

Affichage de Sacred Phoenix dans Godot

Résolution d'écran en map

Après consultation de l'équipe et trois maquettes proposées, le choix de la résolution de base s'est porté sur un **braquet de 960×540** (qHD en 16/9^{ème}) tout en gardant la résolution originale des tiles et OWs (16×16, upscalé ×2).

Cela offrira un FOV très large de 30×16.875 sur les scènes de map ; et côtés menus, permettra d'avoir un peu plus de hauteur disponible comparés au 480p de PSDK.



Le fait de conserver un upscale ×2 nous laissera la possibilité de passer les ressources en HD (base 32×32) si nous souhaitons réaliser cet Idéal dans le futur.

Résolutions proposées au joueur

Lors de son premier démarrage, le jeu se lancera en mode fenêtré ×1 et générera un fichier de configuration (settings.ini) avec des paramètres par défaut. Sur un appareil mobile où le fenêtré n'existe pas, c'est le plein écran qui sera exécuté par défaut. (Je ne sais pas comment ça fonctionne et si le fenêtré est autorisé.)

Il sera possible de changer la résolution en accédant aux options qui seront accessibles depuis le menu principal ou le menu d'aventure. Les changements seront sauvegardés dans le settings.ini et permettront de conserver ces réglages d'une session à l'autre.

Voici ce que ce pourra proposer le réglage de résolution :

Résolution	Braquet	Zoom artworks	Nom	Pixel art net ?
960×540	×1 (défaut)	×0.5	qHD	Oui
1280×720	×4/3	×2/3	HD	Non (lissage obligatoire)
1440×810	×3/2	×0.75		Oui si pixel art en SD
1920×1080	×2	×1	Full HD	Oui (résolution optimale)
2400×1350	×5/2	×1.25		Oui si pixel art en SD
2560×1440	×8/3	×4/3	Quad HD	Non (lissage obligatoire)
2880×1620	×3	×1.5	3K	Oui
3840×2160	×4	×2	Ultra HD 4K	Oui
Plein écran	(Variable)	(Variable)		Variable, voire remarque 2

Remarque 1 : Si le moniteur du joueur a une largeur ou une hauteur inférieure à une des résolutions possibles, elle sera exclue de la liste proposée par les options du jeu. (Par exemple, un moniteur Full HD ne proposera que le ×1, ×1.33, ×1.5 et ×2). Réfléchir aussi si une résolution en mode fenêtré doit être autorisée si elle fait pile la hauteur du moniteur (à cause de l'en-tête de la fenêtre).

Remarque 2 : Un lissage sera automatiquement appliqué au pixel art si la résolution de plein écran fait que le zoom appliqué sur ces éléments n'est pas un chiffre entier, que ce soit en hauteur ou en largeur. Le plein écran occupera toujours tout l'espace disponible, idéalement sans déformer le ratio largeur / hauteur quitte à laisser deux bandes noires.

=> Pour ces 2 remarques, regarder comme j'ai codé sur la démo de PSDK : c'est fonctionnel.

Affichage en map

Ressources non pixel art

- Les éléments en artworks devront être optimisés pour l’affichage en braquet $\times 2$ (1920 $\times$ 1080). Le dézoom / zoom se fera avec un lissage afin d’optimiser la lisibilité. Il faudra faire des tests en $\times 1$ ($= \times 0.5$ pour ces éléments), $\times 1.5$ ($= \times 0.75$), $\times 2.5$ ($= \times 1.25$) et $\times 3$ ($= \times 1.5$) pour déterminer quel sera le meilleur rééchantillonnage.
- Les polices d’écriture seront gérées en vectoriel afin d’avoir toujours un affichage optimal. Elles devront bien entendu être visible dans la taille minimale du jeu.

Ressources pixel art en SD (vanilla) :

- Tiles 16 $\times$ 16, upscalées $\times 2$
- OWs style 4G en base 16 $\times$ 16, upscalés $\times 2$
- Zoom en plus proche voisin si le braquet actif est un multiple de 0.5 ; sinon lissage.

Ressources pixel art en HD (format Pokémon Xenoverse) :

- Tiles 32 $\times$ 32, upscalées $\times 1$
- OWs style 4G en base 32 $\times$ 32, upscalés $\times 1$
- Zoom en plus proche voisin si le braquet actif est un multiple de 1 ; sinon lissage.

Futurs changements à prévoir par rapport à la démo de l’ancien moteur de jeu :

=> Nécessité de passer la taille de chunks de 32 à 24 tiles de côté, un bloc de 2 $\times$ 2 chunks correspondra à 1 chunk de l’ancien moteur de jeu. Cette subdivision des chunks en 4 quarts permettra un meilleur découpage des zones.

=> Pour rappel, une map extérieure (dite « zone ») contient de 1 à plusieurs chunks. La forme d’une map n’est pas forcément rectangulaire et peut avoir une forme de « L » ou de « U » (l’espace concave sera englobé par d’autres zones).

=> Une map affichée en jeu devra être capable d’afficher les évènements des maps adjacentes (ou pour optimiser : seulement ceux des chunks adjacents à la map active, diagonales incluses) ainsi que leurs mouvements et déplacement (ex : oiseaux qui tournent en rond) afin d’éviter tout pop ou dépop d’évènement au changement de map.

=> Les évènements « multi-cases » devront pouvoir être interactifs en utilisant un seul évènement. (Exemple : les arbres à baies qui passeront de 1 $\times$ 1 tiles à 2 $\times$ 2 tiles).

=> Nécessité de refaire toutes les maps à une échelle plus grande, notamment les ponts, routes et rivières (+50% de largeur et longueur)

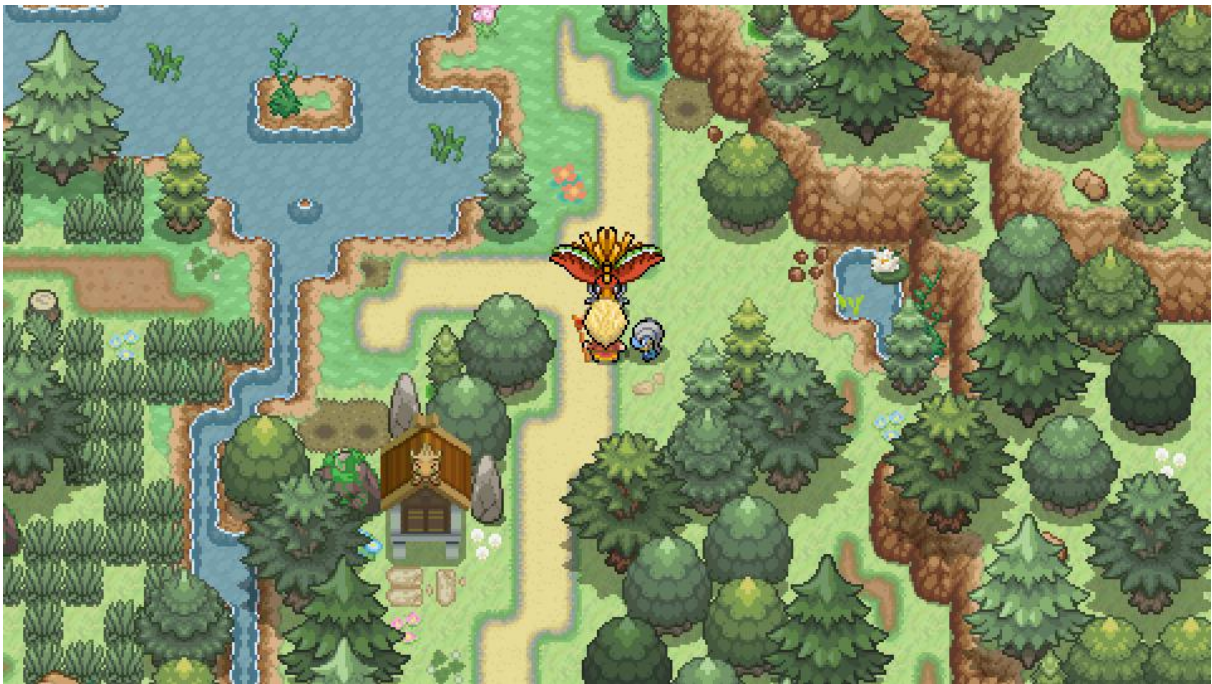
=> Nécessité de retravailler certaines tiles utilisées en extérieur à une échelle plus grande (sont concernés : les maisons, les portes, les arbres fixes offrant des récoltes, et certains monuments). Les maps intérieures ne sont pas affectées.

Fluidité des déplacements et animations

=> Les animations devront être calculées au pixel près sur l’affichage réel, non sur une pseudo matrice de finesse égal au braquet (ou pire : au braquet $\times$ l’upscale comme on le voit sur PSDK). Par exemple, si un personnage se déplace d’une tile en 64 frames, un joueur qui joue en Full HD (et affiche donc des tiles de 16 $\times$ 2 d’upscale $\times$ 2 de braquet = 64 pixels) se déplacera à l’écran d’un pixel par frame (au lieu de 4 pixels toutes les 4 frames).

=> Idem s’il y a des effets de zoom ou de rotation (valable surtout pour les animations). Les animations seront zoomées en plus proche voisin, comme sur RPG Maker (sauf si on est dans un braquet qui ne permet pas un rapport entier sur le pixel art).

Résultat attendu :



Ancienne map du Lac du Temps en résolution 960×540. Point fort : le grand champ de vision du joueur qui met l'emphase sur son environnement.



Mais cela demandera des ajustements : les maps de Sacred Phoenix ont une échelle « étroite », ce qui fait que le joueur voit trop d'éléments sur son champ de vision et peut poser des problèmes pour certaines mises en scène. (Ici, impossible de faire apparaître les Adeptes de Darkrai depuis le bord droite de l'écran, puisque la falaise est déjà à portée de vue). Agrandir l'échelle des bâtiments, des chunks et du relief (de 25 à 50%) peut permettre virtuellement d'agrandir la définition de Keltios tout en gardant les tiles en 16×16.

Et le cas du 1280x720 ?

Il s'agit d'une résolution très courante et qui sert de multiple à de nombreuses autres résolutions (dont le quad HD). C'est celle de la Nintendo Switch en mode portable.

Solution 1 :

Calculer l'affichage en full HD (1920×1080), puis selon la résolution réelle (2/3 pour du 1280×720), calculer un lissage bilinéaire si le zoom final n'est ni un nombre entier, ni supérieur à 4.

- Si la valeur est un chiffre entier, tout lissage entraînera un flou inutile qui dégradera l'aspect du pixel art.
- Au-delà de 4, la distorsion du « plus proche voisin » devient peu perceptible.
- Potentiellement laisser ce choix au joueur dans les options d'affichage. (Lissage du pixel art : Automatique, Plus proche voisin, Bilinéaire... voire proposer d'autres filtres.)

Cette solution a l'avantage de pouvoir s'appliquer à toutes les résolutions inimaginables (notamment en cas de plein écran qui force une résolution « non listée » et peut altérer le ratio largeur / hauteur).

Upscale du pixel art	Braquet	Résolution	Zoom réel	Entier ?	> 4 ?	Lissage
×2 (SD)	×1.333	1280×720	2.667	Faux	Faux	Bilinéaire
×2 (SD)	×1.5	1440×810	3	Vrai	Faux	Plus proche voisin
×2 (SD)	×2.5	2400×1350	5	Vrai	Vrai	Plus proche voisin
×2 (SD)	×2.667	2560×1440	5.333	Faux	Vrai	Plus proche voisin
×1 (HD)	×1.333	1280×720	2.667	Faux	Faux	Bilinéaire
×1 (HD)	×1.5	1440×810	1.333	Faux	Faux	Bilinéaire
×1 (HD)	×2.5	2400×1350	1.5	Faux	Faux	Bilinéaire
×1 (HD)	×2.667	2560×1440	2.667	Faux	Faux	Bilinéaire

Solution 2 :

Réduire le FOV à 26.667 ×15 et upscaler le pixel art en ×3 s'il est en résolution standard. (Attention : ça ne marche pas avec du Pixel art en HD, car l'upscale tombera sur du ×1.5, soit un nombre non entier.)

Ce FOV spécial ne s'appliquerait qu'en map et si on utilise la résolution 720p + éventuellement en 1440p.

En map, on obtiendra ces braquets pour le pixel art :

Upscale du pixel art	Braquet	Résolution	Zoom réel	Entier ?	> 4 ?	Lissage
×3 (SD)	×1	1280×720	3	Vrai	Faux	Plus proche voisin
×3 (SD)	×2	2560×1440	6	Vrai	Vrai	Plus proche voisin
×1.5 (HD)	×1	1280×720	1.5	Faux	Faux	Bilinéaire
×1.5 (HD)	×2	2560×1440	3	Vrai	Faux	Plus proche voisin

Et ce FOV :



Le problème majeur est que les mises en scènes pourront être plus difficiles à adapter (à cause de la bordure qui n'est plus visible comparé au FOV à 30×16.875).

Aussi, la valeur de zoom deviendra peu confortable ($\times 1.5$) si on utilise du pixel art HD.

Résolution d'écran en combat

Battlers (sprites front et back)

Partie sur une base qHD offre une base confortable et permet même l'usage de sprites HD.



Cette maquette montre deux Pokémon ennemis (affichés en front) en HD upscalé $\times 1$, ainsi qu'un Pokémon allié (donc affiché en back) en HD upscalé $\times 2$. Cette différence d'échelle est pour donner un effet de perspective.

Ceci dit, un tel agrandissement en faveur du back peut lui permettre d'occuper jusqu'à 512 pixels de large ou de hauteur. Si on veut avoir la place d'afficher un combat à 3 contre 3 tout en prenant en compte l'espace réservés à l'UI, le canevas idéal maximum pour les battlers serait donc de...

- 256 $\times$ 256 pour les fronts
- 384 $\times$ 384 pour les backs

Soit un ratio de 3/2 entre ceux vus de face et ceux vu de dos. Il s'agit d'un juste milieu entre l'effet de perspective et la place disponible sur l'écran de jeu.

Si on part sur des sprites en SD :

(Nécessité de se fixer sur une taille unique pour les backs)

Sprite	Taille sprite	Si animée	Upscale	Zoom comparé à la SD
Front (SD)	96 $\times$ 96	128 $\times$ 128	$\times 2$	100%
Back (SD)	96 $\times$ 96	128 $\times$ 128	$\times 3$	100%
Back (semi-HD) 🐾	144 $\times$ 144	192 $\times$ 192	$\times 2$	150%

• Avantages des fronts SD en upscale $\times 2$:

- ✓ Le pixel art des fronts restera parfaitement net avec des braquets multiples de 0.5.
- ✓ Aucun besoin de refaire les fronts.

• **Avantages et inconvénients des backs SD en upscale 3 :**

- ✓ Aucun besoin de refaire les backs.
- ✗ L'échelle du pixel entre les fronts et les backs n'est pas respectée.
- ✗ Du pixel art en finesse de 3×3 commencera à déteindre significativement vis-à-vis de l'UI de combat (qui est intégralement composée d'artworks, donc en finesse 1×1).
- ✗ Zoom non entier en cas de braquet à ×1.5 ou ×2.5 (Ceci dit, on peut redimensionner en plus proche voisin, car le zoom total sera supérieur à 4.)

• **Avantages et inconvénients des backs semi-HD en upscale 2 :**

- ✓ Échelle de pixel respectée vis-à-vis des fronts (on reste en finesse 2×2.)
- ✗ Nécessité de consacrer du temps pour les redimensionner à 150 %.

Si on part sur des sprites en HD :

(Là aussi, il faudra se fixer sur un seul choix pour les backs)

Sprite	Taille sprite	Si animée	Upscale	Zoom comparé à la SD
Front (HD) 	192×192	256×256	×1	200%
Back (semi-HD)	144×144	192×192	×2	150%
Back (HD)	192×192	256×256	×1.5	200%
Back (Ultra HD) 	288×288	384×384	×1	300%

L'inconvénient majeur est que ça demandera un énorme travail de pixel art pour redimensionner les sprites, surtout sur les backs si on part en Ultra HD.

Les backs en semi-HD upscalés ×2 pourraient être un compromis, ou alors des HD avec un upscale ×2 (accentuation de la perspective) ou ×1 (pas de perspective).

Du côté de Coromon...



Sur sa résolution bâtarde 768×432 (en upscale ×2), les sprites de Coromon sont assez petites (canevas maximum de 256×256 sur cet upscale, soit 128×128 non upscalé).

Dans la pratique, les Coromons dépassent très rarement les 96×96 non upscalé. Seuls les Titans peuvent dépasser cette limite. Et tous sont animés, donc le canevas inclut la place nécessaire pour l'animation.

Ils ont aussi la même échelle entre les fronts et le backs (pas de perspective).

Rappelons que la taille d'un Pokémon varie en fonction de sa taille.

- 96 est la limite pour un Pokémon « statique » de très grande taille (Méga, Pokémon très grands...)

- L'augmentation à un canevas de 128 de côté correspond à l'espace nécessaire pour l'animation (et permettra de faire des textures en spritesheets avec des puissances de 2.)

- Une frame de Pokémon de petite taille (première évolution) occupera souvent moins de 50% d'un carré de 96×96.

Il faut donc tester le rendu avec des Pokémon petits tels que Roucool ou Chenipan.



L'évolution vers du HD

La conversion des sprites de Pokémon en HD se fera par étape en fonction de la hype du projet et du nombre de personnes qui seront motivées à y contribuer. (Je reste réaliste^^ )

- D'abord rassembler des ressources de sprites non animées en HD. Il en existe déjà en accès libre sur le Web.

<https://www.pokecommunity.com/showthread.php?p=10235281>

- Une fois qu'on en aura réuni assez, initier une période de transition où on aura un mélange SD (zoom ×2) et HD. Prioriser les Pokémon disponibles en début de démo.

- Une fois une grande partie des Pokémon de la démo entièrement disponibles en HD, commencer à les animer la Black and White avec une animation unique qui se boucle. Des logiciels comme Dragon Bone Pro peuvent aider à automatiser ce type de tâche.

- Puis si le projet prend encore plus d'ampleur, faire des animations (toujours en style 5G) représentant les diverses actions possibles du Pokémon (animation d'attaque, d'attaque spéciale, d'attaque de statut lancée, de dégâts subis, de gravement blessé, de tombé KO...) pour avoir tout l'éventail d'animations comme depuis 6G, mais en 2D.

Exemple de Pokémon HD :

Ceci est un Pokémon de taille honorable en résolution « HD », comparé à sa sprite en résolution standard. Ça rentre dans un cadre 160×160, mais si le Pokémon est animé comme dans le mod “Elite Battle System” de *Pokémon Essentials* (ou les jeux 5G), 192×192 ne suffira pas dans les cas extrêmes (comme un Méga-Drattak qui prend beaucoup de largeur). 256×256 sera parfait et pourra gérer tous les cas.



Concernant la cohabitation entre sprites SD et HD, la problématique est la même qu'avec les OW. Il sera donc probablement nécessaire d'agrandir en $\times 2$ toutes les images existantes (mode PSP / Essentials) et de programmer Godot pour les utiliser sans upscale en 960×540 , ou bien d'utiliser un système de classe dictée par une constante qui indique l'upscale à appliquer selon si c'est du SD ou du HD.

Éléments d'UI

Éléments en artworks

Les éléments d'UI – qui seront en grande partie des images matricielles – seront optimisés pour **une résolution en 1920×1080** et devront donc être créés dans ce format.

Ces éléments devront rester lisibles un minimum avec la résolution minimale de 960×540 (soit 50% de zoom).

Si une résolution différente est choisie par le joueur, ces images doivent être idéalement redimensionnées avec un lissage.

Paint.NET conseille le super échantillonnage si $\text{zoom} < 100\%$ et bicubique si $\text{zoom} > 100\%$... après, à voir quel sera l'algorithme qui aura le meilleur rapport qualité / pertes de performances). Cela peut être aussi l'objet d'une option de jeu au choix pour le joueur (style un shader).

Ce qui est au format « artwork » :

- L'interface du sac
- L'interface de combat
- Le menu d'horloge
- Les futures icônes d'action en combat
- Les futures icônes d'attaque
- Les backgrounds de menu
- Les mugshots de Pokémon et de PNJ (au format 384×384)
- Les mugshots d'objets (au format 256×256)
- Les futures icones de types et d'attaque

À terme, les éléments en artworks représenteront la majeure partie des éléments d'UI ; bien que quelques éléments en pixel art subsisteront, notamment ceux hérités de la licence originale. Il faudra donc avoir des « tags » ou des « classes » afin d'appliquer le bon filtre de redimensionnement selon l'élément graphique concerné.

Éléments en pixel art

Si un élément d'UI est en pixel art, il suivra les règles habituelles des autres sprites :

- Si le zoom réel ($\text{braquet} \times \text{upscale}$) donne un nombre entier ou supérieur à 4, ça sera interpolé en Plus proche voisin.

- Sinon, un lissage sera appliqué.

(Là aussi, le type de lissage peut éventuellement être proposé en option : automatique, ou une liste d'interpolations au choix du joueur.)

Ce qui est au format « pixel art » en upscale ×2 :

(Si on décide de faire une version HD de ces éléments, ils passeront en upscale ×1. Idéalement, tout devrait être converti.)

- Les petites icônes des objets issues de Pokémon vanilla (canevas de 32×32, taille idéale de la partie remplie à 24×24)

- Les icônes des de Pokémon dans le style 5G animé (format 32×32, 2 frames ; peut monter exceptionnellement à 40×40 pour des Mégas ou très gros Pokémon)
- Le curseur rouge de choix
- La flèche de sélection bleue « patte de dragon » (un format HD existe pour le menu du starter).
- Les icônes du menu de jeu (ça peut être l'occasion de encore refaire un reboot de ces icônes en HD et d'avoir quelque chose de vraiment libre de droit.)

Ce qui est au format « pixel art » en upscale ×1 :

- Tout élément graphique en pixel art qui apparaît avec une finesse 1×1 sur la démo actuelle de PSDK.
- La flèche de sélection « Cœur d'or ».
- Les barres d'amitié, de mana, d'effroi, ainsi que les icônes liés au menu d'amitié

Textes en image

Actuellement, certains textes faits dans une police celtique sont affichés via une image en dur. Cela est du aux contraintes de PSDK qui :

- N'a rien prévu pour changer la police d'écriture d'un texte
- Affiche les caractères sur une matrice au format timbre-poste de 320×240, ignorant totalement l'avantage vectoriel apporté par les polices d'écriture.

Idéalement, il faudrait que seule image de fond soit affichée, et le texte sur un affichage « texte » dédié tout en profitant de l'avantage vectoriel, quelque soit la résolution choisie par le joueur. (On ne peut se permettre de perdre en lisibilité pour les textes.)

Gestion des tilesets

RPG Maker XP est limité à 3 couches. Cela représente une forte contrainte pour moi qui doit anticiper de nombreuses combinaisons de superposition en usant et abusant du pré-mapping.

Une des forces de Godot est qu'il n'y a plus cette limite. On peut donc empiler autant de couches, et cela me permettra de significativement épurer mes tilesets de toutes les « tiles combinées ».

J'ai entendu dire que c'est compatible avec le logiciel Tiled (à voir si c'est plus pratique que l'éditeur interne de Godot).

Bien entendu il faudra retrouver au minimum toutes les fonctionnalités de bases présentes sur RPG Maker XP et PSDK :

- Gestion des collisions dans les 4 directions
- Niveau de superposition (de 0 à 5, plus n'est pas de refus)
- Tiles d'interaction (permet d'interagir avec un événement derrière un compteur ou une barrière).
- System tag illimité (ceux de PSDK et les 8 combinaisons de RPG Maker se cumulent, cela permet par exemple d'avoir plusieurs types de hautes herbes où les rencontres sont différentes comme herbe 1, herbe 2 ou sable 1, sable 2.) À noter que la fonctionnalité de « carreau obscurcissant » est censée être gérée via System tag (ex des particules d'herbes lorsqu'on marche dans l'herbe, ou les traces de pas dans le sable.)
- Gestion des autotiles
- Gestion des altitudes (z-index) pour pouvoir passer sous des ponts par exemple.
- Bonus (paraît que c'est plus optimisé et que c'est le fonctionnement de RPG Maker MV) : pouvoir utiliser plusieurs tileset sur une map et indiquer à la map quels tilesets sont nécessaires.

Les tiles seront sujettes à un système de tons (effets jours nuits) et de météo (particules avec éventuelle influence sur les tons).

L'eau devra aussi avoir la possibilité de générer un reflet des événements juste au-dessus des tiles d'eau. (Contrairement à PSDK qui utilise la transparence de la tile mais qui fait trop de faux positifs, il faudra trouver autre chose comme par exemple un system tag qui indiquera si cette tile génère un reflet.)

Gestion des maps

À l'instant des jeux Nintendo DS, Sacred Phoenix sera basé sur un système de chunks pour ses maps extérieures. Chaque chunk est un « bloc » carré de 32x32 tiles.

La « map » (en tant qu'unité géographique : une ville, une route...) est composée d'un ou plusieurs chunks.

L'ensemble des maps est prévu pour s'emboîter parfaitement du fait d'une feature de « Grand Envol » à la 6G qui permettra au héros de survoler tout Keltios librement.



Comme tu le constateras, j'ai déjà subdivisé ma carte du monde en cases (1 case = un chunk de 32x32 tiles sur PSDK).

Sur Godot, on passera à du 24x24 tiles, mais le nombre de chunks sera doublé sur chaque sens, ceci afin d'avoir des maps plus grande pour correspondre au FOV plus grand.

- La Colline du Phénix fait 2x2 chunks (4x4 dans le futur format)
- Le Lac du Temps 3x2 (6x4 dans le futur format)
- Bourg Sylvestre 3x1 (6x2 dans le futur format)
- Le Sentier de l'Aigle 3x1 (6x2 dans le futur format)
- Etc...

Idéalement, les chunks occupés par une map devraient pouvoir avoir une forme autre que rectangulaire.

Encore idéalement, les 8 chunks adjacents (et leurs événements) devront être chargés afin que le champ de vision du joueur soit toujours complet. Soit un total de 9 chunks à un instant donné si le joueur est en extérieur.

En cas de grand envol et si une mécanique de « dézoom » est prévue afin de représenter le gain d'altitude, cela pourrait monter à un rayon de 2 chunks (soit $5 \times 5 = 25$ chunks chargés).

Si gérer 25 chunks est possible, prendre ce choix. Cela peut éviter des dépopulations d'événements soudains s'ils sont « mobiles », mais dépendants d'un chunk devenu trop éloigné. (Ex : Roucoups qui se déplacent en cercle dans le ciel.)

Anecdote : avec le maplink natif et des maps qui font 4 chunks en moyenne, on arrive à l'équivalent de 20 chunks chargés par PSDK (avec 4 maps maplinkées). C'est un argument en faveur de charger toutes celles dans un rayon de 2 chunks.

Anecdote 2 : Skyrim utilise un système de chunks pour déterminer l'espace qu'il doit charger autour du joueur. C'est aussi un rayon de 2 chunks autour de celui du joueur qui est chargé, soit 25 blocs au total.

Note : Attention aux system tags qui sont à la lisière exacte des maps adjacentes. Sur PSDK, ils ne sont pas pris en compte, ce qui peut entraîner un bug. (Ex : bord d'eau qui commence pile en bordure de la map voisine, alors que la bordure de la map active est un terrain accessible : le jeu ne détecte pas que c'est un plan d'eau si le joueur veut rejoindre la map voisine à ce point précis, et il pourra alors marcher dessus.)



Le genre de bug qu'on peut avoir si la bordure du plan d'eau est pile entre deux maps.

Ça le fait aussi sur les sauts de petites falaises, ce qui m'a forcé à faire quelques ajustements sur mon mapping.