Université de Cergy-Pontoise

RAPPORT

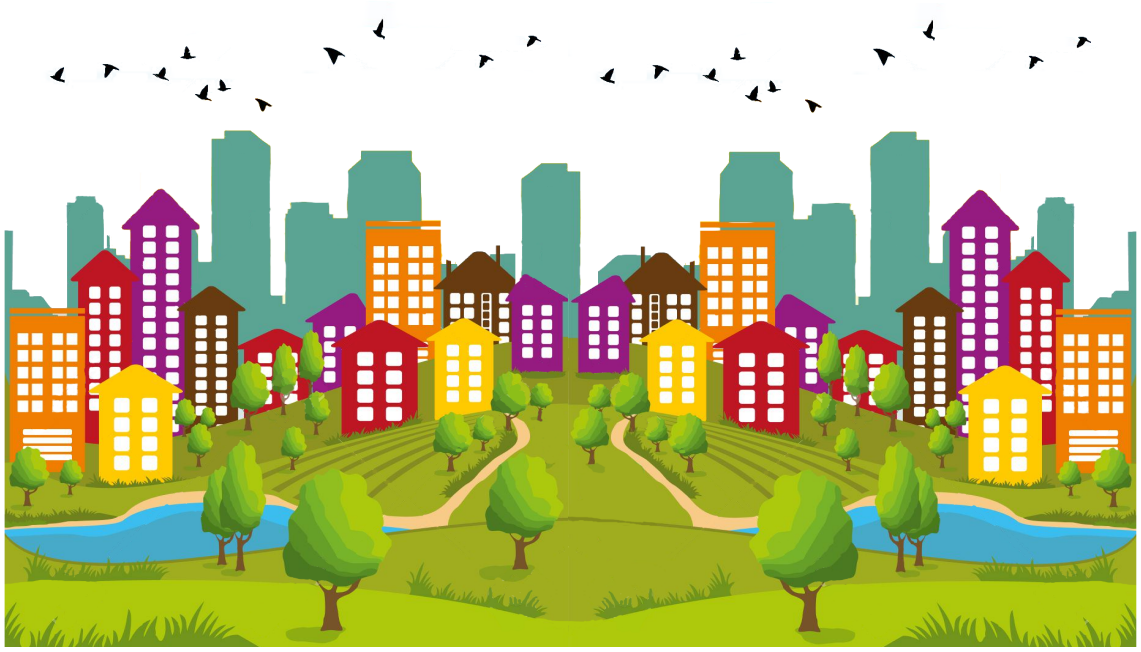
pour le projet Génie Logiciel
Licence d'Informatique deuxième année

sur le sujet

Urbain

rédigé par

Matthieu Vilain - Quentin Gerard



Mai 2017

Table des matières

Table des figures

Liste des tableaux

Remerciements

Les auteurs du projet voudraient remercier...

1 Introduction

Le projet Le projet consiste à la création d ?un jeu vidéo simulant une vie urbaine, dans laquelle plusieurs individus vivent leur vie au sein d ?une ville. L ?utilisateur pourra influer sur le comportement des individus et les paramètres de la ville.

Fonctionnalités Fonctionnalités du programme : Le joueur aura plusieurs actions possibles afin d'infuer sur l ?évolution de la ville. Tout d'abord il pourra agir sur le temps en l ?accélérant ou en le mettant en pause. Ensuite il pourra accéder à des informations sur les personnages comme : leurs informations de bases, leur historique ou leurs objectifs directs (exemple : ce personnage se rend à la piscine). Ces informations pourront être changées par l'utilisateur et il pourra ainsi renommer un personnage, le faire déménager ou le faire rentrer chez lui par exemple. De même pour les bâtiments, l'utilisateur pourra accéder à ses informations et les modifer. Il pourra donc par exemple modifer les horaires d'ouverture d'un lieu ou faire varier son nombre d'utilisateur maximum. Le joueur pourra donc modifer à sa guise ses informations et voir ce que ces modifcations apportent de bon ou de mauvais sur à population de la ville.

Nos motivations Nous avons choisi ce projet car il représente une opportunité pour chacun de nous d ?explorer des domaines/notions qui nous intéressent, et dans lesquelles nous voulons nous perfectionner.

2 Spécification

3 Réalisation

3.1 Le Model - présentation des classes

3.1.1 La ville

La ville est le premier élément qui constitue notre projet. Elle est composée de différentes infrastructures : Les routes, les maisons, les bâtiments de travail et les divertissements. Chaque type d'infrastructure possède une utilité qui lui est propre :

- Les Routes permettent aux personnages de se déplacer
- Les Maisons permettent aux personnages de se reposer le soir et regagner de l'émotion
- Le Travail est une activité imposé à chaque personnages et sera la principale source de baisse d'émotion
- Les Divertissements permettent aux personnages lorsqu'ils ne dorment pas de regagner de l'émotion

image

Toutes les infrastructures possèdent en commun :

- Un nom de type String
- Un type au format int
- Une position dans la Map
- Un taille
- Et un nombre d'utilisateurs courant

Et les bâtiments (hors routes) possèdent tous en commun ces caractéristiques :

- Une adresse, seule point d'entré dans le bâtiments au niveau de la Map
- Une récompense, qui peut être positive ou négative suivant le type de bâtiment
- Un nombre maximum d'utilisateur

En plus de ces caractéristiques, les bâtiments de travail et les divertissements possèdent un temps d'utilisation moyen par les personnages, ainsi que des horaires d'ouvertures durant lesquelles les personnages pourront utiliser ces bâtiments. Il est également impossible pour un personnages d'utiliser un bâtiment lorsque celui ci à atteint son nombre maximum d'utilisateur. Les routes et les maisons sont ouverts 24h/24.

La taille de la Map ainsi que la répartition des infrastructures est déterminé à l'avance dans un fichier CSV. La création de la Map dans la mémoire se fait grâce au design pattern Builder. Chaque ligne du fichier CSV renseigne, le type, l'adresse, la taille et sa position dans la Map. Ainsi l'objet MapBuilder va lire les lignes du fichier une par une grâce à la bibliothèque Apache Common CSV, et faire appelle au autres Builder correspondant à chaque type d'infrastructure.

Durant leur création, les Infrastructures de type Work ou Entertainement, se voient également attribuer un nom, des horaires d'ouvertures et leur récompenses. Ces informations sont aussi renseignées à l'avance dans un fichier CSV.

3.1.2 La population

La population est le second ensemble qui constitue notre projet. Nous l'avons voulu le plus adaptatif possible. Ainsi elle peut contenir autant d'individu que souhaité. Cependant, une trop grande population provoquera des ralentissements de l'interface graphique mais le moteur de jeu est parfaitement capable de faire tourner une grande population. Pour la jouabilité du jeu, nous avons fixé le nombre d'individus au démarrage d'une partie à 5.

Création des personnages Un personnage est composé d'informations de base comme d'un nom, d'un prénom, d'un sexe , d'un age et d'un numéro d'identité.

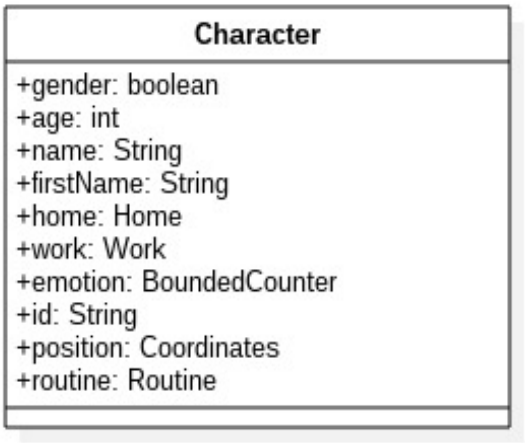


FIGURE 1 – Classe Character

Pour le nom, le prénom et le sexe du personnage, ils sont initialisés à partir de fichier CSV. Le premier fichier contient une liste de 300 noms de familles et le second une liste de 200 prénoms, 100 masculin et 100 féminin. Lors de la création du personnage, le programme va prendre au hasard un nom dans le fichier name.csv et un prénom (associé à un sexe) dans le fichier firstName.csv. L'age du personnage est simplement choisit aléatoirement entre 10 et 100 ans. Le numéro d'identité du personnage est unique, il est calculé à partir de toutes les informations du personnage grâce à un code de hashage. Ce code nous permettra de reconnaître le personnage. Ce grand nombre de choix nous permet de garantire une grande diversité au sein de la population.

Comme le montre la figure 1, un personnage possède également une maison et un travail. Ces deux éléments sont également attribué aléatoirement via une recherche dans la liste de lieux de travail et d'habitation de la carte de jeu.

Un personnage possède également une jauge d'émotion qui peut varier de 0 à 100. C'est comme la barre de vie du personnage. Elle est initialisé a 75 en début de partie mais elle variera en fonctions des actions des personnages. Si le personnage atteint une émotion de 0 il meurt.

Enfin la création de la routine sera expliqué dans la partie moteur.

Nous avons voulu rendre nos personnages le moins statique possible. Ainsi d'une partie à l'autre, la chance de tomber sur des personnages avec les mêmes propriétés est très faible.

L'organisation de la création des personnages se fait grâce au design pattern builder.

La classe PopulationBuilder va construire une population grâce à la carte de jeu (pour assigner les résidences) et grâce au CharacterBuilder. Ce dernier va faire la lecture dans les fichiers CSV grâce à la bibliothèque Apache-commons-csv et va faire les choix aléatoires pour initialiser les différentes composantes de nos personnages.

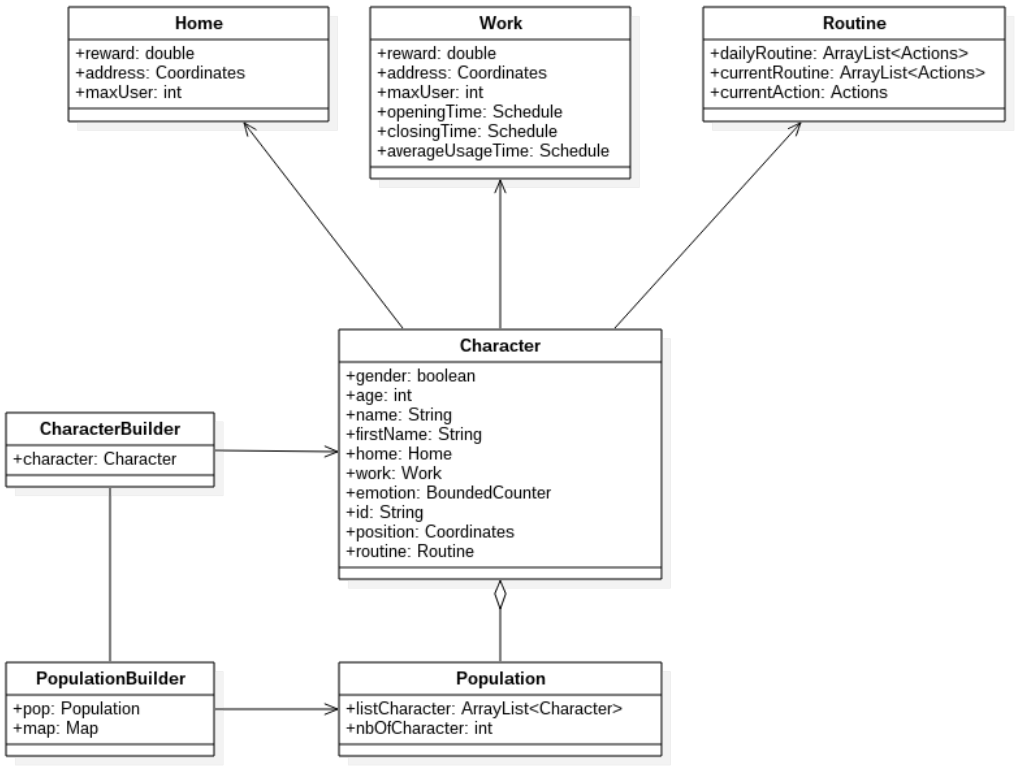


FIGURE 2 – Organisation de la partie population

3.2 Le moteur de jeu

3.2.1 Gestion de l'émotion des personnages

L'émotion représente la barre de vie des personnages, elle varie de 0 à 100 en fonction du temps. Si elle atteint 0, le personnage meurt. L ?évolution de cette jauge d ?émotion est totalement dynamique, elle se fera automatiquement en fonction des actions faites par le personnage. Certaines actions vont apporter un bonus sur la jauge d'émotion du personnage alors que d'autres vont apporter un malus. Pour certaines actions le bonus est constant (par exemple dormir apportera toujours +30) mais pour d'autres le bonus/malus est variable en fonction du lieu dans lequel se fait l'action. Par exemple travailler dans un atelier auto apportera un malus de -25 car la tache est difficile alors que travailler dans une boutique d ?électronique apportera un malus de -15. Même fonctionnement pour les bonus des loisirs. Ces valeurs sont initialiser lors que l'initialisation des bâtiments, elles proviennent donc d'un fichiers CSV. Enfin les rewards de sont pas toujours effectif au même moment. Pour la plupart des actions, le personnage perçoit le reward en fin d'action. Pour l'action de déplacement, dans un soucis de mieux représenter la réalité, le malus est retiré à chaque itération de temps. C'est à dire que plus le chemin que le personnage a à faire est long, plus il perd de l'émotion.

Action	Reward	Effectivité
Sleeping	+30	En fin d'action
Chilling	+5	En fin d'action
Schifting	-1	A chaque itération de temps
Working	Malus variable [-25 ;-10]	En fin d'action
Entertain	Bonus variable [+5 ;+20]	En fin d'action

3.2.2 Gestion de la routine

La routine est un enchaînement d'actions que va exécuter le personnage. Le personnage peut exécuter 5 types d'actions différentes regrouper en familles : les déplacements et les occupations. Les actions d'occupation sont reliées à un lieu alors que les actions de déplacement sont reliées à un lieu de départ,

un lieu d'arrivé et un chemin entre les deux.

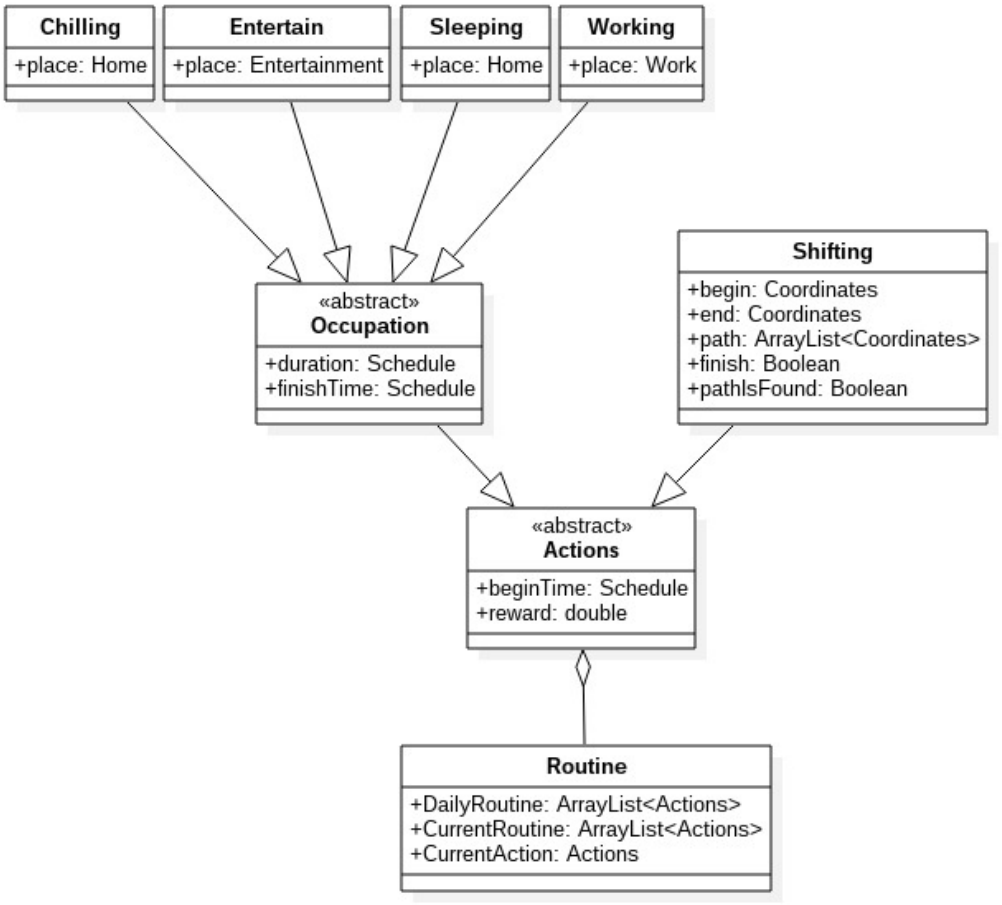


FIGURE 3 – Classe routine

L'un des dilemme du projet est l ?interaction entre les différentes actions pour que chaque personnage puisse avoir puisse suivre une suite d'actions indépendantes mais que l'utilisateur puisse ajouter des actions a faire sans perturber le bonne enchaînement des actions.

Nous avons décidé de décomposer ce problème en 3 partie :

- Une partie statique qui organise les grandes lignes des journées du personnage : métro/boulot/dodo
- Une partie dynamique qui évolue au court de la journée et qui représente la liste d'actions que le personnage a à faire
- Une partie utilisateur associé à différente options d'ajout/suppression d'actions

La liste d'actions communes a chaque journées est stocké dans la liste DailyRoutine. A chaque début de journée, on ajoute toutes les actions de cette liste a la de la journée courante, CurentRoutine. La currentRoutine à les même propriétés qu'une files mais avec plus de possibilité d'ajout d'actions. On défile currentRoutine et on ajoute cette action à l'action courante, currentAction. Cette action est exécuté par le personnage et lorsqu'elle est finie, on défile à nouveau la currentRoutine. Enfin l'utilisateur peut soit ajouter un action en tête ou en queue dans la currentRoutine ou il peut casser l'action courante du personnage mais cela engendra un malus d ?émotion.

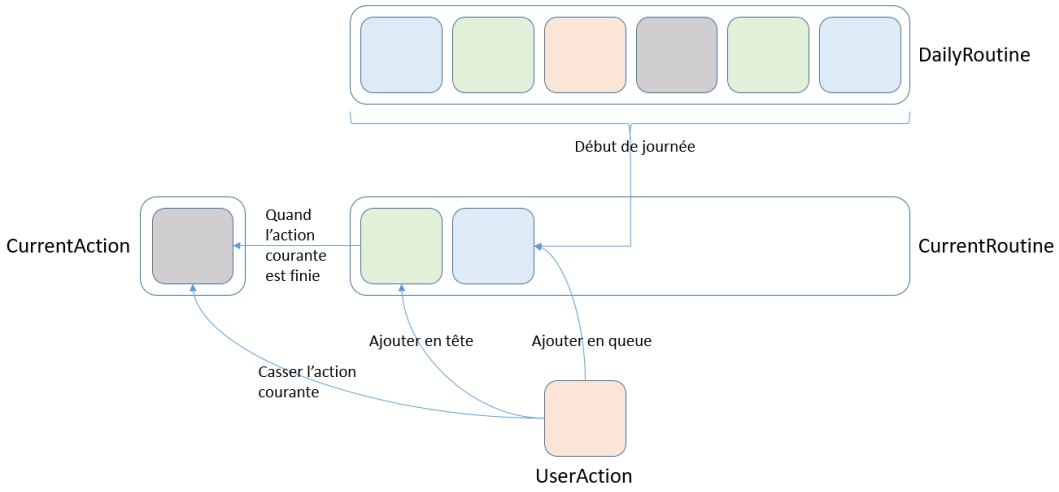


FIGURE 4 – Fonctionnement de la routine

3.2.3 Calcul d'itinéraire

3.3 L'interface homme/machine

L'interface graphique est composé de plusieurs éléments :
- L'horloge du jeu, permettant de savoir quelle est la date et l'heure dans notre ville fictive
- La map, ou nous pouvons voir les infrastructures, et les personnages se déplacer
- La liste des personnages accompagné de l'état de leur barre d'émotion

Chacun de ces éléments sont séparé dans différentes classes qui héritent de la classe JPanel , ce qui permet un assemblage facile de l'interface graphique, et une permutation plus facile entre différentes versions d'un même éléments.

L'interface graphique est réalisé grâce à Java Swing/AWT/Java 2D .

Nous pouvons voir dans le diagramme ci dessus que l'ensemble des éléments graphiques sont rassemblé dans une seule classe principale, qui à pour but de les assembler correctement.

La Map et la liste des personnages sont cliquables et affiche des informations en fonction de la position du curseur au moment du clique. Cela peut permettre d'ouvrir un autre Panel, une autre fenêtre ou autre, contenant les informations souhaitées, en fonction des situations (Pour plus de détail, se reporter au manuel d'utilisation). Ces fonctionnalités ont été implémenté grâce à des MouseListener, qui nous permettent de récupérer des informations sur le curseur ou l'état de la souris/trackpad lors d'événements prédéfini, lors d'un clic de souris par exemple.

Références

[1] L. M. Haas, E. T. Lin, and M. A. Roth. Data integration through database federation. *IBM Syst. J.*, 41(4) :578–596, 2002.