

Travail de bachelor 2008

Filière Informatique de gestion

Projet de simulation multi-agent



Etudiant : Stéphane Claret

Professeur : Michael Ignaz Schumacher

Table des matières

Introduction	4
Le projet Juste-Neige	5
Le travail de bachelor	5
La simulation	6
Motivation personnelle	8
Les bases d'une simulation	9
Premier exemple sous Repast	11
Premier exemple	12
Description	12
Screenshots	13
Comportement des agents	14
Erosion de la couche neigeuse	15
Représentation graphique	15
Résumé du flux	17
Conclusion	18
Choix d'architecture	19
Le problème du routage	20
Une première idée de solution	22
Les points de décisions	23
L'abandon de la diffusion	24
La solution choisie	25
Conclusion	27
Le graphe du domaine skiable	28
Introduction	29
Description générale	31
Comportement des agents	32
Chemin prédéfini	32
Décision à la volée	35
Fin de la simulation	36
Conclusion	36

Table des matières (suite)

Simulation des grilles locales	37
Introduction.....	38
Description générale.....	39
Comportement des agents.....	40
Erosion de la couche neigeuse	42
Arrêt et vue d'ensemble.....	43
Conclusion.....	44
Mise en commun de l'ensemble	45
Besoin d'homogénéisation.....	46
L'échange de données	47
Déroulement du processus.....	49
Conclusion.....	50
Conclusion	51
Proof of concept	52
Déroulement du travail.....	52
Bilan personnel.....	53
Ressources Web	54
Calendrier.....	55

Introduction

Chapitre 1

Projet de simulation multi-agent

Stéphane Claret

Le projet Juste-Neige

Juste Neige est le nom d'un projet de recherche qui a démarré en juillet 2008 auquel participent entre autres l'institut d'économie et tourisme de la HES-SO de Sierre ainsi que l'institut IGAR de l'université de Lausanne en partenariat avec les principales sociétés de remontées mécaniques valaisannes.

Ce projet s'inscrit dans le cadre polémique du futur de l'enneigement des stations de ski concernant les perspectives problématiques du réchauffement climatique et du développement durable.

Son but principal est d'étudier et de concevoir des outils d'aide à la gestion de l'enneigement des stations de skis, afin de permettre l'optimisation de la production de neige de culture. De cette façon il serait possible de

- Minimiser les coûts engendrés par la production de neige artificielle et optimiser l'utilisation des moyens à disposition en ne produisant que les quantités nécessaires.
- Améliorer la gestion des réserves d'eau destinées à la production de neige, diminuer les besoins en ressources énergétiques et ainsi minimiser l'impact environnemental en accord avec la politique du développement durable.

Le travail de bachelor

Le développement des outils prévisionnels du projet Juste-Neige passe par l'implémentation d'une simulation multi-agent permettant de simuler les skieurs se déplaçant sur les pistes de ski et ainsi d'estimer leur impact sur l'état de la neige tout au long de leur parcours.

Il serait possible grâce notamment à une telle simulation de prévoir l'état de la neige plusieurs jours à l'avance et ainsi d'en utiliser les résultats pour évaluer les nécessités en neige artificielle.

Mon travail de diplôme a consisté en la mise en place de la première version de cette simulation. Première version seulement car il était évident dès le départ qu'il ne serait pas possible de la finir entièrement avec le temps à disposition.

Le projet a été suivi par Mr. Michael Schumacher, qui possède déjà de l'expérience dans les domaines de la simulation multi-agent et de l'intelligence artificielle. Les principaux acteurs de travail hormis moi-même ont été :

- Mr. Michael Schumacher, décideur et responsable du suivi de projet.
- Mr. Jean-Christophe Loubier, chercheur en géomatique, expert en systèmes GIS.

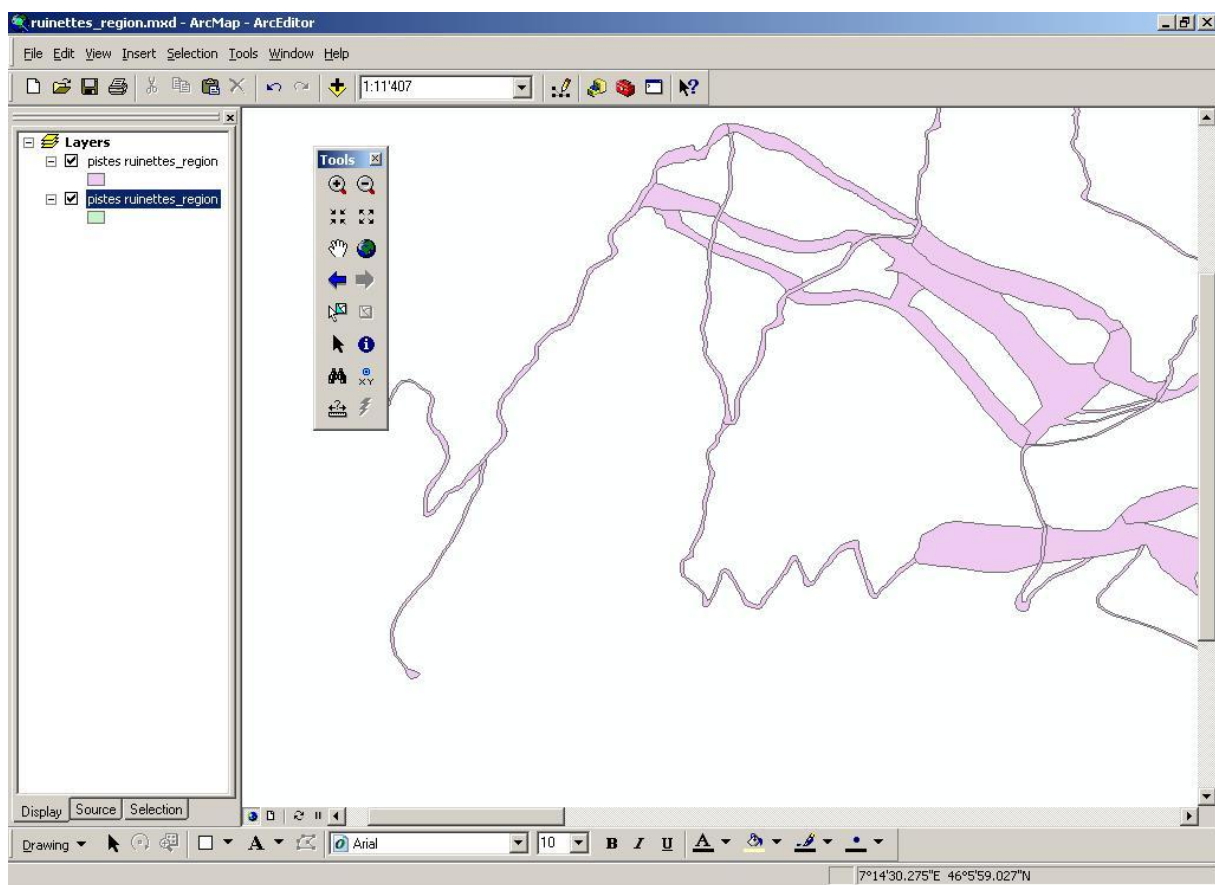
La simulation

Utilisation de données GIS

Cette simulation de skieurs que nous sommes en charge de créer doit naturellement avoir lieu sur un environnement qui correspond à celui d'une véritable piste de ski.

Nous devons pour ce faire utiliser des cartes de domaine skiable provenant d'un système SIG (système d'information géographique), GIS en anglais. Le logiciel SIG utilisé dans le projet juste-neige est ESRI ArcGis, il s'agit d'une vaste suite d'outils de géomatique.

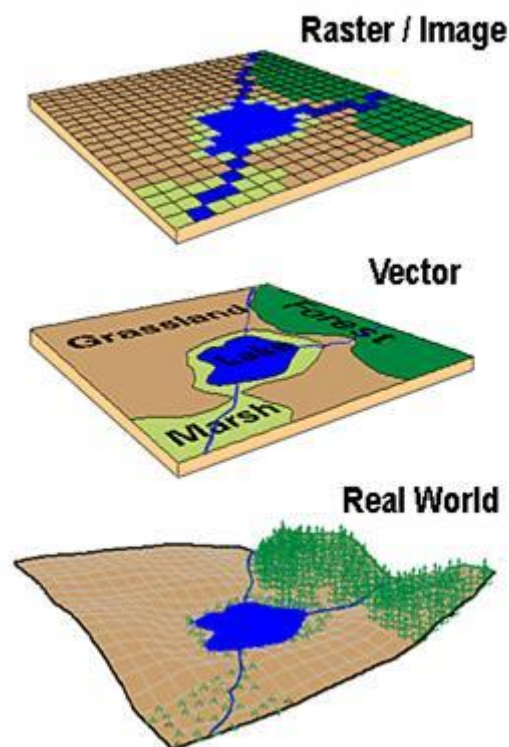
Voici ci-dessous un screenshots de la carte du domaine skiable des ruinettes de Verbier dans ArcGis:



Ces cartes GIS modélisent donc un réseau de pistes (en violet) en deux dimensions, c'est à l'intérieur de ces zones violettes que nos agents skieurs se déplaceront en définitive. Notre rôle sera de leur fabriquer un comportement cohérent dans ce milieu.

La carte de l'illustration précédente est une carte dite vectorielle. Cela signifie qu'elle est construite à partir de points et de polygones. Notre simulation ne travaillera pas sur des cartes vectorielles mais des cartes raster.

La rasterisation est un procédé utilisé lorsque l'on souhaite analyser les attributs d'un espace précis sur une carte. Il consiste à quadriller la carte en lignes et en colonnes, comme pour la "pixelliser".



La différence entre une vue vecteur et une vue raster source (bgis.sanbi.org)

Chacune des petites cases ainsi obtenues est localisable facilement à l'aide de coordonnées X et Y entières et peut contenir un certain nombre d'attributs (type de terrain, altitude, hauteur de neige).

Dans le projet juste neige, les cartes sont découpées en carrés de 5x5m, le terrain sur lequel nous ferons évoluer nos skieurs sera donc en vérité une grille comparable à celle d'un damier.

En résumé...

Le processus de simulation que allons mettre sur pied se compose donc de 3 phases principales:

Importation GIS	Nous initialisons notre carte avec les informations rasters générées par les outils GIS.
Simulation	Nous faisons tourner cette fameuse simulation qui doit simuler le comportement des skieurs et l'érosion neigeuse.
Restitution du résultat	Nous restituons les résultats de l'érosion neigeuse dans un format avalable par les outils GIS.

Echelonnement du travail

La planification initiale de ce travail proposait les étapes suivantes dans l'ordre :

- Prise en main de l'outil de simulation Repast.
- Implémentation d'une simulation de base simple.
- Interfaçage GIS.
- Travail sur l'amélioration progressive du comportement des agents.

Motivation personnelle

Ce travail de bachelor m'a été proposé personnellement par Mr. Schumacher et n'a pas fait partie de la liste proposée aux étudiants.

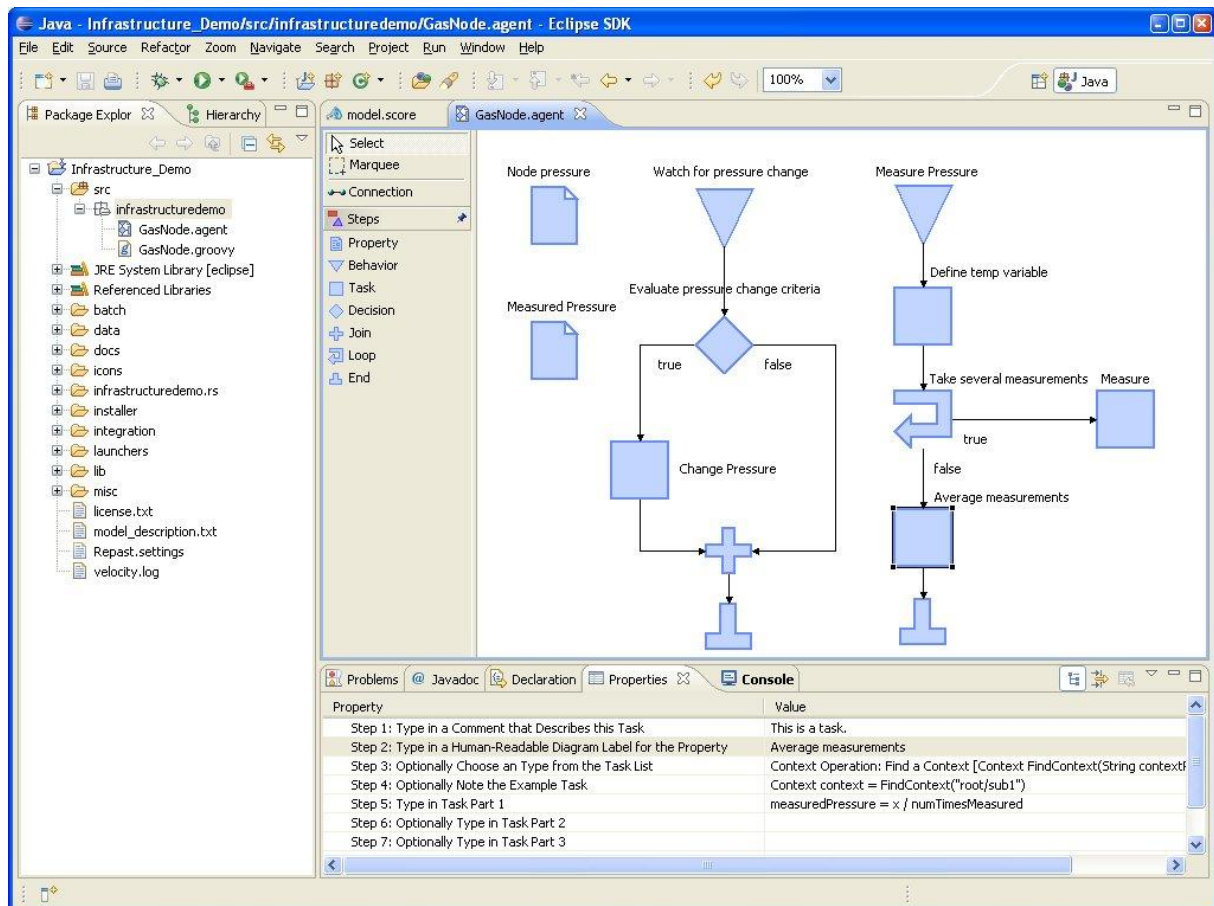
J'ai été immédiatement intéressé par ce travail car il s'inscrivait dans le cadre d'un projet de recherche concret et très sérieux. De plus, j'ai pressenti que ce travail constituerait un challenge de taille et je ne suis pas du genre à choisir les solutions de facilité.

Du point de vue motivation, j'ai laissé ma curiosité prendre le dessus. Je ne souhaitais pas faire un travail de développement similaire à ce que je fais habituellement dans ma vie professionnelle. C'était l'occasion de mettre de coté l'informatique de gestion pour s'essayer à quelque chose de différent et ainsi explorer de nouveaux horizons.

Les bases d'une simulation

Pour ce projet nous travaillons avec Repast simphony (<http://repast.sourceforge.net/>), cet outil a été proposé dès le départ par Mr Schumacher sur la base de ses expériences précédentes sur d'autres projets.

Repast se présente sous la forme d'un plugin pour le célèbre IDE eclipse (v3.4 ganymede) et transforme celui-ci en un environnement de développement pour simulation.



L'éditeur visuel de flow-chart dans l'environnement eclipse transformé.

Il est également accompagné d'une grande quantité de librairies couvrant un large éventail de besoins en tout genre, algorithmie génétique, gestion de graphe, représentation spatiale. Etc...

Au centre de la simulation: une horloge

Dans une simulation, tout tourne autour d'une dimension temporelle symbolisée par une horloge. Il s'agit en fait d'un planificateur responsable du déclenchement des événements dans la simulation.

Cette horloge démarre à 0.0, puis lorsque la simulation est lancée, elle s'incrémente d'unité en unité, nous nous référerons dans ce document à cette unité en utilisant l'expression "tick d'horloge".

A chacun de ces ticks d'horloge, nous avons la possibilité des déclencher des actions (déplacer un agent, générer un phénomène etc...), lorsque toutes les actions planifiées ont fini de s'exécuter, l'horloge passe au tick suivant.

La partie active : les agents

Quelque soit son type, une simulation comprend au moins un agent. Au sens Repast, un agent est un simple objet java. A cela près qu'il contient une méthode planifiée.

Cette méthode planifiée est exécuté automatiquement par le planificateur au moment décidé par le programmeur. Une méthode planifiée peut être exécutée une seule fois à un tick précis, ou alors répétée tous les X ticks. Un exemple d'agent sera notre skieur qui possèdera entre autre une méthode planifiée qui le fera se déplacer.

L'environnement : le terrain

Aussi appelé projections, les environnement représentent le terrain, l'espace dans lequel évoluent les agents. Sans environnement, les agents ne forment qu'une sorte de "soupe" sans organisation spatiale.

Ces environnements peuvent être

- des **graphes**, par exemple dans le cas d'un réseau de conduites forcées ou d'un tableau électrique.
- Des **grilles**, dans le cas d'un espace rectangulaire en 2D non continu tel qu'une carte raster, faite de coordonnées entières.
- Un **plan continu**, dans le cas d'un terrain composé de points, de lignes et de polygones dans lequel tout nombre réel représente une coordonnée différente.

Premier exemple sous Repast

Chapitre 2

Projet de simulation multi-agent

Stéphane Claret

Premier exemple

Après une première phase de prise en main de Repast, j'ai très vite ressenti le besoin de réaliser un exemple didactique afin de pouvoir me familiariser avec les possibilités de l'outil. Cette étape était nécessaire pour pouvoir prendre part aux discussions stratégiques en connaissance de cause.

J'ai pris la décision de le présenter dans ce rapport, d'une part parce qu'il s'agit de la première étape concrète que j'ai franchie dans le cadre de ce travail de bachelor mais aussi pour pouvoir introduire certains concepts par un exemple pratique plutôt que par la théorie.

Description

Ce scénario fait évoluer des agents skieur sur une piste symbolisée par une grille de 50x50 cases. A chaque fois que l'horloge émet un tick, des agents au nombre de 30 vont se déplacer vers l'avant et simuler des dégâts sur la couche neigeuse.

La simulation tourne jusqu'à ce qu'elle soit arrêtée, on peut constater l'évolution du manteau neigeux en regardant certaines portions très fréquentées du terrain virer peu à peu vers le rouge foncé.

Ci-dessous un récapitulatif du contenu de cette simulation.

Environnement :

Type de terrain	Grille 2D multi-occupants 50x50.
Couches	Une couche représentant la santé de la neige 50x50. Pour affichage des nuances de rouge seulement.

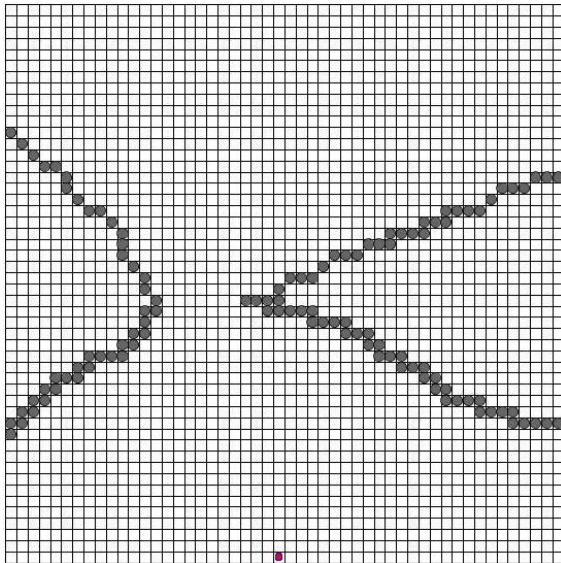
Agents :

Type	Actions	Fréquence d'exécution
Skieur	Se déplace sur le terrain.	A chaque tick d'horloge.
Rocher	Bloque le passage du skieur. Sert de bordure.	-
Neige	Occupe chaque case de la grille.	-

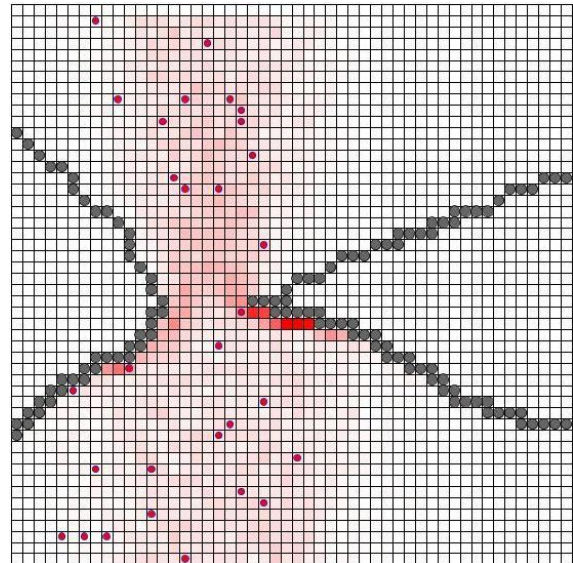
Screenshots

Voici quelques screenshots de la simulation, pris à différents stades. Les ronds rouges représentent les skieurs. La bordure grise délimite une zone non praticable.

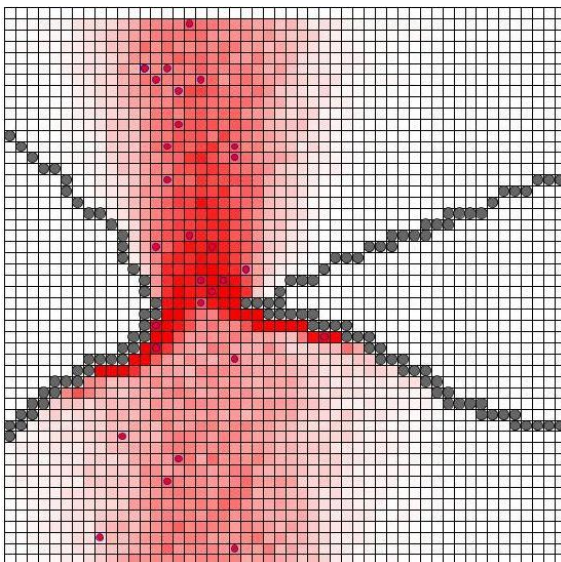
Il est important de préciser que tout cet exemple est basé sur des données fictives.



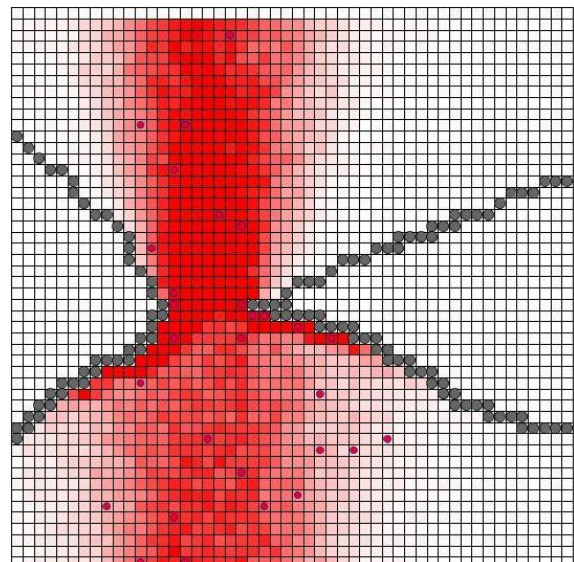
*Aperçu de l'état des choses avant le lancement de la simulation.
En bas au centre se trouvent les agents.*



Etat après 706 ticks d'horloge, ce qui équivaut à 706 déplacements pour chacun des agents.



Après 2950 ticks d'horloge.



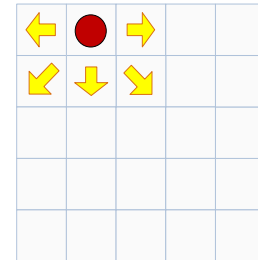
Après 5018 ticks d'horloge, le terrain est en très mauvaise santé.

Comportement des agents

Chaque agent skieur doit se déplacer d'une case à une autre sur la grille d'une manière comparable à celle d'un pion sur un plateau d'échec. Son comportement se résume donc à l'ensemble des critères que nous allons considérer lorsque nous devrons choisir leur destination respective. Pour cet exemple, nous avons choisi de déplacer nos agents aléatoirement vers l'avant afin de simplifier au maximum.

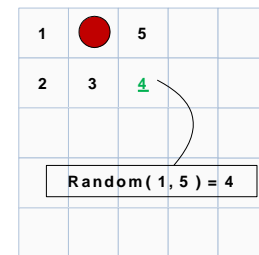
Etape 1 : Repérage des destinations possibles.

L'agent commence par considérer les cases adjacentes à sa position actuelle comme destinations possibles, il élimine éventuellement les destinations hors des limites de la grille ainsi que celles qui sont considérées comme non-utilisables (hors piste).



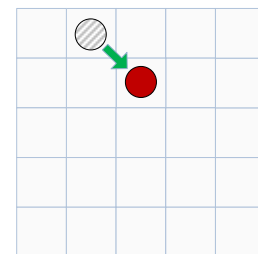
Etape 2 : Choix de la destination.

On désigne aléatoirement une destination parmi celles que nous avons retenues lors de l'étape 1, pour cet exemple nous ne tenons compte d'aucun facteur. Toutes les destinations ont donc une probabilité équivalente.

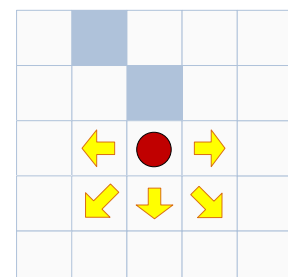
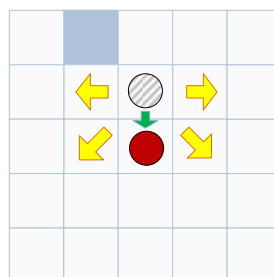
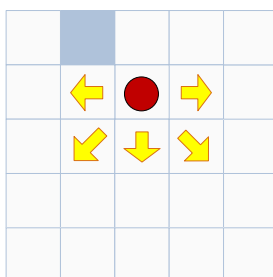


Etape 3 : Déplacement de l'agent.

L'agent est déplacé vers la destination choisie lors de l'étape 2, en réalité nous ne faisons que mettre à jour ses coordonnées dans la grille.



Ces trois étapes sont planifiées pour s'exécuter à chaque tick de l'horloge de la simulation, ce qui a pour effet qu'elles se répètent indéfiniment jusqu'à ce que la simulation soit stoppée. Ainsi au tick suivant nous retrouvons le même processus...



Et ainsi de suite...


Erosion de la couche neigeuse

Il s'agit là de l'un des point-clefs de ce projet, simuler la dégradation progressive du manteau neigeux sur les lieux de passage des skieurs.

Lors de l'initialisation de la simulation, un agent neige est créé et installé dans chaque case de la grille. Contrairement aux agents skieurs, Cet agent occupe la grille de façon passive mais ne dispose d'aucun comportement planifié.

Il possède une propriété état de type double initialisée à 1000. Cette variable d'état correspond à la santé de la neige dans sa case respective. A chaque fois qu'un agent skieur pénètre dans une case, nous allons diminuer cette valeur. Ainsi, plus le nombre de passage dans une zone sera important, plus la neige y sera dégradée.

Dans cette exemple, chaque passage enlève entre 0 et 10 points de santé à la neige, de façon aléatoire.

1K	994	1K	1K	1K
1K	998	990	1K	1K
1K	1K	1K	993	1K
1K	1K	1K		1K
1K	1K	1K	1K	1K

Les agents diminuent la santé de la neige sur leur passage.

Représentation graphique

Nous avons à présent un phénomène de dégradation de la neige, il est intéressant de pouvoir visualiser la chose pendant le déroulement de la simulation.

Repast offre cette possibilité en standard dans son runtime, ceci passe par l'utilisation d'une couche de valeurs (value layer). Une couche de valeur est une simple matrice de variables de type double qui associe à chaque coordonnée XY de notre grille une et une seule valeur.

Nous avons ainsi besoin de créer une matrice de santé de la neige, ce qui en d'autres termes correspond à tableau 2D (double[][]) de dimension égale à celle de notre grille comprenant des valeurs comprises entre 0 et 1000. Nous devons alors tout au long de la simulation nous débrouiller pour que les valeurs de nos agents neige et celles de cette couche valeur soient synchronisées. En vérité ce n'est pas une si grosse contrainte que ça car il nous suffit simplement de mettre à jour les deux choses simultanément dans le setter de l'agent neige.

A présent que nous avons notre couche de santé de la neige fonctionnelle, il nous suffit d'indiquer à repast comment il doit la représenter. Cela se fait en implémentant une petite méthode d'interface qui reçoit en paramètre les coordonnées d'une case et retourne la couleur de fond à utiliser.

Pour ce cas, nous voulons que la couleur de fond commence au blanc et devienne de plus en plus rouge au fur et à mesure que la neige se dégrade. Cela se fait par une simple règle de trois. Si nous posons les données connue nous avons :

santé à 1000 = blanc = `RVB(255, 255, 255)`

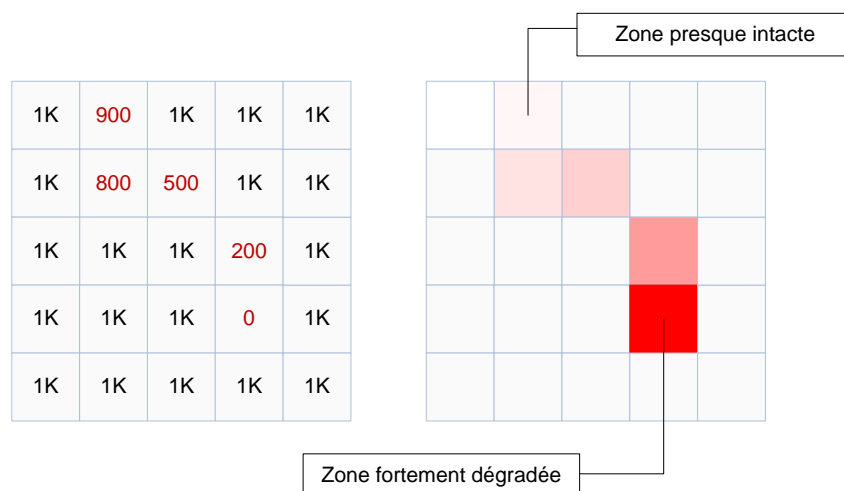
santé à 0 = rouge = `RVB(255, 0, 0)`

Il nous suffit de diminuer la teneur en vert et en bleu progressivement en la calculant comme suit:

$\text{teneurVb} = \text{etatActuel} / 1000.0 * 255.0;$

Nous retournons alors la couleur suivante :

`RVB(255, teneurVb, teneurVb);`

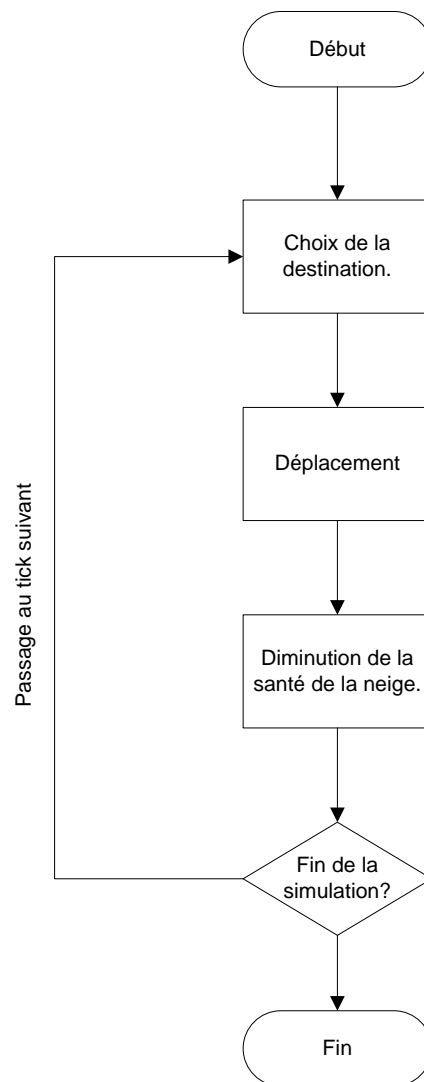


La couche de valeur à gauche est représentée par la figure de droite, les zones abîmées apparaissent clairement.

Résumé du flux

Le mouvement des agents skieurs est la seule action planifiée de cette simulation et se répète indéfiniment. Tout ce qui arrive ensuite (interaction avec la neige, etc...) n'est que la conséquence du déplacement.

Le diagramme de flux suivant représente le déroulement des actions depuis le point de vue de l'agent skieur :



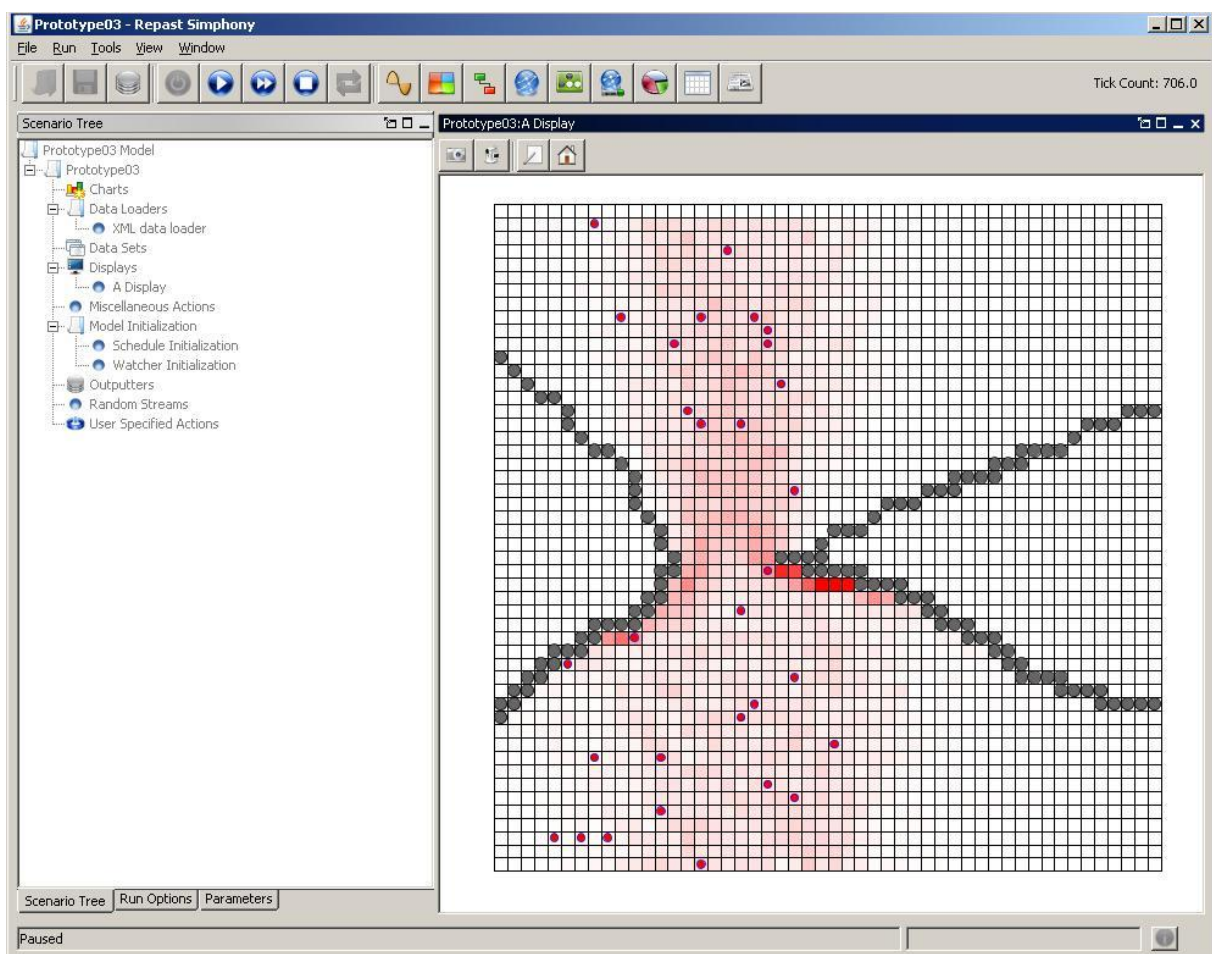
Conclusion

Cet exemple a été ma première réalisation concrète sous Repast, il m'a permis d'acquérir quelques principes sur le fonctionnement d'une simulation, suffisamment pour que je sois en mesure de me rendre compte de ce qui est faisable ou non afin de pouvoir effectuer des choix stratégiques pour la suite.

Le comportement des agents skieurs de ce scénario est clairement trop pauvre pour refléter la réalité, ils ne font même pas l'effort d'éviter les zones dans lesquelles la neige est abîmée et surtout, ils ne se dirigent que vers l'avant ou latéralement.

La grille utilisée n'a d'ailleurs de loin pas la complexité d'une véritable carte qui pourrait comprendre des trajectoires coudées, le besoin de se diriger tantôt au nord, au sud, à l'est etc...

Nous avons utilisé une vidéo de cet exemple pour faire une démonstration à d'autres intervenants du projet.



Exécution de la simulation au sein du runtime repast.

Choix d'architecture

Chapitre 3

Projet de simulation multi-agent

Stéphane Claret

Le problème du routage

Nous avons vu au cours du chapitre précédent une stratégie de déplacement simple pour nos skieurs. Ils se déplacent dans une direction prédéfinie sans jamais se soucier d'autre chose que des 5 cases devant eux.

Notre simulation finale doit s'appliquer à l'entier d'un domaine skiable, il est tout à fait possible que l'orientation du terrain change d'une piste à l'autre. Ainsi au sein d'une même station, il se peut que nous soyons amenés à déplacer nos agents tantôt du nord au sud, tantôt d'est en ouest. Ou l'inverse.

Le GIS nous fournit des informations rasters permettant d'initialiser chaque case de notre grille avec les attributs suivant :

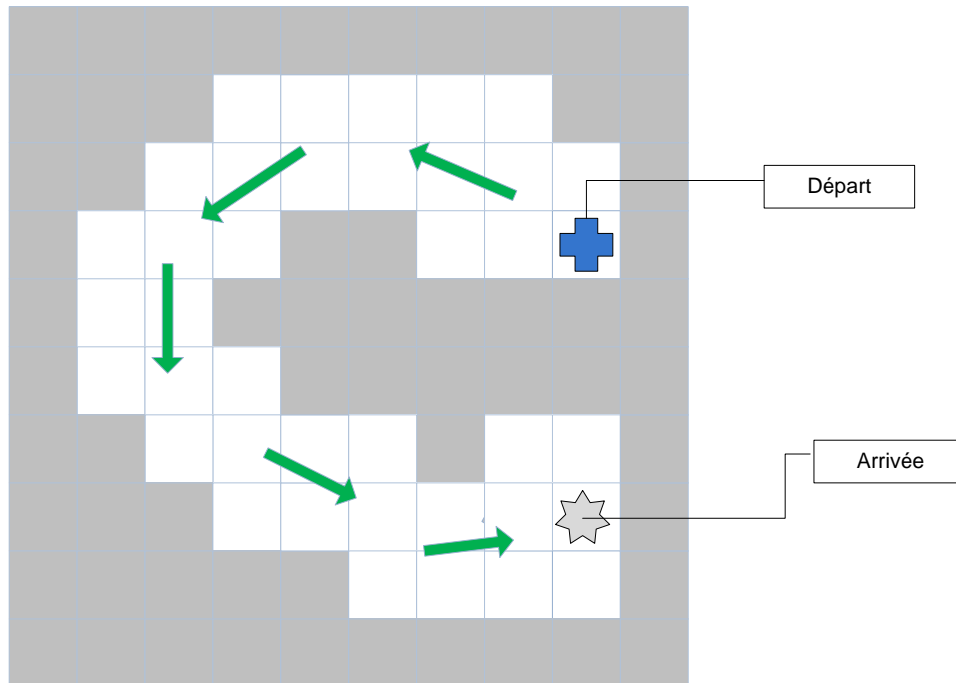
Attributs	Description
X	Coordonnée X réelle de la case
Y	Coordonnée Y réelle de la case
Z	Altitude de la case
Hors piste	Indique si la case est utilisable par un agent ou non.
Pente	Pente en % le long de la case
Arrivée de remontée mécanique	Nous indique si la case contient une arrivée de remontée mécanique.
Départ de remontée mécanique	Nous indique si la case contient un départ de remontée mécanique.

A partir de cela nous créons une grille représentant tout le domaine skiable. Grâce aux informations sur les arrivées et départs de remontées mécaniques, nous savons également quelles sont les points de création d'agent et quelles sont les points d'arrivée.

Nous avons beau savoir les coordonnées des points d'arrivées, cela ne nous indique aucunement la direction à faire suivre à nos agents pour les amener à bon port.

De plus, nos agents skieurs occupent une coordonnées sur une grille mais ils ne sont pas du tout conscients de leur environnement. Il n'ont absolument aucune information sur la géométrie de la zone dans laquelle ils se trouvent.

Si nous considérons l'illustration ci-dessous d'un coude.



La solution montrée par les flèches vertes nous semble tout de suite évidente à nous, êtres humains, car nous avons immédiatement une vision d'ensemble de cette grille. D'un point de vue machine, c'est très complexe car il est très difficile de mettre côte à côte les attributs de chaque case pour se rendre compte qu'il s'agit là d'un coude et que des changements de direction sont nécessaires.

Utiliser une formule basée sur une relation entre les coordonnées du point de départ et du point d'arrivée telle qu'une droite pour déterminer un itinéraire ne nous mènera pas à grand chose.

Par ailleurs le comportement de l'agent doit rester le plus local possible, si l'agent se met à considérer un grand nombre de cases autour de lui pour déterminer un itinéraire, les performances vont diminuer de façon exponentielle. Il est intéressant à ce sujet de noter que notre objectif est tout de même de simuler plusieurs milliers d'agents sur des grilles qui peuvent atteindre des centaines de milliers de cases.

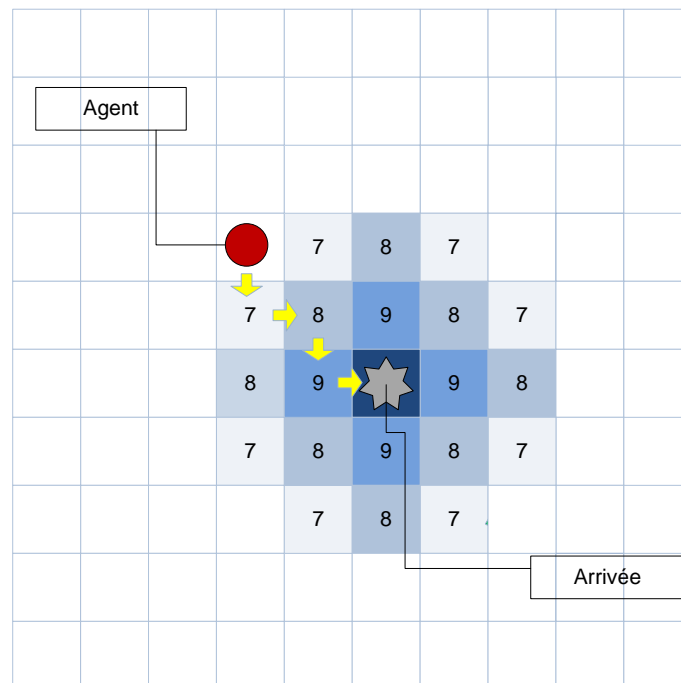
Nous touchons à présent au problème le plus complexe de ce travail de diplôme.

Une première idée de solution

Comment pouvons-nous diriger notre agent d'un point d'arrivée à un point de départ sans être obligé de précalculer un itinéraire?

Une solution que nous avons considérée consiste en la mise en place d'une **diffusion** de valeurs. Le principe en lui-même est assez simple, il s'agit de localiser un point d'arrivée puis de ventiler des valeurs dans les cases adjacentes de façon dégressive.

Lorsque le skieur se déplace sur la grille, il peut considérer ces valeurs dans les cases proches de lui et s'arranger pour se déplacer toujours là où cette valeur augmente, comme attiré par un aimant.



L'agent est attiré par les valeurs que nous diffusons autour du point d'arrivée.

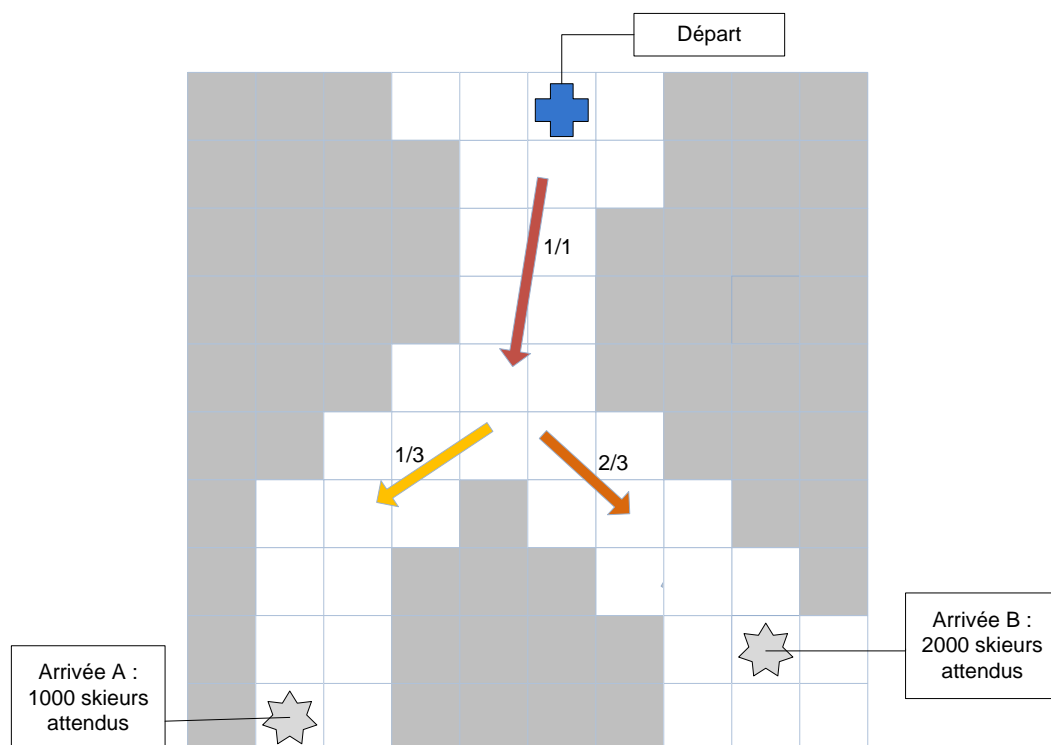
Grâce à cette solution, nous résolvons certains problèmes de routage locaux. Cependant, un problème que nous avons laissé de côté pour plus tard a prématurément surgi : Celui des points de décision.

Les points de décisions

Dans un domaine skiable, il est courant de rencontrer des points de choix, c'est à dire des intersections de piste au sein desquels le skieur réel doit choisir son itinéraire. Son itinéraire détermine également quelle remontée mécanique il utilisera.

Nous ne pouvons pas laisser ceci au hasard dans notre simulation, nous ne devons pas laisser la majorité de nos agents skieurs choisir aléatoirement entre un chemin qui les mène vers un petit télésiège abandonné et un autre qui aboutit sur un grand télésiège 6 places ultra-fréquenté.

Afin de pouvoir résoudre ce problème, nous disposons de données sur les débits de chaque remontée mécanique en entrée. Nous devons à présent diriger nos skieurs vers les bonnes remontées mécaniques. Ce qui s'ajoute en sus des problèmes précédents.

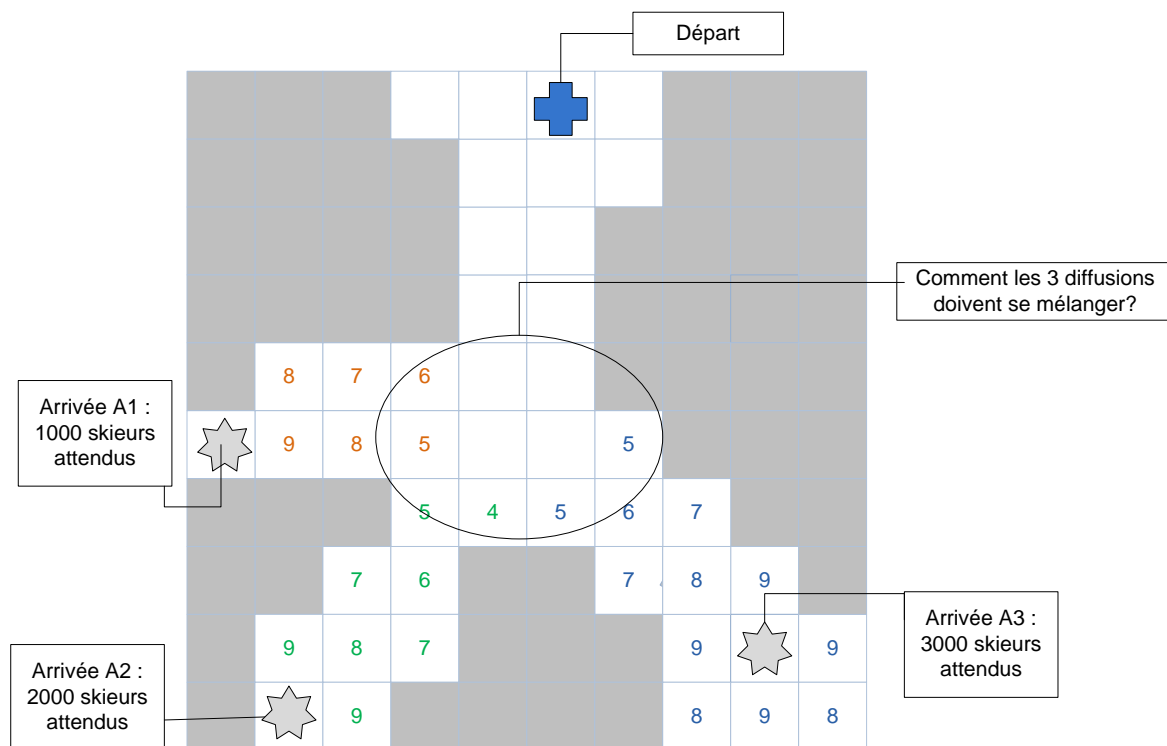


Nous devons répartir nos skieurs sur les itinéraires possibles de façon à respecter la fréquentation des remontées mécaniques. Les flèches représentent le flux de skieurs qui est amené à se diviser à l'intersection proportionnellement à la demande de chacun des 2 chemins.

L'abandon de la diffusion

Malheureusement, je n'ai pas été capable de trouver une solution permettant de résoudre le problème de ces points de décision à l'aide d'une diffusion.

Nous avons réfléchi longuement à la manière dont nous devons organiser nos diffusions pour résoudre ces problèmes d'intersection. Nous devons trouver un algo qui s'applique à toutes les situations possibles, tous les domaines skiables, dans une géométrie de piste qui n'est pas connue à l'avance et qui serait ultra-complexe à découvrir par rétro-ingénierie.



Quelles règles appliquer pour gérer une telle intersection?

L'illustration ne met en évidence qu'une partie du problème, il est possible aussi que l'influence des points d'arrivée A1, A2, A3 s'étende plus loin que l'intersection, par exemple s'il existe d'autres intersections plus haut sur la piste.

Je suis personnellement convaincu qu'un tel problème est insolvable avec le temps dont nous disposons. De plus, vu la quasi-infinité de situations différentes que nous devons être en mesure de gérer, valider un algorithme avec certitude serait très difficile, sinon impossible.

La solution choisie

Au vu de l'ampleur des problèmes rencontrés, j'ai fait la proposition d'une autre approche qui a mis toute l'équipe d'accord.

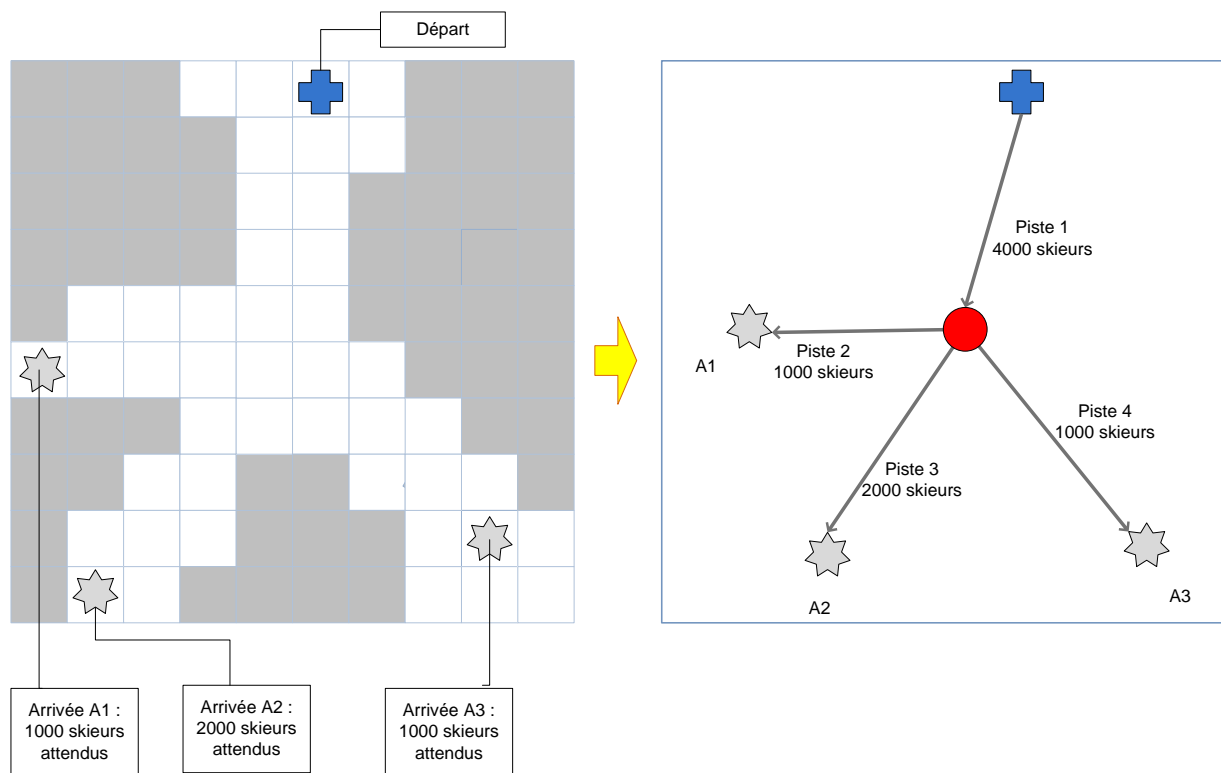
Jusqu'ici nous cherchions à gérer une simulation en grille directement sur l'ensemble du domaine skiable et nous nous heurtons aux problèmes de décisions et d'intersections évoqués précédemment.

La nouvelle approche propose de résoudre d'abord les problèmes de décisions d'itinéraire des agents dans le domaine skiable au moyen d'un graphe des pistes dirigés, puis d'exécuter une simulation en grille individuelle pour chaque piste.

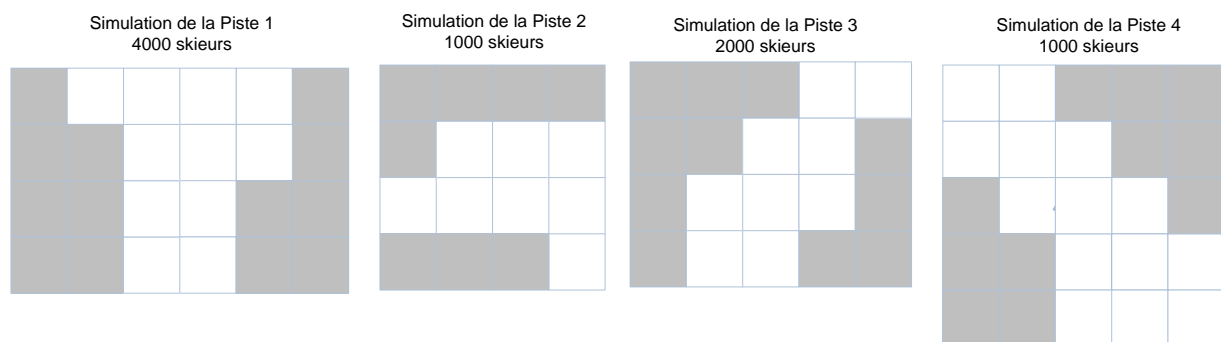
Cette idée comporte donc 2 étapes :

Etape 1	<p>L'objectif de cette étape est de savoir combien de skieurs emprunteront telle ou telle piste du domaine skiable.</p> <p>C'est une simulation à part entière qui travaille sur un graphe dirigé dans lequel chaque arc équivaut à une piste. Avec un tel graphe nous sommes capables de résoudre beaucoup plus aisément les problèmes de routage et d'intersection à l'échelle du domaine.</p> <p>L'output de cette première simulation sera pour chaque piste le nombre de skieurs l'ayant empruntée.</p>
Etape 2	<p>En utilisant les résultats de l'Etape 1, nous lançons une simulation en grille individuelle pour chaque piste dans laquelle nous simulerons respectivement le nombre de skieurs l'ayant empruntée.</p>

Cette solution nécessite la création d'un graphe des pistes par dessus la carte du domaine skiable, c'est une opération manuelle faite par l'expert GIS. Pour notre simulation, nous pourrions profiter de ces données directement sans avoir besoin de calculer nous-mêmes le graphe.



Etape 1 : Nous calculons l'occupation de chaque piste à l'aide d'un graphe.



Etape 2 : Nous faisons tourner des simulations locales sur chacune des pistes avec les résultats obtenus lors de l'étape 1.

Conclusion

Je pense que la complexité des problèmes évoqués durant ce chapitre nous a tous un peu surpris et a quelque peu dépassé nos attentes.

Mon travail de diplôme a désormais pris une tournure très différente de ce que nous avions prévu au départ. La solution retenue a remis en cause nos plans et nous a forcé à reconsidérer nos objectifs.

Dorénavant, notre priorité sera de démontrer la faisabilité de cette solution en mettant en place une architecture utilisable. Cette remise en condition apparaît malheureusement un peu tardivement dans le travail de diplôme, mais nous avons pris la décision de faire le maximum pour parvenir à mettre cette structure en place.

Comme nous le verrons, certains problèmes de finition sur le comportement des agents seront ignorés. Ce sont des aspects certes d'importance mais qui passeront en priorité 2. Nous aurons la possibilité de les travailler, mais ceci est inutile tant que nous n'avons pas la certitude que notre architecture en deux étapes est bel et bien réalisable et répond aux attentes.

Le graphe du domaine skiable

Chapitre 4

Projet de simulation multi-agent

Stéphane Claret

Introduction

Cette simulation en graphe est le point névralgique de notre nouvelle architecture, elle doit nous permettre de modéliser tout le réseau de piste d'un domaine skiable sous forme d'une succession d'arcs représentant tous les itinéraires possibles d'un point d'arrivée de remontée mécanique jusqu'à un point de départ.

Grâce à ces données, nous devons être en mesure de simuler un nombre donné de skieurs parcourant les pistes et de ressortir les statistiques individuelle d'utilisation de chaque piste.

Pour effectuer ce travail, nous disposons des données suivantes sur les arcs et les sommets :

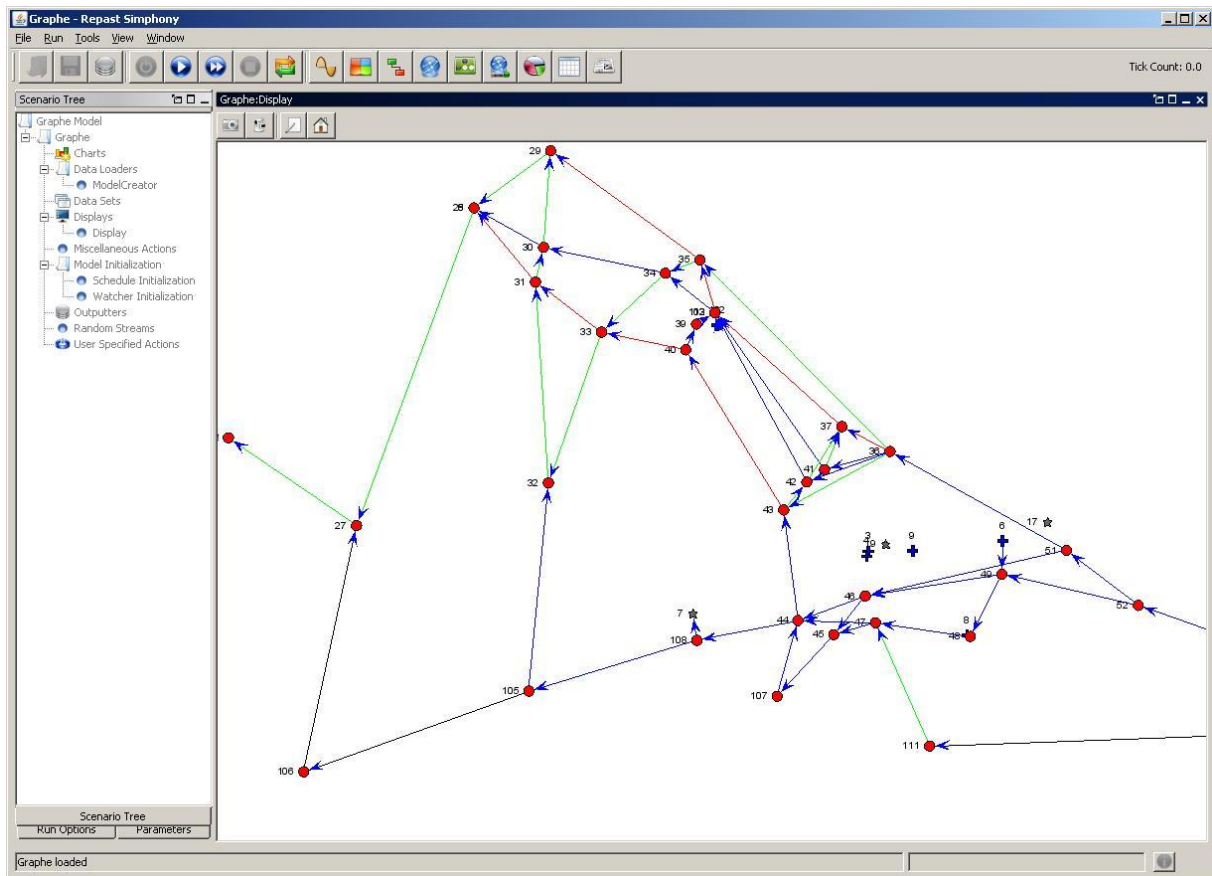
Sommets :

Identifiant	L'identifiant numérique unique du noeud.
Coordonnées X et Y	Les coordonnées des sommets sur la carte. Ils nous serviront uniquement pour l'affichage graphique.
Type de sommet	Trois valeurs sont possibles : <ul style="list-style-type: none">• Départ de remontée mécanique• Arrivée de remontée mécanique• Simple point de passage ou intersection de piste

Arcs :

Identifiant	L'identifiant unique de l'arc.
ID des sommets source et destination	Les sommets reliés par l'arc.
Type d'arc	Deux valeurs sont possibles : <ul style="list-style-type: none">• Arcs de piste, parcourable par le skieur• Arc de remontée mécanique, relie le point de départ et le point d'arrivée d'un télésiège ou télési. Seuls les Arcs de piste nous intéressent pour l'instant.
Longueur	La longueur en mètres de l'arc.
Difficulté	La couleur exprimant la difficulté de la piste. Vert représente une piste pour débutant, Noir une piste potentiellement dangereuse pour skieur averti.

En utilisant la puissante librairie Jung (<http://jung.sourceforge.net/>) incluse dans Repast, nous sommes déjà en mesure de construire le graphe complet.



Une visualisation du graphe ainsi construit dans Repast, les couleurs des arcs représentent la difficulté des pistes.

Description générale

Maintenant que nous disposons d'un graphe, nous allons devoir créer des agents skieurs sur les points de départ des pistes.

Une fois créé, un skieur devra se déplacer dans le graphe, d'arc en arc (ce qui revient à dire de piste en piste) jusqu'à ce qu'il atteigne un point d'arrivée.

Voici un récapitulatif des éléments qui composent notre scénario :

Environnement :

Type de terrain	-Graphe (network) construit à partir des informations fournies sur le domaine skiable. Un plan continu (continuous space) est utilisé en arrière-plan pour tenir compte des coordonnées des sommets et permettre un affichage sympathique.
-----------------	--

Agents :

Type	Actions	Fréquence d'exécution
Sommet simple	-	-
Sommet départ RM (arrivée de piste)	-	-
Sommet arrivée RM (départ de piste)	Crée un agent skieur.	A chaque tick d'horloge
Skieur débutant	Emprunte un arc.	A chaque tick d'horloge.
Skieur moyen	Emprunte un arc.	A chaque tick d'horloge.
Skieur d'élite	Emprunte un arc.	A chaque tick d'horloge.
Outputter	Teste si tous les skieurs sont arrivés au fond et stoppe la simulation si tel est le cas.	Tous les 100 ticks d'horloge

Comportement des agents

Tous les sommets qui sont un point de départ de piste fabriquent (littéralement) un skieur à chaque tick d'horloge. L'agent ainsi fabriqué peut être un débutant, un moyen ou un élite, les chances de chaque niveau sont d'1/3 pour commencer mais le code a été prévu pour permettre facilement l'implémentation d'autres formes de distribution.

Une fois le skieur créé, il va devoir se déplacer d'un sommet à l'autre en utilisant les arcs jusqu'à un point d'arrivée. Une fois le point d'arrivée atteint, l'agent est détruit, c'est à dire supprimé de la simulation.

Chaque fois que le skieur emprunte un arc, nous incrémentons le compteur de passage de cet arc, ceci constitue nos données de sortie.

Tout le problème à présent est de déterminer selon quelles règles les skieurs vont se déplacer, Nous avons pour l'instant implémenté deux stratégies de comportement différentes.

Chemin prédéfini	-Le skieur décide du chemin qu'il va utiliser dès sa création, puis il se contente de le suivre.
Décision à la volée	-Le skieur décide de sa prochaine piste à la volée, et effectue des choix au fur et à mesure qu'il progresse dans le graphe.

Chemin prédéfini

Dans ce comportement, l'agent créé va décider de son itinéraire dès sa création. Cette stratégie comporte 3 étapes :

1. L'agent choisit aléatoirement un point d'arrivée parmi ceux atteignables depuis son point de départ.
2. L'agent choisit un chemin entre son point de départ et le point d'arrivée choisi.
3. Le chemin choisi est transformé en une collection d'arcs (un itérateur pour être plus précis) que le skieur va suivre au fur et à mesure jusqu'à son point d'arrivée.

Choix du chemin

Les étapes 1 et 2 nécessitent que depuis un sommet de départ donné, on sache exactement quels points d'arrivées sont atteignables ainsi que tous les chemins différents capables de nous y emmener.

Ces informations ont du être précalculées avant le démarrage de la simulation au moyen d'un algorithme d'exploration.

L'agent choisit aléatoirement son point de destination, mais il examine la difficulté des chemins qui s'offrent à lui pour atteindre ce point. Son choix se base sur la piste la plus difficile contenue dans chaque chemin.

S'il est débutant, sa probabilité de choisir un chemin ne comprenant que des pistes vertes est élevée, en revanche celle de s'attaquer à un chemin comprenant une piste noire est très faible.

Pour pouvoir modéliser ce phénomène, l'agent possède des facteurs de probabilité adaptés à son niveau. Voici ci-dessous un tableau illustrant les facteurs utilisés actuellement, notez qu'ils sont éditables hors programme afin de permettre de tester facilement différents jeux de valeurs.

Niveau	Vert	Bleu	Rouge	Noir
Débutant	1.5	1.0	0.9	0.3
Intermédiaire	1.0	1.1	1.2	0.8
Elite	0.7	1.0	1.3	1.0

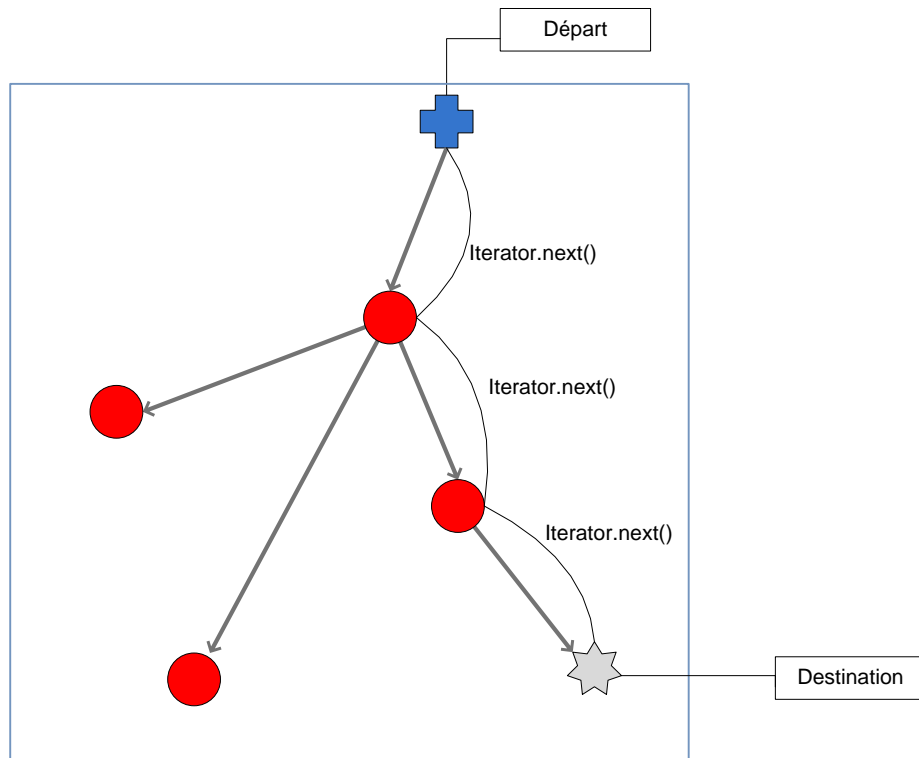
Ces nombres sont utilisées de la façon suivante: A l'aide d'une fonction `random()` nous générons pour chaque chemin disponible un nombre entre 0 et 1. Ce nombre est ensuite multiplié par le facteur de probabilité correspondant à la difficulté du chemin associé, le chemin qui fait le meilleur score l'emporte sur les autres.

Pour prendre un exemple, si un débutant doit choisir entre un chemin de piste verte ou un chemin de piste bleu, il effectuera le calcul suivant :

```
ScoreCheminVert = Random() * 1.5;  
ScoreCheminBleu = Random() * 1.0;
```

```
gagnant = Max( ScoreCheminVert, ScoreCheminBleu);
```

Ensuite, une fois son chemin décidé, le skieur se contente de le suivre sans se poser de question jusqu'à ce qu'il atteigne son point d'arrivée, où il sera détruit.



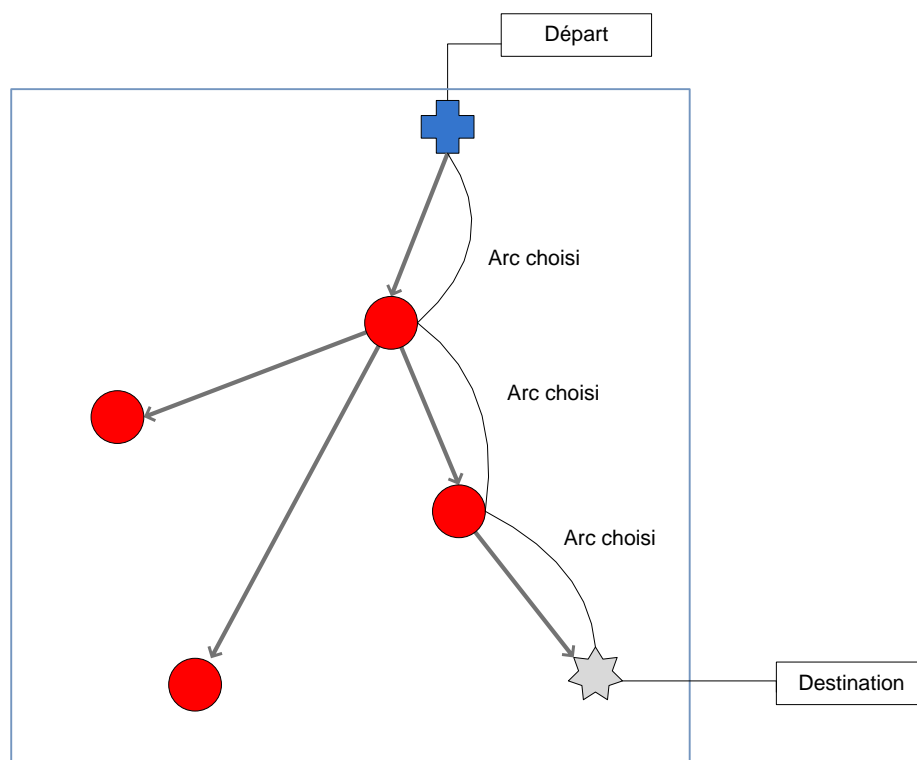
L'agent skieur suit son chemin choisi sous la forme d'un itérateur d'arc jusqu'à ce qu'il arrive à destination.

Le routage par chemin prédéterminé est fonctionnel mais possède le désavantage de nous forcer à pré-calculer toutes les possibilités de chemins et de destinations pour chaque sommet de départ.

Décision à la volée

Dans cette stratégie de comportement, l'agent ne planifie pas son chemin complet lors de sa création. Il choisit sa destination au fur et à mesure qu'il progresse dans le graphe, en ne considérant que ses possibilités immédiates.

Le système de probabilité basé sur la difficulté décrit précédemment est utilisé de la même manière, chaque arc reçoit un score, l'agent détermine le meilleur arc puis l'emprunte. Arrivé à destination, il répète le processus jusqu'à tomber sur un point d'arrivée, ce qui provoquera sa destruction.



L'agent skieur décide de chaque étape à la volée.

Fin de la simulation

La fin de cette simulation doit intervenir lorsque tous les agents skieurs sont arrivés à destination. Un compteur est incrémenté chaque fois qu'un agent est détruit, ce compteur doit être surveillé périodiquement et lorsqu'il devient égal au nombre d'agent que nous souhaitons simuler, le scénario doit prendre fin.

Pour faire cela, nous avons ajouté à la simulation un agent spécial appelé "outputter" qui vérifie ce compteur tous les 100 ticks d'horloge seulement pour ne pas gêner les performances.

Si les conditions de fins sont remplies, cet agent va ordonner l'arrêt de la simulation et aussitôt écrire les résultats de fréquentation de chaque arc dans une base de donnée Derby.

Conclusion

Cette simulation en graphe constitue la première pierre de notre architecture, elle n'est pas encore pleinement fonctionnelle car il lui manque les données de débits des remontées mécaniques.

A l'aide de ces données, nous devons influencer le choix des pistes effectué par nos agents, Mr. Schumacher pense utiliser un réseau bayésien pour propager des probabilités à travers les différents noeuds du graphe.

Cependant je ne peux pas couvrir ce sujet car mes connaissances ne sont pas actuellement suffisantes. Toutefois le design pattern strategy a été utilisé de façon exhaustive pour permettre l'implémentation de nouveaux comportements sans trop toucher à la structure globale.

Simulation des grilles locales

Chapitre 5

Projet de simulation multi-agent

Stéphane Claret

Introduction

Maintenant que nous disposons des statistiques d'utilisation des pistes, nous devons simuler localement l'érosion du manteau neigeux sur chaque piste. Pour ce faire, nous lancerons une simulation en grille pour chaque piste et simulerons exactement la quantité d'agent l'ayant fréquentée.

Pour créer notre environnement, nous disposons en entrée des données rasters suivantes pour chaque maille de la piste :

Mailles de la pistes :

Identifiant d'arc	L'identifiant de l'arc correspondant à la piste sur laquelle la maille se situe.
Coordonnées X et Y	Les coordonnées de la maille.
Altitude (Z)	L'altitude normalisée de la maille.
Pente	La pente exprimée en pourcentage.

Pour rappel, ces mailles proviennent d'une analyse spatiale faite sur les cartes géographiques du domaine skiable par l'expert GIS. Chaque maille représente une zone de 5x5m.

Dans la simulation, chacune de ces mailles sera considérée comme une case de notre grille.

Nous ne simulons qu'une seule piste à la fois, c'est à dire qu'un seul arc.

Description générale

Nous allons travailler de façon similaire à notre exemple du chapitre 2, mais en y ajoutant un peu de complexité.

Voici un récapitulatif des éléments qui composent notre scénario :

Environnement :

Type de terrain	Grille selon taille de la piste, bordures strictes.
Couches	<ul style="list-style-type: none"> • Case hors-piste ou non (valeurs 1.0 ou 0.0) • Etat de la neige de la case (valeurs 1000.0 à 0.0) • Altitude de la case • Zone de sortie (valeur 1.0 ou 0.0)

Agents :

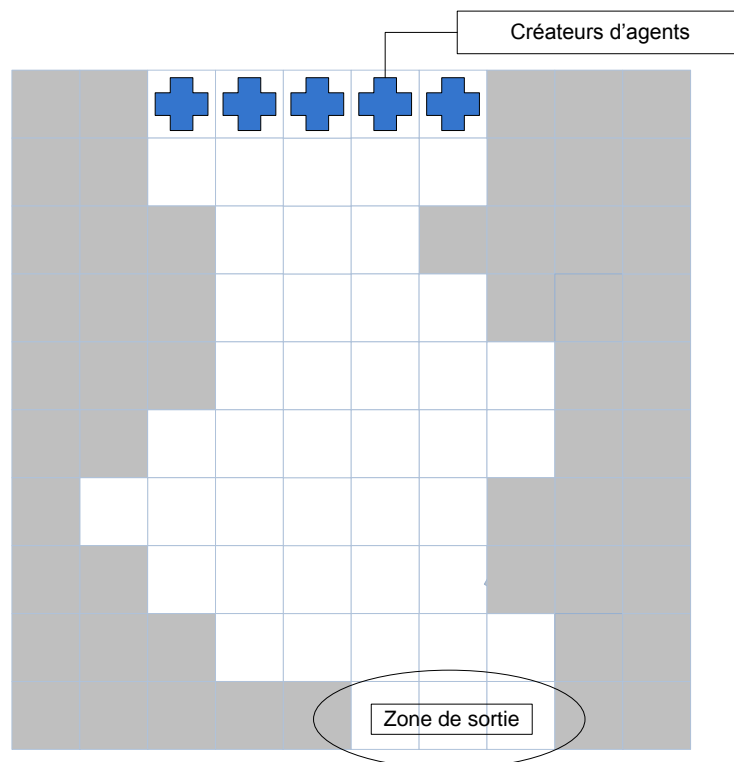
Type	Actions	Fréquence d'exécution
Skieur	Se déplace sur la grille	A chaque tick d'horloge
Créateur d'agent	Fabrique des skieurs	A chaque tick d'horloge
Outputter	Teste si tous les skieurs sont arrivés au fond et stoppe la simulation si tel est le cas.	Tous les 100 ticks d'horloge

Contrairement à l'exemple simple présenté au chapitre 2, nous avons renoncé à insérer dans toutes les cases un agent neige. Nous lisons directement l'état de la neige dans une couche valeur, cette solution est bien plus rapide car elle nous évite de devoir sans arrêt analyser tous les agents contenus dans une case pour repérer celui d'entre eux qui est de type "Neige" pour le mettre à jour. Tout se règle ainsi en une seule opération de lecture ou d'écriture.

Comportement des agents

Tout d'abord, nous devons repérer sur notre grille quels seront les points d'entrée et de sortie pour nos skieurs. Ceci est obtenu en parcourant les bordure de la grille à la recherche de deux zones praticables distinctes.

Nous comparons les altitudes des deux zones ainsi trouvées pour déterminer laquelle sera la zone de départ et laquelle sera la zone d'arrivée. La zone de départ est alors recouverte de créateurs d'agents.



Les zones d'entrée et de sortie s'obtiennent en analysant les bordures de la grille.

A chaque tick de l'horloge, les créateurs d'agent fabriquent un skieur. Les skieurs fabriqués se déplacent dans la grille et sont détruits lorsqu'ils tombent sur une zone de sortie.

Pour se déplacer, les agents analysent les 8 cases autour d'eux, ces 8 cases forment un ensemble appelé "voisinage de Moore". Les cases hors pistes sont immédiatement éliminées à l'aide d'un test sur notre couche valeur. Pour les cases restantes, la stratégie de déplacement utilisée exige qu'une note soit donnée à chacune d'entre elles.

La case ayant reçu la note la plus élevée est sélectionnée comme prochaine destination.



L'agent considère les cases possibles autour de lui, leur donne une note puis choisit la meilleure comme destination.

L'implémentation actuelle évalue les cases de la façon suivante :

- Les cases dont l'altitude est supérieure à celle de la position actuelle voient leur note forcée à Double.MinValue. Ceci empêche les agents de remonter la pente. Il s'agit bien sûr là d'une solution provisoire car il existe des pistes qui comportent des zones de prises de vitesse suivies d'une légère pente montante (on les appelle communément "combes" ou "schuss!").
- Pour les autres cases, c'est l'état de la neige qui influence les probabilités. La note donnée à une case correspond à l'état de la neige qui s'y trouve multiplié par un nombre aléatoire compris entre 0.0 et 1.0.

`NoteCase = EtatNeige * Random();`


Il s'agit là d'un comportement plutôt simple, mais le système d'évaluation des cases mis en place laisse la porte ouverte à des implémentations plus complexe dans le futur.

Erosion de la couche neigeuse

Comme dans notre exemple du chapitre 2, nous devons simuler la dégradation progressive du manteau neigeux sur les lieux de passage des skieurs.

La santé de la neige est initialisée uniformément à 1000 par case sur toute la surface. A chaque fois qu'un agent skieur pénètre dans une case, nous allons diminuer cette valeur. Ainsi, plus le nombre de passage dans une zone sera important, plus la neige y sera dégradée.

Dans l'implémentation actuelle, chaque passage enlève entre 0 et 10 points de santé à la neige, de façon aléatoire.

1K	994	1K	1K	1K
1K	998	990	1K	1K
1K	1K	1K	993	1K
1K	1K	1K		1K
1K	1K	1K	1K	1K

Les agents diminuent la santé de la neige sur leur passage.

Il faut bien admettre que cette valeur à 1000 ainsi que la diminution aléatoire est quelque chose de complètement fictif à ce stade. La version finale de cette simulation utilisera des données de hauteur de neige qui seront importées régulièrement, sans doute quotidiennement.

L'algorithme de "rabotage" de la couche neigeuse bénéficiera sans doute lui aussi d'une étude plus poussée. Il est possible qu'à l'avenir, le passage d'un skieur provoque par exemple la dispersion d'un certain pourcentage de neige sur les cases voisines.

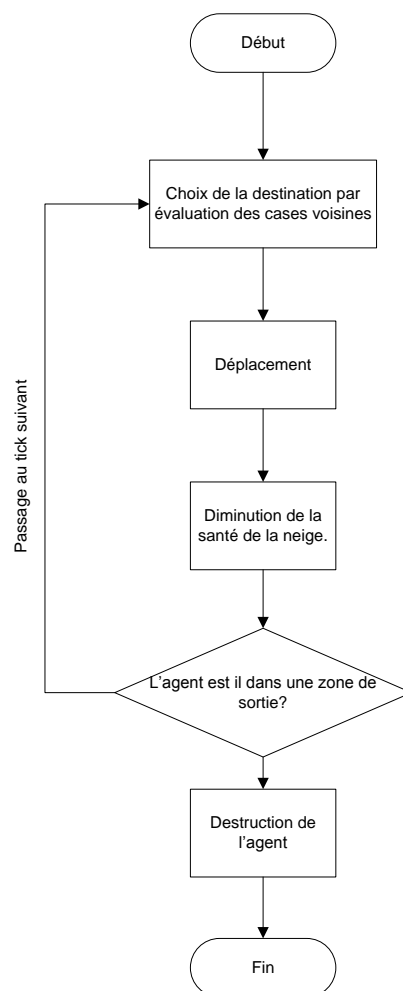
Arrêt et vue d'ensemble

A l'instar de notre simulation en graphe, l'arrêt de la simulation est confié à un agent unique qui teste tous les 100 ticks s'il reste des agents n'ayant pas encore atteint une zone de sortie.

Il se charge ensuite d'inscrire dans notre base de donnée les informations de résultat, elles se composent de l'identifiant de la piste, d'une paire de coordonnée XY ainsi que de l'état de la neige à cette coordonnée. Ceci pour toute case praticable de la grille. En voici l'aperçu.

Identifiant d'arc	L'identifiant numérique de l'arc représentant la piste simulée
Coordonnées X et Y	Coordonnées réelles de la case
Etat de la neige	L'état de la neige dans cette case à la fin de la simulation

Pour résumer cette simulation, le diagramme de flux ci-dessous représente le déroulement des actions depuis le point de vue de l'agent skieur :



Conclusion

Nous avons à présent mis en place une structure permettant de simuler les dégâts provoqués par les skieurs sur la couche neigeuse d'une piste individuelle.

La faible complexité des comportements implémentés dans cette simulation sont le résultat du très court délai dont j'ai disposé pour la réaliser (seulement quelques jours). On ajoute à cela que je ne disposais pas de données rasters réelles utilisables dans le format adapté et que j'ai du moi-même inventer des données d'entrées.

Heureusement, le know-how acquis lors de mon apprentissage de Repast et la réalisation du premier exemple expliqué au chapitre 2 m'ont permis de progresser suffisamment vite pour obtenir le minimum.

Même si cette simulation est loin d'en être à son stade définitif, elle est suffisante pour démontrer que nous sommes dans le juste avec notre approche de découpage en deux étapes.

Mise en commun de l'ensemble

Chapitre 6

Projet de simulation multi-agent

Stéphane Claret

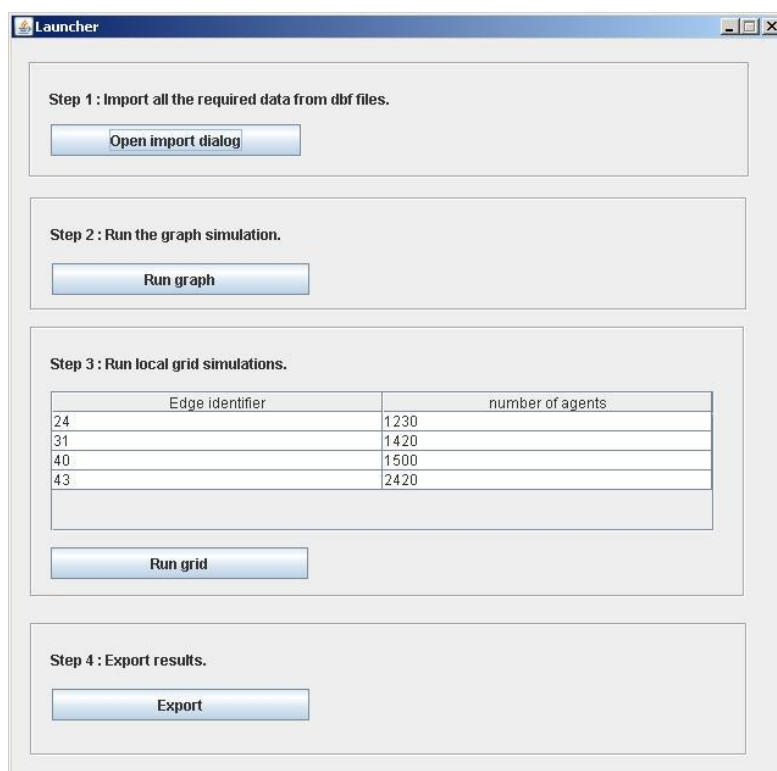
Besoin d'homogénéisation

Il était prévu au départ que nous gérons l'ensemble de la problématique du travail à l'aide d'une et une seule simulation exécutée sous Repast qui aurait assuré la totalité du processus.

Les problèmes rencontrés dans ce travail nous ont contraint à séparer notre travail de simulation en plusieurs étapes interdépendantes.

Etape 1	Importation des données en provenance du GIS nécessaires à la simulation.
Etape 2	Exécution de notre simulation en Graphe.
Etape 3	Exécution des n simulations en Grille en utilisant les résultats de l'étape 2.
Etape 4	Exportation du résultat.

Nous avons alors ressenti le besoin d'écrire une application java standard ayant pour but de fournir une interface utilisateur permettant le pilotage du processus.



L'application créée fournit une interface utilisateur pour diriger le processus

L'échange de données

Nos simulations sont basées sur des données générées par les outils de la suite ESRI Arc GIS, nous avons essayé sans succès de lire ces données directement dans leurs fichiers GIS.

Arc Gis étant un outil propriétaire, ses fichiers sont non-standards et ne laissent rien apparaître en clair. Il est théoriquement possible de lire ces fichiers directement en utilisant l'API propriétaire d'ESRI "ArcObjects", mais celle-ci est extrêmement vaste et confuse, d'autant plus que mes faibles connaissances de l'outil ne me permettaient pas de la comprendre convenablement.

Nous avons perdu passablement de temps dans ce projet à chercher une solution d'échange de données viable, mais finalement nous en sommes parvenus à la conclusion que nous devions passer par des exportations / importations.

Restait alors la question du format à utiliser pour les échanges. Personnellement j'étais en faveur de l'utilisation d'une base de données telles que PostgreSQL, gratuite et multi-plateforme. Cependant, les experts de l'IGAR, consultés à ce sujet, ont manifesté une préférence pour le format Dbase 4.

Dbase 4

Le format Dbase IV est un format de stockage de données qui était surtout populaire dans les applications des années 90. Il se présente sous forme de fichiers comprenant plusieurs colonnes typées (Caractère, Date, Numérique etc...).

Ces fichiers DBF structurés sont à peine meilleurs que des fichiers plats comma-separated dans le sens où ils ne possèdent pas d'index. Ils n'offrent d'ailleurs pas de prise en charge du langage SQL en standard.

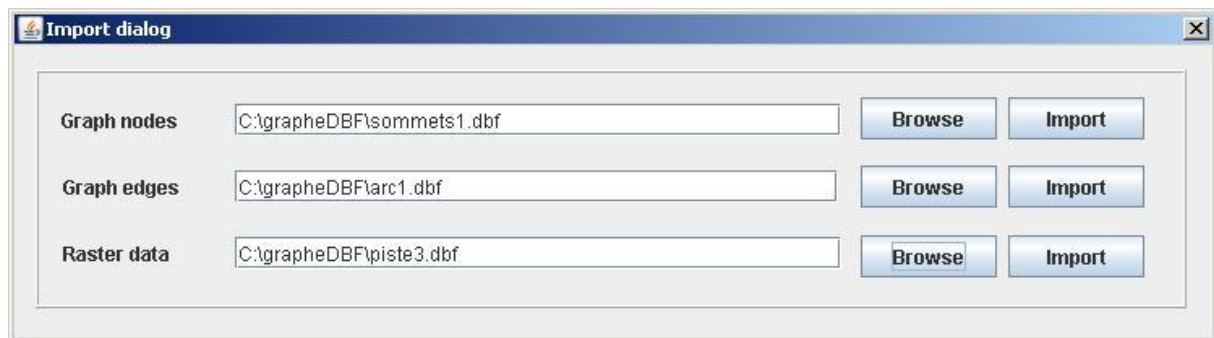
Nous avons été en mesure de les déchiffrer en utilisant l'API open-source javaDbf. (<http://javadbf.sarovar.org/>).

Apache Derby

Les fichiers DBF fournis par les outils GIS doivent ainsi être utilisés par nos simulations. Mais avant cela, nous avons décidé de les importer dans une base de données Derby (<http://db.apache.org/derby/>).

Derby est une base de données SQL légère et open-source écrite entièrement en java provenant de la fondation Apache.

La première étape de l'opérateur consistera donc à importer les fichiers DBF du GIS dans notre base de données. Pour cela, il a à sa disposition l'écran d'importation suivant :



Les tables dans lesquelles les DBF sont importés ont une structure IDENTIQUE à celle des fichiers sources, alors à quoi bon les copier dans une base de données au lieu de laisser les simulations taper directement dedans?

Il y a deux raisons à cela:

- L'importation permet la validation des données grâce aux contraintes d'intégrité entre les tables, une fois que les données ont été insérées dans notre base, nous sommes quasi-certain de leur cohérence ce qui nous évite de devoir complexifier le code de nos simulations avec une gestion poussée de cas de bord.
- Le requêtage est plus aisé, nous disposons de toute la puissance du langage SQL, ce qui nous permet de rapidement extraire certaines données telles que les coordonnées MIN et MAX d'une grille raster. Faire ce genre de recherche à la main nous forcerait à écrire du code complexe qui nuirait à la maintenance du travail.

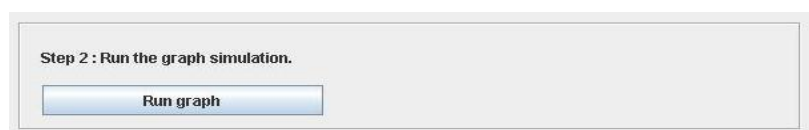
Déroulement du processus

Une fois les données DBF importées dans notre application, nous allons pouvoir lancer le processus. Voici dans l'ordre les étapes que l'opérateur doit effectuer.

Simulation du graphe

Il suffit pour cela de presser le bouton "Run Graph", la simulation s'exécute aussitôt de façon automatique, sans interface graphique dans le même process que l'application.

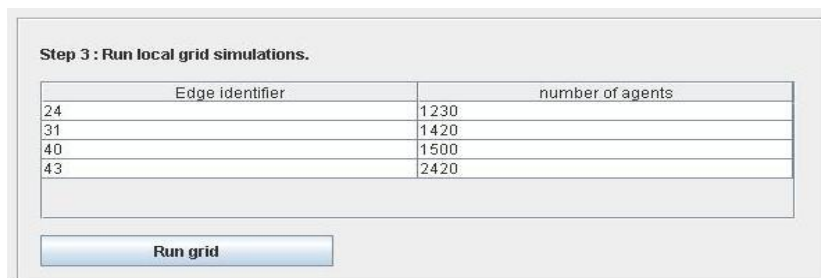
Exécuter la simulation dans le même processus possède un avantage majeur : les exceptions java non gérées par la simulation peuvent être interceptées par notre application swing. Ceci jouera sans doute un rôle important lorsque l'automatisation sera recherchée.



Simulation des grilles locales

A présent que le graphe s'est exécuté et que l'on dispose des statistiques d'utilisation de chaque piste, nous pouvons passer à l'étape suivante : les simulations individuelles des pistes.

Les arcs disponibles, c'est à dire ceux pour lesquels nous disposons à la fois de statistiques d'utilisation et données rasters s'affichent dans une table.



Il suffit de sélectionner une ligne, puis de presser "Run grid" pour une exécution silencieuse de la simulation correspondante.

Exportation du résultat

Une fois que toutes les grilles locales ont été exécutées, il ne reste plus qu'à générer un fichier DBF contenant nos résultats en cliquant le bouton "Export", le processus est à présent terminé.



Conclusion

Cette application swing simple constituée de 2 dialogs sert de liant afin de cimenter nos deux précédentes simulations.

Il ne s'agit pas là d'une application "state of the art", elle n'est pas véritablement aussi dynamique et user-friendly que l'exigent les standards actuels en informatique de gestion mais elle constitue la pierre finale de notre "proof of concept".

Nous avons en effet démontré un point clef de la faisabilité technique de notre approche, la possibilité de piloter une simulation depuis une application externe. L'intérêt de cette démarche ne saute peut être pas aux yeux avec les exemples d'aujourd'hui. Mais lorsque nous travaillerons sur des domaines skiabiles comportant plusieurs centaines de pistes, il sera possible de s'inspirer des travaux effectués sur cette application pour automatiser le traitement.

La mise en place d'une interface utilisateur finale complète, intuitive et features-rich (historisation des résultats, wizards en tout genre) sera laissée à mon successeur.

Conclusion

Chapitre 7

Projet de simulation multi-agent

Stéphane Claret

Proof of concept

Ce travail de diplôme touche à présent à sa fin, les applications fournies fonctionnent, nous avons couvert l'intégralité du processus, de l'importation des données en provenance du GIS jusqu'à l'exportation de nos résultats vers celui-ci.

A travers ce travail, nous avons rempli notre objectif qui était de prouver la faisabilité de cette méthode de simulation en 2 temps. C'est là notre "proof of concept".

Tout ceci n'est cependant que la première brique d'un haut mur, les comportements implémentés dans nos simulations ont encore besoin de polissage, les échanges de données avec les outils GIS doivent encore être complétés... Il est évident que le projet Juste-Neige ne se terminera pas avec mon travail mais nous en étions conscients depuis le début.

Cependant, toute cette structure, ce squelette que nous avons mis en place servira pour la suite, nous avons déjà passablement dégrossi le problème. Nous avons pu nous rendre compte des difficultés, des limites techniques de nos outils et finalement la chose plus importante : Nous avons choisi une direction!

Déroulement du travail

Ce projet était un travail d'exploration et de recherche, nous avons constamment remis en question nos objectifs et les techniques utilisées, allant jusqu'à s'écarter assez fortement de l'idée initiale.

Nous ne devions réaliser qu'une seule simulation, au final nous avons construit 2 simulations de nature différente plus une application swing.

La seule phase à s'être véritablement déroulée comme prévue a été la prise en main de Repast, par la suite nous avons fait évoluer nos plans de semaine en semaine.

Rareté de la documentation

Le monde de la simulation reste petit, la documentation ne se trouve pas en abondance et les problématiques sont souvent très spécifiques par projet. A cause de cela, nous n'avons pas pu trop compter sur des recherches google pour répondre à nos interrogations.

J'ai tout de même personnellement participé au forum de repast pour obtenir de l'aide sur certains points quelque peu sous-documentés. J'ai été jusqu'à contacter par email l'un des auteurs de Repast, Nick Collier de l'université de Chicago.

Bilan personnel

Je suis personnellement enchanté des résultats que nous avons obtenu avec ce travail de diplôme.

Mon principal regret est de ne pas avoir disposé de plus de temps pour peaufiner les détails de mon application ainsi que ce document que vous lisez. Tout ceci est dû au sentiment d'urgence induit par le remaniement des objectifs.

Sur les dernières semaines, la pression était grande, je voulais à tout prix fournir un travail de grande qualité comme couronnement de mes efforts et de mon implication dans ce projet. Car tout au long de son déroulement j'ai eu à coeur que nous réussissions.

Etant un étudiant en filière d'emploi, prendre le pari de finir cette année était déjà un grand risque, j'ai une vie professionnelle que je dois assurer en parallèle, le timing était serré mais j'ai fait ce choix en connaissance de cause et je l'ai assumé jusqu'au bout, bien qu'une bonne partie de mon temps libre y soit passée.

Quand je regarde en arrière, je me dis qu'il est regrettable d'avoir perdu tant de temps et d'énergie à tenter de lire les fichiers rasters d'ArcGis directement. Pourtant la démarche était légitime avant tous ces changements de direction, mon expérience professionnelle m'a déjà démontré à plusieurs reprises que ce genre d'erreur est parfois inévitable.

Tout ce travail a représenté un challenge conséquent. Je dois à présent passer la main à mon successeur pour la suite. Une bonne partie des idées que j'ai mises en place accompagneront ce projet jusqu'à son aboutissement et je fais de cela ma grande satisfaction personnelle.

Remerciements

Je tiens à adresser mes remerciements à messieurs Michael Schumacher et Jean-Christophe Loubier pour ce remarquable travail d'équipe ainsi que pour la confiance témoignée tout au long de ce projet.

Ressources Web

Site internet de repast, la plate-forme de simulation utilisée dans ce projet.

<http://repast.sourceforge.net/>

Un groupe de discussion autour de Repast sur lequel j'ai passé de nombreuses heures.

<http://www.nabble.com/Repast-f3965.html>

Quelques tutoriaux sur une ancienne version de repast, mais d'actualité sur le fond.

<http://complexityworkshop.com/tutorial/RePast/index.html>

Un exemple de simulation faite sous Repast en modèle GIS, impressionnant.

<http://crimesim.blogspot.com/2008/04/introduction-my-model.html>

Un blog comportant une belle série d'exemple de scénario en projection GIS.

<http://gisagents.blogspot.com/2007/04/creating-slider-bars-in-repast.html>

Le site qui m'a le plus aidé dans mon apprentissage de Repast.

<http://portal.ncess.ac.uk/access/wiki/site/mass/symphony%20tutorial.html>

Agent analyst, une solution considérée mais vite laissée tomber.

<http://www.institute.redlands.edu/agentanalyst/Tutorials.html>

Site d'arcGis, l'outil utilisé pour l'aspect géomatique de "Juste Neige"

<http://www.esrifrance.fr/>

La documentation d'arcObjects, l'API propriétaire d'arcGis.

<http://edndoc.esri.com/arcobjects/9.1/>

Geotools, une librairie open source gérant un grand nombre de formats GIS.

<http://geotools.codehaus.org/>

Le site de derby, base de donnée utilisée dans ce projet, doc et référence SQL.

<http://db.apache.org/derby/>

Le site de javaDbf, la librairie utilisée pour lire les fichiers DBF.

<http://javadbfsarovar.org/>

DbUtils, la librairie utilisée pour l'accès aux données.

<http://commons.apache.org/dbutils/>

ApacheIO, une librairie utilisée pour simplifier la gestion de fichiers en java.

<http://commons.apache.org/io/>

Illustration rasters

<http://www.edc.uri.edu/criticallands/raster.html>

Calendrier

Le calendrier ci-dessous résume nos rencontres ainsi que le suivi général du projet.

08.07.2008	Réunion de kick-off du projet à Bellevue. -Commencement de mon apprentissage de Repast.
17.08.2008	Rencontre avec Mikail Kanevski à l'IGAR. -Installation d'ARC GIS sur mon portable.
12.09.2008	Séance avec Michael Schumacher et Jean-Christophe Loubier à l'EPFL. -Nous prenons la décision de réaliser l'exemple du chapitre 2.
26.09.2008	Séance avec Michael Schumacher au technoArk. -Je présente l'exemple du chapitre 2, nous axons nos recherches sur l'interface GIS.
17.10.2008	-Echec des tentatives d'interfaçage avec Arc Gis, Nick Collier programmeur de repast contacté à ce sujet recommande un couplage plus faible par import/export.
20.10.2008	Rencontre à l'EPFL avec Michael Schumacher. -Nous lançons l'idée du guidage de nos skieurs à l'aide d'une diffusion.
23.10.2008	Séance avec Michael Schumacher et Jean-Christophe Loubier à l'EPFL. -Nous recherchons ensemble une solution pour implémenter cette diffusion.
27.10.2008	Séance avec Mikail Kanevski et Jean-Christophe Loubier à l'IGAR. -Discussion sur l'échange de données, les gens de l'IGAR préconisent le format Dbase. Séance avec Michael Schumacher et Jean-Christophe Loubier à l'EPFL. -Adoption de l'idée de la simulation en deux temps et abandon de la diffusion.
06.11.2008	Séance avec Michael Schumacher au TechnoArk. -Présentation du prototype de ma simulation en graphe.
13.11.2008	Séance avec Michael Schumacher au TechnoArk. -Présentation de la seconde version de ma simulation en graphe.
20.11.2008	Séance avec Michael Schumacher à Bellevue. -Décision d'inclure les simulations en grille dans l'objectif -Décision de créer une application swing liante.
13.12.2008	Rendu du travail de diplôme.

Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué le travail de diplôme ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de diplôme, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail.

Claret Stéphane

13 décembre 2008