5D RPG with Multiverse Time Travel

GLenPLonk

Un JdR textuel & asynchrone, avec voyages temporels et multiversels, à MJ variable, jouable via git.

1 Contexte

Votre univers est en danger. Peut-être s'agit-il d'une guerre thermoentropique de civilisations, d'une invasion de robots géants, d'une rupture innée (ou pas) mais néanmoins malencontreuse de la continuité des espaces-temps, mais votre univers est en danger. Vous seul·es pouvez le sauver. Enfin, peut-être...

2 Outils

2.1 Outils informatiques

Les joueurses s'assurent d'avoir un client git qui fonctionne et un éditeur de texte (de leur choix). Iels s'accordent sur une forge git à laquelle toustes peuvent accéder.

L'unité d'action est le `commit`. La Contrainte est la règle suivante :

> Aucunes paires de `commits` successifs ne doit avoir
> læ même auteurice.

Chaque joueurse joue quand iel le souhaite, en respectant la Contrainte.

2.2 La communication externe

Il est nécessaire d'avoir un moyen de communication entre joueurses autres que git (qui, soit dit en passant, n'est *pas* un moyen de communication ...). Cela permet entre autre de communiquer sur des éléments hors-jeu, de vérifier que les sujets abordés ne blessent, gênent ou mettent mal à l'aise un·e joueurse, de rappeller aux autres de jouer, ...

Avant de jouer, chaque joueurse précise aux autres si des sujets ne *doivent* pas être abordés, si certains sujets *peuvent* l'être mais avec précaution, ... La liste des sujets bannis/limités est amenée à évoluer, elle n'est pas figé dans le marbre. Il suffit qu'un·e joueurse le veuille pour qu'un sujet y entre, mais toustes doivent être d'accord pour en faire sortir un.

Par ailleurs, ces règles sont un guide plutôt qu'un carcan contraignant, vous pouvez décider de changer, modifier, ignorer ou ajouter des règles.

3 Comment jouer ?

3.1 Mise en place

3.1.1 L'Univers

Læ premièr·e joueurse (désigné·e par la méthode de votre choix : plus âgé·e, plus grand nombre d'heure d'écoute d'Evanescence, plus grand nombre d'ongles, autodésignation, ...) créé un fichier texte nommé `univers` avec éventuellement quelques mots expliquant que l'Univers est, puis `commit` ce fichier et le `push` sur le repo du jeu. L'Univers est alors créé, sa chronologie est décrite par l'ensemble des fichiers du repo.

Toustes choisissent ensuite l'univers dans lequel les personnages vont évoluer : peut-être y a-t-il de la magie, des humains, des elfes, des robots, des voyages interstellaires, ... Notament, les joueurses décident comment et pourquoi leurs personnages pourront voyager entre les branches temporelles. Enfin, le Danger pesant sur l'Univers est décrit. Le Danger doit être *quelque chose* dont le but est la disparition de la diversité de l'Univers : son objectif est l'existence d'une unique branche temporelle, grise et triste et morne.

Toute la mise en place se fait dans une unique ligne temporelle, incrémentalement, sans qu'aucun·e joueurse ne contrôle un personnage en particulier (d'ailleurs, *les personnages joués n'existent pas encore* !). En pratique, chaque élément est un fichier texte séparé, comportant simplement la description de cet élément et/ou son évolution depuis le dernier `commit`. Si un fichier est créé, un élément vient d'apparaître : naissance, construction, découverte, ... Si un fichier est supprimé, cela signifie que l'élément n'existe plus : la personne est morte, la planète a été détruite, la maison s'est effondrée, ... Dans tous les cas, les joueurses `commit` les changements (sans oublier de `push` sur le repo distant commun). Si nécessaire, le message associé permet d'expliquer les circonstances de l'apparition ou de la disparition d'un élément. Ne pas hésiter à créer une arborescence pour facilement s'y retrouver, en prenant garde à ce qu'il ne faille pas déplacer les fichiers trop souvent.

À ce stade du jeu, la commande `git log --graph --all --oneline` doit n'afficher qu'une unique ligne, *il ne doit y avoir qu'une unique branche* : la ligne principale.

Chaque joueurse joue au moins 1 fois et au plus autant de fois qu'il n'y a de joueurses.

Une fois l'Univers assez étoffé, vous pouvez passer à la création de personnages.

3.1.2 Les personnages

Enfin, chaque joueurse créé une nouvelle branche au nom de son personnage (`git branch <nom>`) à partir du commit de tête de la ligne principale, s'y positionne (`git switch <nom>`) et y fait un `commit` en décrivant son personnage.

3.2 Le jeu

Quand iel joue, un·e joueurse a le choix soit d'incarner son personnage, soit un·e agent·e du Danger.

3.2.1 Jouer un personnage

Læ joueurse agit sur l'Univers comme iel le souhaite, tout en respectant le fait qu'iel n'a de contrôle que sur son personnage. Notamment, si des choix ayant des conséquences sur un autre personnage sont réalisés, ils doivent être fait en collaboration avec læ joueurse le contrôlant. Il peut être utile pour cela de communiquer via le moyen de communication choisi.

Si (et seulement si) le personnage d'un·e joueurse meurt dans un `commit`, ce·tte joueurse crée un embranchement temporel, c'est-à-dire qu'iel crée un deuxième `commit` ayant le même parent (où son personnage meurt également). Dans chaque `commit`s, un nouveau personnage contrôlé par læ joueurse dont c'est le tour doit être créé. Les deux branches doivent ne pas contenir le même personnage. En pratique, on utilise les commandes `git branch <nouveau perso 1>`, (on joue,) `git commit`, `git checkout <ancienne branche>`, `git branch -m <nouveau perso 2>` (et on joue) et `git commit`.

3.2.2 Jouer un·e agent·e du Danger

Différentes actions sont possibles :

— Commit : læ joueurse choisit une branche, s'y positionne (`git switch <branche>`) et joue sur cette branche d'univers, en décrivant les actions du Danger et explicitant ses effets dans les bons fichiers. Une fois tout cela fais, iel `commit` et `push`.

— Merge : deux branches (et pas plus) sont fusionnées. Læ joueurse se positionne sur la branche ayant le Chaos le plus élevé (`git switch <branche 1>`) puis tente de fusionner la seconde branche (`git merge --no-ff --commit <branche 2>`). Iel gère la résolution des conflits comme iel le souhaite. Il faut considérer que les actions du Danger doivent être dangereuses, mais pas mortelles, pour les personnages dans la résolution de conflits. Toujours passer l'option `--no-ff` : on veut un `commit` dans tous les cas. Enfin, iel change le nom de l'unique branche en la juxtaposition des nonms des deux anciennes branches : `git branch -m <branche1 - branche2>`.

4 Quelques conseils

— Attention, au début de la partie, de bien mettre en place la `config` locale de git : bon repo distant, bon email, bon nom (ou pseudo), ...
— Toujours `git fetch --all` avant de jouer une action.
— Attention, bien que `git reset --hard` existe, il est difficile de supprimer un `commit` (c'est-à-dire une action de jeu) qui a déjà été `push`.
— Une bonne façon d'avoir une vue d'ensemble du bazar temporel dans lequel les personnages évoluent est `git log --all --graph --oneline` ou `git log --all graph --pretty=short`.
— Ne pas oublier la Contrainte.
— Il est tout à fait possible de placer des fichiers non textuels dans l'Univers (des images, des pdf, des exécutables, des fichiers compressés, ...), mais il sera plus difficile de comprendre leur évolution via `git diff`. Un bon compromis est l'utilisation de SVG pour représenter des cartes, des plans, des personnages, ...