MODULES POO/SD

RAPPORT : PROJET TNCITY

Guilain Leduc, Matta Karczewski, Marie-Lys Rousseau

Date de rendu : 28 Mai 2017

Sommaire

INTRODUCTION

Ce rapport est la synthèse de la deuxième phase du projet des modules POO/SD, dont la date de rendu est fixée au 28 mai 2017.

Rappel du sujet

Ce projet a pour objectif de produire un jeu SimcityLike jouable et ambitieux. La réalisation de ce projet est divisée en plusieurs sous-objectifs dont les premiers sont :

1. La création d'un réseau électrique et d'eau potable, avec des centrales et des réservoirs
2. Différents types de terrains : eau, forêt...
3. Un réseau routier ou ferroviaire
4. Événements aléatoires avec des mesures de protections
5. L'amélioration de l'interface utilisateur

La liste est non exhaustive, mais il nous était rappelé de ne pas sous-estimer le temps de réalisation du travail. L'objectif est de développer un jeu équilibré en fixant les prix, taxes, valeurs de production et de consommation, ainsi que de calculer certaines statistiques comme le bonheur, la santé et la sécurité qui doivent impacter l'équilibre du jeu.

Contenu

Ce document contient dans un premier temps une synthèse de la première phase de notre projet, puis l'architecture de notre jeu, à savoir les classes que nous avons créées et modifiées et les structures de données que nous avons utilisées. Nous expliquons ensuite les choix que nous avons fait. Il contient également la description des tests que nous avons déployés ainsi que les éléments de gestion de projet.

1 SYNTHÈSE DE LA PREMIÈRE PHASE

Lors de la première phase, nous avions décidé d'implémenter des zones industrielles, commerciales et résidentielles. Concernant les infrastructures, nous voulions que l'eau soit extraite de la mer, puis passe dans une station d'épuration avant d'être délivrée, et nous souhaitions traiter le rejet des eaux usées. L'objectif concernant l'électricité était de proposer différents types de centrales plus ou moins rentables et polluantes. Nous voulions implémenter une infrastructure routière améliorable pour éviter les embouteillages. Pour les bâtiments, nous visions à implémenter différents types de bâtiments publics, notamment une mairie pour gérer les impôts, ainsi que des hôpitaux, commissariats, casernes et écoles, et des infrastructures de loisir.

Notre projet n'ayant pas été redirigé par l'encadrant lors de la soutenance, nous avons lancé la deuxième phase en poursuivant les objectifs que nous nous étions fixés.

2 ARCHITECTURE

Nous sommes partis du squelette donné, dont il a fallu saisir le fonctionnement avant de pouvoir le modifier.

2.1 Les Classes

2.1.1 Diagramme de classe

Les diagrammes de classes simplifiés - sans les attributs - sont disponibles en annexes pour plus de lisibilité. Les diagrammes de classes complets se trouvent dans le dossier "diagrammes de classe" du projet.

2.1.2 Javadoc

Une Javadoc est disponible pour chaque fonction en commentaire dans le code. Sinon une Javadoc est disponible en Html dans le dossier doc.

2.1.3 Explications des classes

Voici les différentes classes que nous avons modifiées ou ajoutées au cours du projet, groupées par package.

— Package calculMat

La classe **EtatRes** est une énumération qui donne l'état local du réseau : actif, producteur, inactif, ou pas de réseau. Elle est appelée dès qu'il y a un appel au réseau, donc dans toutes les tuiles qui sont classées "pas de réseaux", les routes qui sont notées par défaut "inactives" et les bâtiments "producteurs".

L'instance de la **matRes** est créée dans CityResources. Elle fait appel aux deux énumérations TypeRes et EtatRes et elle est modifiée dès que l'on pose une tuile. Son fonctionnement est expliqué plus en détails dans la partie Choix techniques.

La classe **TypeRes** est une énumération qui indique s'il s'agit d'un réseau routier, électrique ou d'eau, et elle contient une fonction qui permet de passer de l'énumération à des int et des int à des énumérations pour être utilisés dans la matrice.

— Package dialog

GestionImpots est une classe permettant d'ouvrir une fenêtre grâce à laquelle le joueur va pouvoir changer les taux d'impositions pour les impôts locaux et la TVA, ainsi que les dépenses qu'il fait chaque mois pour les hôpitaux, les commissariats et les parcs et jardins. Le résultat modifiable par des spinners est enregistré au moment de la validation dans les variables correspondantes à ces taux dans la classe Impôts.

Les classes **LuckyYouMessage** et **RiotMessage** permettent l'ouverture de fenêtres de dialogue lorsqu'un de ces deux événements se produit. Elles permettent d'avertir le joueur de ce qui est arrivé. Les classes **WarningMessage** et **DebtMessage** sont des classes dont les instances sont créées dans nextState : des fenêtres d'avertissement s'ouvrent lorsque le joueur a des ressources négatives, lui anonçant qu'il ne peut plus poser de tuiles, et lorsqu'il a plus de 200 habitants pour l'inciter à poser un hôpital et un commissariat. Avant que le seuil des 200 habitants ne soit dépassé, le joueur ne peut placer ces deux tuiles : s'il essaie, le GameBoard créée une instance de **NoNeedHospital** ou de **NoNeedPolicestation**.

— Package launcher

La classe **panneau** permet d'afficher le fond du menu d'ouverture du jeu. Nous avons modifié la classe **SimCityUI** pour prendre en compte le menu d'ouverture qui permet de lancer le jeu ou de changer la langue, grâce aux boutons. Les boutons de langue appellent UKTexts ou FRTexts selon le choix du joueur. De plus, nous avons modifié cette classe pour avoir les outils situés en haut du GameBoard et créer une instance de toolsView.

— Package localization

Les classes **LocalizedTexts** sert à afficher différentes informations du jeu et les labels

associés dans la fenêtre, comme par exemple le nombre de travailleurs, la capacité de soin des hôpitaux, le bonheur, la quantité d'eau et d'électricité disponible. Elles permettent au joueur de connaître ses ressources. Nous les avons modifiées à mesure que nous avons ajouté des éléments dans le jeu comme les puits, les centrales, le bonheur... **UKTexts**et **FRTexts** sont tous les labels associés, en anglais ou en français, qui apparaîtront sur le GameBoard.

— Package model

Nous avons beaucoup modifié **CityResources** pour rajouter les différentes ressources du jeu ainsi que les fonctions liées, à savoir des get et des set pour chaque ressource. La méthode resetEphemeral a été modifiée : les notes de sécurité et de santé de la ville y sont calculées, ainsi que les informations du nombre de bâtiments en état "riot" et elle appelle la méthode refreshHappiness(). La méthode refreshMatRes a été ajoutée et appelle le refresh() établi dans MatRes pour calculer la matrice du prochain tour.

Nous avons beaucoup modifié le **GameBoard**. Tout d'abord, dans son constructeur, nous avons créé une instance de la classe Random afin d'obtenir une carte différente à chaque partie, avec de l'eau, des forêts et des plaines. Nous avons rajouté les différents outils des tuiles supplémentaires. Nous avons également créé un certain nombre de "get" afin d'obtenir dans le GameBoard des informations se trouvant dans CityResources, au fur et à mesure que nous avons ajouté des paramètres, comme par exemple le bonheur, la santé, le nombre de produit, l'eau disponible... Nous lui avons ajouté une fonction hasCityHall qui renvoie un booléen et vérifie sur chaque tuile de la carte s'il s'agit d'une instance de la mairie. Elle est appelée dans effectTile lorsque l'outil courant est la mairie afin d'empêcher que plusieurs mairies ne soient placées sur la carte. Toujours dans effectTile, nous créons une nouvelle instance de GestionImpots dès que la tuile sélectionnée par le joueur est l'instance de la mairie, et une nouvelle instance de NoNeedHospital ou NoNeedPolicestation si le seuil de 200 habitants n'est pas dépassé -présentées plus haut. Dans nextState, nous avons ajouté plusieurs paramètres que nous souhaitions voir évoluer chaque mois, comme les impôts à recalculer, les modifications des ressources courantes, le rafraîchissement de la matrice réseau en fonction des modifications des bâtiments avec l'appel de refreshMatRes, le calcul du bonheur....

— Package model.event

Nous avons créé trois événements : le premier est un coup de chance. Sur le modèle du monopoly, la banque verse au joueur une certaine somme. Cet événement est implémenté dans la classe **LuckyYouEvent**. La classe **RiotEvent** produit des émeutes lorsque le bonheur

de la population est trop bas. Elle fixe un nombre de bâtiments à mettre dans "l'état d'émeute" dans le GameBoard, ce qui les empêche alors de produire dans le cas des usines. Ces classes créent des instances de LuckyYouMessage et de RiotMessage.

Nous avons également implémenté un événement qui fait tomber une pluie de météorites aléatoirement. Un compteur aléatoire qui choisi une tuile : s'il s'agit d'une instance de Buildable, le bâtiment est détruit.

Dans **EventFactory**, nous avons choisi les différentes probabilités des événements. Probabilité de chance 3/100
Probabilité de la pluie de météores 1/100
Probabilité d'une émeute 20/100
Rien : 76/100

— Package model.tiles

Nous avons modifié le **BuildableTile** pour ajouter la gestion de l'eau calquée sur le modèle de l'électricité et nous avons modifié "evolve" pour que les tuiles ne passent de l'état "under-construction" à "build" que s'il y a un accès aux réseaux.

Nous avons créé la classe **CityHall** afin d'avoir une mairie, point central de notre jeu. Elle hérite de BuildableTile. Elle est le point d'activation du réseau.

Nous avons créé la classe **IndustrialTile** et instancié les méthodes comptant le nombre d'habitants faisant partie de la population active, ceux au chômage. En fonction de la production d'eau, d'électricité et de la santé, les usines sont plus ou moins productives. La capacité de production augmente en fonction du nombre d'usines et ne dépasse pas un certain seuil. Lorsqu'il y a une émeute, l'usine ne produit plus, jusqu'à ce que le bonheur remonte au-dessus de 20.

Nous avons créé **CommercialTile** qui fonctionne sur le même principe que la classe IndustrialTile en ce qui concerne la gestion des employés. Des habitants y travaillent et en fonction du nombre de travailleurs, de la population globale et de son bonheur, le magasin a la capacité de vendre plus ou moins de produits.

Les classes conductorTile et conductorTilonroad ont été implémentées pour permettre d'avoir des tuiles dédiées à la gestion de l'eau et de l'électricité. Cependant, cela représentait un problème de lisibilités sur le GameBoard, donc nous avons préféré les mettre de côté.

Les **forestTiles** sont des tuiles sur lesquelles on peut construire monnayant un coût supplémentaire. On ne peut pas construire sur les **waterTiles** ni sur les **shoreTiles**, qui sont plus esthétiques et rendent la carte plus ou moins évidentes à jouer.

La classe **HospitalTile** fonctionne sur le principe d'une capacité d'accueil de malades.

Si cette capacité est assez importante, cela améliore la note de santé "IllnessRate" de la ville et cette note affectera la production des usines et le bonheur de la population. La capacité des hôpitaux changent en fonction des budgets alloués par le joueur dans la fenêtre GestionImpots.

Nous souhaitions implémenter au début du jeu une classe abstraite pour tous les bâtiments publics. Nous n'avons pas réussi à factoriser le code correctement : nous l'avons créé tôt dans le projet, alors que nous ne savions pas encore comment nous allions gérer ce type de bâtiments. Nous ne l'avons donc pas gardée.

La classe **ResidentialTile** a été modifiée pour être adaptée à la matRes et au RiotEvent.

RoadTile est une des première classe que nous avons créé : il s'agissait donc de comprendre le code, la façon dont étaient générées les tuiles ainsi que les outils dans la barre d'outils. Elle a ensuite été modifiée pour s'adapter à la matrice.

Dans la classe **Tile**, nous avons rajouté un attribut tilePosition qu'on utilise pour la gestion du réseau, ainsi qu'un vecteur réseau dans chaque tuile pour définir sa caractéristique sur le réseau -en fonction des TypeRes et EtatRes. Nous avons ensuite créé des get et des set afin d'obtenir pour chaque nouvelle instance d'une tuile les vecteurs réseaux et leurs coordonnées.

Nous avons créé **WaterPlantTile** qui fonctionne comme **PowerPlantTile**. Nous n'avons pas beaucoup modifié cette dernière, nous y avons juste ajouté une initialisation du vecteur réseau sur le mode "producteur".

PolilceStationTile est une classe créée sur le même modèle que HospitalTile, avec une CellCapacity au lieu de CareCapacity et SecurityRate à la place de IllnessRate. La différence entre ces deux classes provient de l'effet qu'elles ont sur le bonheur : cosntruire trop de commissariats fera baisser le bonheur, mais réduira le nombre d'émeutes.

La classe **PublicGardenTile** permet d'augmenter le bonheur de manière fixe et implémente Destroyable.

— Package model.tools

Dans le package Tool, nous avons rajouté les outils correspondant aux tuiles que nous avons créées.

— Package ui

Nous avons modifié **IconFactory** et **PropertiesView** pour la gestion des tuiles par rapport à leur environnement et que le rendu du jeu soit plus esthétique. Il y a eu notamment des changements dans la méthode getTileID qui prend maintenant en compte les points cardinaux et affectent différents noms aux tuiles en fonction de leur orientation.

La classe **Impot** est la classe qui permet de calculer les impôts locaux, la TVA et les dépenses pour les hôpitaux, les commissariats et les parcs et jardins, ainsi que les totaux des dépenses et des recettes. Liée à GestionImpots pour pouvoir modifier ces données selon le choix du joueur, elle possède une méthode refreshImpot qui est appelée dans nextState du GameBoard. De cette manière, elle incrémente les ressources courantes du joueur chaque mois.

Dans la classe **RefreshView**, nous avons ajouté un refresh automatique du jeu qui envoie une instruction au GameBoard pour réaliser l'appel de la méthode nextState toutes les deux ou cinq secondes selon la vitesse choisie. Le bouton pause bloque l'appel de nextState.

La modification de **TileUI** concerne la prise en compte de l'environnement par les tuiles : il s'agit de donner les bons arguments à IconFactory pour avoir les points cardinaux.

Nous avons modifié ToolsView pour afficher la barre d'outils du jeu à l'horizontal.

2.2 Choix Techniques

L'idée d'utiliser une matrice à trois dimensions est venue pour gérer indépendamment trois réseaux, voir même quatre puisque nous pensions un moment installer un réseau internet, et ces réseaux devaient à l'origine disposer chacun de leur système de distribution propre. Ce système permettait de traduire chaque besoin par des opérations matricielles et d'obtenir localement l'état d'un réseau. Chaque case peut se décomposer en quatre état : productrice du réseau, relié au réseau et transmettant le réseau - active -, non relié au réseau mais pouvant transmettre - inactive -, et n'intervenant pas dans le réseau - non relié. A chaque tour, toutes les cases reliées au réseau passent à l'état pouvant transmettre, puis chaque case génératrice active les cases transmissibles adjacentes, puis récursivement, chaque case qui arrive à un état actif va activer les cases adjacentes. Ainsi toutes les cases reliés entre elles et à un bâtiment producteur vont s'activer, les autres cases seront inactive ou non relié. Pour savoir si une case est reliée au réseau, il suffira juste qu'elle soit adjacente à une case productrice. Cette méthode nécessite donc la création de bâtiments générateurs : les centrales électriques pour l'électricité, les puits pour l'eau et la mairie pour la route Les bâtiments doivent être reliés à ces trois réseaux pour fonctionner. Pour que ce système fonctionne, il doit y avoir toujours existence de la mairie, c'est pour cela que l'on a décidé de rendre notre mairie indestructible. Par la suite, ces matrices devaient être généralisées au bonheur, à la sécurité, à la santé et à la pollution. Cette fois, au lieu de contenir des énumérations, les matrices auraient eu pour contenu des valeurs, puis à la suite d'opérations matricielles, les valeurs des facteurs aurait été déterminées d'un point de vue local.

Nous avons choisi la forme d'une matrice car la carte est elle-même une matrice.

Nous avons ensuite choisi de jouer avec un timer plutôt qu'au tour par tour. Cela se rapproche davantage d'un SimCityLike de cette manière et permet au joueur de mener ses actions sans penser à refresh le GameBoard. Il doit cependant relancer le timer après avoir ouvert la fenêtre GestionImpot - c'est un problème qui n'a pas pu être corrigé.

2.3 Graphismes

Nous avons choisi de changer les graphismes pour avoir un aspect plus champêtre. Après avoir envoyé une demande aux créateurs de Dust Sprites, nous avons choisi d'utiliser leur bibliothèque Stardew Valley pour créer nos images.

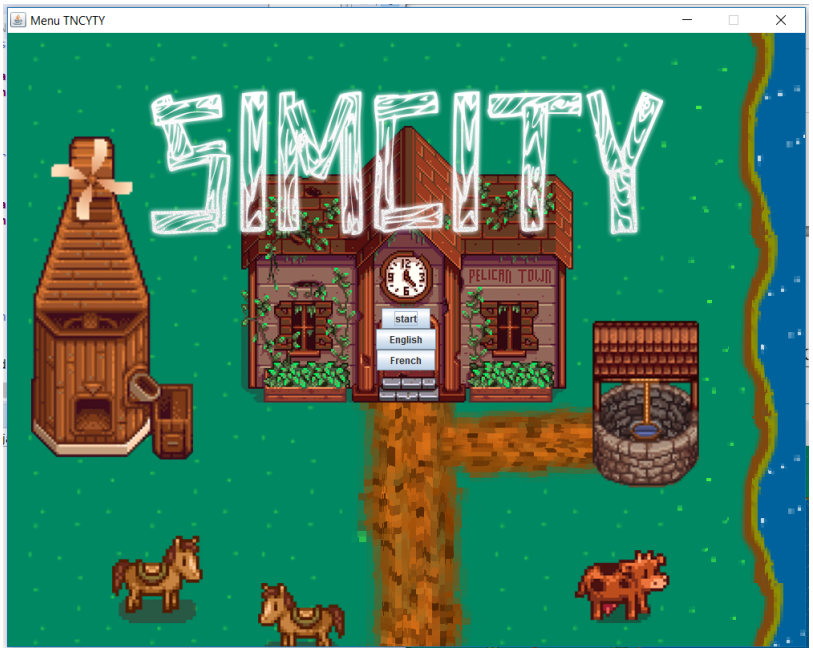


FIGURE 1 : Menu de démarrage

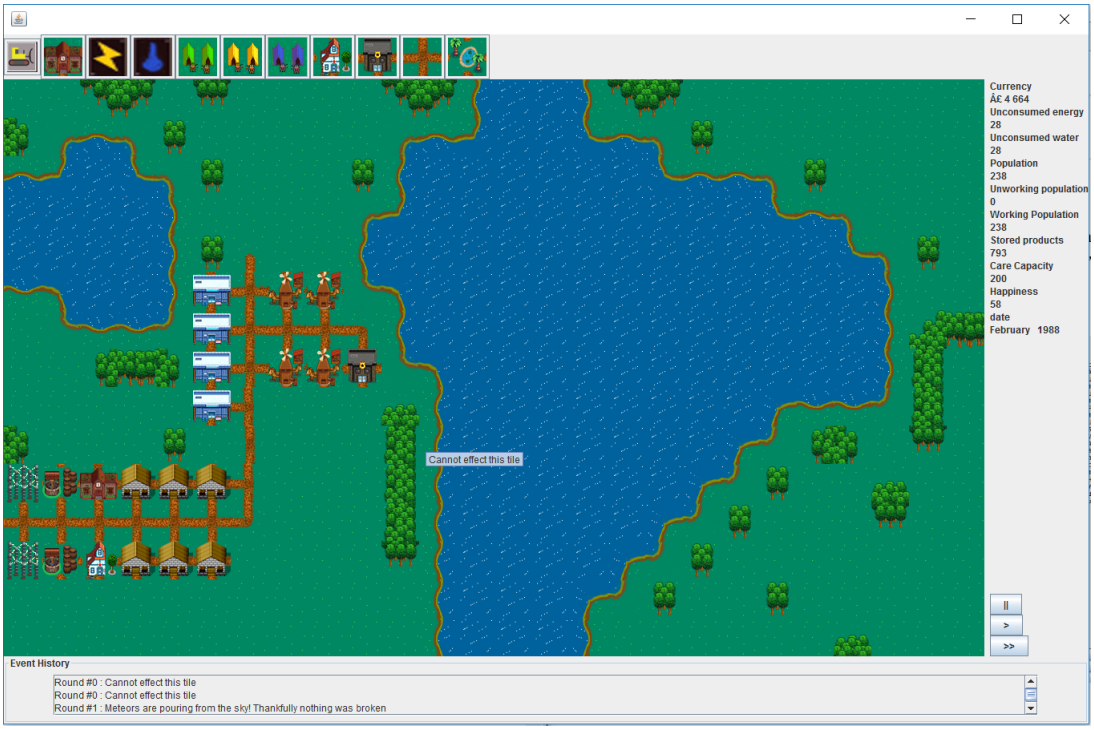


FIGURE 2 : Exemple de GameBoard

3 Les Tests

3.1 Tests Unitaires

Les tests se divisent en deux parties :

— Les premiers testent les fonctions de la matrice MatRes et vérifient si son rafraîchissement correspond bien aux valeurs attendues avec TestMatRes, puis TestTileReseaux vérifie que chaque classe de bâtiments supporte le réseau. Enfin, TestGameboard va vérifier l'intégration au GameBoard en testant si les mises à jour de construction ont bien lieu dans MatRes.

— Les seconds vérifient les classes de bâtiments. Ils vérifient chacun le constructeur, si le bâtiment se construit bien et a les répercussions attendues, si pour que le bâtiment se construise, toutes les conditions sont réunies (accès à la route, à l'électricité, à l'eau), si le bâtiment se démonte correctement, en enlevant les bonus générés, et si le bâtiment a un fonctionnement régulier à chaque tour.

3.2 Tests Fonctionnels

Durant toute la création du jeu, nous avons lancé des tests fonctionnels afin de vérifier que nos implémentations étaient correctes et avaient l'effet recherché pour le joueur. Voici les tests les plus significatifs que nous avons fait :

- La mairie est indestructible : la sélectionner implique l'ouverture de la fenêtre de gestion d'impôt, peu importe l'outil courant => OK

- Test descendre en dessous de 0 en current ressources :
La fenêtre Warning apparaît une première fois. On ne peut pas placer de tuiles=>OK
En revanche, parfois on peut utiliser le bulldozer et d'autres fois non => BUG
Une fois l'argent remonté au-dessus de 0, puis repassé en-dessous, la fenêtre apparaît une nouvelle fois => OK

- Tests hôpitaux/commissariats :
Si l'on essaie d'ouvrir de poser ces tuiles avec une population inférieure à 200 : impossible de poser ces tuiles et ouvertures d'une fenêtre dialog => OK
Dès le seuil de 200 habitants dépassés, ouverture d'une fenêtre annonçant la nécessité de poser ces tuiles et bonheur baisse jusqu'à ce qu'elles soient posées => OK

-Test Résidences, Commerces, Industries : Ces bâtiments doivent être connectés à la route pour fonctionner. Si la route n'est pas connectée, ils n'évoluent pas. Si une des ressources eau ou électricité est manquante, un panonceau apparaît. Dès que le bâtiment est construit : les résidences produisent de la populations, la population travaille dans les usines et les commerces, les usines produisent des biens jusqu'à une certaine limite, les commerces consomment les biens jusqu'à 0 => OK

-Test Routes : Tout bâtiment (hormis les jardins), doit être construit près d'une route pour fonctionner, si une portion de route est coupée entre le bâtiments et la mairie, un puits ou une centrale électrique, le bâtiment arrête de produire et un panonceau affichant le manque d'eau ou d'électricité apparaît => OK

- Test des commerces et des industrie :
Si l'on pose une usine et un commerce et que l'on contrôle leur production et leur consommation, la différence des deux doit donner l'augmentation du nombre de produits stockés => OK

- Test de la population :
Le total de la population est égal à la somme de la population active et de la population au chômage => OK

- Test du bonheur :
Une augmentation significative du montant des impôts, de la TVA, des hôpitaux, des stations

de police, de la population sans travail et des commerces doit impacter significativement le bonheur => OK

4 Gestion de Projet

Pour cette deuxième phase du projet, nous avons commencé par suivre les jalons indiqués par notre diagramme de PERT :

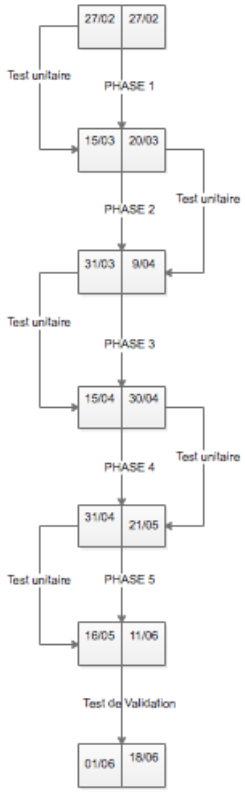


FIGURE 3 : PERT du Projet

La phase 1 regroupait la définition des routes qui constituent un réseau, la définition d'un mode inactif pour les bâtiments en manque d'eau et d'électricité, la définition des zones industrielles, commerciales et résidentielles de différentes densités.

Dans un second temps, nous devions implémenter le timer, la gestion des impôts, la définition des lignes électriques et des canalisations ainsi que la gestion de l'arrivée des habitants.

Ensuite nous devions définir le bonheur et ses impacts ainsi que la santé, la criminalité et l'éducation. Nous souhaitions que le joueur puisse afficher une carte différente pour chacun de ces statistiques (cf Cities Skylines).

Dans un quatrième temps nous devions gérer les catastrophes et les bâtiments pouvant prévenir ces catastrophes : hôpitaux, casernes.

Finalement, nous devions travailler sur des décrets politiques et la possibilité de l'évolution automatique de la densité des zones.

Une erreur dans notre diagramme de PERT dans lequel la date de rendu était prévue beaucoup plus tard, nous a obligé à revoir nos objectifs à la baisse. Nous avons également sous-évaluer le temps de travail sur certains aspects du jeu, et notamment sur tout ce qui concerne la compréhension de l'existent. Nous avons commencé par suivre le PERT sur les premières phases puis nous avons dévié des jalons prévus : tout était planifié trop strictement pour que le PERT soit exactement suivi. La méthode de donner à chacun des objectifs à réaliser seul n'a pas été probante : sur la fin du projet, nous avions des contacts plus réguliers et nous nous entraidions beaucoup plus ce qui nous a permis d'avancer plus vite. Les compte-rendus de nos réunions se trouvent en annexes.

Voici le tableau de répartition des heures effectuées par chaque membre de l'équipe ainsi qu'un tableau spécifiant l'apport de chacun dans les différents package.

		Matta	Guilain	ML	Commun
Phase 1	Total	10h45	6h15	12h20	4h35
Geston de projet	Réunion				16h15
	CR	1h30		4h30	
Rapport	Rapport		2h	7h30	
	Manuel Utilisateur	6h		1h	
	Manuel développeur	5h	3h		
Code	calculMat		12h40		
	dialog			14h	
	launcher	5h			
	localization	20min		10min	
	model	6h	8h	4h	
	model.event	1h			
	model.tiles	10h30	11h20	4h20	
	icones	23h			
	ui	14h		7h	
Tests	tests unitaires		8h		
	tests fonctionnels/ debugging	2h	8h	5h	
	TOTAL	**85h05**	**59h15**	**59h50**	**225h**

FIGURE 4 : Répartition des heures par lots

Nom du Package	Matta	Guilain	ML
calculMat		toutes les classes	
dialog			toutes les classes
launcher	Panneau SimCityUI		
localization	LocalizedText UKTexts FRTexts		LocalizedText UKTexts
model	CityResources GameBoard	CityResources GameBoard	CityResources GameBoard
model.event	Event LuckyYouEvent RiotEvent MeteorEvent		
model.tiles	IndustrialTile CommercialTile ResidentialTile ForestTile ShoreTile WaterTile HospitalTile	BuildableTile Tile WaterPlantTile PublicGardenTile CommercialTile IndustrialTile ResidentialTile	CityHallTile RoadTile PoliceStationTile
ui	InconFactory RefreshView TileUI ToolsView		Impots
Tests		test.model.tiles testCalculMat	

FIGURE 5 : Apport de chaque membre

5 Bilan et Conclusion

Nous avons rencontré plusieurs difficultés au cours du projet.

Tout d'abord, comme nous l'avions soulevé dans la matrice SWOT, nous formons une équipe provenant de classes préparatoires. Nous n'avons donc pas beaucoup d'expérience en terme de code et peu de connaissances et de pratique du Java et de la programmation orientée objet. Cela a donc représenté une difficulté mais ce projet nous a apporté de nouvelles compétences.

En ce qui concerne les difficultés techniques, nous avons voulu créer une classe abstraite PublicServices intermédiaire dont auraient hérité les bâtiments comme les hôpitaux et les commissariats afin de factoriser le code. Nos services publiques devraient implémenter destroyable et hériter de Tile, alors qu'ils héritent actuellement de buildableTile. Nous avons commencé la classe PublicServices très tôt dans le projet mais elle ne nous a pas paru utile alors : nous ne l'avons donc pas utilisée. Nous nous ne sommes pas rendus compte immédiatement qu'elle était vraiment importante car notre modèle est fonctionnel même s'il n'est pas factorisé et manque de clarté. Nous nous en sommes rendus compte dans les

dernières semaines du projet et nous n'avons pas corrigé cette erreur, de peur de générer trop de bugs.

Nous avons compris que aurions dû mieux préparer les différentes classes à implémenter avant de commencer à coder ; nous avions peur de prendre du retard dès le début du projet, mais cela nous aurait permis de gagner en efficacité par la suite. Une des difficultés du projet a été de préparer une gestion de projet sur plusieurs mois sans connaître la quantité de travail qui serait imposée en plus du projet. La gestion du temps à long terme est très complexe.

Certains bâtiments que nous avions prévu d'implémenter n'ont pas été codés, à cause du retard que nous avons pris. Cependant, nous avons passé du temps sur des aspects que nous n'avions pas prévu dans le PERT, en particulier l'interface graphique et la génération aléatoire de cartes. Concernant cette dernière, certains défauts peuvent encore apparaître. Comme ce n'était pas le point le plus important du projet, nous n'avons pas réglé tous les bugs et la carte n'est pas toujours complètement fonctionnelle.

Par rapport à l'utilisation de GitHub, nous avons eu quelques difficultés avec les merges. Cependant, nous aurions dû faire des merges plus réguliers pour éviter de perdre du temps lors des merges manuels.

Pour conclure, ce projet nous a apporté de l'expérience en programmation orienté objet et en java et a souligné l'importance d'un travail préliminaire avant l'implémentation du code. Notre jeu est constitué d'une mécanique assez simple mais qui est équilibrée. Les interractions entre les différentes statistiques auraient pu être plus complexes mais nous trouvons le rendu satisfaisant et le jeu est jouable.

Nous avons eu une très bonne cohésion d'équipe tout au long du projet, ce qui nous a permis de nous entre-aider et les membres étaient tous très motivés.

REFERENCES

Voici les sources que nous avons utilisé lorsque nous rencontrions un problème lors du développement.

[1] https ://docs.oracle.com/javase/8/

[2] http ://bbclone.developpez.com/fr/java/tutoriels/

[3] https ://stackoverflow.com/

[4] https ://www.developpez.net

[5] https ://www.generation-nt.com

[6] http ://www.java2s.com/Tutorial/Java

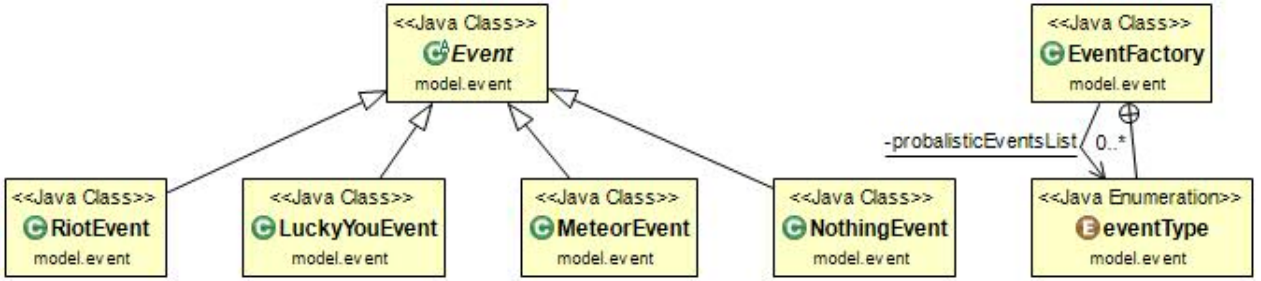
[7] https ://www.jmdoudoux.fr/java/dej/

Pour les images :

[8] http ://stardewvalleywiki.com/Stardew_Valley_Wiki

ANNEXES

1. Diagramme de classe des événements
2. Diagramme de classe du GameBoard
3. Diagramme de classe général
4. Diagramme de classe de MatRes
5. Diagramme de classe de Tiles
6. Diagramme de classe de Tools
7. CR 1
8. CR 2
9. CR 3
10. CR 4
11. CR 5
12. CR 6
13. CR 7
14. CR 8
15. CR 9
16. CR 10
17. CR 11



Events short.jpg Events short.jpg

Diagramme 1 : Events

Diagramme 2 : GameBoard

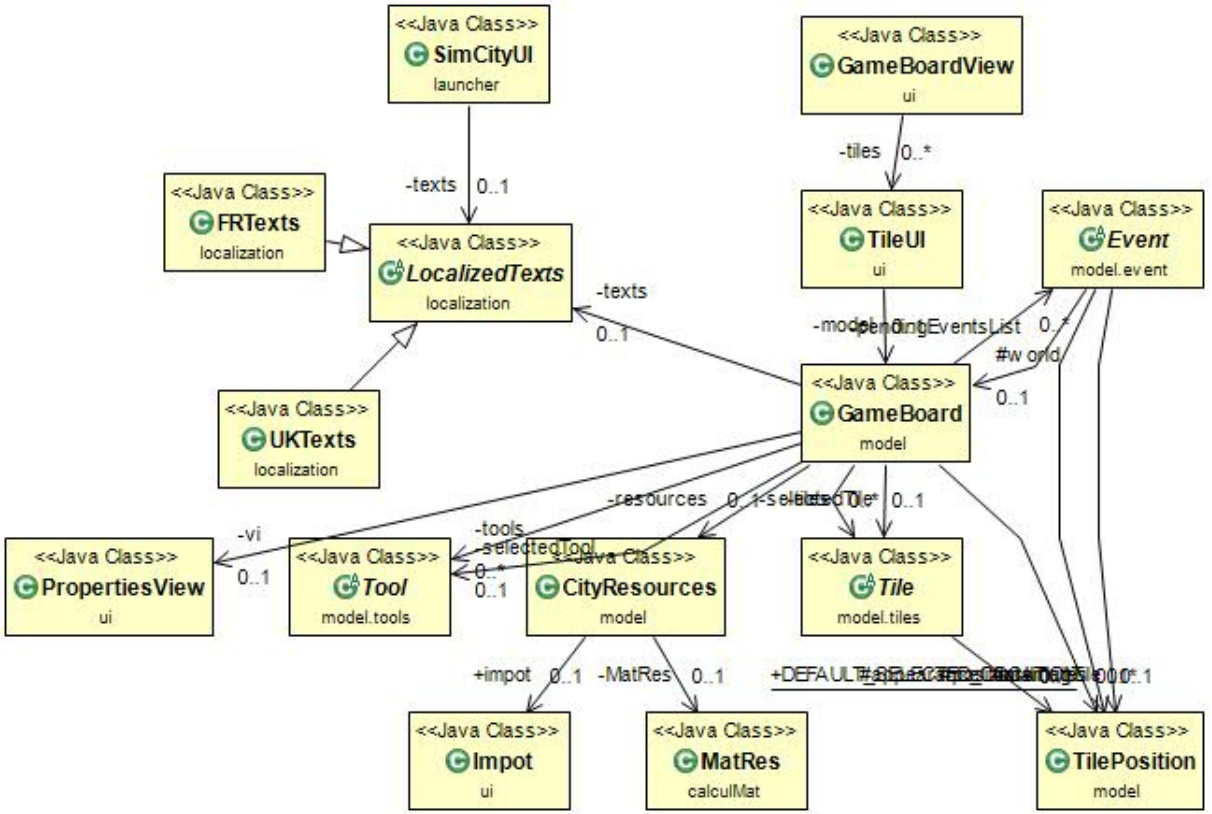


Diagramme 3 : Général

general short.jpg general short.jpg

MatRes short.jpg MatRes short.jpg

Diagramme 4 : MatRes

Diagramme 5 : Tiles

Tools chort 2.jpg Tools chort 2.jpg

Diagramme 6 : Tools

Gestion du projet de POO phase 2: CR 01

Minutes for 28 Fevrier 2017

Present: Matta Karczewski, Marie-lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Lancement de la phase 2	PI
Heure de début	**Durée**
15h50	45 mins

Ordre du Jour

1. Retour sur la soutenance
2. Mise en place du github
3. Répartition des tâches de la phase 1

Résumé

Nous avons effectué une spécification rapide des bâtiments que nous souhaitions proposer lors des phases suivantes.

1. Retour sur la soutenance

Retour globalement positif vis à vis de la soutenance et du contenu que l'on a proposé. Les principales inquiétudes ne reposaient pas dans le contenu que nous proposions mais toujours dans nos capacités de programmation. On ne nous a pas donné de redirection particulière donc nous allons poursuivre dans la voie que nous nous étions fixée dans la ligne globale.
L'équipe s'est avoué soulagée de voir enfin le squelette et l'ampleur de ce qui est déjà programmé.

2. Mise en place du github

Le github a été initialisé pour commencer le travail

2. Répartition des taches de la phase n1

Lors de la phase 1 nous nous attelons à l'élaboration du moteur de jeu : il est nécessaire de coder les zones commerciales et industrielles, peaufiner la zone résidentielle si possible. D'un point de vu graphique pour le moment, permettre le placement de routes.

Décision

-Coder une classe globale bâtiments qui se subdivise en classe étendue zones puis bâtiments spécifiques. - coder les zones industrielles et commerciales - Revoir le code des zones résidentielles - Définir les routes (d'un point de vue graphique pour le moment)

Description	**Responsable**	**Délais**
Se connecter au github créé et installer le squelette du jeu	tous	1 semaine
Tester le jeu pour se familiariser aux mécaniques proposées à l'heure actuelle	tous	1 jours
Coder les zones industrielles et commerciales	Matta et Guilain	1 semaine
Définir les routes (d'un point de vue graphique pour le moment)	Marie-lys	1 semaine

Next Meeting: Mardi 7 Mars

Gestion du projet de POO phase 2: CR 02

Minutes for 28 Fevrier 2017

Present: Matta Karczewski, Marie-lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	Bar
Heure de début	**Durée**
12h05	25 mins

Ordre du Jour

1. Retour sur le travail effectué
2. Discussion sur l'implémentation des routes

1. Retour sur le travail effectué

L'équipe a rencontré des difficultés à lancer le jeu sur les différentes machines, en fonction des environnements utilisés. Il y a eu aussi beaucoup de lecture de code afin de comprendre son organisation et son fonctionnement.
Matta a créé les classes zone industrielle et zone commerciale avec leur implémentation graphique. Il ne leur a pas encore donné de spécifications propres.

2. Implémentation des routes

Discussion sur la façon de lier les routes aux bâtiments à l'aide de contrainte : une solution serait de donner à chaque instance de la classe route un booléen : pour qu'il soit vrai, il doit être placé près d'une autre instance route dont le booléen est vrai. Un bâtiment ne pourrait être placé que près d'une route au booléen vrai. La première route posée serait initialisée à vrai.
Nous n'avons pas encore bien compris les structures de données utilisées pour créer la carte de jeu.

Décision

-Commencer à penser aux tests de cette première partie -Coder les routes, leur implémentation graphique -Définir les caractéristiques des zones industrielles et commerciales -Le chef de projet a déterminé que nous pouvions prendre trois jours de plus pour finir cette partie

Description	**Responsable**	**Délais**
Réfléchir aux tests	Guilain	10 jours
Implémenter les routes de façon graphique et leur associer un booléen	ML	10 jours
Coder les zones industrielles et commerciales	Matta	10 jours

Next Meeting: Vendredi 10 Mars

Gestion du projet de POO phase 2: CR 03

Minutes for 10 Mars 2017

Present: Matta Karczewski, Marie-Lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	PI
Heure de début	**Durée**
16h10	40 mins

Ordre du Jour

1. Retour sur le travail effectué

1. Retour sur le travail effectué

La classe Road a été implémentée et les images ont été créées pour les différentes étapes de construction.
Les booléens restent à implémenter.
Les zones industrielles et commerciales ont été implémentées.
Guilain a commencé à regarder pour les tests.

Décision

Il faudrait implémenter un timer pour éviter de refresh à chaque tour et avoir un jeu plus semblable aux SimCityLike.
Il faut coder le réseau routier : nous avons décidé de rendre une mairie indispensable : elle serait le point central du réseau routier. Si une route n'est pas raccordée à la mairie, les bâtiments posés au bord de cette route ne pourra pas évoluer. Marie-Lys va implémenter la classe CityHall et Guilain va coder le réseau routier.

Description	**Responsable**	**Délais**
Coder le réseau routier	Guilain	18 jours
Implémenter CityHall	ML	18 jours
Timer	Matta	18 jours

Next Meeting: Mardi 28 Mars

Gestion du projet de POO phase 2: CR 04

Minutes for 28 Mars 2017

Present: Matta Karczewski, Marie-Lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	PI
Heure de début	**Durée**
16h10	40 mins

Ordre du Jour

1. Bibliothèque graphique pour les icônes
2. Retour sur le travail de Matta timer dans le jeu
3. Retour sur le travail de ML et Guilain

1. Bibliothèque graphique pour les icones

Nous avons fait le choix d'utiliser la bibliothèque des Sprites de Stardew valley (nous avons contacté les auteurs pour les questions de droits et attendons une réponse). La bibliothèque propose un aspect plus rural du jeu que nous devrons prendre en compte.

2. Retour sur le travail de Matta timer dans le jeu

Le jeu possède désormais plusieurs vitesses (le jeu s'update toute les 2 ou 5 secondes selon la vitesse). Les vitesses sont calibrables dans le jeu où 3 boutons (vitesses 1, 2 et pause) sont présents et "clickables".
3. Retour sur le travail de ML Guilain
Guilain a codé un package calculMat qui implémente une matrice donnant des informations sur les réseau électrique, d'eau...
Marie-lys a codé une classe "CityHall" et les instances permettant de la placer. Le bâtiment ne peut être posé qu'une seule fois dans le jeu

Décision

-ML va coder l'apparition de la fenêtre de la mairie et des outils
-ML va voir pour la gestion des impôts
-Matta va créer un bâtiment similaire aux centrales pour l'eau
-Guilain va continuer de travailler sur la matrice et les problèmes d'informations protégées.
-Matta va procéder à la création d'autres bâtiments dans le style de celui du "townhall" et de ligne électrique "flottante"

Description	**Responsable**	**Délais**
Débbuguer la matrice matRes	Guilain	27 jours
Coder l'apparition des fenêtres et les outils, commencer à coder la gestion des impôts	ML	27 jours
Créer d'autres bâtiments publics et les lignes électriques	Matta	27 jours

Next Meeting: 24 Avril

Gestion du projet de POO phase 2: CR 05

Minutes for 24 avril 2017

Present: Matta Karczewski, Marie-Lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	PI
Heure de début	**Durée**
18h	45 mins

Ordre du Jour

1. Retour sur le travail effectué

1. Retour sur le travail effectué

Matta a changé les tuiles de terrain, pour donner un aspect plus esthétique au jeu. Il a également implémenter un menu, qui permettra de choisir la langue et le niveau de difficulté.
La matrice matRes fonctionne et Guilain a adapté toutes les classes de bâtiments concernées. Il a également codé la classe pour les puits. Guilain a également implémenté une classe PublicServices pour factoriser le code et être la classe mère de nos futurs services publics.
La fenêtre pour la gestion des Impôts est codée et s'ouvre quand on clique sur la mairie, mais elle n'est pas encore relié aux impôts qu'il faut implémenter.

Décision

ML doit finir la fenêtre Gestion Impôts et doit implémenter la classe concernant les impôts.
Matta doit afficher la date pour le timer et harmoniser les graphismes pour les bâtiments, signaler les manque d'eau et d'électricité.
Guilain doit coder les lignes à haute tension et poursuivre l'adaptation de matRes au reste du jeu.

Description	**Responsable**	**Délais**
Débbuguer la matrice matRes	Guilain	10 jours
Coder les impôts	ML	10 jours
Affichage du timer et harminisation des graphismes	Matta	10 jours

Next Meeting: 4 Mai

Gestion du projet de POO phase 2: CR 06

Minutes for 4 mai 2017

Present: Matta Karczewski, Marie-Lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	PI
Heure de début	**Durée**
14h	3h

Ordre du Jour

1. Retour sur le travail effectué

1. Retour sur le travail effectué

ML a implémenté la classe Impots et a fini la présentation de la fenêtre Gestion Impots.
Matta a affiché le timer et changer les icônes de tous les bâtiments et créé celles qui apparaissent en cas de manque d'eau ou d'électricité.
Guilain a terminé la matRes et son adaptation au jeu ainsi que la classe des lignes à haute tension : elles n'ont pas encore de graphisme.

Décision

Nous avons fait le premier merge complet : beaucoup de difficulté.
Guilain doit commencer les tests sur les différentes classes que nous avons créé. ML doit poursuivre le travail sur les impôts et commencer le rapport. Matta doit changer les routes pour qu'elles changent en fonction de l'environnement.

Description	**Responsable**	**Délais**
Adapatation des routes	Matta	11 jours
Poursuivre le travail sur les impôts. Commencer le rapport	ML	11 jours
Commencer les tests des différentes classes	Guilain	11 jours

Next Meeting: 15 Mai

Gestion du projet de POO phase 2: CR 07

Minutes for 15 mai 2017

Present: Matta Karczewski, Marie-Lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	PI
Heure de début	**Durée**
18h	30mins

Ordre du Jour

1. Retour sur le travail effectué

1. Retour sur le travail effectué

Matta est parvenu à adapter les routes en fonction de la proximité des bâtiments : elles ne sont plus rectilignes, elles se rejoignent et s'orientent.
ML a fait le calcul des impôts pour les impôts locaux, TVA et hôpitaux.
Guilain a avancé sur les tests de classe.

Décision

ML doit poursuivre les ajustements de la fenêtre des impôts. Elle doit voir M Da Silva pour obtenir des informations sur le contenu attendu.
Matta doit implémenter les hôpitaux.
Guilain doit avancer encore sur les tests.

Description	**Responsable**	**Délais**
Implémenter les hôpitaux	Matta	2 jours
Finaliser code impôts et voir M Da Silva	ML	2 jours
Poursuivre les tests	Guilain	2 jours

Next Meeting: 17 Mai

Gestion du projet de POO phase 2: CR 08

Minutes for 17 mai 2017

Present: Matta Karczewski, Marie-Lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	E.1.21
Heure de début	**Durée**
14h	2heures

Ordre du Jour

1. Retour sur le travail effectué

1. Retour sur le travail effectué

Matta a terminé les hôpitaux, mais il manque des ajustements pour impacter sur la consommation des habitants. Il a commencé le manuel utilisateur.
Les impôts sont fonctionnels : le joueur peut choisir ses taux d'imposition. La structure du rapport est faite et disponible sur Latex. Il faut penser à faire un manuel développeur à partir de la JavaDoc et des tests fonctionnels.
Guilain a avancé dans les tests unitaires.

Décision

Nous avons commencé la réunion en faisant des merges.
ML doit faire les derniers ajustements sur les impôts et faire en sorte que la fenêtre de gestionImpôts arrête le timer.
Matta doit implémenter les hôpitaux et finir le manuel utilisateur.
Guilain doit avancer encore sur les tests.

Description	**Responsable**	**Délais**
Manuel utilisateur à terminer et finaliser les hôpitaux	Matta	2 jours
Finaliser code impôts et poursuivre le rapport	ML	2 jours
Poursuivre les tests	Guilain	2 jours

Next Meeting: 19 Mai

Gestion du projet de POO phase 2: CR 09

Minutes for 19 mai 2017

Present: Matta Karczewski, Marie-Lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	E.1.21
Heure de début	**Durée**
16h	2h

Ordre du Jour

1. Retour sur le travail effectué

1. Retour sur le travail effectué

Matta a terminé les hôpitaux et le manuel utilisateur.
Les impôts sont finalisés : l'ouverture de la fenêtre gestionImpôt arrête le timer : ne le remet pas en route à la validation cependant.
Guilain a avancé dans les tests unitaires.

Décision

On a transféré impôts dans CityRessources au cours de la réunion. Nous avons relié les hôpitaux aux impôts. Nous avons revu toutes les classes que nous avons modifié au cours du projet pour remplir la partie arhitecture du rapport et le compte d'heures.
ML doit faire la classe commissariat et le raccorder aux Impôts.
Matta doit créer des événements : émeutes et chance, ainsi que voir pour une map random.
Guilain doit implémenter le bonheur et le raccorder à la santé et à la consommation.

Description	**Responsable**	**Délais**
Créer les événements et la random map	Matta	3 jours
Doit faire la classe commissariat.	ML	3 jours
Implémentation du bonheur	Guilain	3 jours

Next Meeting: 22 Mai

Gestion du projet de POO phase 2: CR 10

Minutes for 22 mai 2017

Present: Matta Karczewski, Marie-Lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	Local TNS
Heure de début	**Durée**
19h15	2h30

Ordre du Jour

1. Retour sur le travail effectué

1. Retour sur le travail effectué

Matta a terminé l'implémentation de la carte random. Les événements des émeutes et de chance sont fonctionnels : doivent être reliés au bonheur.
Guilain a implémenté le bonheur influançant la consommation. Il est influencé par la population, les impôts,santé, le chômage et la consommation.
Les commissariats sont implémentés sur le modèle des hôpitaux. Des fenêtres s'ouvrent lorsque les événements se produisent.

Décision

Nous avons commencé par faire des merges et régler quelques bugs, notamment l'arrêt du timer à chaque tic, provenant des fenêtres des événements.
Guilain doit implémenter un parc pour améliorer le bonheur et relier le bonheur à la sécurité. Explication des différents tests dans le rapport. Explications sur la matrice matRes.
Matta doit régler bug graphique et soucis d'affichage. Les connecteurs haute tension et eau doivent avoir leur graphisme. Changer les boutons du menu pour la langue et la difficulté. Rajouter les derniers éléments au manuel utilisateur.
ML doit régler l'affichage des labels dans les fenêtres événements. Mise au propre du rapport, relecture des CR, explication des différentes classes et modifications.

Description	**Responsable**	**Délais**
Régler bugs de la map, faire les gaphismes des lignes hautes tensions Ajouter bouton au menu, finaliser le manuel utilisateur	Matta	4 jours
Régler l'affichage des labels des fenêtres événements, mettre le rapport au propre Relire les CR, faire les explications des classes et des modifications	ML	4 jours
Implémenter un Parc et remplir le rapport	Guilain	4 jours

Next Meeting: Vendredi 26 Mai

Gestion du projet de POO phase 2: CR 11

Minutes for 26 mai 2017

Present: Matta Karczewski, Marie-Lys Rousseau, Guilain Leduc

Motif de la réunion	**Lieu :**
Réunion d'avancement	Salle PI
Heure de début	**Durée**
14h	3h

Ordre du Jour

1. Retour sur le travail effectué
2. Merge global
3. Finalisation du projet et dernières missions

1. Retour sur le travail effectué

Guilain : Implémentation des parcs et jardins, suppression des connecteurs et lignes à hautes tensions. Un bug détecté dans la matrice a été résolue ainsi. A rempli une partie du rapport.
Matta : Rajout de la JavaDoc, relie les riots au bonheur, création d'un événement météorites, règle la plupart de problèmes d'affichage de la carte.
ML : j'ai relu tous les CR, j'ai rajouté javadoc, commencé les explications des différentes classes dans le rapport. Régler les affichages dans les fenêtres événements.

2. Merge global

Le merge a mis en évidence un soucis avec le bonheur qui ne monte pas au dessus de 75% et ne descend pas en dessous de 25%. De plus les tests ne marchent plus, il faut les adapter aux modifications.

3. Finalisation

Création du diagramme de classe Guilain. Nous avons choisi les prix de nos tuiles et la somme de départ du jeu en comparant les éléments à ceux de SimCity2000. Réparation des Tests Explication des différentes classes

A faire

Description	**Responsable**	**Délais**
JavaDoc des fonctions restantes et finaliser le manuel utilisateur Rendre le projet	Matta	1 jour
Mettre le rapport au propre	ML	1 jour
Génère javadoc, ajusterl le bonheur, préciser les tests dans la javadoc	Guilain	1 jour
Relire le rapport et faire des tests fonctionnels	Tout le monde	1 jour

Next Meeting: Pas de prochaine réunion