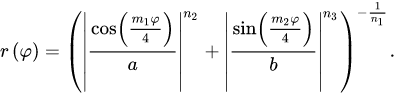
**Procedural Weapon Generator**

**Sethiel Jann Tantay Cosio**

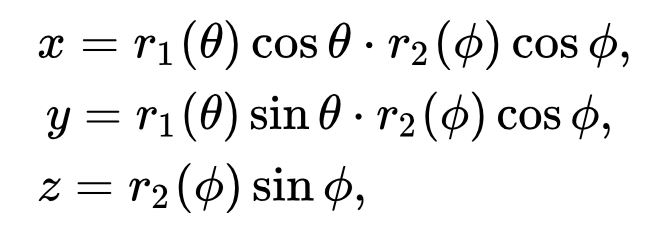
**DUB11735**

I have made a procedural, random weapon generator that displays stats: Durability (how long the weapon can be used effectively), Damage, Rarity, Weight and Speed. Initially, the generator was supposed to be a level generator with a terrain that will change using the superformula for procedural generation that was proposed by Johan Gielis around 2000 (Wikipedia, 2018)



**Superformula (Wikipedia, 2018)**

I managed to code this superformula into my script and give its parameters random inputs using Random.range to get a completely random outcomes using a generic float method called ‘ranVals,’ that took in 2 float inputs that the Random.range will use to randomize the parameters of the formula. But this formula alone wouldn’t define the shape, so I used the second part which took into account the space the shape would be in – 3D space.



**Superformula (Wikipedia, 2018)**

This can also go into the 4th dimension, or nth dimension of space, but I only needed 3D. Using a the ‘Test,’ scene in my game, I was able to see the code in action when I attached it onto a cube and made it affect the scale of the shape, which worked. I then tried to implement the code onto my weapon generator to give me a wide range of values for the weapon statistics. In the beginning, since I only needed 3 stats to be randomly determined by the formula: Durability, Weight and Rarity, the formula worked fine and gave me the results I needed to make realistic values for the 3 stats. I had to use if statements to keep the values from going into unrealistic values like -1000 for weight. These statements limit them and by checking if the values are higher or lower or in between a certain amount and Random.range will determine another set of values for it. For example, if the value for weight was equal to or above 100, then the new range for weight will be between 80 and 100 which the Random.range will pick out a number from that range. This was where problems came as I added Damage and Speed that were calculated using the Rarity and Weight. To calculate Speed, I used the Inverse Square Law (NDT Resource Centre, 2018) that took in a float variable called ‘highestSpeed,’ and Weight.

Speed = highestSpeed / (Weight)2

The problem I face here was that it would take the original, calculated value by the superformula and not the value that the if statements would give. I face the same problem when calculating Damage which used the Newtonian formula involving Force and Mass and Acceleration.

F = m \* a **(The Zooland Education, n.d.)**

Damage = Weight \* Speed

I used Speed instead of Acceleration because in my game time of movement from point A to point B is always 1 second, and the initial speed of the weapon would be 0. So, if we were to input this in the Acceleration formula:

a = (v – u) / t

a = (speed – 0) / 1

We’d get Speed as Acceleration as an answer. To tackle the problem, I had to cut out the superformula and just use Random.range instead to find Durabilit, Weight and Rarity. The superformula is still present in my code. It is the block of code from line 98 to line 116 that has been commented out. This allowed me to get realistic values for all my stats. The bonus stats, Fire and Poison were determined by the rarity. 1 would be added at every rarity level alternately from fire to poison; from 0 for both in common to 2 for both in legendary.

To generate my weapon, I used an if statement that checks for a number from 1 to 4 from the ranVals function containing the Random.range, which will spawn the weapon and display the statistics onto a panel.

**References:**

**Wikipedia. (2018, April 12). Superformula. Retrieved April 30, 2018, from** [**https://en.wikipedia.org/wiki/Superformula**](https://en.wikipedia.org/wiki/Superformula)

**NDT Resource Centre. (n.d.). Inverse Square Law. Retrieved April 30, 2018, from** [**https://www.nde-ed.org/GeneralResources/Formula/RTFormula/InverseSquare/InverseSquareLaw.htm**](https://www.nde-ed.org/GeneralResources/Formula/RTFormula/InverseSquare/InverseSquareLaw.htm)

**The Zooland Education. (n.d.). The Mighty F = ma. Retrieved April 07, 2018, from** [**http://zonalandeducation.com/mstm/physics/mechanics/forces/newton/mightyFEqMA/mightyFEqMA.html**](http://zonalandeducation.com/mstm/physics/mechanics/forces/newton/mightyFEqMA/mightyFEqMA.html)