Snorlax's tools

Spell System Version 0.01

Can't believe I actually finished it

This will be small documentation exploring the current systems and features implemented and the general knowledge needed in operating them. This project was started god knows when and has been shelved more times than I have fingers due to lack of knowledge and motivation.

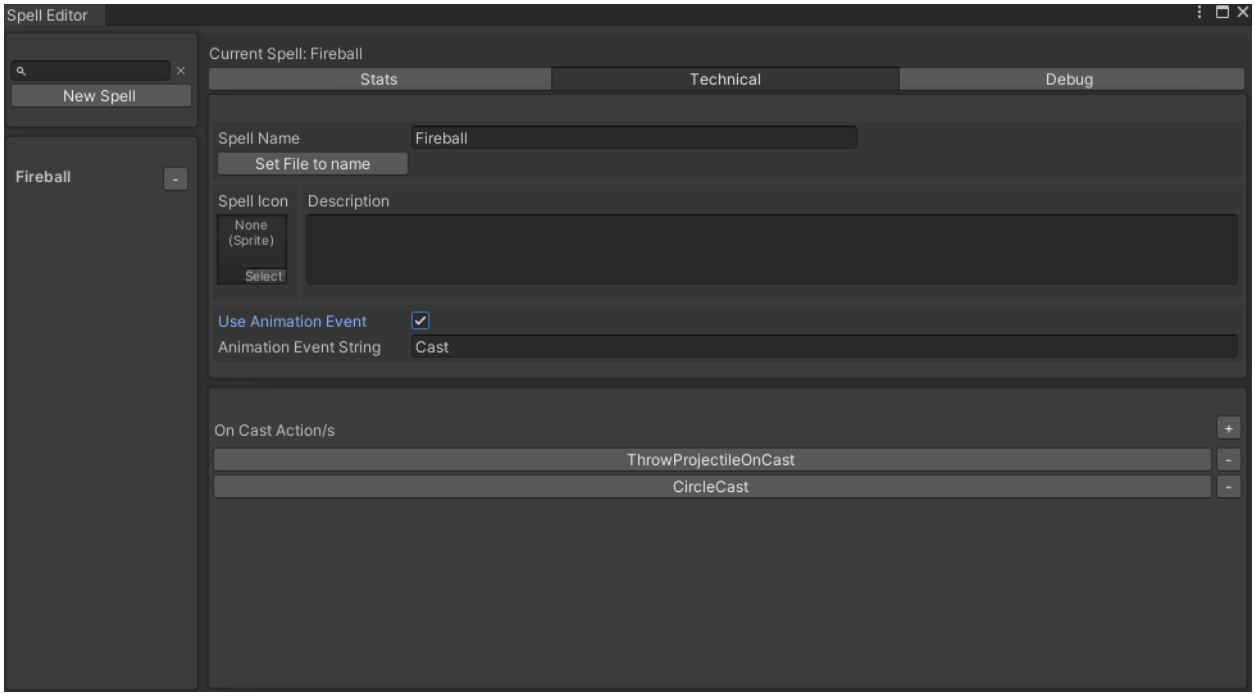
The modular spell system is my solution to their unattractive cousin the hard-coded systems while being flexible and easy to use, with more features and UI updates planned in the future.

Currently, the demo scene used is one from Sebastian Grave's demo scene, script, and animations hence this rendition of the system will be limited to friends for testing. Link to Sebastian's asset: (Will only allow smiths to use this package)
https://assetstore.unity.com/packages/3d/animations/straight-sword-animation-set-220752

Magic System

The crème de la crème, the magic system uses abstract class scriptable objects for two types of actions, OnCast and OnHit. OnCast as the name implies, activates the given list of OnCast actions on the cast with the same occurrence happening with the OnHit actions however, the OnHit actions have the ability to trigger on the max range.

In order to make creating spells and actions without flooding the project files with scriptable objects, the actions are parented to the spell giving the ability to remove actions with ease.



To the left panel in the search bar, a button to create new spells, and a list of spells that can be selected and or deleted. To the right panel, variables are split to keep the editor clean and will be explained below. However, below that, the On Cast actions can be added, selected, edited, and deleted from the spell.

Stats Variables

Cooldown - The number of seconds before the spell is able to be cast again.

Range - Distance of spell targeting allowed, it won't interfere with the max projectile distance.

Cast Direction - Determines how spells will react with direction.

- None - No cast direction hence spells will start at the caster's location.
- Forward - Cast direction will go in the direction of the camera forward (specific for the player with the demo aim script).
- Target - A form of lock-on that targets an entity.
- Indicate - Can direct cast direction and targeted position with ray cast from the camera.

Radius - Determines the radius of the cast for some actions.

Detect Layer - Targeted layers for detection in both the aimer script as well as some actions.

Surface Indicator - Instantiates an object to be used for visual targeting.

Surface Indicator Offset - Gives the indicator at an offset when visualizing.

Technical Variables

The variables here are pretty self-explanatory except:

Use Animation Event - If true, blocks normal casting and plays a selected animation state that can trigger an animation event.

Animation Event String - This is the string name input for the animation state to play.

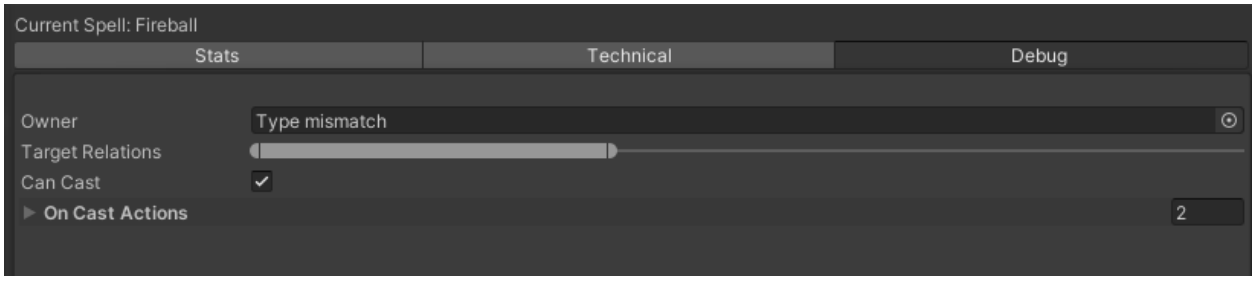
Debug Variables

Owner - owner or caster of the spell.

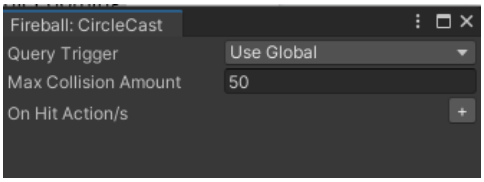
Target Relations - The min and max variables for the target relation value (can target enemy or ally which is based on their relationship value).

Can Cast - Boolean controlled by the cool down variable.

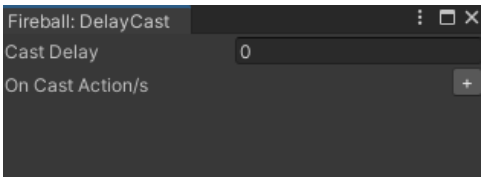
On Cast Actions - The base list of On Cast actions for the spell.



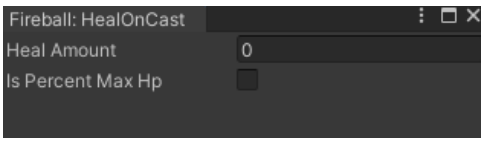
On Cast Action List



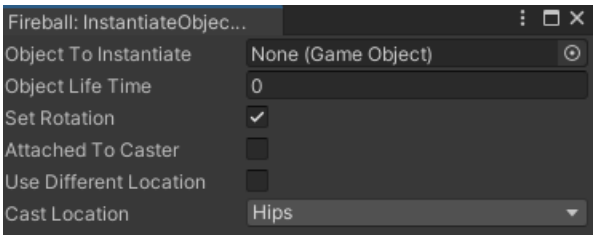
- Circle Cast
  Casts a OverlapSphereNonAlloc at target or indicated position.



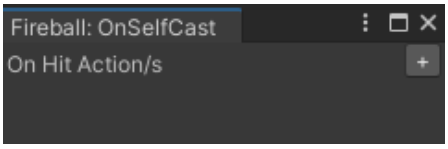
- Delay Cast
  Any On Cast actions in delay cast will activate after a set amount of time.



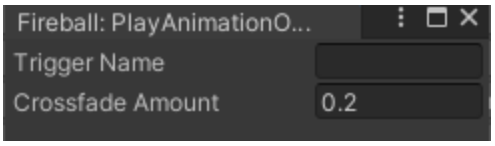
- Heal On Cast
  Heals the caster based on the amount inputted however, if the boolean 'Is percent Max HP' is ticked the value will then act as a percentage.



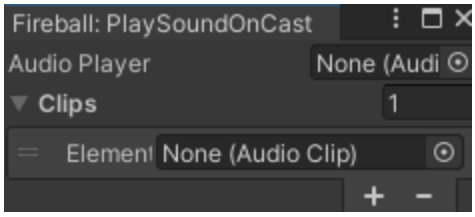
- Instantiate Object On Cast
  Instantiates an object at the position of the caster's origin (set from aimer) however, the boolean to use a different location will take cast location as the new origin. This uses the animator getbonetransform function. The object will also be destroyed after a given amount of time except if the number is 0, in which the object will be permanent. There are also options to set rotation based on cast direction and the ability to parent the object to the caster.



- On Self Cast
  Will activate a list of On Hit Actions with the caster as the target.



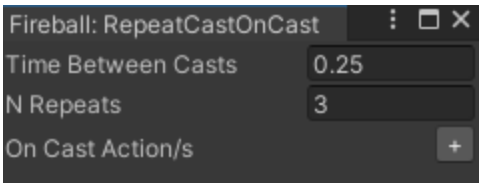
- Play Animation On Cast
  The caster will play the animation state inputted in the trigger name with the desired crossfade amount.



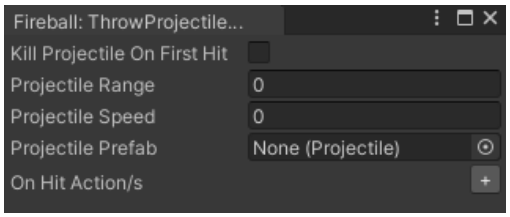
- Play Sound On Cast
  Plays a random sound clip from an array given from the base audio source attached to the caster.

On Cast Action List - continue

- Repeat Cast On Cast
  Repeats a list of Cast Actions a desired number of times with an inputted delay between each case.

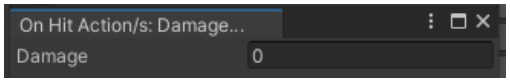


- Throw Projectile On cast
  Creates a projectile based on a premade prefab with inputted range and speed and can be enabled to be destroyed on first contact.

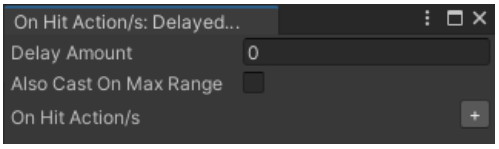


On Hit Action List

- Damage On Hit
  Deals a certain amount of damage to the target.

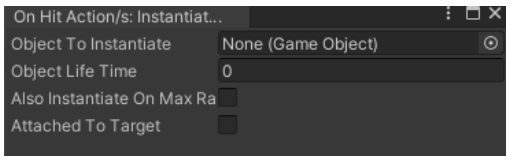


- Delayed On Hit
  Works the same as the On Cast counterpart with uses On Hit actions and allows to enable the On Max Range method of On Hit actions.

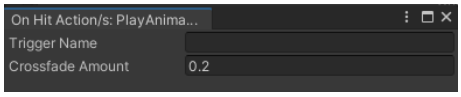


- Heal On Hit
  It works the same as On Cast counterpart however, the target is healed.

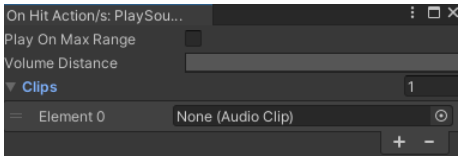
- Instantiate Object On Hit
  It works the same as On Cast counterpart however, attached to the target and can instantiate on the max range. Some variables are also missing due to working with a target instead.



- Play Animation On Hit
  Makes target play an animation.

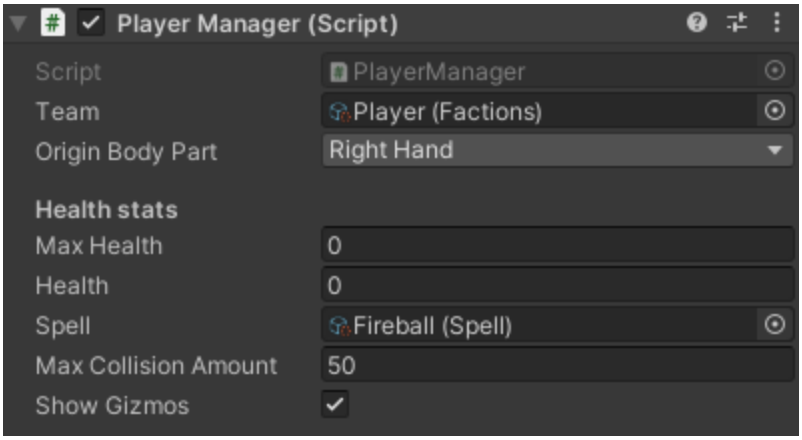


- Play Sound On Hit
  Creates a game object with an audio source to function like the On Cast counterpart with an animation curve to control volume via distance.



How to implement

Spell Aimer (Player Manager script - inherits from Character Manager)



The demo scripts provided give an idea of how to implement spell aiming with Character Manager being the way spells can identify and interact with entities. Aside from the Team variable and Origin body part which controls where the spell is cast/instantiated from, everything is self-explanatory.

Player Manager script - Methods

The script itself has some summaries explaining its functionalities as well as the general naming convention making it easy to identify the purpose however, I will be mentioning the essential methods though not to say the entire script isn't essential.

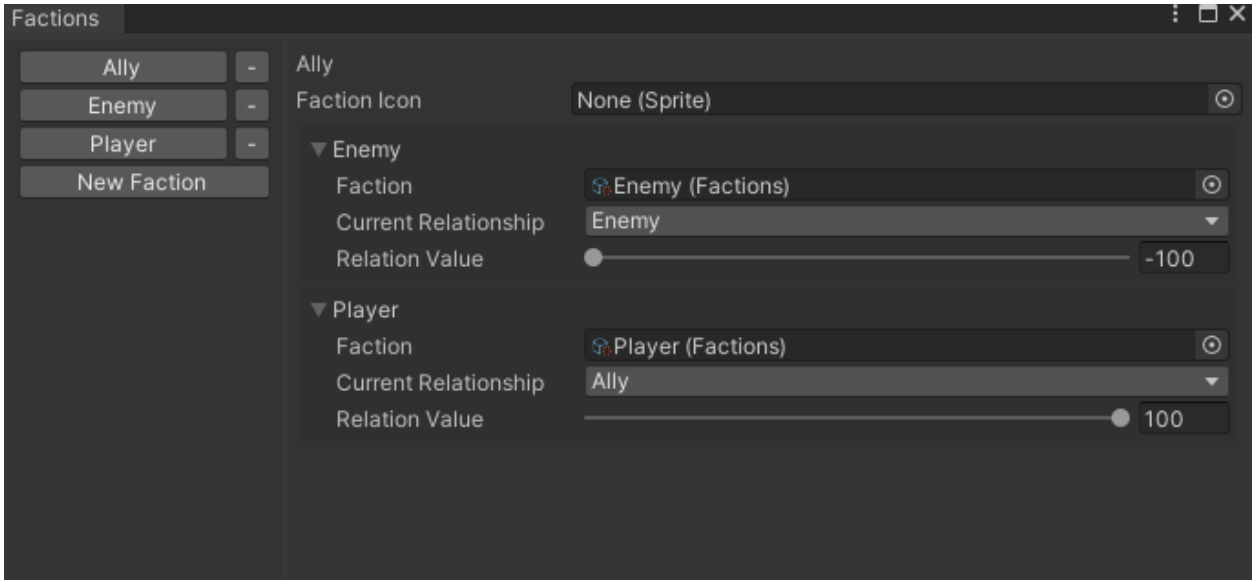
#Region Main Methods ChangeSpell(Spell newSpell)
Changes out the spell as well as instantiate the surface indicator in the spell while destroying the old one. If no indicator is found, an empty game object is created instead.

#Region Main Methods CastInputMethod()
The brains of the script and activated in Update(), determines inputs and information from the current spell.

#Region Cast Methods CastSpellEvent()
This method is used for the animation events in animation clips to have a better result of casting spells at a specific frame of an animation. (Im assuming you know how to add an animation event.)

Faction System

Getting a two for one, the faction system is just used to easily identify whether a target is a friend or foe based on their relationship value. This system comes with its own easy to use editor and will create scriptable objects which hold the faction data and relationship values with all other factions.



It follows the same design principle as the spell system editor, however, simplified like the lack of a search bar. The right panel indicates which faction is currently selected alongside the relationship with each other faction in the list.

Relationship Value

Relation Value controls the Current Relationship based on the number value as seen in the picture below.

```
public RelationStat valueRelationship =>
    relationValue < -75 ? RelationStat.Enemy
    : relationValue > -75 && relationValue < -25 ? RelationStat.Wary
    : relationValue > -25 && relationValue < 25 ? RelationStat.Neutral
    : relationValue > 25 && relationValue < 75 ? RelationStat.Indifferent
    : relationValue > 75 ? RelationStat.Ally
    : RelationStat.Neutral;
```

Settings Editor

This controls the folder location for both the spell and faction system when a scriptable object is created via either editors. This uses PlayerPref to set and get keys which are the folder path in stream form.

In truth, I stole this when reverse engineering some Unity assets which had editors. “Hippity hoppity, your code is my property.” - Man about to be sued 2022

Its bloody 2:38 am. Imma make a better version when the editor tool is more updated :u