Digital Dice Roller

Validation Testing

Author: Dillon Linville

**Summary:**

Due to the inherently random nature of the outputs produced by the Digital Dice Roller (DDR) it can be difficult to develop a standard data set for validation testing. This is further complicated by the 7 different inputs needed to make a calculation, thankfully 5 of the inputs have a small, fixed range of possible values that have been predefined. Only two of the inputs, “Number of Shots” and “Damage” have true text entry fields for a user to define a new option.

**Method:**

The first step in the validation process was to determine a set of default values for the 7 inputs.

A screenshot of a game

Description automatically generated

The second step in the validation process is a series of tracing print statements built into the calculation function. This means that with every calculation all the relevant variables and roll lists are printed to the terminal and can allow review and verification of the calculation process.

A computer screen with text on it

Description automatically generated

1. In this example the ShotList has 10 integers with values 1-6 to represent the 10 shots or die rolls.
2. Any shot must meet or exceed the value of the Ballistic Skill, 3 in this case, to be successful.
3. The HitList represents the rolls or values from the ShotList that were successful.
4. The length of this list, or the Number of (successful) Hits, is 6.
5. WoundChance is the value that a Wound roll must meet or exceed to be considered successful, in this case 3. This is determined by comparing Attack Strength vs. Target’s Toughness.

- If Strength is twice Toughness: wound on 2+

- If Strength is > Toughness: wound on 3+

- If Strength is = Toughness: wound on 4+

- If Strength is < Toughness: wound on 5+

- If Strength is half Toughness: wound on 6+

1. For each successful hit, or Number of Hits, a new die will be rolled, this is a Wound Roll. These values are stored in WoundList.
2. The ConfWoundList, are the confirmed or successful wound rolls, their value met or exceeded the WoundChance.
3. NumWound is the length of ConfWoundList, or the number of confirmed/successful wound roll.
4. Then for each confirmed wound, the target will roll a new die. These save rolls are stored in SaveList.
5. The Enemy Save, 4 in this case, is the value which each save roll must meet or beat to be successful.
6. The Ap or Armor Penetration value of the attack modifies the value of each roll in SaveList before it is compared to the Enemy Save value. So, in this instance (save – 1) >= Enemy Save.
7. The Save Counter indicates the number of successful saves.
8. Save Counter is then subtracted from NumWounds to determine the number of wounds that damage the target.
9. Then each wound is multiplied by the Damage value and summed to equal the output Amount of Damage.

**Edge Cases:**

These apply only to the two inputs that have a text entry field, “Number of Shots” and “Damage”.

**Zero or Negative Numbers**:

* Zero for Number of Shots results in all other outputs being zero, no crashes
* Negative for Number of Shots results in all other outputs being zero, no crashes
* Negative for Damage returns a negative value, no crashes

**Text Entries:**

* Text entries into Number of Shots yields an error message, no crashes

A screenshot of a computer

Description automatically generated

* Text entries into Damage yields an error message, no crashes

A screenshot of a computer

Description automatically generated