# Concept

## Idée générale

Un afficheur / gestionnaire de carte hexagonale pour Oltréé, qui doit permettre d’afficher la carte du jeu actuel, sélectionner chaque hexagone pour en voir les informations, et donner aux joueurs un visuel sur le territoire, pour informer et faciliter leurs décisions.

## Etapes

### La Base

Offre le minimum vital.

* Afficher la carte aux dimensions pré-fixées.
* Charger / sauvegarder
* Visualisation d’Hexagone:
  + Couleur & Image (Carte)
  + Degré d’exploration (Carte)
  + Label de coordonnées en accord avec celles de la carte initiale (Carte)
  + Description (Interface – si sélectionné seulement)
* Modification d’Hexagone :
  + Couleur
  + Image
  + Description
  + Degré d’exploration
* Sélection d’hexagone :
  + Par clic dans le polygone (Carte)
  + Applique un liseré vert pétant à l’intérieur (Carte)
  + Affiche la Description de l’Hexagone (Interface)
* 3 Modes :
  + Création de la carte : modification des éléments graphiques
  + Edition de la carte : modification des éléments descriptifs
  + Visualisation de la carte : sélection d’hexagone 🡺 visualisation des éléments descriptifs

### Evolution – BDD

Disposer d’une BDD accessible en jeu, mutuelle, permettant de sauvegarder / recharger la carte.

### Evolution – Lignes 1/2

* Création d’une Ligne :
  + Mode de placement (point cliqué, proche sommet, point fixe)
  + Nom (qui sera placé au milieu du segment milieu)
  + Définition d’une Couleur (liste habituelle, probablement à généraliser donc) et d’une Largeur (liste statique)
  + Définition des points par clics :
    - Sur le point cliqué, simple
    - Sur le « sommet polygonal » le plus proche (on prend le polygone cliqué, on cherche son point le plus proche)
    - Sur le « point fixe » le plus proche : sommet polygonal, centre polygone, milieu des côtés, milieux sommet-centre.
* Modification d’une Ligne :
  + Mode de placement (point cliqué, proche sommet, point fixe)
  + Nom
  + Couleur
  + Largeur
  + Note : Pas de déplacement des points
* Sélection d’une ligne :
  + Par clic directement dessus
  + Indiqué par un liseré vert

### Evolution – Lignes 2/2

* <http://csharphelper.com/blog/2014/10/let-the-user-draw-polygons-move-them-and-add-points-to-them-in-c/>
* Déplacement des points d’une ligne :
  + Sélection de la ligne 🡺 modification des « Points » de façon à les rendre plus facilement sélectionnables
  + Déplacement du point + relâchement : selon Mode.

### Evolution – Calendrier 1/x

* Elément Calendrier – Top
  + Jour, Mois, Année
  + Clic dessus pour afficher une fenêtre Calendrier
* Fenêtre Calendrier
  + Indique mois et année
  + Tableau des Jours
    - Indication du jour en cours
  + Boutons pour avancer / reculer dans les jours
* Pas d’agenda

### Evolution – Calendrier 2/x

* Elément Agenda
  + Contient des jours
    - Date
    - Description
  + Seuls les jours où quelque chose a été renseigné existent
* Sur le Calendrier
  + Les jours avec une correspondance dans l’Agenda apparaissent différemment, cliquables.
  + Cliquer sur un Jour Agenda ouvre une autre fenêtre (ou modifie celle du Calendrier), et afficher la Date et la Description

### Evolution – Météo 1/x

* Possibilité d’afficher une interface Météo
  + On fournit la Saison et le Temps Actuel
  + Génère, sur clic, la Météo suivante
    - Temps
    - Durée (s’il y a)
  + Prend en compte les Evènements météorologiques particuliers
  + Indique les effets météorologiques

### Evolution – Météo 2/x

* Lien entre l’Interface Météo et le Calendrier / Agenda
  + Ajout d’une donnée Météo dans les Jours de l’Agenda (ce qui créée un Jour s’il n’y en a pas)
  + Un bouton « Inscrire dans l’Agenda » permet d’ajouter une météo donnée sur un jour donné.
  + Il peut être utile de rendre visible un symbole météo sur les jours du calendrier – si existant. Sinon, un indice identifiant que le jour a, ou non, une Météo remplie
* Réfléchir sur l’utilité d’une fonctionnalité de remplissage automatique sur toute une période (et qui ne surcharge pas une météo déjà remplie)

### Evolution – Météo 3/x

* Lien entre Météo et Déplacements
  + Tableau de coûts en déplacements selon la météo
  + Prendre en compte les modes de déplacement : route embouée, neige, etc.

Evolution – Météo 4/x

* Effets Météo cumulatifs :
  + Compteurs incrémentables pour les « situations » :
    - Neige : augmente par jour de neige, reste dans certaines conditions, etc.
    - Pluie : autant une pluie fine n’a guère d’effet, autant des pluies diluviennes impactent divers éléments
    - Chaleur : jour après jour de sécheresse a un impact
  + Ces compteurs doivent avoir des effets, probablement par « crans »
    - Handicaps de déplacement
    - Hausse de la consommation de ressources
    - Dégâts de fatigue
    - Impacts sur les bâtiments / productions / populations
    - Etc.

### Evolution – Patrouilleur 1/x

* Elément Patrouilleur
  + Informations générales
    - Nom
    - Joueur
    - Peuple (liste)
    - Motivation (liste)
    - Initiation (liste)
  + Vocations & Sauvegardes
    - Vocations
    - Sauvegardes (sous forme de bonus)
  + Carnet de Patrouille
    - DV, PV max, PM max
    - Cartes
    - Vigilance
    - Déni
    - CA
    - Initiative
  + Métiers
    - Un tableau avec tous les métiers, score, bouton +/-
    - Tous les métiers ont un score, même à 0
  + Harnois : des cases à cocher
  + Langues : un tableau, deux cases chacune (écrit / parlé)
  + Ressources immobilisées
* Interface « Feuille de Personnage »
  + Mode « Edition »
  + Mode « visualisation »
* Cette 1e version n’a pas à afficher grand-chose en dehors du nom de chaque élément, ni fournir de calcul automatique
* Elle permet par contre de renseigner l’objet Patrouille

### Evolution – Patrouilleur 2/x

* Elément Patrouilleur
  + Informations générales :
    - Initiation : fournit les points lors de la Création, ainsi que les Cartes.
  + Vocations & Sauvegardes
    - Vocations : boutons + / - avec Source
    - Sauvegardes : boutons + / - avec Source (sauf le -)
  + Carnet de Patrouille
    - PV max : fournis par le joueur, seulement à la Création et en cas de Gain de DV
    - PM max : calcul auto
    - Cartes : automatiquement, par initiation. Possible de faire en sorte qu’en cas de Gain de DV, il y ait choix Carte / normal.
    - Vigilance (calcul auto)
    - CA : à partir des sources sur la fiche (armure, bouclier, guerrier, custode), et faire le calcul automatique
    - Initiative : calcul auto
  + Métiers
    - Tableau 🡺 Liste : un bouton Ajout + Source
    - Eléments déjà ajoutés : boutons + / - & Source.
  + Harnois : des cases à cocher
  + Langues : un tableau, deux cases chacune (écrit / parlé)
  + Ressources immobilisées (le max étant calculé auto)
  + Notes – les Sources :
    - Gains : Création, Dé de vie, Objectif de Motivation, Carte d’Exaltation, Trésor, Guerrier (pour les sauvegardes).
    - Pertes : erreur (n’enregistre rien, supprime juste), perte de Dé de Vie, autre (champ à remplir)
    - Cet aspect implique de sauvegarder un historique des modifications
* Interface « Feuille de Personnage »
  + Mode Création, qui va demander le nombre de Vocations & Métiers bonus, et ne pas demander de Source.
  + Mode « Edition », pour la modification « en jeu »
    - Il peut être utile d’avoir des boutons « Objectif de patrouille réussi », « Objectif de Motivation Réussi », « Carte d’exaltation – Vocation », « Carte d’Exaltation – Métier », « Trésor – Vocation », « Trésor – Métier »
    - Ces boutons fourniront un certain nombre de « points d’amélioration » des types idoines, ouvriront la modification des PV, etc.
    - C’est vraiment secondaire
  + Mode « visualisation complète »
  + Mode « visualisation simple » (identique à la précédente)
* Ajout majeur : automatisation

### Evolution – Patrouilleur 3/x

Le but ici est d’ajouter la gestion de certains métiers à sélection simple :

* Druide : liturgies chtoniennes (sélection simple)
* Exorciste : exorcismes (sélection simple)
* Maître de guerre : ordres de bataille (sélection simple)

### Evolution – Patrouilleur 4/x

Le but est de s’occuper des métiers à sélection moyenne

* Aberrant : pouvoirs (sélection simple, possible de la perdre) et faiblesses (sélection simple, impossible à perdre)
* Marqué : type de marque (sélection définitive, seule la suppression pour Erreur permet de s’en débarrasser) et marques (sélection définitive, suppression dernière en date)
* Prêtre : panthéon (sélection définitive, seul la suppression pour Erreur permet de s’en débarrasser), divinité (sélection définitive, suppression dernière en date)

### Evolution – Patrouilleur 4/x

Le but est de s’occuper des métiers à sélection complexe

* Alchimiste : différents types de préparations
* Magicien : les sorts, avec obligation des rangs inférieurs, prix en mana, etc.

### Evolution – Patrouilleur 6/x

Le but est de s’occuper des métiers ajoutant des créatures au personnage

* Chevalier : montures, leurs améliorations
* Maître des bêtes : compagnons animaux, leurs augmentations
* Noble : suivants, type, DV, etc.

Evolution – Patrouilleur /x

Evolution – Patrouilleur /x

Evolution – Patrouilleur /x

### Evolution – Patrouille 1/x

* Ajout d’un élément « Patrouille »
  + Patrouilleurs : nombre de Patrouilleurs, de Montures, d’animaux de bât, de packs de suivants
  + Ressources transportables et transportées – à remplir à la main (Interface)
  + Consommation journalière – calcul automatique
* Dans l’interface
  + Indique et modifie le nombre de Patrouilleurs, de Montures, d’animaux de bât, de packs de suivants (Interface)
  + Indique et modifie Ressources transportables et transportées – à remplir à la main (Interface)
  + Indique la consommation journalière – calcul automatique (Interface)
  + Bouton « Fin de journée », « Consommer X », « Combat », « Blessé », « Rajout X » (Interface)

Evolution – Patrouille 2/x

* Lie les Patrouilleurs à l’objet Patrouille, qui désormais utilise ces informations pour renseigner tout ou partie de ses données

Evolution – Patrouille 3/x

* Rajout d’un élément sur la carte
  + Pion (avec image ?)
  + Déplaçable

Evolution – Patrouille 1/x

Evolution – Déplacement 1/x

Evolution – Déplacement 1/x

Evolution – Déplacement 1/x

Evolution – Déplacement 1/x

### Evolution –

Evolution –

Evolution –

Evolution –

Evolution –

Evolution –

Evolution –

Evolution –

# Roadmap

## Hexmap

### Données

* Nombre de colonnes (25)
* Nombre de lignes (25)
* Option de comptage des lignes et colonnes ? (0,1, début à gauche, à droite, etc.)

### Fonctionnalités

## Hexagone

### Données

* Coordonnées
* Couleur
* Image
* Description
* Degré d’exploration

### Fonctionnalités

* Sélection (avec liseré vert)

# Divers

\* [X] Lire https://blog.rsuter.com/recommendations-best-practices-implementing-mvvm-xaml-net-applications/

\*\* [X] Mettre en place une ViewModelBase et une INotifyPropertyChanged base classes, qui implémentent les fonctions nécessaires

\*\* [X] Toujours assigner le ViewModel en DataContext

\*\* [X] Ajouter Mytoolkit au projet : https://github.com/RSuter/MyToolkit/blob/master/README.md - le récupérer par NuGet

\*\* [X] Sa logique de commande ==> Menu ?

\*\* [X] Ne pas set le datacontext par binding dans xaml qui instancie la vue; Cf son code ==> méthode perso : get set vm dans le view, view.viewmodel = xxx pour le rajouter

\*\* [X] Explorer sa méthode de comm' pr message, surtout callback ! => à mettre en place d'abord dans le clic sur polygon pour les échanges

\*\* [X] Le choix entre user control et templated control peut attendre d'avoir plusieurs applications

\* [ ] Interface UI : Menu en haut

\*\* [X] Première implémentation

\*\* [X] Menu en haut : récupérer le bouton cliqué , Sinon une fonction différente par bouton

\*\* [X] Menu en haut : command = msg

\*\* [ ] Menu en haut : gérer la structure du menu par un fichier de configuration ? Clairement pas urgent

\* [X] Lien Models <=> Data : on récupère Data, on créée les Models, mais met-on à jour Data quand Model change ? Idéalement, V => VM => M

\* [X] ABSOLUMENT : un système pour gérer toutes ces commandes lancées de partout. On va possiblement devoir disposer de classes supplémentaires pour composer l'application. On a déjà le système des EventMessage d'un côté, EventLogic de l'autre. Il faudrait rassembler les fonctions ainsi : EventMessage les messages, EventLogic les liens EventMessage / GameLogic, et GameLogic les fonctions

\* [X] Sauvegarde / rechargement de Map par Sérialisation

\* [ ] Préparation passage BDD

\*\* [ ] Récupération enum par string (pour BDD)

\*\* [ ] Fichier "constant" pour les couleurs et les images, gestion par ID

\* [X] Faire un MVVM pour la Carte

\* [ ] Faire un MVVM pour l'Interface Gauche (LeftPanel) (au lieu de TileEditor)

\*\* [X] UserControl LeftPanel (simple pour le moment)

\*\* [ ] Menu déroulant contenant les "modes". Edition Hex, Selection Hex, Edition Road/River, etc.

\*\* [ ] Rajouter un champ de texte contenant la Description

\* [ ] Rivières et Routes

\*\* [ ] Réfléchir à comment enregistrer les données de localisation des points d'une rivière. Possiblement s'organiser avec les points centraux comme référence, et des fonctions de traduction

\*\* [ ] River(Data,Model,ViewModel,View)

\*\* [ ] Road(Data,Model,ViewModel,View)

\*\* [ ] Dessiner Rivière, route

\*\* [ ] Sélection Rivière, route

\*\* [ ] Rivière : possibilité d'identifier les hexagones où elle passe ? La direction ?

\* [ ] Au lancement, demander dimensions (et type terrain de base ?)

\* [ ] Regénération d'une carte (avec demande dimensions)

\* [ ] Créer un objet TileSet, contenant la liste des types de terrain, référencé dans Map ?

\* [ ] Ajouter toutes les tiles d'origine dans l'objet TileSet (interface ?), et en faire une version pour chaque set de tiles que j'ai, et faire un Enum + ue fonction pour récupérer le bon

\* [ ] Séparer les fonctions de Save / Load / New dans un autre fichier

\* [ ] Redimensionnement avec boutons +/-

\* [ ] Redimensionnement en cas de redimensionnement de fenêtre

\* [ ] Les Models s’initialisent eux-mêmes, avec des évènements qui indiquent leur chargement effectif et récupère les données à mettre dedans (en théorie, l’interface est « statique »)