Deputy (Dungeon) Master – User's Manual

v. 2017-12-04

Short description

Deputy Master (DM) allows the Dungeon Master to focus more on the important stuff during battle – by rolling the rolls for you, and keeping track of various stats and status effects of each monster you're controlling.

You will only need to prepare a text file for each monster in your battle, and Deputy Master does the rest for you.

System requirements

You will need Python 2.x or 3.x installed on your computer:
https://www.python.org/downloads/

Verified info (if you have trouble, check these):

- Deputy Master works with Python 2.7 and Python 3.5.
- It also works on Windows 10.
- Module Tkinter/tkinter might not start when running on MacOS.

Starting up

After you installed Python 2.x or 3.x , you have two options:

1. Simply start `DM.pyw` (by double-clicking, or equivalent; pair it with the `pythonw.exe` you've just installed). This will scrape the folder `DM.pyw` is in for text files, and open them up in separate windows (provided they have the right format).
2. Open the console, and type in:
   `python DM.pyw`
   this does the same as #1; or:
   `python DM.pyw <foldername>`
   this scrapes the subfolder <foldername> instead.

Example scenario

Using the console with
`python DM.pyw christmas`
will result in Deputy Master opening the example subfolder "christmas", and finding the three files:

`snowman.txt`
`tree1.txt`
`tree2.txt`

They all have the following structure:

```
Name    Mr. Snowman
AC      27
HP      225
Init    2
Fort    16
Refl    7
Will    8
Weapon Slam,17,3,8,7,20,2
Weapon Slam,17,3,8,7,20,2
Weapon Broomstick,19,2,8,6,19,2
```

Text file requirements

- Each different stat is a new line.
- Each stat is preceded by a keyword (e.g. `Name`, `AC`, `Will`), then a tab character (`\t`).
- The order of the stats does not matter.
- All above stats need to be included.
- The number of `Weapon` lines can be any > 0 number.

Name, AC, HP

Simply provide a name/number.

Init, Fort, Refl, Will

This is the appropriate Initiative/Save modifier.

Weapon

This is a list of stats for one specific weapon, separated by commas, e.g.:

`Slam,17,3,8,7,20,2`

1. 2. 3. 4. 5. 6. 7.

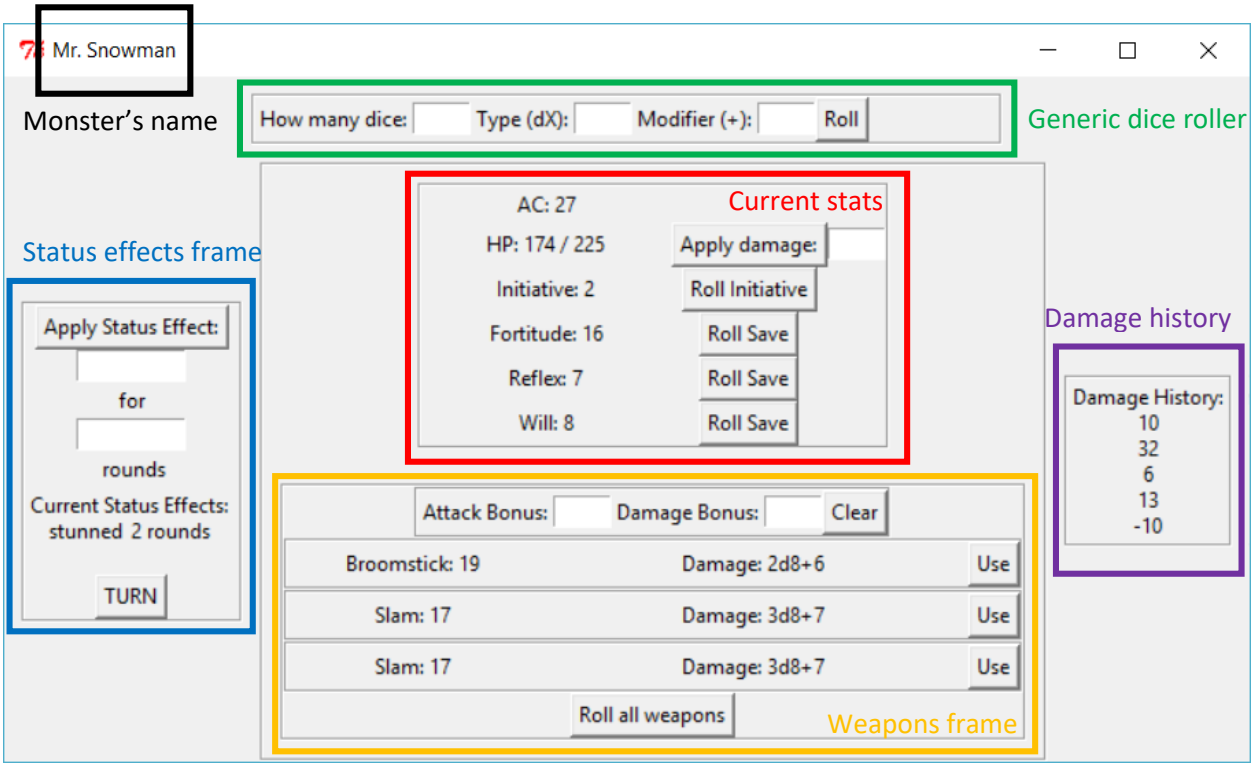
1. Name of the attack
2. Attack modifier
3. Number of dice in damage roll
4. Type of dice in damage roll
5. Damage modifier
6. Lowest attack roll that is considered a critical hit
7. Multiplicative factor for critical hit

Thus, if a character with a +7 attack modifier (+5 from base attack, +2 from strength) has a Longsword and a +2 Greataxe, this would be represented in the text file as:

```
Weapon Longsword,7,1,8,2,19,2
Weapon Greataxe,9,1,12,4,20,3
```

Features

An open window of Deputy Master looks like this:



Generic dice roller

You can roll any (positive) number of dice that have X number of sides, with a + or – modifier that is applied once at the end.

Status effects frame

Write any status effect into the first entry box and the number of rounds it'll last into the second box, then push the "Apply Status Effect" button.

Current effects are listed at the bottom of the frame. The "TURN" button decreases the duration of all effects by 1 round, this helps you keep track of when will each expire.

Current stats

Shows AC and current/original HP. The "Apply damage" button allows for healing as well (use negative numbers), however, you can't heal your monster above its original HP.

Roll initiative and saves in this frame.

Weapons frame

Attack Bonus and Damage Bonus are optional values, and once entered they are applied to all weapon rolls, until cleared with the Clear button (or deleted/replaced manually).

Each weapon is a new row. You can use them one-by-one, or all at once ("Roll all weapons" button at the bottom).

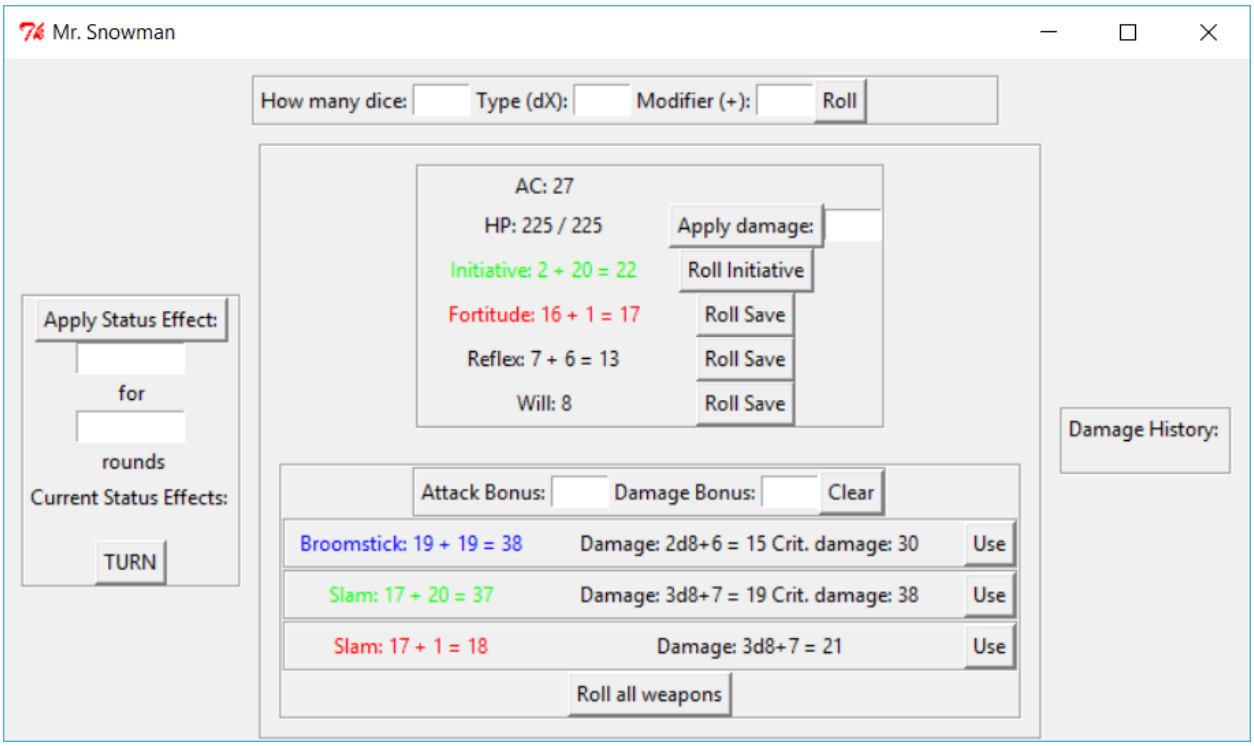
Damage history

Shows the last 10 damage made to the monster, including healing (signified by negative numbers).

Handling of critical rolls

Critical rolls are color-coded, for the convenience of the Dungeon Master:

- Critical fail
- Proper critical hit/natural 20
- Critical hit, but with a roll less than 20



Last updated: 2017-12-05
by A.Komar

Developer's Guide

v. 2017-12-04

Notation conventions

All functions, methods and classes start with capital letters,
and all (most) variables and instances of classes start with lowercase letters.

Each new word in a name is capitalized.

For consistency and readability, please adhere to these rules when adding to the code.

The GUI: Tkinter module

Deputy Master is a Python program, utilizing the module Tkinter (tkinter for Python 3.x) to realize a GUI. Importing this module allows you to place frames for visual organization purposes, as well as buttons, labels, entry boxes, etc. to realize the functionality you desire.

High level structure

Launching the script `DM.pyw` starts up the function
`Main(directory)`
with input from the input line argument (or with input="" if no input line argument given). This reads and processes all the text files in the the folder `directory`, by calling the functions
`ReadAllMonsters(folder)`
`ReadMonster(filename, sep, sepWeapon)`

Then, there is a new instance of the class `MonsterGUI` launched for each text file/monster, with the appropriate stats; this generates the separate windows for the monsters.

Class `MonsterGUI`

This is where the meat of the code is. It has the following methods, most corresponding to the action of pushing a certain button:

`__init__` : this method is immediately called when creating an instance of the class. It defines the frame structure of the window, defines and places the objects of the GUI.

`RollDice` : invoked when pushing the "Roll" button in the Generic dice roller frame.

`ApplyStatusEffect` : invoked when pushing the "Apply Status Effect" button in the Status effects frame. It gets the information from the entry boxes, stores them in the dictionary `self.statusEffects`, then calls the method `UpdateStatusEffectText`.

`TurnStatusEffect` : invoked when pushing the "TURN" button in the Status effects frame. It decreases the remaining duration of each effect in `self.statusEffects`, deletes any that have expired, then calls the method `UpdateStatusEffectText`.

UpdateStatusEffectText : called by both the ApplyStatusEffect and TurnStatusEffect methods, this updates the label showing the list of current status effects.

UpdateHP : invoked when pushing the "Apply damage" button in the Current stats frame. It gets the amount of damage from the entry box, decreases the variable self.currentHP appropriately, appends the new damage to the list self.damageHistory, then updates both the label showing the HP and the Damage History.

ShiftDamageHistory : called by UpdateHP, when appending the new damage to the list self.damageHistory, and that list has reached its maximum length (10). This function shifts all previous damages to the left, then appends the new damage to the far right.

UpdateInitiative / Update<save> : invoked when pushing the "Roll Initiative" / "Roll Save" button in the Current stats frame. It generates a random 1d20 number, adds the stored modifier to it (self.initiative / self.<save>), then updates the appropriate label.

UseAllWeapons : invoked when pushing the "Roll all weapons" button in the Weapons frame. It goes through self.weaponList, the list of instances of the class Weapon, and invokes each ones UseWeapon method.

ClearModifiers : invoked when pushing the "Clear" button in the Weapons frame.

Class Weapon

A new instance of this class is created for each weapon, inside the instance of MonsterGUI. Its methods are:

__init__ : this method is immediately called when creating an instance of the class. It passes the weapon's characteristics to class variables.

PlaceButtons : invoked during the __init__ method of MonsterGUI, for all instances of Weapon. It defines a new frame for the current weapon, and places the relevant labels and buttons.

UseWeapon : invoked when pushing the "Use" button of the current instance of Weapon. It gets the miscellaneous attack and damage modifiers from the entry boxes, generates the appropriate random numbers for the attack and damage rolls of this weapon, based on the class variables of Weapon, then updates the labels of this weapon to show the results.

Last Updated: 2017-12-05
by A.Komar