

Rapport de projet

- Incidence -

*CHAMBONNET Kevin
GAUTHIER Silvère
MARTINEZ Thierry
MOKHRETAZ Amin*

May 15, 2014

Contents

| | |
|---|-----------|
| I Remerciements | 5 |
| II Cahier des charges | 6 |
| 1 Introduction | 6 |
| 1.1 Philosophie | 6 |
| 1.2 Questions fréquentes | 6 |
| 1.2.1 Qu'est-ce que ce jeu ? | 6 |
| 1.2.2 Qu'est-ce que je contrôle ? | 6 |
| 1.2.3 Quelles sont les conditions de victoire et de défaite ? | 6 |
| 2 Mecanismes de jeu | 6 |
| 2.1 Les Pouvoirs | 6 |
| 2.2 Choix de la nuit | 7 |
| 2.3 Sauver et Charger une partie | 7 |
| 3 Structure du jeu | 7 |
| 3.1 Moteur multi-agent | 7 |
| 3.1.1 Primitives | 7 |
| 3.1.2 Actions | 8 |
| 3.2 Les Entites | 8 |
| 3.2.1 Les métiers des citoyens | 8 |
| 3.2.2 La santé des entités vivantes | 8 |
| 3.3 Moteur de carte | 10 |
| 3.3.1 La carte | 10 |
| 3.3.2 Les Cases | 10 |
| 3.3.3 Diagramme de transitions des différentes cases | 11 |
| 3.3.4 Les incidences sur l'environnement : | 11 |
| 3.4 Les Ressources | 11 |
| 3.4.1 Ressources utilisées par les citoyens | 11 |
| 3.4.2 Ressources utilisées par le joueur | 12 |
| 3.5 La météorologie | 12 |
| 3.6 Le cycle jour/nuit | 12 |
| 3.7 Scripts | 13 |
| 4 Elements graphiques | 13 |
| 4.1 Les tuiles | 13 |
| 4.2 Les entités | 13 |
| 4.3 Interface utilisateur | 13 |
| III Gestion du projet | 14 |
| 5 Gestion de l'équipe | 14 |
| 6 Découpage en tâches | 14 |

| | | |
|-----------|--|-----------|
| 7 | Assignation | 15 |
| 8 | Gestion du temps | 15 |
| 9 | Profil de risques | 15 |
| 10 | Choix technologiques | 15 |
| 10.1 | Langages de programmation | 15 |
| 10.2 | Bibliothèques | 16 |
| 11 | Gestion des fichiers | 16 |
| 11.1 | Format des Fichiers | 16 |
| 11.1.1 | Animation | 16 |
| 11.1.2 | Tileset | 17 |
| 11.1.3 | Carte | 17 |
| 11.1.4 | Sauvegarde | 17 |
| 11.2 | Commentaires | 17 |
| 11.3 | Convention de Nommage | 18 |
| 11.4 | Gestion du code source | 18 |
| IV | Développement | 19 |
| 12 | Moteur de jeu | 19 |
| 12.1 | Gestion des états | 19 |
| 12.2 | Gestion de l'interface utilisateur | 19 |
| 12.3 | Gestion des ressources et animations | 19 |
| 13 | Moteur de carte | 19 |
| 13.1 | Les Tuiles | 19 |
| 13.1.1 | Les Sols | 20 |
| 13.1.2 | Les Eléments | 20 |
| 13.2 | Le Tileset | 20 |
| 13.3 | Incidences | 21 |
| 13.3.1 | Dilatation | 21 |
| 13.3.2 | Erosion | 21 |
| 13.3.3 | Aléatoire | 22 |
| 14 | Moteur multi-agent | 22 |
| 14.1 | Gestion des entités | 22 |
| 14.1.1 | La santé des entités vivantes | 22 |
| 14.1.2 | Incidences | 22 |
| 14.2 | Gestion des scripts | 23 |
| 14.3 | Liste des scripts | 23 |
| 14.3.1 | Les citoyens alliés | 23 |
| 14.3.2 | Les citoyens ennemis | 24 |
| 14.3.3 | Les animaux | 24 |
| 15 | Météo | 24 |

| | |
|--------------------------------------|-----------|
| 16 Architecture | 24 |
| 16.1 Machine à états | 24 |
| 16.1.1 Main menu state | 24 |
| 16.1.2 Game state | 25 |
| 16.1.3 Night state | 25 |
| 16.1.4 End state | 25 |
| 16.1.5 Game menu state | 25 |
| 16.1.6 Save menu state | 25 |
| 16.1.7 Load menu state | 26 |
| V Post-Mortem | 27 |
| 17 Réalisations non abouties | 27 |
| 18 Améliorations réalisables | 27 |
| VI Annexes | 28 |
| 19 Diagramme de Gantt | 28 |
| 20 Comparatif de performances | 28 |
| 21 Incidences sur la carte | 28 |

Part I

Remerciements

Un grand merci à Fabien Hervouet pour son encadrement.

Part II

Cahier des charges

1 Introduction

1.1 Philosophie

Du point de vue d'un observateur, un monde peut être qualifié de complexe sans que les individus y évoluant ne le soient forcément. Dans ce TER, il est question d'aborder cette thématique sous forme de jeu. Autrement dit, le problème qui nous intéresse ici est de déterminer comment faire émerger un comportement global en mettant en oeuvre des contraintes extérieures dans l'environnement pour influencer les interactions locales d'agents. De plus il faudra, pour le joueur, analyser le comportement des ces derniers afin d'optimiser la réalisation des buts demandés. Plus précisément, il s'agit de mettre en scène des agents (disposant d'un comportement précis et défini en terme d'interaction), évoluant dans un environnement modifiable par un joueur humain (qu'il s'agisse d'ajouter des ressources exploitables, de modifier le relief, etc.) devant atteindre un certain objectif (stabilité du système écologique, récolte d'une certaine quantité de ressources, développement d'un chemin).

1.2 Questions fréquentes

1.2.1 Qu'est-ce que ce jeu ?

C'est un jeu de type God-Like dans lequel on doit aider une civilisation à survivre le plus longtemps possible grâce à différents pouvoirs et actions divines.

1.2.2 Qu'est-ce que je contrôle ?

Vous ne contrôlez rien directement, tout se fait de façon indirecte grâce aux pouvoirs. Modifier l'environnement et aider les citoyens sont des actions qu'il faudra effectuer pour faire survivre la civilisation.

1.2.3 Quelles sont les conditions de victoire et de défaite ?

Il n'y aura pas de condition de victoire particulière dans le jeu de base, le but sera de survivre le plus longtemps possible.

La partie se terminera quand le joueur n'aura plus de citoyen.

2 Mecanismes de jeu

2.1 Les Pouvoirs

Le joueur aura la possibilité d'interagir sur l'environnement via différents pouvoirs.

- **Placer un Arbre :** Fait apparaître un Arbre sur une case choisie. Coût : 3 PI.

- **Placer un Arbre Fruitier** : Fait apparaître un Arbre Fruitier sur une case choisie. Coût : 6 PI.
- **Placer de la Pierre** : Fait apparaître de la Pierre sur une case choisie. Coût : 5 PI.
- **Placer de l'Eau** : Fait apparaître de l'Eau sur une case choisie. Coût : 2 PI.
- **Placer une Falaise** : Fait apparaître une Falaise sur une case choisie. Coût : 7 PI.
- **Placer un Buisson** : Fait apparaître un Buisson sur une case choisie. Coût : 4 PI.
- **Placer de la Terre** : Fait apparaître de la Terre sur une case choisie, la Terre placée s'adapte en fonction des autres Terres autour. Coût : 2 PI.
- **Soigner** : Améliore l'état de santé d'un citoyen. Coût : 50 PI.
- **Naissance** : Crée un citoyen supplémentaire, hors naissances quotidiennes. Coût : 200 PI.

2.2 Choix de la nuit

- **Météo** : Le joueur peut choisir la météo du jour suivant (ensoleillée ou pluvieuse).
- **Métier** : Le joueur peut orienter la distribution des métiers, sans définir exactement la proportion de chaque métier.

2.3 Sauver et Charger une partie

Le joueur pourra sauvegarder sa partie à tout moment, mais la sauvegarde se fera au moment de la dernière nuit.

Le chargement d'une partie placera donc le joueur soit au début de la partie soit à la dernière nuit passée, avec la possibilité de modifier de nouveau les différentes options du jour (météo, métiers...).

3 Structure du jeu

3.1 Moteur multi-agent

La gestion multi-agent se fera à l'aide d'un appel au script de chaque catégorie d'entité pour chaque tick. Chaque catégorie d'entité aura donc un script propre, qui décrira son comportement et pourra faire appel aux primitives que fournira le moteur pour décider de l'action à faire.

3.1.1 Primitives

- Connaître les entités et cases alentours
- Connaître l'emplacement du village
- Changer de direction
- Connaitre sa santé

- Si leur sac est plein
- S'il est attaqué
- Par qui sont-ils attaqués

3.1.2 Actions

- Avancer
- Couper un Arbre
- Ramasser des Baies
- Casser de la Pierre
- Rentrer au village
- Attaquer une entité

3.2 Les Entités

3.2.1 Les métiers des citoyens

Chaque citoyen aura une tâche à accomplir durant la journée et ne pourra pas en changer avant la nuit. La nuit, un métier sera attribué à chaque citoyen selon les choix du joueur et les besoins des citoyens (cf. section 3.6, page 12).

- **Bûcheron** : Coupe les arbres et récolte les ressources associées (Le Bois en général mais aussi de la nourriture sur les Arbres Fruitier).
- **Mineur** : Casse les rochers et récolte la Pierre.
- **Chasseur-Cueilleur** : Chasse les animaux, cueille les baies ou cultive des champs pour récolter la Nourriture.

3.2.2 La santé des entités vivantes

Chacune des entités possède une gestion de la santé avec plusieurs états.

- **Bonne santé** : Si l'entité est un citoyen, elle offre de plus grands bonus de PI.
- **Normal** : L'entité est dans son état par défaut.
- **Blessé/Malade** : L'entité agit avec un léger malus de vitesse. Si l'entité est un citoyen, elle offre de plus petits bonus de PI.
- **Gravement blessé/malade** : L'entité agit avec un malus de vitesse plus important. Si l'entité est un citoyen, elle n'offre plus de PI.
- **Mort** : L'entité disparaît.

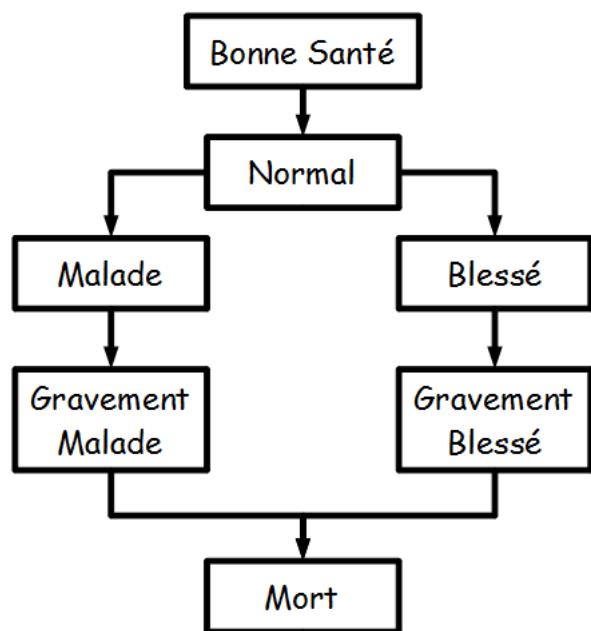


Figure 1: Diagramme de transitions entre les états de santé

3.3 Moteur de carte

3.3.1 La carte

La carte sera composée de 150x150 cases, mais ne sera affiché à l'écran qu'une vingtaine de cases en largeur sur une quinzaine en hauteur. Le déplacement sur la carte pourra se faire grâce aux touches fléchées mais aussi en plaçant la souris sur un des bords de l'écran.

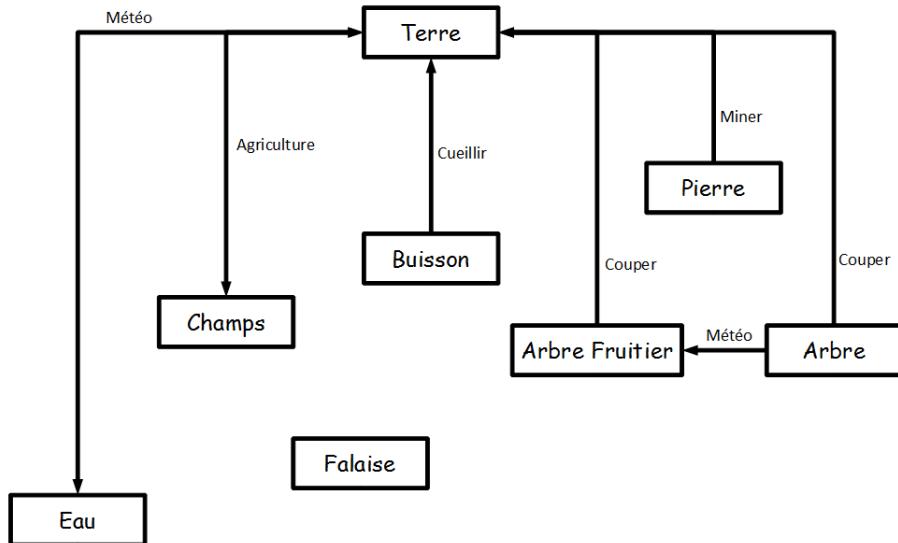
Le joueur ne pourra pas voir au delà des limites des 150x150 cases composant la carte.

3.3.2 Les Cases

La carte sera découpée en cases. Chaque case aura un type de base en début de partie, qui pourra ensuite être modifié selon le déroulement du jeu (cf. section 3.3.3, page 11). Un élément neutre est un type de case ne donnant pas lieu à une ressource quelconque.

| Type de case | Ressources | Franchissable | Description |
|----------------|--|---------------|---|
| Terre | - | Oui | Type par défaut. |
| Terre Aride | - | Oui | Terre ne pouvant pas être cultivée. |
| Terre Innondée | - | Oui | Terre ne pouvant pas être cultivée. |
| Terre Fertile | - | Oui | Terre pouvant être cultivée pour devenir un Champs . |
| Champs | Nourriture (1 à 3 unités) | Oui | Terre cultivée possédant plusieurs stades de maturité. Le maximum atteint, la récolte peut être effectuée et offrir de la nourriture. |
| Arbre | Bois (3 unités) | Non | Peut être coupé pour récolter du bois. |
| Arbre Fruitier | Bois, Nourriture (2 unités de chaque) | Non | Peut être coupé pour récolter du bois et de la nourriture. |
| Eau | - | Non | Des poissons peuvent s'y trouver permettant de récolter de la nourriture. |
| Pierre | Pierre (2 unités) | Non | Peut être cassée pour récolter de la pierre. |
| Falaise | - | Non | Elément neutre. Peut faire apparaître de la pierre à ses pieds. |
| Buisson | Nourriture (2 unités) | Non | La récolte de ses baies permet d'obtenir de la nourriture. |

3.3.3 Diagramme de transitions des différentes cases



3.3.4 Les incidences sur l'environnement :

- Une étendue d'eau peut faire apparaître des poissons.
- Une étendue d'eau peut faire apparaître de la végétation dans les environs.
- Une zone de végétation très dense augmente les chances de faire apparaître des animaux herbivores.
- Une grande concentration d'animaux herbivores peut faire apparaître des animaux carnivores.
- Une forêt très dense augmente les chances de faire apparaître des arbres fruitiers.
- Les falaises peuvent faire apparaître des pierres par éboulement.
- La météo peut modifier la taille des étendues d'eau, assécher ou humidifier la terre.

3.4 Les Ressources

3.4.1 Ressources utilisées par les citoyens

Ces ressources pourront être stockées dans une quantité illimitée, et les citoyens les utiliseront pour des constructions ou pour se nourrir.

- **Bois** : Utilisé pour la construction des bâtiments.
- **Pierre** : Utilisé pour la construction de certains bâtiments.
- **Nourriture** : Consommé par les citoyens chaque nuit pour se nourrir.

Comment ces ressources sont-elles récoltées ?

Elles sont récoltées par les citoyens, ainsi chaque métier correspond à la récolte d'une de ces ressources (cf. section 3.2.1, page 8).

3.4.2 Ressources utilisées par le joueur

Cette ressource est la seule que le joueur pourra utiliser, elle sera stockée dans une quantité illimitée.

- **Point d'Incidence (PI)** : Utilisé à chaque action ou pouvoir divin.

Comment cette ressource est-elle récoltée ?

Elle sera obtenue grâce aux citoyens qui nous en feront gagner une petite quantité tout au long de leur journée. La nuit, chaque citoyen rapporte des points bonus.

3.5 La météorologie

La météo sera présente et sera contrôlée par le joueur. Elle aura une incidence sur l'environnement et les citoyens. Elle sera basique : ensoleillée ou pluvieuse, chacune des deux aura une incidence différente.

- **Temps ensoleillé** : Améliore la pousse des champs mais un excès de soleil assèche les terres et récoltes, peut réduire les étendues d'eau et une sécheresse trop longue peut faire brûler certaines ressources.
- **Temps pluvieux** : Permet de faire pousser les champs. Un surplus de pluie inonde les terres et récoltes, augmente les probabilités de maladie et peut augmenter la taille des étendues d'eau.

3.6 Le cycle jour/nuit

Un cycle jour/nuit sera présent, avec des journées longues et des nuits courtes. Le Jour, les citoyens se voient à leur métier, jusqu'au soir. La Nuit, tous les citoyens retournent au village, plus aucune action n'est possible. Lorsque la nuit tombe, toutes les actions du jour ont une incidence sur la nature et les entités, et seront visibles au début de la nouvelle journée.

- Le terrain est mis à jour, toutes les actions de la journée auront une incidence sur l'environnement.
- Tous les citoyens se nourrissent, la nourriture diminue en fonction du nombre de citoyen (*-3 de nourriture par citoyen*). S'ils manquent de la nourriture, certains citoyens peuvent tomber malade.
- Certains citoyens tombent malade en fonction des anciennes météos.
- S'il y a assez de nourriture, de nouveaux citoyens peuvent naître.
- Gain des points bonus d'incidence en fonction de la taille de la population et de sa santé.
- Mise à jour du métier de chaque citoyen, choisi en fonction des choix du joueur et des besoins.

3.7 Scripts

Chaque catégorie d'entités déterminée selon leur classe d'appartenance aura un script propre. Ce script sera appelé à une fréquence variable en fonction de la situation dans laquelle se trouvera l'entité.

4 Elements graphiques

4.1 Les tuiles

Les tuiles seront des images de 32x32 ou 16x16 pixels au format PNG.

- **Le sol :** Il y aura 16 tuiles pour chaque type de sol, correspondant à toutes les possibilités de jonction avec le type voisin. En effet, chaque type de sol ne pourra être en contact qu'avec deux types différents suivant l'ordre hiérarchique suivant :
Eau → Terre innondée → Terre fertile → Terre → Terre aride
- **Les ressources :** Il y aura 4 tuiles pour chaque ressource, correspondant au type de sol sur lequel elle se trouve.
- **Les bâtiments :** Chaque bâtiment sera constitué d'un ensemble de tuiles.

4.2 Les entités

Chaque entité aura plusieurs positions possibles. Pour chacun d'elle, un ou plusieurs éléments graphiques seront créés.

Les positions basiques : Face, dos, profil droit, profil gauche.

Les positions spécifiques : Coupant du bois, ramassant des baies, chassant, cultivant, construisant, se déplaçant sans ressource, se déplaçant avec des ressources.

4.3 Interface utilisateur

Part III

Gestion du projet

5 Gestión de l'équipe

Tous les membres se connaissant et étant supposés être capable de travailler en équipe, nous n'avons fait aucune élection de chef de projet.

Nous avons opté pour travailler de manière collégiale, et ainsi garder une cohésion de groupe sans pour autant avoir de hiérarchie instaurée au sein du groupe, qui pourrait au contraire déservir la réalisation de nos objectifs.

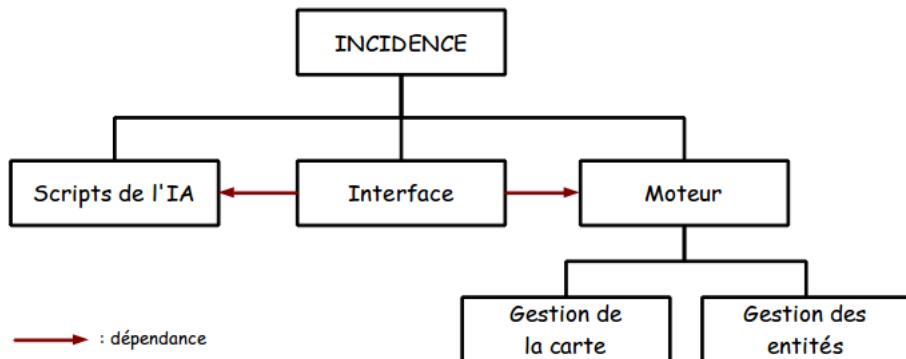
Chaque membre a donc autant de pouvoir que les autres, et peut donc participer activement au projet, autant lors de la conception que du développement. Toutes les décisions seront prises suivant la majorité lors de votes.

Pour ce qui est des réunions de projets, nous avons convenu avec notre tuteur d'une réunion, allant d'environ trente minutes à une heure, toutes les une à deux semaines, afin de mettre au point l'avancement du projet. En parallèle, tous les membres de notre équipe se retrouvent une fois par semaine afin de discuter des points clés effectués ou à venir, donner lieu aux votes pour les prises de décisions, ou encore, lors de la phase de développement, travailler en collaboration afin d'optimiser notre travail.

Au niveau du travail collaboratif, nous avons mis en place un dépôt sur github, contenant tant la documentation telle que ce rapport que les sources de notre jeu. Par ailleurs, nous mettrons sur ce dépôt uniquement les fichiers sources, les images et les sons, mais en aucun cas les fichiers temporaires ou les exécutables. Un fichier "makefile" sera disponible pour quiconque voudrait compiler le programme chez lui. Les seuls fichiers binaires disponibles seront les PDF de la documentation, pour un soucis de facilité d'accès.

6 Découpage en tâches

Afin de préparer le développement du jeu, il était nécessaire de séparer les fonctionnalités les unes des autres. Nous avons abouti à ce diagramme, qui résume notre choix de découpage :



7 Assignation

Le projet étant maintenant découpé en un certain nombre de modules, il ne restait plus qu'à assigner chaque tâche à un ou plusieurs membres de l'équipe. Nous nous sommes organisés comme ceci :

- **Scripts de l'IA** : MARTINEZ Thierry, MOKHRETAR Amin.
- **Moteur** :
 - **Gestion de la carte** : GAUTHIER Silvère.
 - **Gestion des entités** : CHAMBONNET Kevin.
- **Interface** : Tous les membres.

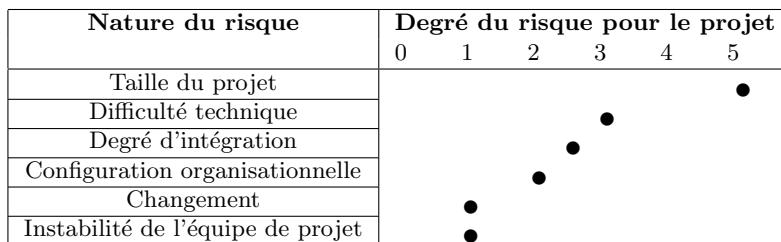
Bien entendu, cette répartition n'est pas totalement fixée, elle concerne en réalité l'affectation de responsables de parties, qui seront en charge de celle-ci mais pourront évidemment faire appel aux autres membres pour trouver une solution à un problème par exemple.

Le détail complet des tâches et assignations se situe dans la section Gestion du temps, page 15.

8 Gestion du temps

Afin de clarifier notre gestion du temps, un diagramme de Gantt est disponible en annexe et dans la documentation de notre projet, et sera mis à jour en fonction de l'avancée du projet.

9 Profil de risques



10 Choix technologiques

10.1 Langages de programmation

Pour des besoins de performances, nous avons comparé différents langages. Pour réduire le temps de recherche et de comparaison, nous nous sommes appuyé sur des tests déjà effectués par d'autre.

Des tests de performances concernant un large panel de langages, comparés

dans quatre contextes différents, sont fournis en annexe.

Nous pouvons observer que globalement, le langage le plus rapide est ici C++. L'utilisation de ce langage étant très fréquente dans les jeux vidéos, de part sa réputation d'un des langages les plus performants, et tous les membres de notre équipe sachant l'utiliser, nous avons fait le choix de programmer le moteur du jeu en C++.

Afin d'optimiser encore la rapidité du moteur, nous avons cherché à associer son cœur écrit en C++ avec un langage de scripting qui permettra de mettre en place les différentes actions du jeu.

D'après les graphiques ci-dessus, nous avons opté pour le langage LUA, performant et facile d'utilisation (syntaxe proche du C++). En effet, même si Python est très prisé et offre beaucoup plus de possibilités que LUA, nous l'avons estimé bien trop lourd pour l'utilisation que nous allons en faire.

Les deux langages C++ et LUA sont souvent associés dans les jeux vidéos, notre choix suit donc la tendance, ce qui nous offre une certaine assurance.

10.2 Bibliothèques

Pour la gestion graphique des tuiles composant la carte et des différentes entités, nous avons cherché une bibliothèque relativement simple d'utilisation mais surtout performante afin de garder la fluidité gagnée avec le choix des langages de programmation.

Connaissant la bibliothèque OpenGL, qui est bas niveau et performante dans les affichages deux et trois dimensions, nous nous sommes tournés vers deux bibliothèques utilisant OpenGL : SDL et SFML.

D'après plusieurs sites web et forums, les dernières versions (respectivement 2.0 et 2.1) de ces deux bibliothèques se valent en terme de performance.

En confrontant nos préférences personnelles quant au choix de l'une ou l'autre, nous nous sommes finalement mis d'accord pour utiliser la bibliothèque graphique SFML 2.1, qui paraît plus simple d'utilisation que la SDL. De plus, elle permet une gestion aisée des fichiers audio, ce qui sera un plus pour la finalité de notre jeu.

11 Gestion des fichiers

11.1 Format des Fichiers

Le moteur de jeu étant écrit en C++, nous utiliserons des fichiers d'en-tête au format HPP et des fichiers de définition au format CPP. Pour les scripts d'IA des entités, le langage étant Lua, ceux-ci seront au format LUA.

Toutes les images nécessaires au jeu seront au format PNG afin de pouvoir utiliser la transparence et garder la pleine qualité d'image (contrairement à JPEG qui perd de l'information à la compression).

Les sons quant à eux seront au format WAV afin d'éviter toute gestion de la compression de fichier pour de petits fichiers qui n'en ont aucunement besoin.

11.1.1 Animation

```
Chemin/vers/image.png size_x_sprite size_y_sprite play?(1/0) loop?(1/0)
+Frame position_x position_y temps_affichage
```

```
+Frame position_x position_y temps_affichage
+Frame position_x position_y temps_affichage
```

11.1.2 Tileset

Le tileset sera une unique image au format PNG représentant l'ensemble des tuiles possibles. Chaque type de sol constituera une ligne du tileset, avec toutes les variantes dépendant des rebords entre tuiles (ce fichier sera présenté plus en détails dans la section 13.2, page 20).

11.1.3 Carte

Les différentes cartes sauvegardées seront enregistrées dans des fichiers IMS (Incidence Map Save), dans lesquels seront stockés le tileset utilisé et la conformation de la carte (dimensions, sols et éléments).

11.1.4 Sauvegarde

Un fichier sera créé pour la sauvegarde du jeu et un autre pour la sauvegarde de la carte, cela permettant dans le futur de créer des cartes réutilisables, comme par exemple dans un éditeur de carte.

Le fichier de carte contiendra uniquement le lien vers le tileset ainsi que les types de sols et éléments présents sur la carte. Le fichier sera donc de la taille : $2 * \text{nombre_de_cases} * \text{sizeof}(\text{int}) + \text{sizeof}(\text{chemin_du_tileset})$.

11.2 Commentaires

Si une méthode ou fonction, voir même un bloc, dépasse une certaine taille (environ 10 lignes) ou devient trop compliquée, un commentaire sera ajouté avant celle-ci expliquant brièvement son processus :

```
/** Description :
 *** Entrée : ...
 *** Sortie : ...
 **/
```

| Marqueur spécifique | Signification |
|---------------------|--|
| TODO | A mettre à la place du code d'une fonctionnalité à implémenter |
| RECODE | A mettre au dessus du bloc d'une fonctionnalité à refaire |
| FIXME | A mettre au dessus du bloc d'une fonctionnalité contenant un bug |

11.3 Convention de Nommage

| Type de variable | Format du nom |
|---------------------------|---|
| Classe | Majuscule suivit de minuscules |
| Méthode et Fonction | Minuscules (pour les mots composés, chaque mots suivant est une majuscule suivit de minuscules) |
| Attribut de classe | m_- |
| Variable globale | g_- |

11.4 Gestion du code source

Afin de faciliter le travail collaboratif, nous utilisons un dépôt utilisant le gestionnaire de version GIT, hébergé sur le site <https://github.com/>. Sur ce dépôt seront présents tous les fichiers sources nécessaires au développement du jeu ainsi que les documentations au format Latex et PDF (même si aucun fichier binaire ne devrait être présent, il est plus pratique de récupérer directement un tel fichier que de le compiler soit-même). De plus, y seront stockées toutes les données utilisées par le jeu telles que les images et les sons. Seuls les fichiers temporaires, exécutables et fichiers de sauvegarde ne seront pas stockés.

Part IV

Développement

12 Moteur de jeu

Pour simplifier le développement et les réalisations de certaines tâches, un moteur de jeu a été réalisé. Il servira surtout dans la gestion des différents états du jeu, de toutes les données utilisées (image ou son) mais aussi d'une interface utilisateur.

12.1 Gestion des états

Chacun des états possibles du jeu est une classe héritant d'un class mère abstraite, State, qui définit le comportement de base d'un état, soit une méthode de mise à jour de l'état, une affichage des informations et deux dernières gérant les événements clavier/souris et les events liés à l'interface utilisateur (clique sur un bouton, ...).

Tous ces différents états sont stockés dans une pile et contrôlés par un gestionnaire d'état, le StateManager. C'est un singleton accessible partout, ce qui permet à n'importe quelle instance d'état d'ajouter ou supprimer un état.

12.2 Gestion de l'interface utilisateur

Different widget permettant la création d'un interface utilisateur simple à être réalisé. On peut donc facilement ajouter des boutons et des barres de saisie de texte personnalisable qui seront liés à une instance d'UI contenu dans un état. Les événements associés aux boutons seraient automatiquement envoyés à l'état qui pourra traiter cet événement pour réaliser une action.

12.3 Gestion des ressources et animations

Pour la gestion de toutes les ressources, tels que les nombreuses images, on a réalisé un gestionnaire de ressource qui s'occupe de charger toutes les données nécessaires quand elles sont requises tout en évitant les doublons. Pour cela, chaque ressource est stockée dans un tableau associé à son nom, ce qui permet une recherche plus rapide et une impossibilité d'avoir des doublons.

Une classe permettant de créer des animations facilement a aussi été faite, ce qui facilite grandement la gestion de celle-ci et leurs utilisations.

13 Moteur de carte

13.1 Les Tuiles

Les différents types de tuile seront tous recensés dans le tileset, qui se chargera de fournir les textures et attributs associés à ceux-ci. La carte n'utilisera que des pointeurs vers ces instances constantes de tuile, ce qui évitera le stockage inutile d'instances identiques.

13.1.1 Les Sols

Chaque sol aura différents attributs définis dans le fichier de configuration du tileset (cf section 13.2, page 20). Comme définis dans le cachier des charges, certains sols seront incompatibles avec d'autres.

| Attribut | Description |
|---------------|--|
| Type | Entier définissant un identificateur du type de sol. |
| Coût | Entier définissant le coût en PI de la pose du sol. |
| Nom | Chaîne de caractères utilisées lors des affichages dans l'interface. |
| Comportement | Enumération de trois comportements : DEFAULT : comportement par défaut (tous les types de Terre). FLUID : les sols forment des zones (étendues d'eau). CLIFF : les sols sont fixes et forment des zones (falaises). |
| Franchissable | Booléen indiquant si le sol est franchissable ou non par les unités. |
| Rebords | Tableaux d'entiers indiquant tous les types de sol compatibles avec les bordures de l'image (cf section 13.2, page 20). |

13.1.2 Les Eléments

Chaque sol aura différents attributs définis dans le fichier de configuration du tileset (cf section 13.2, page 20).

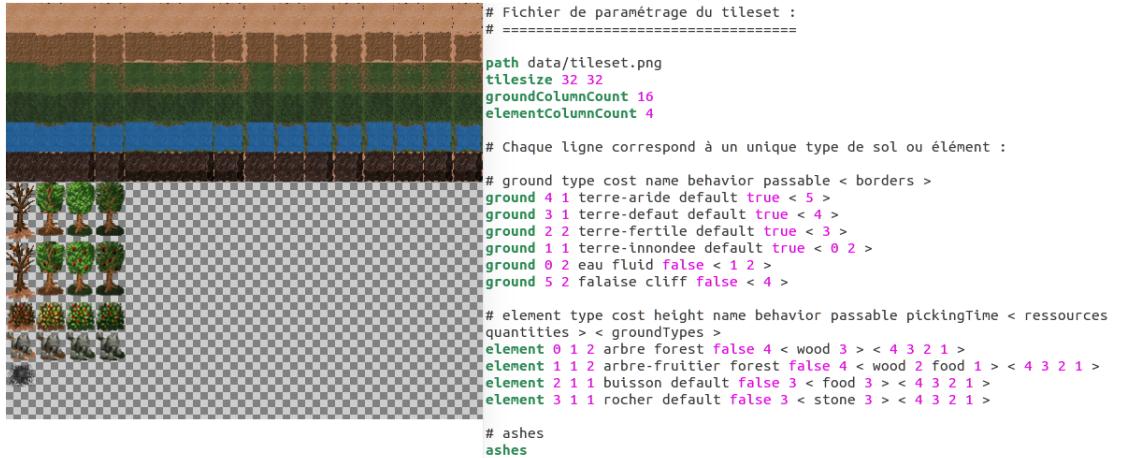
| Attribut | Description |
|------------------|--|
| Type | Entier définissant un identificateur du type d'élément. |
| Type de sol | Entier définissant un identificateur du type de sol sur lequel se place l'élément. |
| Coût | Entier définissant le coût en PI de la pose de l'élément. |
| Nom | Chaîne de caractères utilisées lors des affichages dans l'interface. |
| Comportement | Enumération de deux comportements : DEFAULT : comportement par défaut. FOREST : les éléments forment des zones (arbres). |
| Franchissable | Booléen indiquant si le sol est franchissable ou non par les unités. |
| Temps de récolte | Valeur de temps durant laquelle le citoyen doit agir afin de récolter les ressources. |
| Ressources | Ressources (et leur quantité) associées à l'élément. |

13.2 Le Tileset

Un unique fichier image au format PNG contiendra l'ensemble des tuiles utilisées dans le jeu pour les sols ou éléments. Cette image sera utilisée comme texture et certaines parties seront extraites par le programme au gré des besoins.

Dans le but de permettre à l'utilisateur d'aisément créer des mods de jeu ou changer de tileset, nous avons créé un fichier au format INI détaillant toutes les particularités du tileset.

Voici le tileset que nous utiliserons :



Fichier de configuration :

Chaque ligne commençant par "#" ne sera pas lue, ce sont les commentaires. Les mots en vert sont les mots clés utilisés pour la lecture qui se fait ligne par ligne :

- path : indique le chemin du fichier de texture du tileset
- tileSize : est suivi des dimensions (en pixel, largeur puis hauteur) d'une tuile
- groundColumnCount : nombre de colonnes pour chaque sol dans le fichier de texture (toutes les associations de rebords)
- elementColumnCount : nombre de colonnes pour chaque élément dans le fichier de texture (tous les types de sols)
- ground : définit un sol et tous ses attributs
- element : définit un élément et tous ses attributs (height est la hauteur en nombre de cases dans la texture)
- ashes : fourni une texture pour les cendres (lorsque des éléments brûlent) (cf GDD)

Remarque : chaque ligne commençant par ground, element ou ashes incrémente un compteur afin de connaître la position de la zone de texture correspondante. L'ordre est donc important (définition des lignes de haut en bas).

13.3 Incidences

13.3.1 Dilatation

Les incidences basées sur la dilatation des zones prennent en compte les types de sols ou éléments afin d'élargir les étendues voulues en propageant les sols si besoin afin de garder les contraintes définies dans le GDD.

Quelques présentations des résultats de ces dilatations sont présentes en annexe.

13.3.2 Erosion

Les incidences basées sur l'érosion des zones prennent en compte les types de sols ou éléments alentours afin de choisir les types en propageant les sols si

besoin afin de garder les contraintes définies dans le GDD.

Quelques présentations des résultats de ces érosions sont présentes en annexe.

13.3.3 Aléatoire

Les dernières incidences concernant le territoire sont basées sur un choix aléatoire de cases. Il s'agit ici d'apparition ou disparition de certaines ressources.

Quelques présentations des résultats de ces incidences particulières sont présentes en annexe.

14 Moteur multi-agent

14.1 Gestion des entités

14.1.1 La santé des entités vivantes

Chacune des entités possède une gestion de la santé avec plusieurs états.

- **Bonne santé** : Si l'entité est un citoyen, elle offre de plus grands bonus de PI.
- **Normal** : L'entité est dans son état par défaut.
- **Blessé/Malade/Fatigué** : L'entité agit avec un léger malus de vitesse. Si l'entité est un citoyen, elle offre de plus petits bonus de PI.
- **Gravement blessé/malade** : L'entité agit avec un malus de vitesse plus important. Si l'entité est un citoyen, elle n'offre plus de PI.
- **Mort** : L'entité disparaît.

14.1.2 Incidences

La météo influe sur les entités en fonction de la valeur de la jauge.

- **Jauge positive(1 ou plusieurs jours de temps ensoleillé consécutifs)** : Améliore la pousse des champs mais un excès de soleil assèche les terres et récoltes, peut réduire les étendues d'eau, une sécheresse trop longue peut faire brûler certaines ressources et augmenter la probabilité de fatigue .
- **Jauge négative(1 ou plusieurs jours de temps pluvieux consécutifs)** : Permet de faire pousser les champs. Un surplus de pluie inonde les terres et récoltes, augmente les probabilités de maladie et peut augmenter la taille des étendues d'eau.

L'état de santé joue un rôle capital au niveau de la vitesse de déplacement, du temps de récolte et aussi sur la fréquence à laquelle une entité va vous apporter des points d'incidences(PI).

Un état de santé faible , une maladie ou un état très faible peut entraîner le décès d'une entité avec une probabilité de plus en plus forte dans ce même ordre de citation. Aussi chaque entité a une légère chance de donner naissance à un petit.

14.2 Gestion des scripts

Les scripts LUA implementent les différents comportements de toutes les catégories d'entités. Nous avons utilisé une librairie LUA permettant de redéfinir des méthodes de classe implémenté en C++ et de définir des méthodes dans les classes C++ qui pourront être appelé par le script LUA. On utilise donc un script LUA pour redéfinir une méthode de la classe "Entity" qui sera appelé pour simuler l'intelligence de l'entité. Par contre, cette librairie et les mécanismes de C++ nous ont imposé certaines contraintes. La plus importante est qu'il est impossible que 2 instances d'une même classe possède 2 définitions différentes d'une de leurs méthodes. Il nous a donc fallu implémenter une sous-classe pour chaque catégorie d'entités afin d'avoir un comportement différent pour toutes ces catégories. Cette problématique empêche les joueurs de pouvoir créer de nouvelles catégories d'entité, il pourra cependant modifier celle déjà existante.

14.3 Liste des scripts

Le comportement des agents, en ce qui concerne le déplacement, a soulevé le problème de la gestion des obstacles étant donné que certains éléments du décor ne sont pas franchissables. En effet, certaines actions nécessitant un trajet spécifique utilisent le résultat d'un algorithme de recherche de chemin et se déroulent sans interruption sur plusieurs ticks, c'est-à-dire sans rappeler le script de l'agent. Par conséquent, les directives des scripts sont en partie des objectifs à atteindre et non de simples actions isolées.

14.3.1 Les citoyens alliés

- **Récolteurs(bûcherons, cueilleurs, mineurs) :**

Si un bûcheron faible, un mineur faible ou un cueilleur perçoit un animal agressif ou un citoyen ennemi celui-ci va fuir.

Si un bûcheron ou un mineur se fait attaquer, il va répliquer en attaquant en priorité l'entité face à lui si celle-ci l'attaque.

A chaque fois que l'entité récolte une ressource elle remplit son sac puis se dirige vers la base pour la déposer.

Lorsque son sac est vide, elle regarde dans son rayon de perception si une ressource est disponible. Si au moins une ressource est repérée, elle se dirige vers la plus proche pour la récolter. Sinon, elle se dirige de manière aléatoire.

- **Chasseurs :**

Le chasseur reste relativement proche de la base et adopte un rôle de défenseur. S'il est attaqué, il va regarder le nombre de chasseurs et le nombre d'entités hostiles autour de lui, s'il y a moins d'entités hostiles que de chasseurs auxquels on ajoute 3, il va attaquer l'entité qui l'attaque avec dans l'ordre de priorité : l'entité en face de lui, un citoyen ennemi et un animal agressif. Sinon il adopte la fuite.

S'il n'est pas attaqué, tout en gardant une distance proche de la base, il va chercher une cible autour de lui dans cet ordre de priorité, un citoyen ennemi, un animal agressif puis un animal passif. Encore une fois s'il n'y a pas trop d'ennemis aux alentours, le chasseur va aller attaquer la cible sinon il va fuir. S'il n'y a ni citoyens ennemis ni animaux autour de lui et s'il n'est pas trop loin de la base, il va avancer aléatoirement sinon il va revenir vers la base.

14.3.2 Les citoyens ennemis

Les citoyens ennemis ont un comportement agressif. S'ils sont attaqués, il vont adopter le même comportement qu'un chasseur sauf que dans l'ordre de priorité ils vont attaquer: l'entité en face d'eux, un chasseur et un récolteur. S'ils ne sont pas attaqués, même démarche que le chasseur en changeant l'ordre de priorité par un chasseur puis un récolteur.

Si aucun chasseur ou récolteur n'est perçu dans leurs rayons de perception, ils vont avancer aléatoirement.

14.3.3 Les animaux

- **Agressifs :**

Dès qu'un animal agressif perçoit une entité excepté un animal agressif il va l'attaquer, en attaquant en priorité l'entité qui est en face de lui. Si aucune entité n'est perçue ou s'il ne perçoit que des animaux agressifs , l'animal va se déplacer aléatoirement.

- **Passifs :**

L'animal passif va tout simplement fuir toutes les entités à l'exception des animaux passifs.

Dans certaines situations vues précédemment, l'entité se déplacera de manière aléatoire et pourra marquer un arrêt de temps à autre afin de simuler un comportement naturel.

15 Météo

La météo est gérée comme une jauge comprise entre cinq et moins cinq. Au commencement d'une partie la jauge est à zéro. A la fin de chaque journée nous comparons le temps du jour terminé avec le temps du jour précédent, si celui-ci est le même et est ensoleillé la jauge augmente , si par contre le temps est pluvieux la jauge descend. En revanche, si la jauge est entre -3 et -5 et que la journée fut ensoleillé ou inversement si la jauge est entre 3 et 5 et qu'il fit un temps pluvieux la jauge revient respectivement à -1 ou 1. Plus la valeur de la jauge est grande positivement ou négativement, plus il y aura d'incidences en rapport avec la pluie pour les valeurs négatives et inversement pour le soleil.

16 Architecture

16.1 Machine à états

16.1.1 Main menu state

Le menu principal est le premier état de l'application, il sert d'écran d'accueil et dispose de trois boutons :

- "Start" qui crée une nouvelle instance de jeu et appelle l'état "Game State".
- "Load" qui appelle l'état "Load Menu State".
- "Exit" qui quitte l'application.

16.1.2 Game state

L'état de jeu est le primordial au sein de l'application, il sert de support à l'instance de jeu afin d'en gérer le déroulement. Il contient l'affichage de la partie visible de la carte du monde et des entités ainsi qu'une interface, permettant au joueur de gérer la partie, ayant deux fonctions. La première consiste à informer de la valeur de certaines variables inhérentes à la partie comme le nombre de points ou d'une certaine ressource. La seconde permet au joueur d'indiquer les actions qu'il souhaite réaliser via un ensemble de boutons.

16.1.3 Night state

L'état de nuit revient régulièrement au cours de la partie, pendant celui-ci ont lieu les incidences sur le territoire (cf section 13.3, page 21) et sur les entités (cf section 14.1.2, page 22). De plus, cet état permet au joueur de déterminer directement la météo de la prochaine journée de jeu et d'influer indirectement sur les proportions des rôles des entités. En plus de cela, un bouton de validation supprime l'état courant afin que la partie puisse reprendre son cours dans l'état de jeu.

16.1.4 End state

L'état de fin de partie signale au joueur qu'il a atteint la condition de défaite, il dispose simplement d'un bouton :

- “Back” qui supprime l'état courant.

16.1.5 Game menu state

Le menu de jeu est l'état de pause du jeu, il dispose de quatre boutons :

- “Back” qui supprime l'état courant.
- “Save” qui appelle l'état “Save Menu State”.
- “Load” qui appelle l'état “Load Menu State”.
- “Quit” qui supprime tous les états jusqu'à l'état “Main Menu State”.

16.1.6 Save menu state

Le menu de sauvegarde est l'état qui permet de sauvegarder son avancement, il dispose d'un champ de saisie du nom de sauvegarde et de deux boutons :

- “Back” qui supprime l'état courant.
- “Save” qui crée un fichier de sauvegarde à partir de l'instance de jeu en cours et le nomme d'après le texte entré.

16.1.7 Load menu state

Le menu de chargement est l'état qui permet de charger une partie précédemment sauvegardée, il dispose d'une liste de boutons, représentant les fichiers de sauvegarde répertoriés à partir desquels l'instance de jeu sera modifiée, de deux boutons permettent de parcourir la liste et d'un dernier bouton :

- “Back” qui supprime l'état courant.

Part V

Post-Mortem

17 Réalisations non abouties

- Utilisation de la pierre et du bois pour la construction de bâtiments

18 Améliorations réalisables

Part VI

Annexes

Dans cette partie seront placés les éléments trop volumineux pour être inclus directement dans le texte, tels que les images ou graphiques.

19 Diagramme de Gantt

Le diagramme est fourni page 29 - 30.

20 Comparatif de performances

Le graphique est situé page 31.

21 Incidences sur la carte

Les présentations sont présentes de la page 32 à la page 42.

| | Nom | Durée | Début | Nom de la Ressource |
|-----------|------------------------------|------------------|-----------------------|-------------------------------|
| 1 | Etude Préalable | 30 jours | 20/01/14 08:00 | CHAMBONNET Kevin;GAUTHIER .. |
| 2 | ■Moteur | 42 jours? | 03/03/14 08:00 | |
| 3 | ■Gestion des Entités | 42 jours? | 03/03/14 08:00 | |
| 4 | Moteur multiagent | 20 jours? | 03/03/14 08:00 | |
| 5 | Implémenter les Entités | 5 jours | 14/04/14 08:00 | CHAMBONNET Kevin |
| 6 | Placement des Entités | 1 jour | 21/04/14 08:00 | MOKHRETAR Amin |
| 7 | Gestion des Maladies | 2 jours | 25/04/14 08:00 | MARTINEZ Thierry |
| 8 | Incidences de la Santé | 1 jour | 29/04/14 08:00 | MARTINEZ Thierry |
| 9 | ■Gestion de la Carte | 41 jours | 03/03/14 08:00 | |
| 10 | ■Moteur de Carte | 39 jours | 03/03/14 08:00 | |
| 11 | Conception | 5 jours | 03/03/14 08:00 | GAUTHIER Silvère |
| 12 | ■Implementation des Classes | 26 jours | 17/03/14 08:00 | |
| 13 | Gestion du TileSet | 4 jours | 17/03/14 08:00 | GAUTHIER Silvère |
| 14 | Sols et Elements | 3 jours | 21/03/14 08:00 | GAUTHIER Silvère |
| 15 | Génération de Carte | 3 jours | 26/03/14 08:00 | GAUTHIER Silvère |
| 16 | Optimisations Diverses | 1 jour | 15/04/14 08:00 | GAUTHIER Silvère |
| 17 | Méthode "freePlace" | 1 jour | 21/04/14 08:00 | GAUTHIER Silvère |
| 18 | Affichages | 1 jour | 14/04/14 08:00 | GAUTHIER Silvère |
| 19 | Gestion des Bâtiments | 2 jours | 22/04/14 08:00 | MOKHRETAR Amin |
| 20 | Affichage des Bâtiments | 1 jour | 24/04/14 08:00 | MOKHRETAR Amin |
| 21 | Incidences sur le Territoire | 3 jours | 16/04/14 08:00 | GAUTHIER Silvère |
| 22 | Incidences sur les Entités | 2 jours | 22/04/14 08:00 | GAUTHIER Silvère |
| 23 | Sauvegarder / Charger | 3 jours | 24/04/14 08:00 | GAUTHIER Silvère |
| 24 | ■Cycle Jour/Nuit | 4 jours | 21/04/14 08:00 | |
| 25 | Jour | 1 jour | 21/04/14 08:00 | CHAMBONNET Kevin |
| 26 | Nuit | 3 jours | 22/04/14 08:00 | CHAMBONNET Kevin |
| 27 | Météo | 1 jour | 21/04/14 08:00 | MARTINEZ Thierry |
| 28 | ■Points d'incidence | 5 jours | 22/04/14 08:00 | |
| 29 | Ajout | 3 jours | 22/04/14 08:00 | MARTINEZ Thierry |
| 30 | Suppression | 2 jours | 25/04/14 08:00 | MOKHRETAR Amin |
| 31 | Sauvegarder / Charger | 2 jours | 25/04/14 08:00 | CHAMBONNET Kevin |
| 32 | ■Scripting | 35 jours? | 03/03/14 08:00 | |
| 33 | Lister les Scripts | 5 jours | 03/03/14 08:00 | MARTINEZ Thierry;MOKHRETAR .. |
| 34 | Etude préalable | 5 jours? | 10/03/14 08:00 | MARTINEZ Thierry;MOKHRETAR .. |
| 35 | Concevoir les Scripts | 25 jours? | 17/03/14 08:00 | MARTINEZ Thierry[50 %];MOKH.. |
| 36 | Implémenter les Scripts | 25 jours? | 17/03/14 08:00 | MARTINEZ Thierry[50 %];MOKH.. |
| 37 | ■Interface Graphique | 12 jours | 01/05/14 08:00 | |
| 38 | Widgets | 3 jours | 01/05/14 08:00 | CHAMBONNET Kevin |
| 39 | Menus | 5 jours | 05/05/14 08:00 | MOKHRETAR Amin |
| 40 | Gestion Animations | 3 jours | 07/05/14 08:00 | MARTINEZ Thierry |
| 41 | Gestion Sons | 2 jours | 01/05/14 08:00 | MARTINEZ Thierry |
| 42 | Informations contextuelles | 5 jours | 12/05/14 08:00 | CHAMBONNET Kevin;MARTINEZ .. |
| 43 | ■Graphismes | 45 jours | 08/03/14 08:00 | |
| 44 | Elements de Carte | 5 jours | 08/03/14 08:00 | GAUTHIER Silvère |
| 45 | Entites | 2 jours | 29/04/14 08:00 | GAUTHIER Silvère |
| 46 | Elements d'interface | 7 jours | 01/05/14 08:00 | GAUTHIER Silvère |
| 47 | Equilibrage du Jeu | 5 jours | 19/05/14 08:00 | CHAMBONNET Kevin;GAUTHIER .. |

Figure 2: Diagramme de Gantt (page 1)



Figure 3: Diagramme de Gantt (page 2)

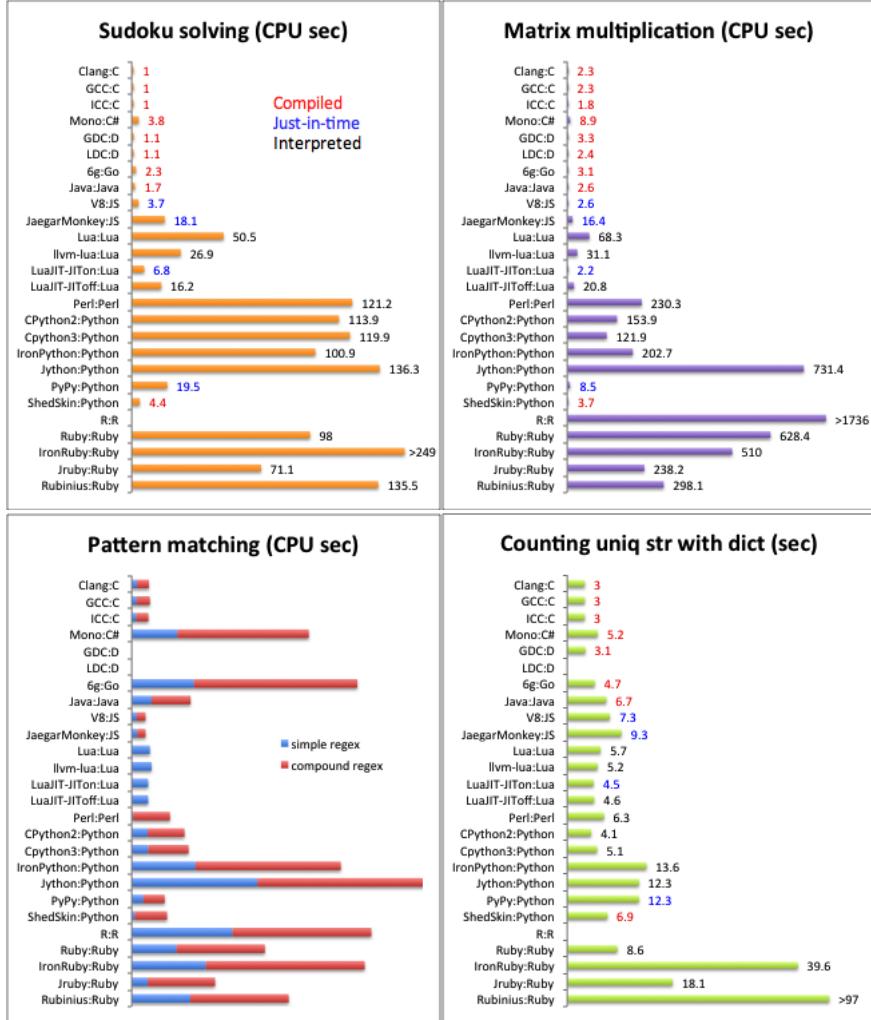


Figure 4: Comparaisons de performances de divers langages dans des cas donnés



Figure 5: Dilatation des étendues de fluides



Figure 6: Dilatation des zones humides



Figure 7: Dilatation des zones sèches



Figure 8: Dilatation des forêts



Figure 9: Erosion des étendues de fluides



Figure 10: Erosion des zones humides



Figure 11: Erosion des zones sèches



Figure 12: Erosion des forêts



Figure 13: Apparition de tout type de ressource



Figure 14: Disparition par le feu des ressources contenant du bois



Figure 15: Apparition de rochers uniquement