Combat

The combat script

Entering the Combat Script

There are a number of functions that are intended as entrypoints into the combat script, which are called by spell scripts. The appropriate function must be called to properly proceed through a combat event.

combat_event_entrypoint
This function serves as the entrypoint for simple combat events, where damage is dealt or healing is done. The function uses the spell, the source, the target and an optional value as arguments, and checks whether the spell is a damage or healing spell to call the next function, which is either combat_event_damage or combat_event_heal.

A value can be prescribed, which will cause the combat event to use that specific value, instead of calculating the value from stats. If no value is prescribed, the default value of -1 is used, which causes the following functions to calculate the actual value.

combat_event_aura_entrypoint
This function serves as the entrypoint for the application and removal of auras, i.e. dots, hots, buffs, debuffs, and absorbs. The function uses the spell, the source, the target, and a boolean as arguments. The boolean is optional, with a default value of false. If the boolean is set to true, the aura is removed instead of applied.

buff_application
This function serves as the entrypoint into the application of a buff or debuff, and leads to the recalculation of stats based on buffs and debuffs. The function uses the spell, the source, the target, and a boolean as arguments. The boolean is optional, with a default value of false. If the boolean is set to true, the buff is removed instead of applied, and the stat calculation is done appropriately.

value_query
This function calculates the damage or healing value of a spell without the following application of this value. It is useful to calculate the magnitude of absorb shields, and possibly for snapshotting values of dots and hots. It uses the value_modifier of the spell, the value_base stat and the active modifier (i.e., damage_modifier or heal_modifier) of the source, and the passive modifier (i.e., defense_modifier or heal_taken_modifier) of the trget to calculate the resulting value of a spell. This is the same function that combat events that deal damage or healing eventually use to calculate the value of a spell.

Combat Events

Combat events are damage, healing and aura events that are reached via the entrypoints into the combat script. They calculate the magnitude of a spell, unless prescribed, check for avoidance and ciritcal hits, apply the hp change or aura to the target, and log the interaction.

combat_event_damage
This function handles damage events. It uses the spell, the source, the target, and the value as arguments. If the value was not prescribed when calling the entrypoint, it is -1 at this point, which causes this function to calculate the actual value as a first step. It then checks whether the spell is avoidable, and if so, if it is avoided. If it is not avoided, it checks whether the spell can hit critically, and changes the value accordingly if the hit is critical. This leads to the finalized value, which is then first applied to the absorb shields on the target, if they are present, and then to the health of the target. The interaction is then logged.

combat_event_heal
This function handles healing events. It uses the spell, the source, the target, and the value as arguments. If the value was not prescribed when calling the entrypoint, it is -1 at this point, which causes this function to calculate the actual value as a first step. It then checks whether the spell can hit critically, and changes teh value accordingly if the hit is critical. This leads to the finalized value, which is appriel to the health of the target. The interaction is then logged.

combat_event_aura
This function handles the instantiating and adding of aura scenes. It uses the spell, the source, and the target as arguments. First, the listed name of the aura is created, which is the name that the child scene will have, and the name that will appear in the aura array of the target. It consists of the name of the spell, and if the aura is not unique (i.e., multiple sources can apply the same aura on one target), the name of the source is appended. This array is sued t track which auras are present on a target. If the aura is already present, it is reinitialized by calling the reinitialize function in the already present aura script. If the aura is not yet present, the appropriate aura scene (dot, buff or absorb) is instantiated, the name is changed to the previously generated listed name, the aura is initialized and added to the appropriate aura container, and the aura name is appended to the aura array of the target. The interaction is then logged.

combat_event_aura_remove
This function handles the removal of auras from a target. It uses the spell, the source, and the target as arguments. First, the listed name of the aura is created, which is the name that the child scene will have, and the name that will appear in the aura array of the target. It consists of the name of the spell, and if the aura is not unique (i.e., multiple sources can apply the same aura on one target), the name of the source is appended. The aura name is then erased from the target's aura array, and the scene is removed from the aura container. The interaction is then logged.

Checks

There are a few checks that the combat script requires to function.

is_critical
This function checks whether a hit is critical. It uses the crit modifier of a spell and the base critical hit chance of the source as arguments. A random number generator is initialized, and a float between 0 and 1 is generated. If the generated float is less than or equal to the sum of the base critical hit chance and the crit modifier, the hit is critical. The result for a critical hit is 1, and the result for a non-ciritcal hit is 0, as the return value is used for a calculation.

is_avoid
This function checks whether a hit is avoided. It uses the avoidance chance of the target as the argument. A random number generator is initialized, and a float between 0 and 1 is generated. If the generated float is less than or equal to the avoidance chance of the target, the hit is avoided.

Value Application

The final part of the combat event sequence, before the logging, is the application of the final spell value to the absorb shields and health of the target.

apply_damage
This function applies damage to the health of the target, and returns the difference between the final spell value and the target's current health before the damage is dealt as overkill. It uses the value to be applied and the target as arguments.

apply_absorb
This function applies damage to the currently active absorb shield on the target, in order of the duration-sorted array of absorbs. It takes the value to be applied, the target, the spell name, the source name and the target name as arguments. The function attempts to deplete all absorb shields in order, until either the remaining damage value reaches zero, or all absorb shields are depleted. For every absorb shield that has damage applied to it, a combat event is logged to display which shield from which source is absorbing how much damage. If a shield is depleted, it is scheduled for removal after the iteration during which the damage is applied. The depleted absorb shields are then removed directly afterwards, as doing so during the iteration leads to unpredictable results. Finally, the remaining damage value is returned. If the remaining value is 0, the combat_event_damage function is exited. Otherwise, it continues with the application of the remaining damage to the target's health.

apply_heal
This function applies healing to the health of the target, and returns the overheal value. It uses the value to be applied and the target as arguments. A min() function is used, as healing cannot heal a target beyond maximum health.

Stat Modificiation

Stat modification occurs when a buff or debuff is applied to a target, and an array of stats is changes either additively (positive or negative) or multiplicatively (with values greater than or less than 1).

apply_buff
This function applies the stat modifications of a buff to the stats_add and stats_mult dictionaries of the target, and calls the calc_current_from_base_partial function, which recalculates the modified stats with the newly updated modifiers. It uses the spell, the source name, and the target as arguments.

remove_buff
This function removes the stat modifications of a buff or debuff from the stats_add and stats_mult dictionaries of the target, and calls the calc_current_from_base_partial function, which recalculates the modified stats with the newly updated modifiers. It uses the spell, the source name, and the target as arguments.

calc_current_from_base_partial
This function calculates the current stat values from the base value and modifiers. The word partial refers to the fact that not all stats are recalculated. It is intended for use when the character is affected by a buff or debuff, or a talent that increases a singular or only few stats. The function uses the target and an array of modified stats as arguments. The array of modified stats are looped over, and the additive and multiplicative modifiers are first then determined, and then applied to the base stats. For changes in maximum health and resource, the difference to the previous value is stored. If this difference is negative, i.e., the maximum values decrease, the current value of health or resource are set to the minimum of either itself or the new maximum, to avoid overcapping.

calc_current_from_base_full
This function calculates the current stat values from the base value and modifiers. The word full refers to the fact that all stats are recalculated. It is intended for use when the character class or role is changed, as this affects most or all stats. The function uses the target and an array of modified stats as arguments. The array of modified stats are looped over, and the additive and multiplicative modifiers are first then determined, and then applied to the base stats. For changes in maximum health and resource, the difference to the previous value is stored. If this difference is negative, i.e., the maximum values decrease, the current value of health or resource are set to the minimum of either itself or the new maximum, to avoid overcapping.

Log Messages

All combat events are logged, to ensure that combat encounters can be analyzed. No in-game analysis exists currently, but is planned to be implemented at some point. It is not actively being developed at the moment, however.

In the future, it is intended that combat log messages can be saved as a text file, such that the file can be subjected to post-processing. Some post-processing scripts will probably be developed by cheese at some point, but are not currently being actively developed.