Creating New Spells

New spells are intended to be simple to add, if they do not do esoteric things. In particular spells that deal damage, heal, apply a buff, debuff or absorb shield, or combine these elements, should be fairly simple to implement. To add a new spell, a few steps are required.

Creating the Database Entry
The spell database data/db_spells.json needs to be appended with an entry for the new spell. This is done by creating a new entry with a new unique ID, and specifying the key and value pairs. Which keys are required depends on the type of spell that is added. More information can be found here.

The spell also needs to be added to the spell_list of the unit or interactable that should use the spell, in either the data/db_stats_player.json file, the data/db_stats_npc.json file, or the data/db_stats_interactable.json file.

Creating the Spell Script
Every spell scene needs an attached spell script that handles the triggering of the spell. The script should inherit from the BaseSpell class, which contains some core functionality. The _ready function of the spell script should set the ID as a string, and call initialize_base_spell with the spell ID as the argument. This creates the dictionaries that contain the spell data, and the cooldown timer. The script should further contain a trigger function, which determines the source and target of the spell, and goes through a number of checks that determine whether the spell can actually be used when it is triggered. The standard checks, such as whether the spell is already on cooldown, whether the target is within range, etc., are contained in the BaseSpell class.

After the checks, the start_cast function of the BaseSpell class can be called, which sets the is_casting state to true and triggers the gcd if applicable. This requires the cast_success function to be passed as an argument, as it links the timeout of the cast timer to this function. The cast_success function then needs to be added to the spell script to determine what happens when the spell is successfully cast. This usually consists of applying the resource cost, sending the combat event, and calling finish_cast. finish_cast is a function in the BaseSpell class, which plays an animation, disconnects the cast timer timeout signal from the cast_success function, starts the spell cooldown if it exists, and sets the is_casting state to false.

Instant cast spells can forego the call to start_cast, as the cast finishes immediately. Instead, the trigger function can send a combat event, trigger the gcd, and call finish_cast. finish_cast requires the cast_success callable as an argument, for which the BaseSpell class has a dummy function that does nothing, and is intended for use in this case.

Note: Spells that increase the maximum health of a target by applying a buff should in principle also send a second combat event to heal the target for that amount. This is done to avoid reducing the health percentage every time such a buff is applied, and to make sure that a tank increasing his maximum health is also healed for that amount, as doing otherwise might feel clunky. If the heal component is not added, it should be carefully reasoned why not. The heal spell requires a separate spell dictionary to be created, which requires the spelltype key to be set to "heal", and requires the name and can_crit keys. If no value is prescribed (which can be constructed from the buff data), more keys are required to function as a healing spell. Spell ID 8 (Player Test Buff) is an example of a buff that increases maximum HP.

Revision #9
Created 2024-03-20 09:25:19 UTC by cheese
Updated 2024-10-06 08:45:29 UTC by cheese