Julien Le Goff
Florian Pineau

Baptiste Samoyault
Sébastien Schaan

Compte rendu

WOTAH

Introduction

Qu'est-ce que le jeu **Wotah** ? c'est un jeu qui est réalisé dans le style StarCraft et de Galèrapagos.

StarCraft est un jeu de stratégie en temps réel qui consiste à contrôler des unités pour récolter des ressources, construire des bâtiments et conquérir le territoire ennemi.

Capture d'écran de StarCraft avec différents bâtiments et unités.

Galèrapagos quant à lui est un jeu de société qui demande aux joueurs se retrouvent de coopérer pour récupérer assez de ressources afin de s'enfuir de l'ile où ils se trouvent naufragés.

Plateau de jeu de Galèrapagos.

Notre jeu a pour base la coopération des joueurs, les joueurs choisissent les actions de leur personnage sur une Grille 2D via un système de Drag and Drop à la souris, ils incarnent des naufragés

et doivent planter et récolter des ressources à la fois pour survivre (eau, nourriture, bois) mais aussi mais pour s'échapper de l'île avant la fin du temps imparti, car pendant que les joueurs jouent, une tempête arrive ! (Game Over évidemment si les conditions de victoire ne sont pas remplies).

Capture d'écran de Wotah avec un joueur, et différentes ressources.

Analyse globale

Voici des fonctionnalités que l'on va retrouver dans le projet.

Pour la partie Interface Graphique :

- Affichage de la grille de l'île
- Affichage des joueurs (naufragés) sur ladite grille
- Affichage de zones spécifique sur l'île (Sol, Buisson, Arbre, Eau, Mer, Epave, Rocher)
- Affichage des ressources sur la grille
- Affichage des zones

Pour la partie Génération de ressources :

- Planter d'arbres :
    - Commande via Drag and drop qui permet de replanter un arbre après avoir récolté la ressource des arbres.
- Planter graine de buisson :
    - Commande via Drag and drop qui permet de replanter un buisson après avoir récolté la ressource des buissons.
- Pousse des arbres et des buissons.

Pour la partie Collecte de ressources :

- Stockage des personnages
- Taux de drop des ressources et taux de drop des pousses
- Récolte des ressources

Pour la partie Commande

- Drag and Drop (saisie à la souris) -> ouverture d'un menu pour le joueur
- Le déplacement des personnages (path finding)

- Saisie et plantation des ressources
- Vitesse de déplacement
- Actions réalisables sur les ennemis

Le système de drag and drop du personnage a été le plus compliqué à réaliser, de par l'affichage du personnage à faire et de sa fluidité.

Les Catégories Interface Graphique et Commandes sont essentielles pour le bon fonctionnement du jeu.

Plan de développement

Tableau de temps de travail estimé :

Fonctionnalité	Analyse	Conception	Développement et test
La grille	45 minutes	1h	2h
Stockage des zones	1h	1h	3h
Les personnages (création)	1h	30 minutes	20 minutes
Stockage des personnages	10 minutes	20 minutes	10 minutes
Les Zones	2h	3h	4h
Les personnages (affichage)	1h	1h	1h
Les ressources	15 minutes	1h	30 minutes
Timer des actions	2h	2h	4h
Timer de la tempête	1h	2h	2h
Drag and Drop	4h	4h	10h
Menu des actions	3h	1h	4h
Récolter	1h	30 minutes	30 minutes
Stockage ressources	30 minutes	25 minutes	1h
Taux de drop des graines	5 minutes	10 minutes	10 minutes
Planter	30 minutes	30 minutes	1h
Déplacement	2h	1h	4h
Pousse	1h30	2h	3h
Victoire	10 minutes	1h	30 minutes
Défaite	20 minutes	2h	1h
Pause	45 mins	1h	3h++

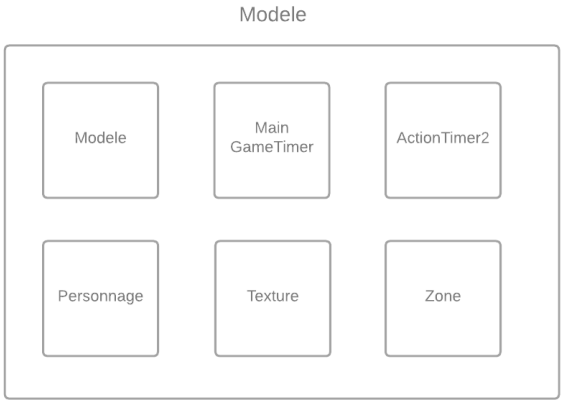
Le système de drag et drop pour décider d'une zone ou un personnage doit effectuer une tâche est la chose la plus compliqué du projet, associé à ceci, le path finding pour les déplacements des personnes regroupent ce qu'il y a de plus lourd.

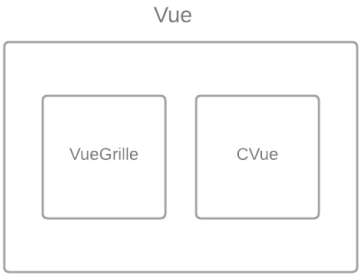
Le diagramme de Gantt avec les différentes affectations aux membres de l'équipe est disponible sur le Github.

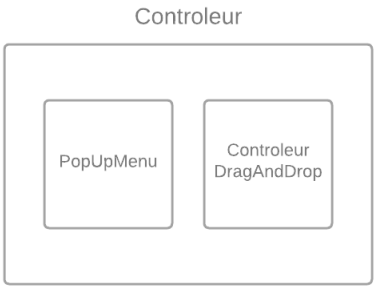
Conception générale

Pour ce projet nous utilisons le modèle MVC pour l'interface graphique.

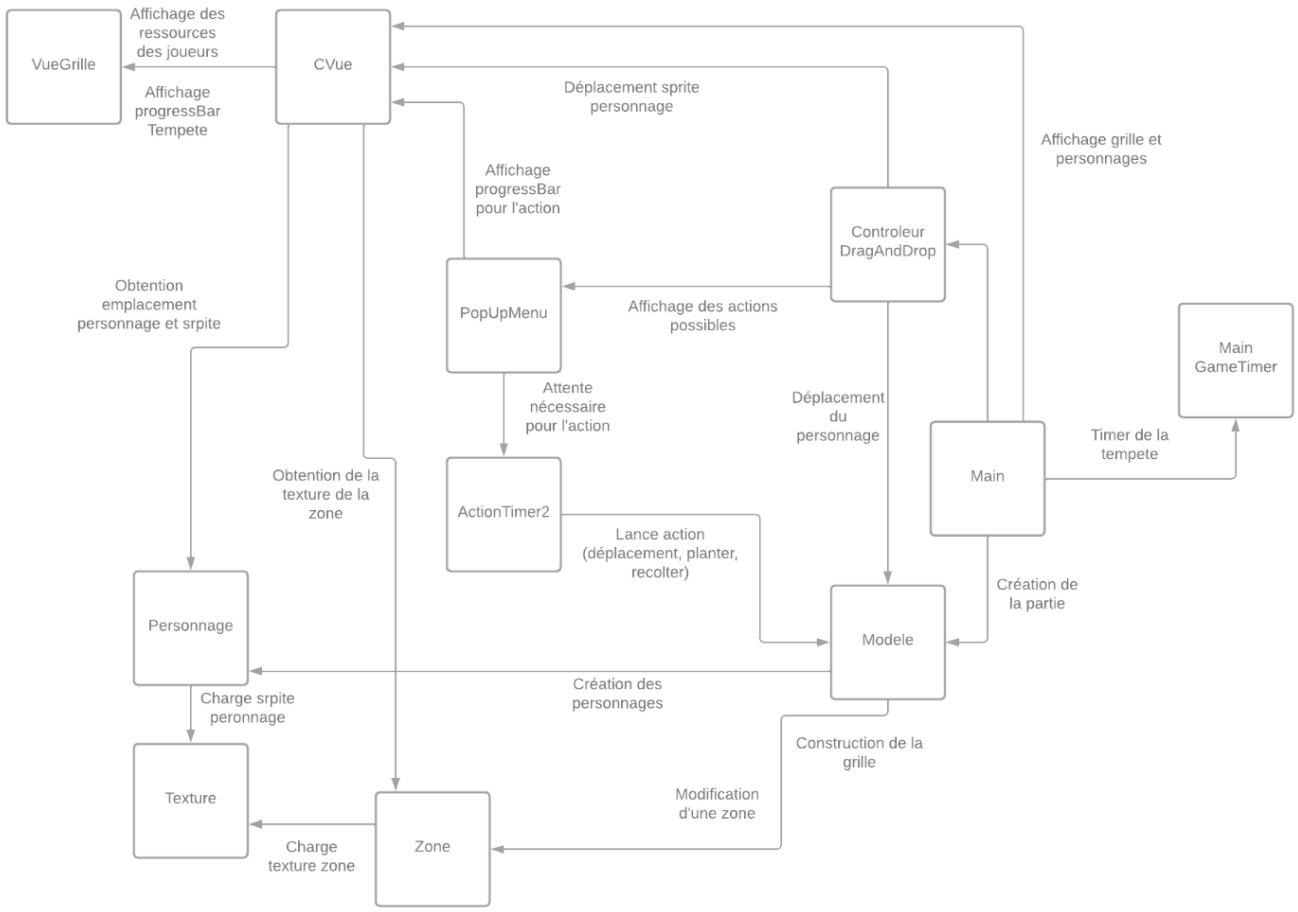
Les différentes classes sont reparties comme tel :







La conception générale est comme ceci :



Conception détaillée

Création de la partie

La grille

Les variables utilisées sont :

- « hauteur » et « largeur » pour définir la taille de l'Ile
- « hauteurEpave » « largeurEpave » pour définir l'emplacement de l'épave
- « zones » pour le stockages des différentes zones

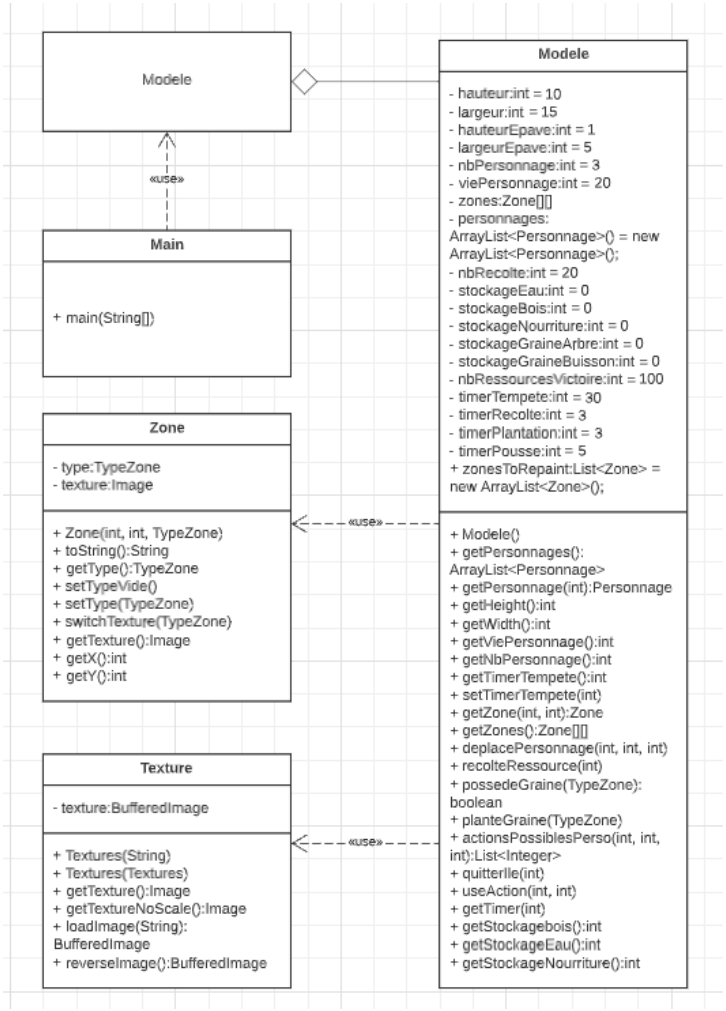
On utilise le type énumérée « TypeZone » présent dans la classe « Texture ». Ce type peut se voir rajouter de nouvelles zones.

Le « Main » appelle le « Modele » lors d'une partie ce qui génère la grille dans le constructeur de la classe « Modele », on utilise une part d'aléatoire pour l'emplacement des différentes ressources. L'ile est arrondie sur les côtés gauches on l'on y défini du sable comme type de zone. On évite également

qu'un rocher se retrouve sur le point de spawn des joueurs (celui-ci se trouve une case à droite de l'épave). Chaque case de notre grille est une zone de la classe « Zone ».

Stockage des zones

Le stockage des zones dans la variable « zones » crée une grille de « Zone » puisqu'il s'agit d'un tableau à deux dimensions.



Les personnages

Les variables utilisées sont :

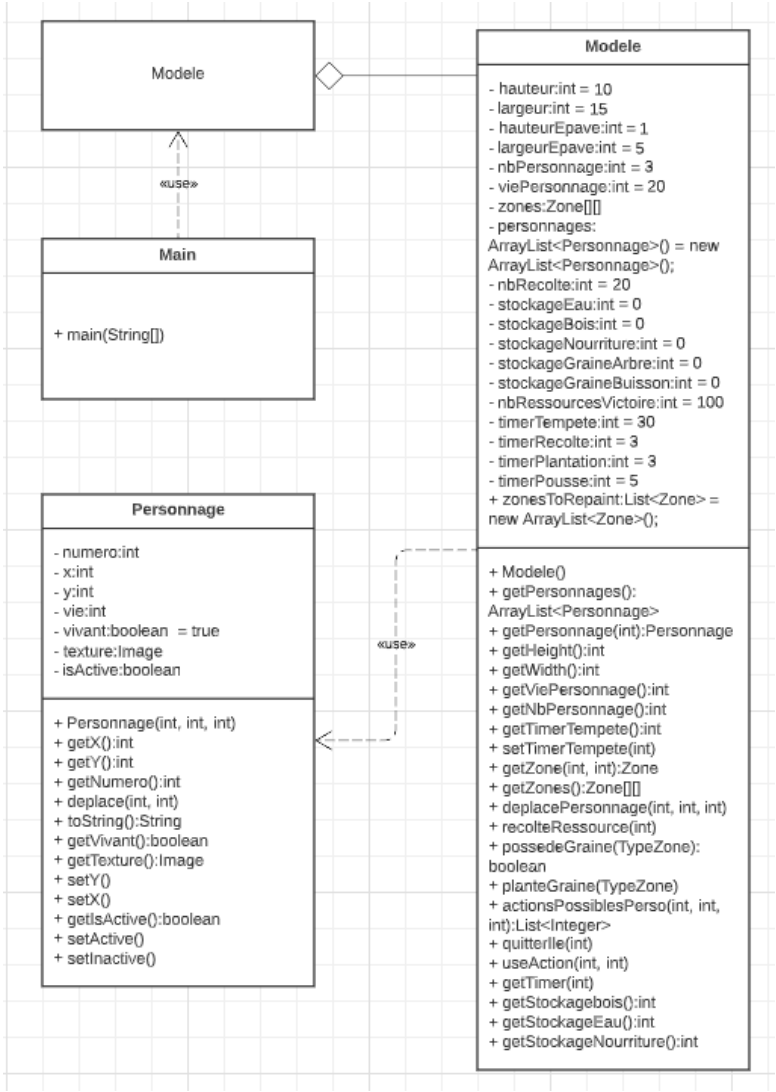
- « nbPersonnages » pour le nombre de personnages
- « viePersonnage » pour la vie d'un personnage (fonctionnalité non implantée)

Lors de la création du « Modele » par le « Main », les personnages sont générés dans le constructeur de « Modele », leur emplacement de spawn se trouve une case à droite de l'épave.

Stockage des personnages

Les personnages sont stockés dans la variable « personnages » qui est une liste de « Personnage ».

Les emplacement des personnages sont uniquement des attributs de « Personnage ».



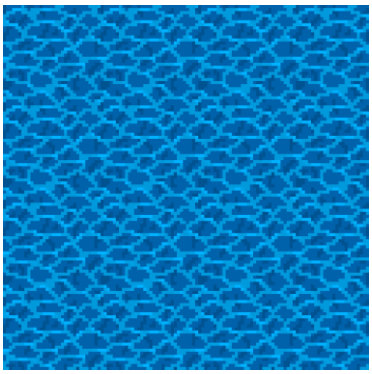
L'affichage

L'affichage en générale a été produit avec Swing et ses différentes classes.

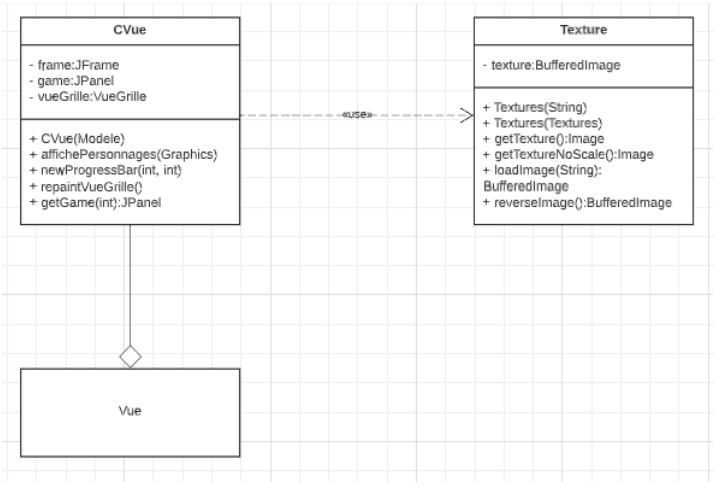
Le background de l'ile c'est-à-dire la mer est géré dans le constructeur de la classe « CVue » en récupérant la « texture » dans la classe « Texture ».

Julien Le Goff
Florian Pineau

Baptiste Samoyault
Sébastien Schaan



Sprite utilisé pour le background.

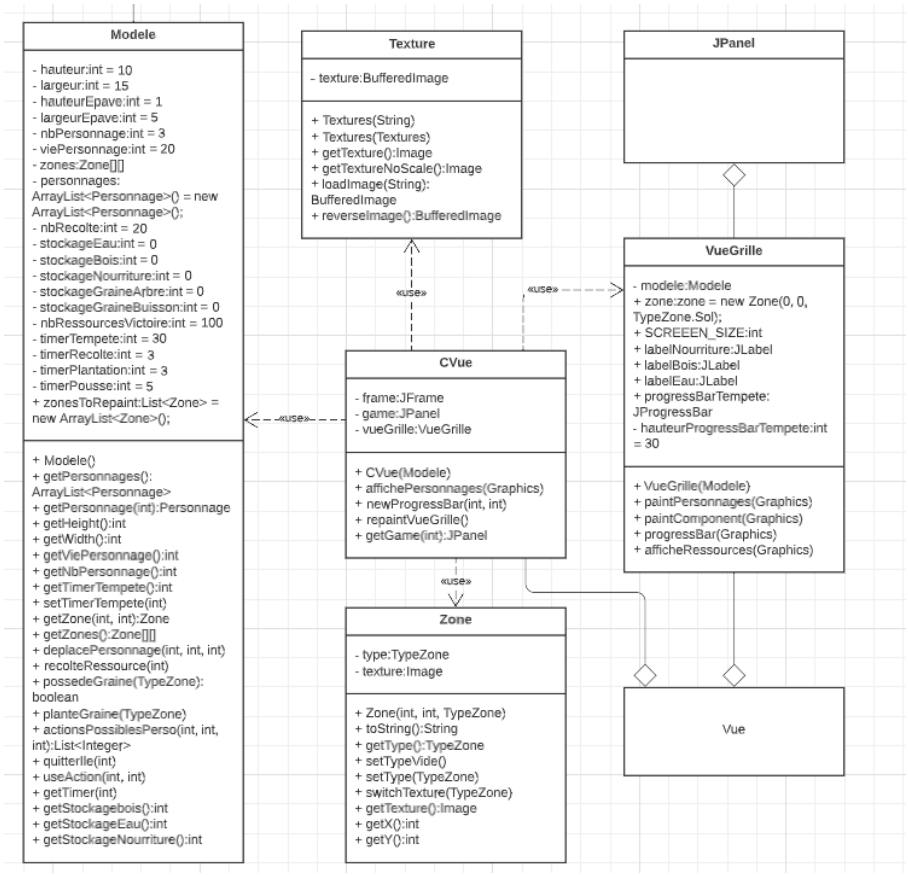


Les zones

Les variables sont « size » qui définir la taille d'une zone et « zonesToRepaint » (dans Modele) la liste des zones à réafficher.

L'affichage des zones et donc de l'Ile est fait dans la classe « Vuegrille » avec la fonction « paintComponent » qui explore et affiche la « texture » des différentes zones à afficher. Cette fonction est appelée par « repaintVueGrille » dans la classe « CVue » qui est ensuite appelé a chaque fois qu'une modification de l'affichage a été fait.

La classe « Modele » gère les zones à ajouter dans « zonesToRepaint ».

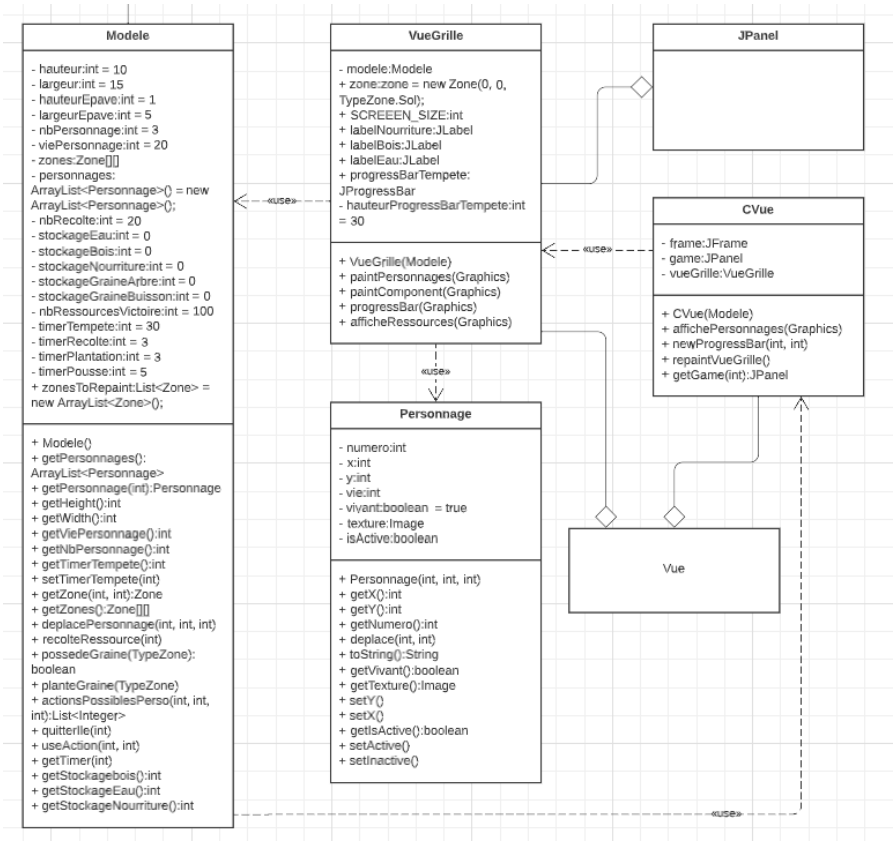


Les différentes zones crées pour l'affichage.

Les Personnages

La constante utilisée ici est « size » la taille d'une zone, pour faire correspondre le personnage avec le reste de la grille. Quant aux variables, on utilise « personnages » pour obtenir les différents personnages ainsi que la « texture » du personnage et sa position.

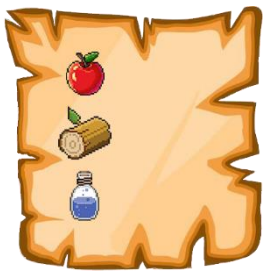
L'affichage des personnages est réalisé avec la fonction « paintPersonnage » dans la classe « VueGrille ». Cette fonction est appelée par « paintComponent » (dans VueGrille également) qui est à son tour appelée par « repaintVueGrille » dans « CVue » qui est par la suite appelé par différentes classes à chaque fois que l'on a modifié quelque chose à afficher.



Les ressources

L'affichage des différentes ressources des personnages est fait dans « VueGrille » avec la fonction « afficheRessources ».

Cette fonction affiche une feuille avec les trois ressources, ainsi que trois labels pour les différentes quantités.



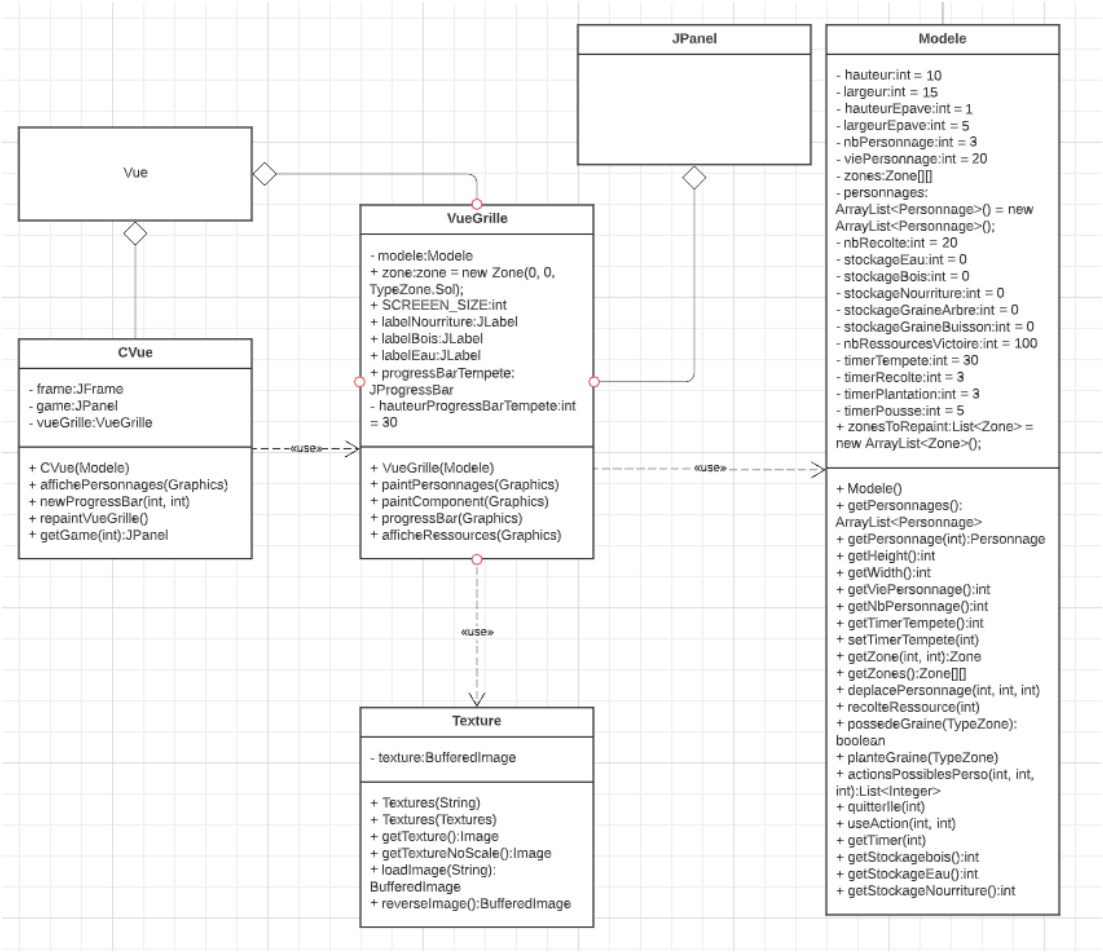
La feuille servant de support pour afficher les ressources.

Les variables utilisées ici sont « SCREEN_SIZE » la taille de l'écran de l'ordinateur (dans la classe VueGrille) et « sprite_sheet » qui est la texture de la feuille se trouvant dans la classe Texture.

On utilise pour la récupéré la fonction « getTextureNoScale » (dans Texture) car on ne souhaite pas que sa taille soit dimensionnée par celle d'une case.

On accède aussi aux getter « getStockageNourriture », « getStockageBois » et « getStockageEau » (de Modele) pour les quantités des ressources.

La fonction « afficheRessources » est appelée par le « paintComponent » de « VueGrille » qui est lui-même appelé avec la fonction « repaintVueGrille » dans « CVue » qui est par la suite appelé à chaque fois que l'on souhaite recharger l'affichage.



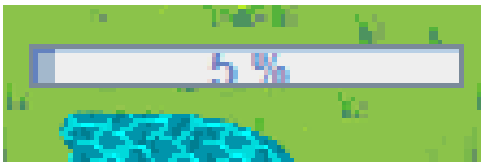
Timer des actions

Les variables utilisées sont « SCREEN_SIZE » de « VueGrille » pour l'emplacement du timer.

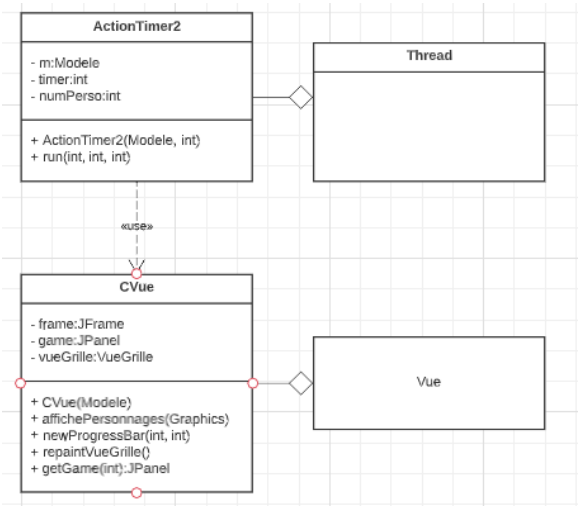
La classe Thread a été utilisé pour géré le temps d'exécution sans bloquer les autres personnages.

Cette partie se trouve dans la classe « CVue », plus exactement la fonction « newProgressBar ». Cette fonction crée une progress bar à usage unique qui est affiché à l'emplacement qu'on lui définit avec les paramètres x et y. Un thread est utilisé pour incrémenter à la barre du temps pendant le temps nécessaire et la supprime ensuite.

C'est la classe « ActionTimer2 » qui l'appelle a chaque fois qu'une action va être lancée.



Un exemple de timer qui apparait lors du lancement d'une action.



Timer de la tempête

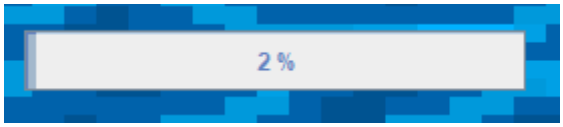
Le timer de la tempête se trouve dans la classe « VueGrille », c'est la fonction « progressBar » qui permet de lui incrémenter du temps.

On utilise la classe Thread ici également pour les mêmes raisons que pour le timer des actions.

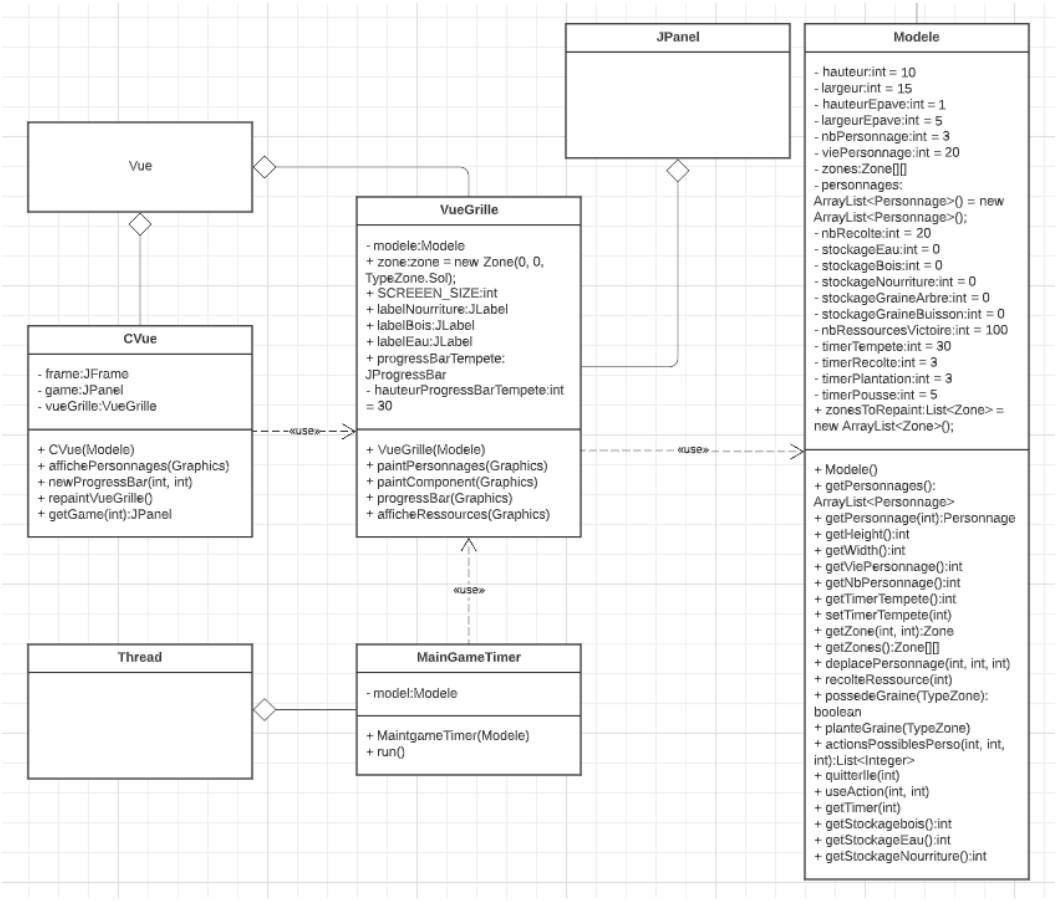
« progressBar » est appelé dans « paintComponent » (dans VueGrille) qui est lui-même appelé par « repaintVueGrille » (dans CVue), cette fonction est appelée pour chaque rechargement de l'affichage.

C'est la classe « MainGameTimer » qui appelle notre fonction dans « run ».

On utilise la variable « SCREEN_SIZE » (dans VueGrille) pour définir l'emplacement de la barre.



La timer de la tempête.



Les contrôles

Drag and Drop

Les variables utilisées sont :

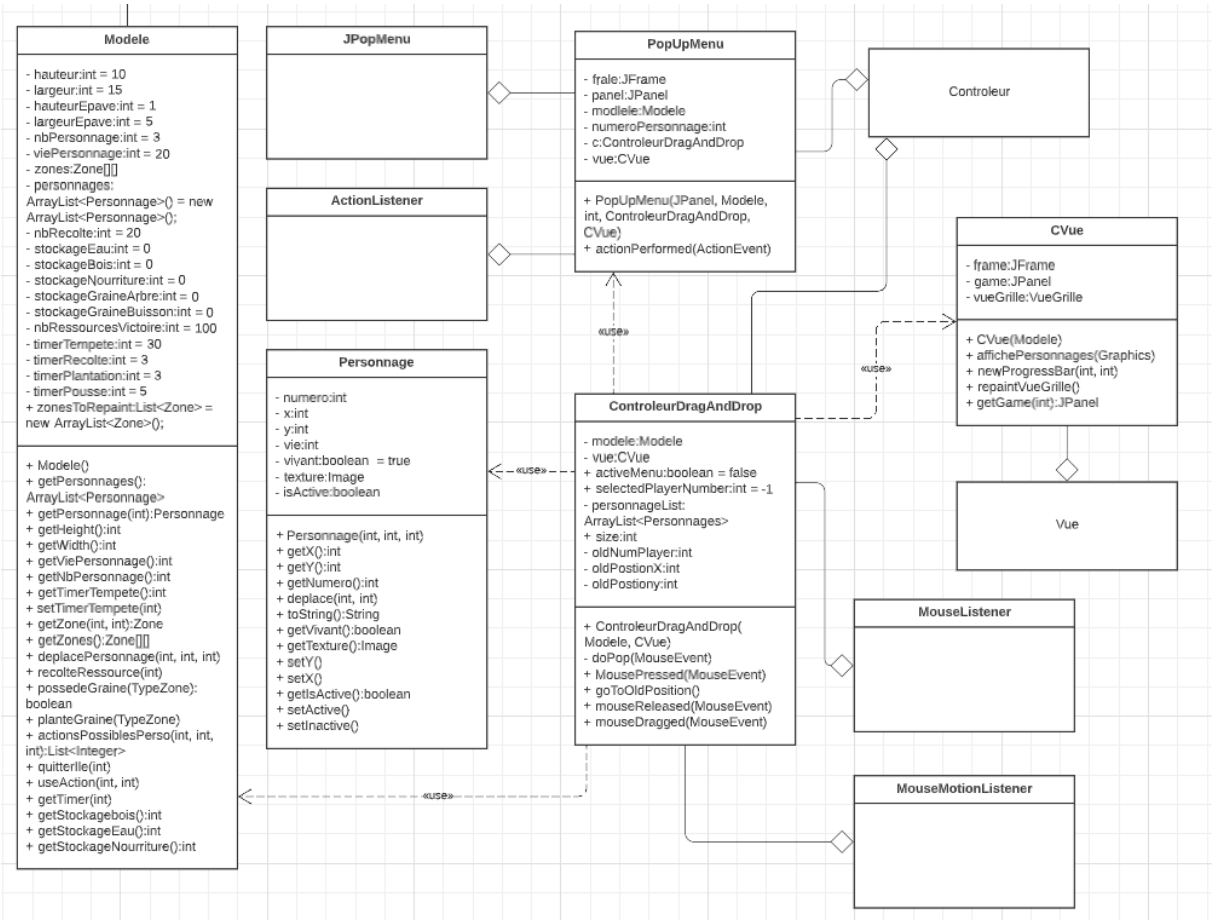
- « selectedPlayerNumber » pour désigner le numéro du personnage sélectionné
- « activeMenu » pour savoir si le menu d'action est ouver ou non
- « size » pour connaitre la taille d'une zone
- « oldNumPlayer » qui garde en mémoire le joueur déplacé par drag and drop même après son déplacement par le joueur
- « oldPositionX » et « oldPositionY » qui stock la position du joueur avant son déplacement par drag and drop
- « personnages » afin de vérifier si un joueur se trouve sur l'emplacement de la souris
- « isActive » pour savoir si un joueur est déplaçable

On utilise ici les classes « MouseListener » et « MouseMotionListener » pour les déplacements de la souris.

Le code du contrôleur de drag and drop se trouve dans la classe « ControleurDragAndDrop ». Lors d'un clic de la souris par un joueur « mousePressed » est appelé et vérifie la présence d'un joueur inactif sur l'emplacement de la souris avec la liste des personnages, dans ce cas on donne le numéro du joueur à « selectedPlayerNumber » ainsi que son emplacement actuel. La fonction « mouseDragged » est ensuite utilisée lors d'un déplacement de souris. Cette fonction vérifie qu'un personnage a été

sélectionné par le joueur et le déplace ensuite de case en case en suivant la souris et en actualisant l'affichage (avec repaintVueGrille de CVue). Finalement après le relâchement du clic « mouseReleased » appelle le menu d'action si un joueur a bien été déplacé et réinitialise le joueur sélectionné tout en stockant le personnage anciennement sélectionné dans « oldNumPerso ».

Lors de chaque clic on vérifie que le menu d'actions n'est pas ouvert, s'il l'est on le ferme et on renvoie le personnage à sa position initiale.



Menu des actions

La variable utilisée est « personnages » pour obtenir la position du personnage qui souhaite faire une action.

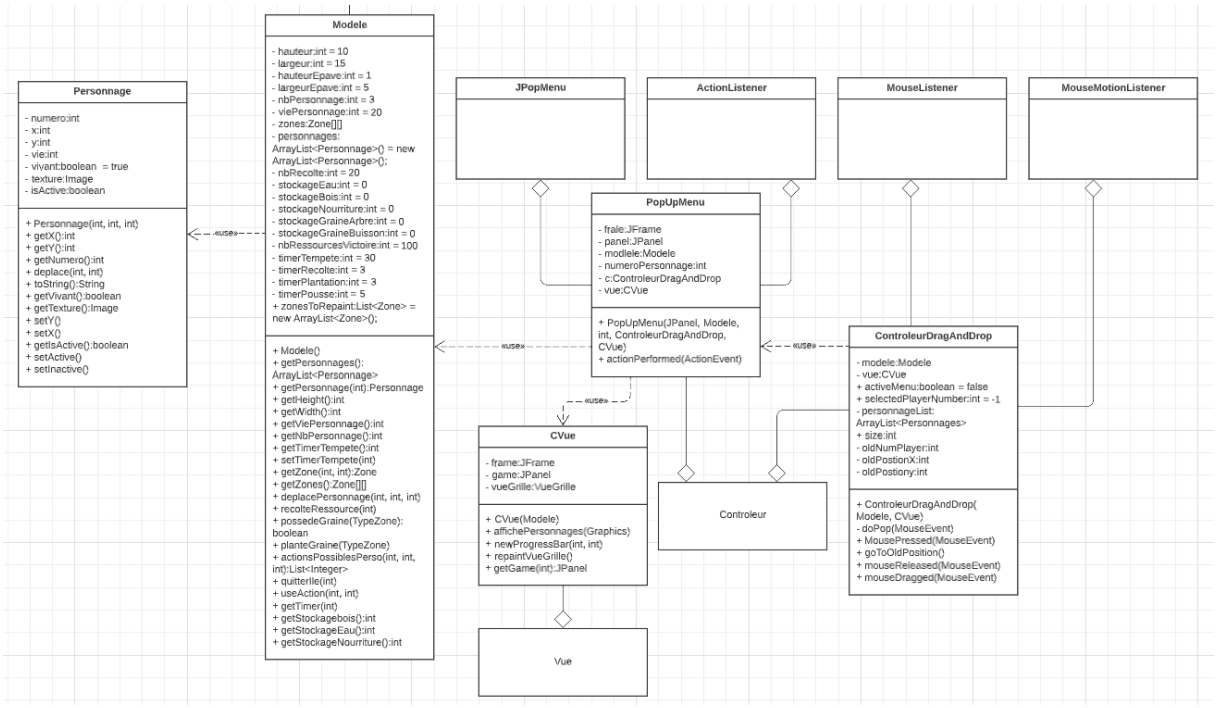
On utile ici la classe « JPopupMenu » et « ActionListener » pour l'affichage et la sélection dans le menu.

La classe « PopUpMenu » est celle qui gère l'intégralité du menu des actions. « PopUpMenu » défini les actions possibles en grisant le numéro des actions qui ne ressortent pas de la fonction « actionsPossiblesPerso » via son constructeur. La classe « PopUpMenu » est utilisée par la classe « ControleurDragAndDrop », lorsqu'un joueur lâche un personnage après un drag and drop la fonction « doPop » (dans ControleurDragAndDrop) ce qui a pour effet de créer un nouveau « PopUpMenu » dans « CVue » jusqu'à ce que le joueur clic à côté du menu ou bien sur l'une des actions proposées.

Julien Le Goff
Florian Pineau

Baptiste Samoyault
Sébastien Schaan

Le menu des actions, dans le cas présent sur une zone contenant des arbres.



Actions

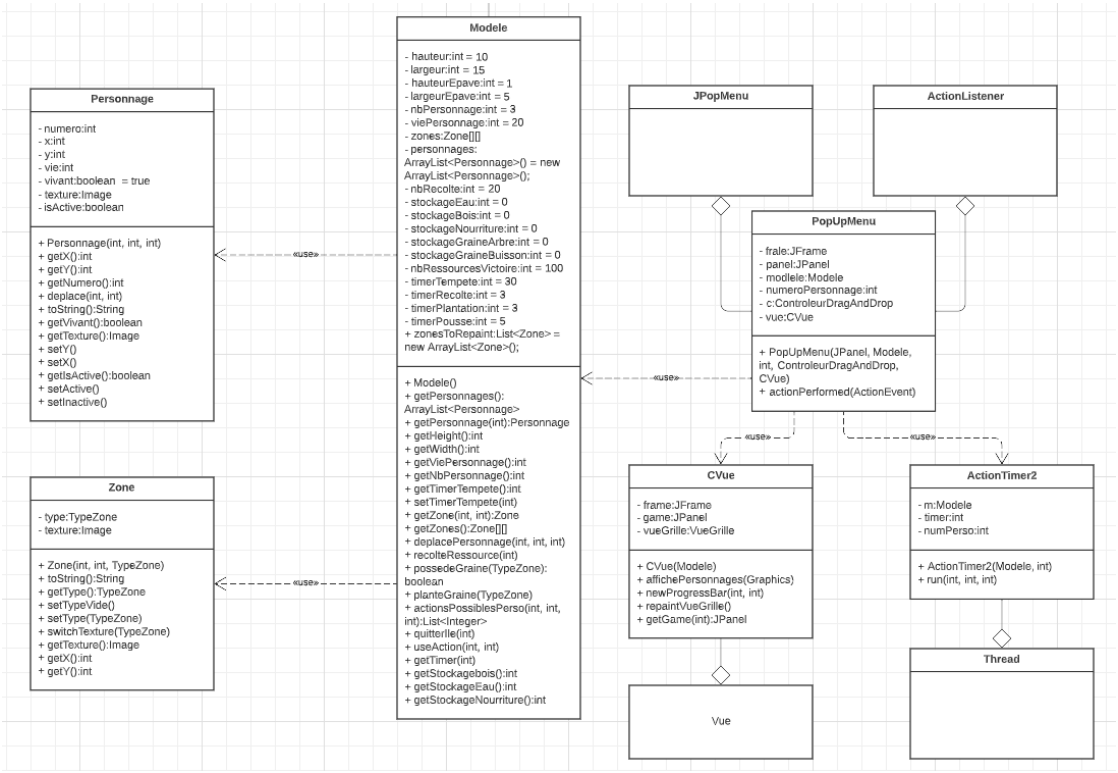
Récolter

Les variables utilisées sont :

- « nbRecolte » pour la quantité de ressource a ajouter au stockage après la récolte
- « zones » pour obtenir le « TypeZone » ou se trouve le personnage et donc le type de ressource qui va être récupéré
- « personnages » pour la position du personnage sur la grille afin de connaitre sa zone
- « timerRecolte » indique le temps en minute que dure une récolte de ressource

La fonction « recolteRessource » du « Modele » est utilisé en étant appelé par la fonction « useAction » (du Modele) elle-même appelée par la classe « PopUpMenu » après un clic sur l'un des 3 choix de récolte du menu d'action (et donc l'utilisation de actionPerformed). Cette fonction récupère et vérifie la zone où se trouve le joueur et ajoute les ressources récoltées ainsi que les éventuelles graines acquises.

Avant l'appelle de « useAction » par « PopUpMenu » on appelle la fonction « newProgressBar » de la classe « CVue » pour créer une barre de chargement et nous créons un nouveau « ActionTimer2 » qui va gérer le temps de récolte grâce au timer de la récolte.



Stockage ressources

Le stockage des ressources est effectué dans les variables « stockageNourriture », « stockageBois » et « stockageEau ».

Taux de drop des graines

Les variables « dropGraineBuisson » et « dropGraineArbre » représentes les pourcentages de chances de drop une seule graine. « stockageGraineBuisson » et « stockageGraineArbre » le stockage des deux types de graines.

Pour cette partie, elle se trouve dans le « Modele » dans la fonction « recolteRessource », lorsque l'on récolte du bois ou de la nourriture, une faible chance de drop 2 graines au lieu d'une est testée, le nombre de graine est ajouté dans le stockage.

Planter

Les variables utilisées ici sont « stockageGraineArbre » et « stockageGraineBuisson » qui represénte respectivement la quantité de graine d'arbre et de buisson.

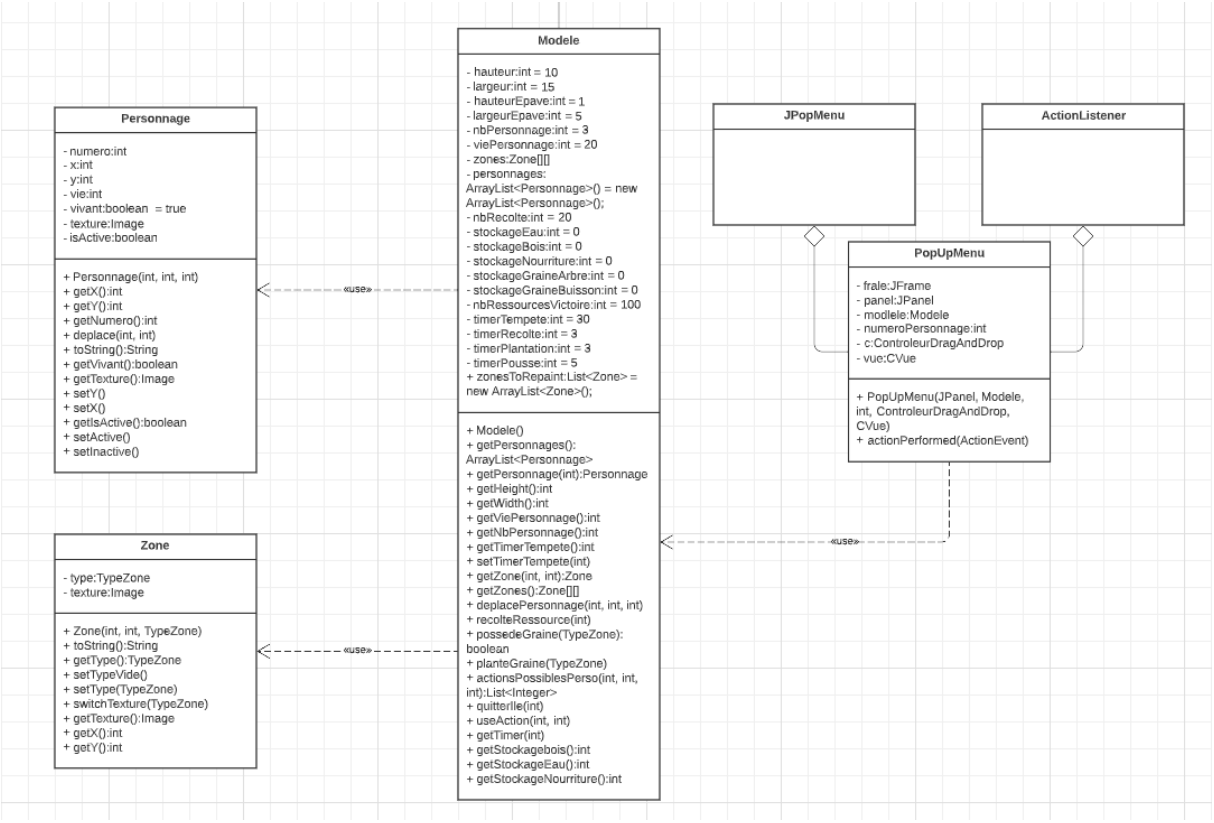
Planter une graine est maitrisée par la fonction « planteGraine » dans le « Modele » cette fonction appelle « possedeGraine » qui vérifie la présence de la graine voulue. « planteGraine » retire une graine et modifie ensuite la zone cible en « TypeZone » de pousse.

Déplacement

Les variables utilisées ici sont « personnages » afin de récupérer le personnage concerné par le déplacement et « zones » pour obtenir la zone ou le joueur souhaite se déplacer et vérifier qu'elle n'est pas impraticable (Rocher ou mer).

Le déplacement d'un personnage est géré par la fonction « deplacePersonnage » dans la classe « Modele » celle-ci effectue déplace le personnage premièrement sur l'axe des X et ensuite sur l'axe des Y en utilisant la fonction « deplace » de la classe « Personnage », un temps d'attente est réalisé entre chaque déplacements afin de ne pas le voir se téléporte en allant plus vite que le taux de rafraîchissement de l'affichage. La fonction a chaque petit déplacement ajoute la zone visée dans « zonesToRepaint » afin de la réafficher.

La fonction « deplacePersonnage » est utilisée par la classe « PopUpMenu » dans plusieurs cas, le premier est le choix « Se déplacer » dans le menu d'actions (après un drag and drop sur une zone), le second corresponds à n'importe quel autre choix que « Annuler ». En effet le personnage doit toujours se déplacer pour aller sur la zone ou il doit effectuer une action.



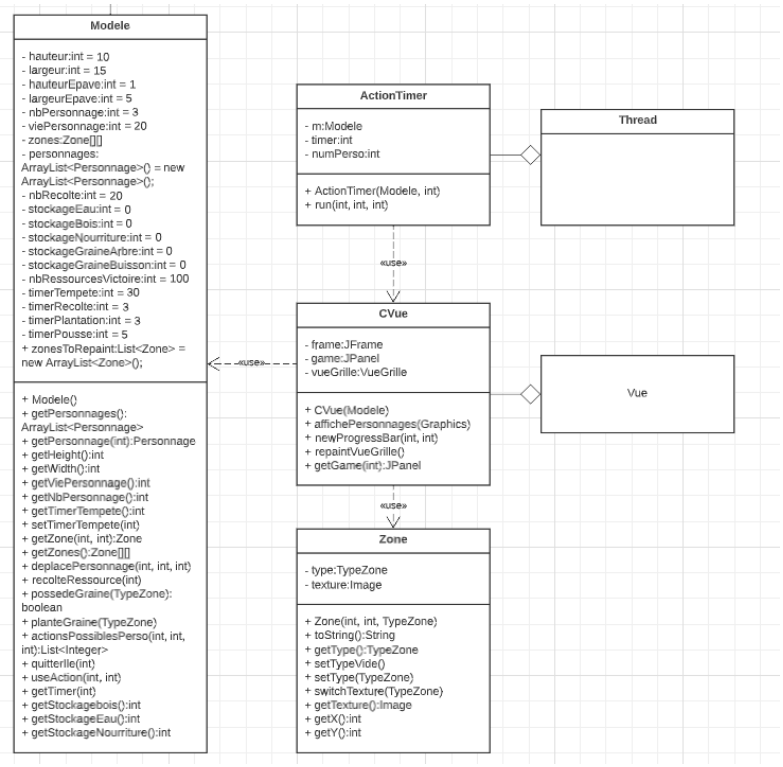
Vitesse de déplacement

La vitesse de déplacement est définie par la variable «vitesseDeplacement » (dans Modele) qui indique le temps d'attente entre chaque petit déplacement de zone.

Pousse

La variable « timerPousse » dans « Modele » correspond au temps en minute nécessaire pour qu'une graine pousse.

La pousse d'une graine est gérée par la fonction « run » dans la classe « ActionTimer », elle permet de modifier la zone cible par un « TypeZone » de pousse souhaitée et appelle la fonction « NewProgressBar » dans « CVue » pour afficher un chargement, à la fin du timer, la zone est modifiée en « TypeZone » classique et une action peut à nouveau être réalisée.



Fin de partie

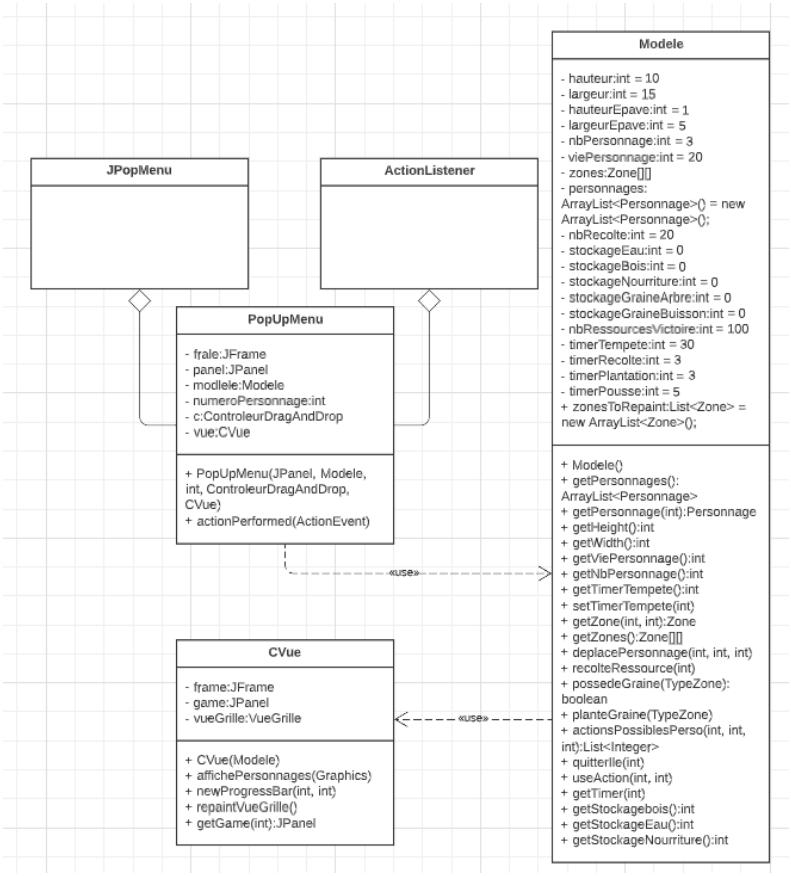
Victoire

Les variables utilisées sont « stockageEau », « stockageBois » et « stockageNourriture » pour savoir si la quantité requise par « nbRessourcesVictoire » (dans la classe Modele) est respectée. On utilise également « personnages » pour obtenir le nombre de personnages présents sur l'épave.

La classe JFrame est utilisée pour l'écran de victoire, ainsi que JPopupMenu et ActionListener pour le menu de choix d'actions.

Cette fonctionnalité est appelée par « PopUpMenu » lorsque l'on clic sur l'option « Tenter de fuir » qui apparait quand on veut déplacer un joueur vers l'épave. Un clic sur cette option du menu appelle « useAction » avec le numéro de l'action souhaité (en l'occurrence le 7) ce qui a pour effet d'appeler « quitterIle ». La fonction « quitterIle » vérifie que la quantité de nourriture, de bois et d'eau des joueurs est supérieur ou égale à 100, le cas échéant, la fonction « win » est appelée et affiche l'écran de victoire. La phrase de victoire est influencée par le nombre de joueurs présents sur l'épave.

Il n'est pas nécessaire d'utiliser cette fonctionnalité quand aucun joueur ne se retrouve sur l'épave, une vérification est faite dans cette « quitterIle » pour empêcher tout crash.

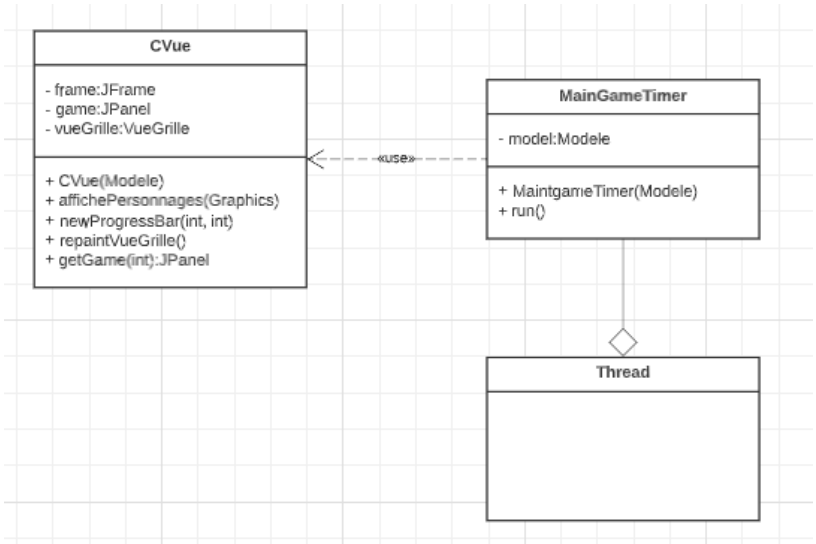


Défaite

La constante utilisée est « timerTempete » (dans la classe Modele), elle indique le nombre de minute avant que la tempête n'arrive et donc que la partie ne se finisse.

La classe Thread est utilisée pour le timer de jeu ainsi que la classe JFrame pour l'écran de victoire.

Cette fonctionnalité est gérée par la classe « MainGameTimer », lorsque le timer atteint le temps voulu la fonction « gameOver » (classe CVue) est appelée ce qui affiche l'écran de fin de partie.



Julien Le Goff
Florian Pineau

Baptiste Samoyault
Sébastien Schaan

Résultat

Fenêtre de jeu avec un joueur.

Documentation utilisateur

- Prérequis : Java avec un IDE compatible.
- Mode d'emploi : Importez le projet dans l'IDE, et sélectionner la méthode main de la classe Main, cliquez ensuite sur le bouton « Run ». Le projet est ensuite ouvert, il suffit ensuite de déplacer un joueur en drag and drop à la souris pour voir le menu d'action apparaitre et de cliquer sur celle voulu pour qu'il l'exécute.

Documentation développeur

Les classes les plus importantes du projet sont clairement : Modele, ActionTimer2 et PopUpMenu. Ce sont ces classes qui permettent de lancer les différentes actions. Les principales constantes modifiables se trouvent dans Modele et CVue.

Ainsi la classe Modele contient les constantes :

- hauteur et largeur pour la taille de l'île
- hauteurEpave et largeurEpave pour l'emplacement de l'Epave
- nbPersonnages, le nombre de personnages jouables
- viePersonnage, la vie d'un personnage
- nbrecolte, le nombre de ressource récolté en une fois

- nbRessourcesVictoire, la de chaque ressources nécessaire pour gagner
- timerTempete, le temps en minute d'une partie
- timerRecolte, le temps en minute pour récolter une ressource
- timerPlantation, le temps en minute pour planter une graine
- timerPousse, le temps en minute pour qu'une graine ai fini de pousser
- vitesseDeplacement, le temps nécessaire pour se déplacer sur une autre zone

La classe CVue contient :

- size, la taille que fera une zone ou un pion
- SCREEN_SIZE, la taille de la fenetre

On peut aussi noter que dans PopUpMenu, on peut ajouter d'autres possibilités d'actions en plus de celles déjà présentes.

Les fonctionnalités non implémentées mais qui le mériterais sont l'ajout d'ennemis faisant baisser notre vie en cas d'attaque mais donnant des ressources si on les bat (serpent, sanglier). La possibilité de perdre des ressources est aussi pertinent pour rajouter en difficulté. Un déchainement de la tempête qui ferait perdre des ressources sur les zones de temps en temps pourrait aussi faire en complexité.

Une part sympathique pour le jeu aurait été de rajouter un menu au lancement avec le logo du jeu, une musique durant tout le long de la partie (le thème de The of Secret Monkey Island avait été choisi) mais aussi l'ajout de sons en fonctions des différentes actions. Nous souhaitions aussi ajouter remplacer les personnages par des GIF courant des déplacements.

Conclusion et perspectives

Lors de ce projet seul une partie du jeu a pu être réalisé, ainsi on peut faire déplacer un personnage en drag and drop jusqu'à une zone ce qui affiche le menu d'action, on peut ensuite décider de récolter une ressource mais le déplacement du personnage au fur et à mesure ne se fait pas. La pousse n'a pas non plus été implémentée ainsi que le path finding. Néanmoins l'affichage est convaincant et fluide, et plusieurs Threads ont pu être utilisés pour différentes parties du projet. On peut également retrouver un compteur de ressources et l'affichages des timer d'actions et de tempête. Plusieurs sprites ont été créés spécialement pour Wotah.

Nous avons appris de nombreuses choses durant ce projet, que ce soit sur le code sur l'affichage ou bien sur la partie génie logiciel avec la gestion du digramme de Gantt et la répartition des tâches.

En conclusion ce projet nous a permis de creuser de nombreux sujets et d'en apprendre un peu plus sur le métier de développeur.

L'évolution la plus logique pour notre jeu serait d'abord d'ajouter les fonctionnalités manquantes, mais aussi d'ajouter de la difficulté et finalement des cométiques avec de la musique et des menus.