Necromancer Game design doc

**Bugs-**

|  |  |  |  |
| --- | --- | --- | --- |
| Bugs | | | |
| Severity | Occurrence | description | Potential solution |
| mid | Very rare | Sometimes the mouse doesn’t focus on a card after drawing them | Use a for loop to check for cards in the focusstate in the $hand node, if no cards are in a focus state, warp the mouse cursor a bit |
| low | Common | The mouse sometimes focuses on two cards, causing one card to bug out for half a second | Use a for loop to check for cards in the focusstate in the $hand node, if there is a card in a focus state, do not focus |

**To do list:**

Finish a prototyped battle system:

~~-Change to rich text labels~~ done!

~~-Energy system~~

-Refactor the code to better encapsulate the methods (spellcard functions inside spell card, card functions inside card)

- implement drag and drop (to replace click and assign)

-design effect architecture

-implement ongoing spell cards

Features-

Order of priority:

Effects

Database

Enemies

Map

Characters

Game design

Etc

**Gameplay description:**

The untitled game is a turn based card roguelike that focuses on pitting a person with a crowd of allies against a single enemy.

The player has three energy, energy can be used to play summon cards and spell cards.

Cards cannot be played once out of energy.

Summon cards summon a creature on the board, in which they can be tasked to either defend or attack at the end of a turn.

Spell cards have a variety of effects, they can debuff or deal damage to an enemy, or buff or heal an ally.

Spell cards can also be “ongoing”, which is to say, whenever they are played, instead of moving to the discard, they will be “inplay”, causing their effects to activate every turn.

**Summon cards and end turn:**

Each summon card has an attack stat and hp stat.

The attack stat deals damage if its in an attack position while the hp stat will block damage for you.

After clicking the end turn button, summon cards can be selected to either attack position or defend position. The player can click the “finish setting” button to finalise their decision.

Cards in the attack position will damage the enemy according to their attack. Cards in the attack position will not defend for you.

Cards in the defend position will defend against enemy attacks for you, enemy damage will be spread out based on the number of defending cards, rounded down (eg enemy attacks for 18, 4 cards are in the defence position, damaging each card for 4.5 rounded to 4, every card takes 4 damage). Cards in the defend position will block damage according to the amount of hp they have left. If an enemy’s attack is higher than the total hp of defending cards, you will take damage.

If a summon card runs out of hp, they will be removed from combat permanently. They will be returned to the deck after the combat.

**Balancing concerns:**

The system may favour spell cards more, so spell cards should be made to not deal a lot of damage, instead focusing on the utility of the cards(maybe instead of dealing damage directly, spell cards can add a “damage charge” to a summoned creature, causing them to deal higher one off damage). Spell cards should be used for utility purposes, helping the player make their summons last longer or buffing existing summons.

**Instances of “code magic”:**

Code magic is a term used to describe when code suddenly does weird things or does things it is not supposed to do when you change it, this section will cover instances of code magic and tries to explain them. So that modifying the code will not be so difficult when new features added.

Note that is section aims to explain why the code behaves the way it does for easier modification and is not a section to solve problems in code.

In a way this section is used to test the code’s quality.

Explained - code magic is explained, problem isn’t necessarily solved but documented to allow for easier changes to the code

Magic - code is seemingly magic and there is no explanation for the logic behind it…for now.

|  |  |  |
| --- | --- | --- |
| Incident | status | explanation |
| Changing the code while holding a card to deselect a creature on the summon board does not work even by changing the elif statement. | explained | The first if statement already has a check to check if the left click is pressed and a card is held, therefore no matter how much the elif statement is changed, it wouldn’t be affected as you are already holding a card. To modify the code from here, another elif statement has to be nested inside the first if statement instead of changing the elif statement itself. |
| Desc label in the card’s node tree will not be programmatically affected if named a certain way | magic | Don’t know what causes this, however simply renaming the textnodes desc label to bigdesc seemingly solves the problem |

**Effect structure and descriptions:**

Effects are everything in a card game, this section will discuss how to structure the effects in a card using OOP so that new effects can be added easily.

Before effects can be implemented, first effects are catalogued so that a good idea can be gotten from the interactions between the nodes.

**Planned architecture:**

Each card will have an array, this array will contain “trigger” objects, each Trigger object has variables and a method describing its trigger.

Effect objects start from a base effect object with generic or no methods or variables, then this object can be inherited into a summonTrigger or spellTrigger.

SummonTriggers and spellTriggers are further inherited into trigger types, such as onAttack, onAnyattack etc, the trigger type of the summoneffect decides when the effect activates.

Each effect trigger type will also have another array of effects, these are generic effect objects and can be put inside any summon or spell trigger effect object.

Generic effect objects start from an empty effect object and are further inherited into different types(poison enemy, buff attack etc). However some effects may have to be differently applied depending if the card is a spell or summon. In this case, the effect is further inherited into summoneffect or spelleffect.

A picture containing text, diagram, screenshot, line

Description automatically generated

We can deduce a generic dictionary structure for any effect for the card:

(Array of dict)

[{triggerCondition:{effect:X,effect2:Y…}}]

Essentially speaking, the superclass trigger effect will handle the trigger condition while the effects will handle the status.

Starting simple first…

Methods have to be devised for every trigger condition so that the card will activate it upon fulfilling it.

**Triggers:**

To better understand triggers, we will categorizing them into different categories based on how they would affect the battlefield.

SelfTrigger - easy to understand, this card triggers it’s effect when it does an action itself or its stats are affected. (this card gains attack on summon or on heal)

fieldTrigger- this card triggers it’s effect when other things affect the battlefield(eg spend energy, this card gains attack. Turn end, this card gains attack), may be hard to implement. We can possibly use groups to streamline the coding for this.

conditionTrigger - this card cant do the things it normally does(attack or defend for summon cards) unless a certain condition is fulfilled (eg this card requires one energy to attack or defend)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Summon trigger effect description datasheet | | | | |
| Effect name | Dictionary structure | summary | Interaction flow | Type |
| OnAttack | {onAttack:{effectdictionary}} | When the card attacks, activate effect. | Card activates effects, signals to effectmanager. | SelfTrigger |
| OnSummon | {onSummon:{effectdictionary}} | When card is summoned, activate the effect. | Card activates effects, signals to effect manager. | SelfTrigger |

|  |  |  |  |
| --- | --- | --- | --- |
| Spell trigger condition effect description datasheet | | | |
| Effect name | Dictionary structure | summary | Interaction flow |
| OnPlay | {onPlay:{effectdictionary}} | When playing the card, activate the effect. | Card activates effects, signals to effect manager. |

**Effects:**

It will be assumed that the trigger effect has been activated.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| generic effect description datasheet | | | | |
| Effect name | Dictionary structure | Spell or summon? | summary | Interaction flow |
| draw | {draw:X} | both | Draw X cards into your hand | card signals to effectmanager which signals to drawval method |
| damage | {damageEnemy:X} | both | Damage the enemy for X damage | Card signals to effectmanager which signals to changeenemyhp method |
| attackUp | {attackUp:X} | summon | Buff the summon by X upon trigger | Card buffs itself which then signals to effect manager. |
| HpUp | {hpUp:X} | summon | Increase HP by X upon trigger | Card buffs itself which then signals to effect manager. |
| damageUp | {damageUp:X} | spell | Increase “damage” effects by X upon trigger | Card buffs itself which then signals to effect manager. |

There will be an effect manager singleton, each time a card is loaded in the scene, it will emit a signal to notify the singleton to connect with it.

The singleton attempt to connect an effect to the card based on its type.

**design patterns:**

The decorator design pattern is a pattern in which attributes can be easily added onto a object.

The decorator design pattern makes use of an interface to implement methods. godot however doesn’t have one. Normally, they keyword “implement” would be used to create an interface but godot doesn’t have this feature. In order to implement it, godot would have to use the extend keyword.

If effects are implemented under this design pattern, multiple effects can be assigned to a card and be evoked at the same time, however when an effect has a unique method assigned to it, some weird workaround with ducktyping has to be done in order to evoke the method.

We might have to return to using an array of effects to implement effects for the cards.

Another way that we can consider is adding the component as a child to a card. Though it wont be too different than using an array though.

Helpful resource:

<https://medium.com/@AlexAndDraw/honestly-im-not-entirely-sure-what-this-post-is-949593e8cdb9>

**UI analysis-**

Recording down font size as godot inheritance doesn’t work for fonts

Godot pls fix yo shit

Font sizes for cards-

Type/name – 18 -24

Desc – 14 -18

Energy – 30 - 39