Code Documentation:

When two cards interact each other, in order to apply effects, both cards need to receive information from each other. Notably the effects array and status array, instead of doing all of this complex shit to evoke effects. Perhaps after the attack, it will call a simple applyeffect with the enum as a parameter (so that the game can check if the method is a attack, or take damage ability etc )method from a singleton, then applyeffect will iterate through the effects or status array received by the other card to apply effects accordingly.

Implementing more complex card effects would be harder though, however the best and easiest way that this can be implemented is through a singleton (global)variable and method. The card will send a signal to the singleton method would take in a question and then output an answer to store in the singleton variable. The card will then take form this singleton variable and apply it to appropriate calculations.

-alternative for inheritance

Instead of inheritance and creating 100+ resources, the type of card effect can be applied through aggregation

This effect resource will also have an enum value called trigger, based on this trigger, the card can be selectively be chosen to activate whenever. It will also have a value called delay, where it causes the countdown() function to create timers based on the delay needed (required for card drawing/discarding effects).

To create applyEffects with different applyEffects, the enum value can be used in the applyEffect() method parameter. Thus this applyEffect can be overloaded to have multiple applyEffects with their own execution. An example applyEffect can be applyEffect(card, enum)

Alternatively there is no need for a resource value, the card itself can simply be passed into the method, and then the values changed that way. Assuming that godot passes by value. We can do something like effect.applyEffect(self), and the applyEffect would change the values from there.

Card abilities

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Trigger | description | Actions sequence |
| ApplyEffectOnAttack | onAttack() -> onTakeDamage() | This ability applies an effect on the opponent upon attacking | OnAttack(effect)  onTakeDamage takes effect  effect.applyEffect() |

Status effects are a little hard to implement. Most status effects go away after a certain trigger(onAttack() or onTakeDamage() etc). Therefore a check might have to be done in those methods to activate the effect.

Status effects would also be stored in an array within a card called “status”.

Instead fo apply status, we can just duplicate the array append() or remove() from the duplicated array and set this new array to the variable

Status effects

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | description | Actions sequence | triggers |
| Hex | Debuff | This effect reduces the opponent’s attack for one attack | Receiving the Effect:  Effect.applyEffect(self)-  Adds the status effect to the array  applyStatus() -  Add it to the status array and decrement the card’s attack by X  Losing the effect:  onAttack()-  run activateStatus then applyEffect  activateStatus() –  remove it from array  applyStatus() –  Recheck the card’s status array to remove it from the card. Then increment the card’s attack by X | onAttack() |
| poison | Debuff | This powerful effect causes the opponent to take damage after you take an action. (passing also counts!) | Receiving the Effect:  Effect.applyEffect(self)-  Adds the status effect to the array  applyStatus() -  Add it to the status array | onAction()  use await to delay the action based on no. of cards with poison effect |

Example mockup of a card that applies poison to opposing card:

onTakeDamage(othervariables, effectsarray):

#damage logic

For effect in effectsarray:

If effect.type == applyEffectOnAttack:

Effect.applyEffect(self, applyEffectOnAttack) #evoke the card effect resource onto the card

applyStatus() #evoke the applyStatus() method to apply the logos and appropriate stat changes

applyStatus():

var effectRes = new EffectResource

increment = 0

for status in statuses:

EffectResource.status(self, status)

Status.add\_child(status.symbol)

Status.symbol.position = ???\*

Status.symbol.position.y += increment \* 100

Increment += 1

\*godot 4 is a bit glitchy when setting node positions so a node may need to be set in place to spawn the symbols