Purple :STACKS **CdC** *[V.]*

[Responsables] : Toinon
[Dernière Modif] : 28/06/2025 : 20:00

Sommaire

Système

GAMEPLAY/ Stacks –

V	CONTENU
1	› Tray Behaviour : *TeammateTray_Behaviour, Tray_Data & TrayAccessor, TrayStacks* › Tray View : *Tray_View, Cart* › Stacks Identity : Dash, Sparks, Edge
2	› Stacks Identity : Acid Drops ›
3	›

Specs Externes

- Nextcloud → **pu_GAMEPLAY-06.4**

► Stacks :

Bonus stacks	Shields Modifiers/Ammodifiers	Stacks Négatives
● Edge ● Sparks ● Dashes	● Shields	● Acid Drops

►Stacks Drawer

Element de UI qui s'affiche lorsque les shifters ont des Stacks et/ou des bonus à **distribuer** et qui permettent de répartir ces bonus/stacks sur différents Teammates.
Le Drawer est commun à la
Quand le drawer est vide, il disparaît. Le drawer apparaît dès l'instant ou il y a des bonus/stacks à distribuer.
Dans

Distribution → Distribuer des Stacks doit pouvoir se faire de 2 manières :

A] Drag & Drop : Le bonus/l'élément de Stack est drag à l'aide du curseur de la souris sur une fiche de perso, puis est appliqué sur ce perso lors du drop.

B] Sélection (Click) sur le Modifier/La Stack

→ Le drawer doit être extensible : taille proportionnelle au nombre d'éléments affichés

→ On peut choisir **combien** d'élément d'un type de Stacks on donne à un Teammate :

1. Le joueur sélectionne l'élément à distribuer, ce qui en pré-sélectionne 1 de ce type.
2. En re-cliquant sur ce même élément, ou en utilisant sa molette, le joueur peut choisir la quantité d'élément de ce type à distribuer à un teammate cible.
3. Si le joueur clique sur un autre élément, il l'ajoute au groupe d'éléments à distribuer à un teammate cible

4a. Le joueur **valide** sa distribution en cliquant sur le Teammate cible

OU

4b. Le joueur **annule** sa distribution en cours en cliquant sur le bouton « Cancel »

► On peut distribuer en une fois un groupe d'éléments différents en provenance du Drawer, a un teammate unique

►Tray (View)

→ Le Tray est toujours affiché au niveau de la CharaSheet d'un teammate
→ Le Tray affiche les Stacks & les Modifiers
→ Shields Modifiers ou Ammodifiers
→ ♠ *Refaire les Icônes d'Ammodifiers & Shields Modifiers*

Ordre d'affichage dans le Tray
(*left to right*)
- Tous les Bonus (D/E/S)
- Les Shields
- Les Stacks Négatives (Acid)
- (Optionnel) Target

→ Nécessite une Anim de dépense pour toutes les Stacks
- ● Dashes, Edges, Sparks → Anim de dépense classique
- ● Bouclier → Destruction du Bouclier, ou dépense comme les bonus Stacks
- ● Acid → se déclenche en début de tour puis enchaîne sur l'anim de tir
- ● Ammodifier → Automatique au moment ou le Shot touche

UTILISATION DES STACKS

► Toutes les **Stacks Bonus** sont utilisées par le joueur au moment ou il le choisit, **quand le contexte le permet** → 🔗 Contexte d'Utilisation des Stacks Les contexte valides pour dépenser des Stack bonus dépend du Type de stacks (voir ci-dessous)

► Quand le contexte permet de dépenser des élément d'une Stack, cette stack **s'illumine** dans le Tray.

►On permet au joueur de choisir **combien** d'éléments de la Stacks il choisit de consommer au moment opportun. Pour ça, il clique sur une Stack

Sélection des Stacks →
► Dans le cas général, le fait de continuer l'action a laquelle profite la Stack suffit à valider son utilisation.

DASHES

contexte d'utilisation → S'illuminent quand le joueur est dans un contexte d'Access

- Le joueur vient de sélectionner un node Controllé de départ, et on suppose que c'est pour déclencher un access (ex : pas encore utilisé d'access ce tour-ci)
- Le joueur vient de sélectionner un Skill d'Access
- Le joueur est déjà en train d'Access

EDGES

contexte d'utilisation → S'illuminent quand le joueur est dans un contexte de Contrôle

- Le joueur vient de sélectionner un Node Access pour lequel il peut utiliser un Skill de Ctrl
- Le joueur vient de sélectionner un Skill de Ctrl
- Le joueur a déjà investi du Ctrl sur un Node, il peut alors **a posteriori** sélectionner puis dépenser un Edge pour investir 1 de Ctrl de plus

SPARKS

contexte d'utilisation → Les Sparks peuvent servir n'importe quand pendant le tour du joueur.

→ Les Sparks s'illuminent s'il existe une action non disponible actuellement à cause de son coût en énergie qui deviendrait disponible en utilisant un ou plusieurs Sparks.

Utilisation → Clicker sur un Spark, puis clicker sur la jauge d'énergie

STACKS NÉGATIVES

ACID

Specs Internes

I] TRAY

I.1] GESTION DU TRAY

L'ensemble des Stacks (comprennant les Bonus Stacks, les Shields et les Acids Drops) seront gérées dans un nouveau comportement subsidiare au Teammate : le **Tray** (*TeammateTray_Behaviour* dans le code)

► L'ensemble des Stacks existent toujours dans le Tray, mais elles ne sont affichées (et donc interactives) que lorsque le nombre de Tokens est supérieur à 0.

► On appelle **Token** l'unité contenue dans chacune des Stacks. → *Ex* : La Stack de Spark peut contenir 1,2, 3 ou+ sparks, ces chiffres correspondant à son nombre de Tokens.
De ce fait, ce que l'on gagne ou que l'on dépense pour une Stack donnée sont des Tokens de cette Stack, la Stack elle même étant toujours existante, même **vide** (0 Token).

► Quand on traite la réduction d'une Stack, on fait la distinction entre sa dépense (**expense**) et sa destruction (**destruction**). La destruction consiste toujours à vider la Stack entière, alors que la dépense se fait Token par Token. L'**effet** d'une Stack se déclenche lorsqu'un Token de cette Stack est dépensée → *Ex* : Quand un Shield est dépensé (c'est à dire après un ShieldBreak), l'effet de son ShieldModifier se déclenche.

- Le Tray est conforme au Datapattern : Il possède ses données (🔗 Tray Datas) en lecture seule depuis l'extérieur du Tray, et en lecture/écriture depuis le Tray lui même.
Pour appliquer un changement dans le Tray (et donc affecter les Stacks), les autres systèmes passent par les Game-Instructions (voir ci-dessous).

- Le Tray est accessible depuis les Teammate_Data → *champ Tray*

- Le Tray est initialisé depuis l'Init des Teammates, via *TrayBehaviour.SetTray()*

► Le Tray récupère la **responsabilité des Shields** : possession des données & game instructions afférentes. Cela ne change en rien le comportement des Shields,
Refacto → On gardera le lien vers les Shields qui existe déjà dans les données des Teammates, simplement en les redirigeant vers le Tray

► **Game Instructions :**

Instructions

Ins-Add-Stacks
→ Rajoute des unités de Stacks d'un Type donné

- *int* **ShifterID** → Cible
- *StackType* **StackType** → Type de Stack Concerné
- int **StackCount** → Nombre d'unité de Stacks à Rajouter

Ins-Remove-Stacks
→ Retire des unité de Stacks d'un Type donné

- *int* **ShifterID** → Cible
- *StackType* **StackType** → Type de Stack Concerné
- int **StackCount** → Nombre d'unité de Stacks à enlever
- bool **destroyed** → Les Stacks ont elles été détruites ou simplement dépensées ?

Ins-Raise-Shield

I.1a] Tray Datas

► Le Tray contiennent les données de chacune des Stacks :

- *ShieldStack* **Shield Stack** → *Refacto Shields Stacks atterit ici depuis les TM_Datas*
- *TrayStack* **Dashes**
- *TrayStack* **Edges**
- *TrayStack* **Sparks**
- ??? **Acid Drops** → *A venir*

Ces données sont représentées par une classe commune : **TrayStack**

► **Tray Stack** (hérite de *IStackable)*, contient toute la logique permettant de compter, gagner, dépenser et détruire des Stacks. Elle contient également une référence au « modèle » de configuration statique de chaque stack par le biais d'une **StackIdentity**, et un **StackEffect** (🔗 Effet des Stacks) représentant le comportement de la Stack lors de son utilisation

› TrayStack :

- ShifterID → renseigné *à l'initialisation, utile pour savoir à quel shifter appartient la Stack*
- **StackIdentity**
- **StackEffect** → *StackEffectProcess*
- Stack Count → *compte d'unités dans la stacks; MàJ depuis les fonctions de IStackable*

► Les différentes **StackIdentity** (une par type de Stack) servent de configurations statiques aux Stacks. C'est dans sa StackIdentity qu'est renseigné l'effet de la Stacks , ainsi que ses autres paramètres, notamment le moment ou l'effet de la Stack doit être appliqué (**Stack Trigger**), ainsi que ses différents paramètres de **View** (Ex : Une icône)

› Stack Identity

- *enum* Stack Type → {Edge, Dash, Spark, Acid Drops à venir}
- *enum* **Stack Trigger** → Détermine quand l'effet de la Stack doit se jouer : {Gain, Spending, Destruction, StartTurn, Activation}
- *EffectProcess* **Stack Effect** → *StackEffectProcess*
- *Sprite* Icon
- [...]

I.1b] Effet des Stacks

► L'effet des Stack est représenté par une variante d'*Effect Process* spécifique aux Stacks : le **StackEffetProcess**. Comme tous les *Effect Process*, donc comme pour les Skills ou les Faculty, l'effet d'une Stack peut impliquer :

- De récupérer des informations sur le **contexte de jeu** pour les traiter
- De déclencher n'importe quelle **Game Instruction**
- De demander à la joueuse de sélectionner une cible valide
- De jouer une animation, ou de créer un délai etc...

En plus de cela, le StackEffectProcess permet de spécifier **combien** de Tokens de Stacks ont été dépensées, et d'adapter l'effet au nombre d'unité de Stacks dépensées (cela évite de déclencher plusieurs fois l'effet pour chacun des Tokens dépensés).

► Comme tous les Effect Process, l'effet d'un Token de Stack ne se joue qu'une fois que l'on a **demandé à le déclencher** (*Process()*). Si l'on veut que l'effet d'une Stack réagisse au contexte

d'utilisation (voir Tableau de cas), il faut que cette logique d'analyse du contexte soit séparée de celle d'application de l'effet de la Stack dans le StackEffectProcess().

I.2] CONTEXTE D'UTILISATION DES STACKS

► Le **contexte d'utilisation** d'une Stack, en particulier pour les Bonus Stack (Edges, Dashes et Sparks), définit quand est-ce que la joueuse à le droit de dépenser cette Stack pour appliquer son effet. La distinction des différentes **StackTrigger** ne suffit pas à modéliser ce contexte d'utilisation, il faut pour cela restreindre les moyens/moments de jeu ou la joueuse peut, via l'interface, **dépenser** ses Stacks.

Pour des raisons d'architecture (*Separation of Concern*), on commence par distinguer :
- Le fait de signaler à la joueuse qu'elle pourrait utiliser une Stacks, à un moment ou c'est pertinent.
  → Stack **highlight**
- Du fait de dépenser effectivement ses unités de Stacks quand c'est autorisé, c'est à dire au moment où l'effet de la Stack **peut** s'appliquer.
  → Stack **Permit**

SPARKS	DASHES :	EGDES :
► **Highlight** : *Il existe ≥1 skill actuellement non-disponible à cause de son coût en énergie, qui deviendrait disponible en utilisant ≥1 Spark* ► **Permit :** *n'importe quand pdt le tour de la joueuse*	► **Highlight** : *la joueuse est en cours d'utilisation d'un Skill d'Access OU la joueuse vient de sélectionner un Node Contrôlé (Node de départ) OU la joueuse sélectionne un de ses Skills d'Access* ► **Permit**: *la joueuse est en train d'utiliser un de ses Skills d'Access*	► **Highlight** : *il existe un node dans lequel du Ctrl a déjà été investit ce tour-ci OU la joueuse sélectionne un Skill de Ctrl OU la joueuse est en train d'utiliser un de ses Skills de Ctrl* ► **Permit**: *la joueuse est en train d'utiliser un de ses Skills de Ctrl OU il existe un node dans lequel du Ctrl a déjà été investit ce tour-ci*

► Highlight & Permit seront modélisé par des *StacksFilter()* qui réévaluerons **true/false** d'après une liste de critères d'évaluation du contexte de jeu applicables aux Stacks à chaque fois que le contexte de la partie changera.

► Quand Permit est **true**, alors il est possible de 🔗 Expend une Stack pour en déclencher l'effet. Cependant, en fonction de la Stack et du contexte de jeu dans lequel se trouve la joueuse, l'effet de la Stack sera légèrement différent. Voir les différents cas d'interaction possible pour activer les Bonus Stacks dans 🔗 Tableau des Cas

II] VIEW DES STACKS

II.1] TRAY VIEW

L'affichage du Tray est représenté par un comportement de View spécifique, le **TrayView**. Le TrayView doit :
- Afficher et mettre à jour le **compte des unités de Stacks** pour chacune des Stacks présentes dans le Tray, soit Sparks, Edges, Dashes et Shields.
- Bonus Stacks – Afficher le **Highlight** des Stacks quand c'est pertinent
- Bonus Stacks – Permettre à la joueuse de **Expend** la Stack dans un contexte permis par son **Permit**
- Global Ammunition Modifier → ***Besoin de précisions sur les Specs***
- *Afficher la target ?*

► Le TrayView est divisé en plusieurs sous-comportements de View, qui gèrent individuellement chacune des Stack affichées, et possède aussi un contrôleur (lui-même) qui permet de gérer l'affichage des Stacks les unes relatives aux autres

→ *Ex* : fermer ou écarter temporairement les autres Stack quand une Stack est *Expend*

► Avec les Stacks, on introduit un nouveau sous-comportement des *Selectables*, le **Highlight** :

● Un Highlight n'est pas une animation temporaire, mais plutôt un switch on/off, avec une déformation visuelle qui reste tant que le Highlight est activé.

● Chaque type de *Selectable* peut spécifier quel doit être son feedback de Highlight

● La fonction de déclenchement du Highlight est la même pour tous les Selectable, via le *SelectableManager*.

● La règle de la sélection unique pour un type donné ne s'applique pas, il peut y avoir plusieurs Selectable d'un même type *Highlighted* en même temps.

● Le Highlight est différent de l'Ancrage (qui lui respecte la règle de l'ancrage unique), mais on pourra donner la possibilité lors de l'Ancrage d'un Selectable de déclencher son Highlight également

►Ordre d'Affichage dans le Tray : (*left to right*)

● Tous les Bonus Stacks (D/E/S)

● Les Shields

●Les Stacks Négatives (Acid)

● (Optionnel) Target

+Intégrer Mockup ← ♠

+Nécessite une Anim de dépense pour toutes les Stacks ← ♠

● Dashes, Edges, Sparks → Anim de dépense classique

● Bouclier → Destruction du Bouclier, ou dépense comme les bonus Stacks

● Acid → se déclenche en début de tour puis enchaîne sur l'anim de tir

● Ammodifier → Automatique au moment ou le Shot touche

II.2] BONUS STACKS

II.2a] Select

Select une Bonus Stack consiste à afficher l'élément d'interface permettant a la joueuse de sélectionner un nombre de Tokens de Stack à dépenser pour une Stack. Les interactions avec la view se déroulent comme suis :

1) Sélection de la Stack par la joueuse.

2) Affichage de la Stack dans un état particulier (25 % + grande) et sélection d'un nombre de tokens de Stack à dépenser. On dit que la Stack est *Selected*.

3) Validation de la dépense de la Stack → Input de confirmation dépend du type de Stacks

- Sparks : Clicker sur la fiche de persos ou la jauge
- Edges : Continuer dans le Ctrl pendant qu'un Edge était sélectionné
  OU Sélectionner un Node dans lequel on peut investir un Edge (cf. II.2b)
- Dashes : Continuer dans l'access pendant qu'un Dash était sélectionné

Une Stack ne peut normalement être *Select* que si son Permit renvoie true. Quand une Stack est ainsi Select, certaines action spécifiques **confirment** l'utilisation de la Stack et déclenche son effet puis sa dépense (prenant en compte le nombre d'unités de Stacks utilisées).

► Pour annuler le *Select* d'une Stack, la joueuse click sur le tray, en dehors de la zone de sélection de tokens à dépenser

+Intégrer un Mockup ← ♠

II.2b] Cas des Edges

► Les Eddges peuvent être dépensés **à posteriori** de l'utilisation d'un Skill de Ctrl. Dans ce cas, un Edge ne peut être dépensé que pour investir du contrôle dans un node qui a déjà un score en contrôle côté joueuse (un Skill de Ctrl a été dépensé dessus) **&** si le teammate qui a investit du Ctrl est le même que celui qui possède le Edge.

→ Refacto : Nécessite de rajouter dans le CtrlScore quel est le/les teammates (array ID) qui ont investit du Ctrl sur ce Node ce tour-ci (reset à la fin du tour) *+Mettre a jour Flowchart Ctrl/Contest*

II.2b] Tableau des cas

► Dans certains cas, si une Stack a préalablement été *Selected* mais sans confirmation de dépense, le système peut en déclencher l'utilisation quand-même (**confirmation auto**)

SPARKS	DASHES :	EGDES :
Cas unique : *La joueuse possède ≥1 skill actuellement non-disponible à cause de son coût en énergie, qui deviendrait disponible en utilisant ≥1 Spark* → Elle peut *Select* ses Sparks, puis en confirmer l'utilisation pour gagner autant d'énergie	*Cas unique* : *la joueuse est en train d'utiliser un de ses Skills d'Access* → Elle peut *Select* ses Dashes, puis en confirmer l'utilisation pour gagner augmenter d'autant l'Access de son Skill OU *Select* ses Dashes et finir son skill (**confirmation auto**)	Cas n°1 : *La joueuse est en train d'utiliser un Skill de Ctrl* → Elle peut *Select* ses Edges, puis en confirmer l'utilisation pour augmenter le Ctrl de son Skill d'autant OU *Select* ses Edges et finir son skill (**confirmation auto**) Cas n°2 : *La joueuse n'est pas en train d'utiliser un Skill de Ctrl mais possède des Nodes dans lequel du Ctrl a déjà été investi* → La joueuse peut *Select* ses Edges, puis confirmer, puis Sélectionner un Node sur lequel rajouter un investissement en Ctrl

II.2c] Incidence dans le Skill Menu

II.3] SHIELDS & AMMUNITION

[Twig 3.4] *+Refaire les Icones d'Ammodifier & Shield Modifier ← ♠*

II.4] CART

+Intégrer un Mockup ← ♠

Le **Cart** est un élément de View qui s'affiche lorsque la joueuse a des Stacks et/ou des bonus (Shields Modifiers, Ammodifiers...) à **distribuer**. Le Cart permet de répartir ces bonus/stacks sur différents Teammates.

- Le Cart apparaît dès l'instant ou il y a des bonus/stacks à distribuer, **dès le premier bonus**
- Le Cart est commun à la Team.
- Quand le Cart est vide, il disparaît.

► Tant que le Cart est ouvert, la joueuse ne peut pas déclencher ses actions de jeux.

►A ce titre, le Cart sert d'intermédiaire entre les Effets qui produisent des Stacks/Bonus et le Tray des Teammates. C'est lui qui reçoit une **demande** de Distribution, et qui sera chargé d'envoyer l'Instruction de gain effectif de la Stack et/ou du Bonus pour le bon Teammate.

► Le Cart s'ouvre uniquement lorsqu'un personnage A a besoin de **distribuer** à un personnage B :
- Des Stacks (Tokens), y compris les Shields
- Des Shields Modifiers
- Des Ammodifiers

II.4a] Distribution

Distribuer des Stacks doit pouvoir se faire de 2 manières :

A] Sélection (Click) sur le Modifier/La Stack

→ Le drawer doit être extensible : taille proportionnelle au nombre d'éléments affichés

→ On peut choisir **combien** d'élément d'un type de Stacks on donne à un Teammate :

1. Le joueur sélectionne l'élément à distribuer (stack d'origine), ce qui en pré-sélectionne 1 de ce type. Cette présélection est représentée par une deuxième pile de stack de ce type de Stack, en version Highlight (stack preselect).
2. En re-cliquant sur cette stack preselect, ou en utilisant sa molette, le joueur peut incrémenter la quantité de tokens à distribuer à un teammate cible.
3. Si le joueur clique sur un autre type de Stack, il créé une nouvelle pile de stack preselect qui viendra s'ajouter aux tokens à distribuer à un teammate cible.

→ Les différentes stacks ainsi split viennent s'organiser dans une « zone de départ » représentés visuellement dans le Cart, qui permet de distribuer tous les tokens de Stacks actuellement highlights,

4a. Le joueur **valide** sa distribution en cliquant sur le Teammate cible

OU

4b. Le joueur **annule** sa distribution en cours en cliquant sur le bouton « Cancel », ce qui remet tous les tokens/piles splitées dans la zone de départ sur leur stack d'origine.

► On peut distribuer en une fois un groupe d'éléments différents en provenance du Drawer, a un teammate unique

B] Drag & Drop : Une pile de Stack d'origine est drag à l'aide du curseur de la souris sur une fiche de perso, puis est appliqué sur ce perso lors du drop.

OU On peut drag toutes les tokens/piles présents dans la zone de départ puis drop sur une fiche de perso