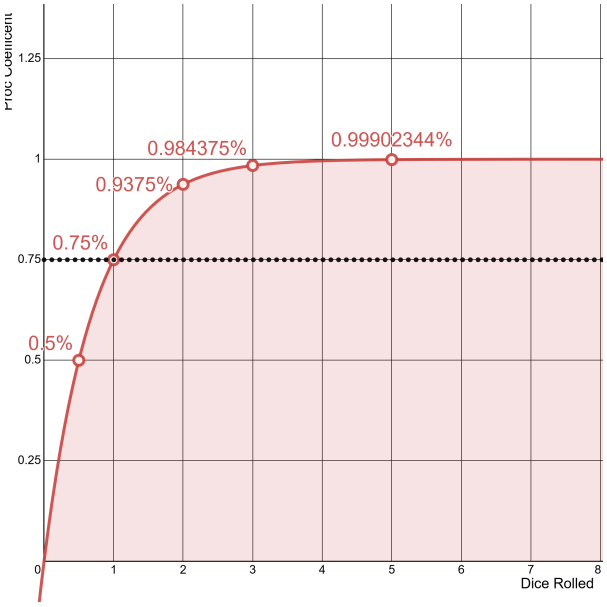
So in the above equation all we would do is

$$1 - 0.5625 = 0.4375$$

Congrats! The math is over and our weary brain muscles can rest. What we are left with is our new coefficient that we can check a single random dice.

Visualization

Lets see what that would look like as a graph :



Desmos Graph Link

You can move the `p` slider to see what different starting coefficient is. The x axis is how many dice are being rolled. Notice how the probability (*The curve*) never goes above 1.

You can also now know what happens when rolling negative dice! Turns out it will always Proc, a little lame - but that does give an idea, what if you had a negative coefficient? I'll let you have fun with that one though.

Implementing - The code

```lua
function DoesPROC(coefficient, dice)
   local new_coefficient = 1 - (1 - coefficient) ^ dice
   return math.random() < new_coefficient
end
```

Well, that is kind of it... see ya.