

Création de urcamod

Avec MCreator





Equipe :

TONNELLE Nathan, THEODORE Jennyfer, FOUQUE Andy,

LAROCHE Tristan, MACZYTA Nicolas, OURDOUILLE Renaud

Encadrant :

Mr Cyril RABAT

Date :

2019 - 2020

Table des matières

[Introduction : 2](#_Toc41099543)

[La conception de base des mods 3](#_Toc41099544)

[La mise en place de ces premières idées 4](#_Toc41099545)

[La concrétisation des idées et des mods 5](#_Toc41099546)

[Conclusion : 7](#_Toc41099547)

# Introduction :

Le jeu Minecraft, que la majorité des joueurs mondiaux connaissent est excellent, ce jeu propose beaucoup de contenus, de quoi tenir quelques longues heures de jeux ! Mais nous avons voulu en rajouter encore plus à ce jeu que nous adorons tous, c'est pourquoi nous avons choisi le sujet de projet suivant : créer un MOD Minecraft. Nous nous sommes donc dirigés vers un MOD alliant exploration et défis. Pour ce faire, nous avons utilisé MCreator qui est un logiciel qui nous a permis de créer facilement notre MOD, cependant, Jennyfer ayant eu des soucis avec ce logiciel, nous avons pris le choix de lui faire faire un second MOD Minecraft, comme elle ne pouvait prendre réellement part au premier, c'est ainsi que nous allons pouvoir comparer les 2 version et les différentes manières de faire un MOD. Nous allons tout d'abord voir les débuts de conception puis les idées de bases et enfin la concrétisation des idées de bases et leurs évolutions. Nous avons choisi de répartir le sujet entre tous les membres du groupe selon leurs capacités, Andy, Nathan, Tristan et Nicolas se sont occupé du premier mod avec MCreator ; Renaud connait bien les logiciels de modèles et textures 3D, ont lui a donc assigné cette tâche ; et Jenny la création du second MOD à partir de l'API Forge.

# La conception de base des mods

Dans cette partie, nous allons voir comment avec quels moyens, interrogations, nous avons mis en place le projet.

## Le questionnement

Nous nous sommes tout d'abord tous concerté, lors de notre première réunion hebdomadaire le lundi 13 avril 2020, afin de savoir qui sera le chef de projet dans le groupe, nous étions 2 à nous proposer et le choix c'est porter sur Nathan. Durant cette première réunion, très importante, il a fallu établir plusieurs fonctionnements, et c'est ainsi que nous nous sommes posés les questions, comment allons-nous pourvoir tous travailler sur le projet en meme temps, avec GitHub, ou via l'envoie de fichiers individuels ? Avec quoi nous allions travailler, nous nous souvenons que cette réunion a durée toute l'après-midi, car il y avait beaucoup de choses à fixer, comme notre manière de travailler, le partage des tâches et autres.

Durant cette première réunion, nous avons commencer par discuter du sujet, qui était d'une part d'utiliser Minecraft Education Edition afin de programmer des activités basées sur les TP de L1 et d'autre part, de développer un mod Minecraft. Selon le plan de travail que nous avait fait Mr Rabat, nous devions faire un choix entre les 2, dans ce plan il nous était également donnée une piste pour le mod, l'incarnation d'un médecin tuant des zombis atteints du COVID avant que l'infection ne se propage. Mais nous avons choisi de partir sur autre chose.

## La manière de faire le mod

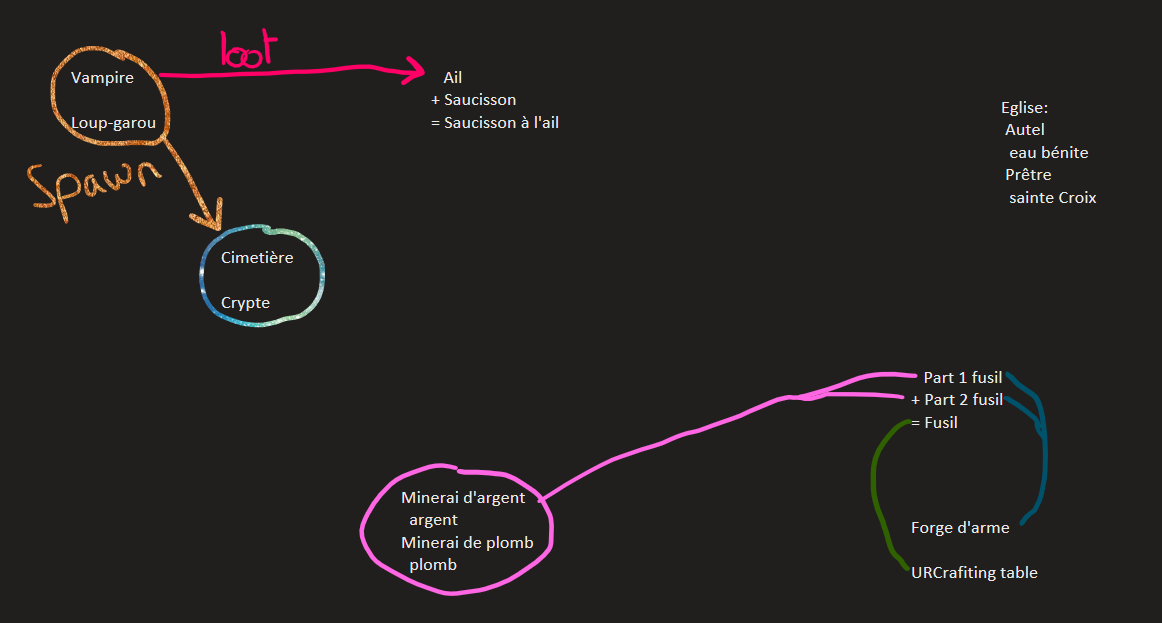
Nous avons également cherché comment peut-on programmer un mod Minecraft en ayant aucunes connaissances dessus, cela allait être un défi. Puis nous avons trouvé MCreator qui est un logiciel qui nous permet de fabriquer un mod de toute pièce plutôt aisément, avec des objets, des items, des monstres, des commandes, des procédures et autres. Nous avons donc pris la dernière version de celui-ci, la version 2020.2 permettant de programmer un mode en 1.14.4. Ce qui est très important car chaque version de Minecraft ajoute des possibilités de programmation. Ce n'est pas la plus récente version du jeu (1.15.2) lorsque nous avons commencé mais elle était entièrement suffisante. Cette version du logiciel est surtout beaucoup plus pratique au niveau de son utilisation face à la version 2019.x. maintenant que nous en sommes à la fin de ce projet, il y a une nouvelle version du logiciel, la 2020.3 pouvant programmer un mod de la 1.14.4 à la 1.15.2. Comme ce n'était pas le cas avant, nous n'allons pas tout changer pour refaire en 1.15.2.

Il est arrivé après 2, 3 semaines de programmation sur le logiciel, un problème que Jennyfer avait, elle avait énormément à travailler avec et à lancer le jeu pour voir ces tests réussir ou échouer, comme nous le faisions. Nous avons donc pris la tangente et entrepris de faire un second mod sur lequel elle fut la seule à travailler, car à causes de problème de partage de dossier via GitHub, il était impossible que Nathan travaille avec elle, malgré le fait que nous avons essayé ensemble, pendant une semaine de faire en sorte, cela n'a pu aboutir et il a fallut choisir entre continuer le projet du premier mod ou essayer encore et encore de programmer et de partager des portions de code. Nous avons fait le choix de la laisser travailler toute seule sur ce second mod.

Pour faire ce second mod, Jennyfer à du partir de l'API Forge et avec l'aide de tutoriels, vidéos, anglais et français, a réussi à produire quelque chose.

## Les premières idées

Durant la première réunion, nous avons posé quelques idées de base, que voici :



Nous pouvons voir que nous avions déjà des idées et les avons mises en relations les unes avec les autres.

Sur le second mod, qui est donc arrivé en milieu de projet, nous avons décidé de partir sur un virus qui peut affecter tous les joueurs.

## Présentation de MCreator

comme toutes le groupe, il a fallu du temps pour s'adapter et comprendre cet outil de travail.

# La mise en place de ces premières idées

Pour différencier plus facilement les 2 mods produit, nous choisissons d'appeler URCAmod1 le premier et URCAmod2, le second.

## URCAmod1

### Les armes à feux

Par rapport, à l'idée de base, nous avons abandonné l'idée de faire fabriquer les armes par le biais d'une forge et en plusieurs morceaux. Mais la réelle difficulté des armes à feux, ne réside pas en leurs conception de textures mais dans le fait qu'il ne fallait pas qu'elles se comportes comme des arcs, c’est-à-dire avec bandaison de la corde et donc zoom sur l'écran du joueur, elle réside également sur le fait qu'il fallait équilibrer les fusils pour qu'ils ne soient ni trop puissant ni trop faible, ce qui rendrait le jeu trop facile. Nous avons rajouté une difficulté en faisant en sorte que les fusils utilisent des balles pour tirer, qui sont fabriqué avec de la poudre à canon et du plomb, celui-ci étant plutôt facile à obtenir, le défis est d'obtenir des poudres pour les balles. L'ajout de monstres puissants comme les vampires ou loup-garou rééquilibre le jeu avec cette arme.

### Les vampires et biome

C'est Andy qui s'est occupé de toute la partie pour les vampires avec leur propre biome, la "Terre Désolée", mais avec le schéma de départ, avant l'ajout de nouvelles idées au fur et mesures des réunions. Il n'y avait qu'une seule classe de vampires, les vampires IO, des créatures rapides avec une IA de Zombie, ce qui fait qu'elle peut vous sentir à 100 blocs et se rapprocher discrètement et rapidement.

Après plusieurs essais, pour régler le nombre de d'arbre, la vie des mobs, passifs ou agressifs, le biome fut opérationnel.

## URCAmod2

### La conception

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Package : com.github.broussk.URCAmod2;   |  | | --- | | **Urcamod2** | | String MODID : urcamod2 | | Logger LOGGER :  LogManager.getLogger(MODID) | | Urcamod2() |  |  | | --- | | **ModEventSubcriber** | | onRegisterItems | | onRegisterBlocks | | onBiomeDecoration |  |  | | --- | | **ContainerModTileEntity** **extend** **container** | | ModTileEntity te | | ContainerModTileEntity | | canInteractWith |  |  | | --- | | **PlayerData** | | entityPlayer player | | Boolean virus | | getVirus() : boolean | | setVirus(boolean virus) : void | | Package : init;   |  | | --- | | ModItemGroup | | Supplier : iconSupplier | | ModItemGroup (final String name, final Supplier<ItemStack> iconSupplier) | | createIcon() | | ModItems | |

On démarre par la première classe urcamod2 appelé simplement main, dans la nouvelle classe on crée une constante string appelé MODID de valeur notre mod id.

Ensuite on importe @Mod de net.minecraftforge.fml.common.Mod l'annotation indique a forge que cette classe est un mod et doit être chargé.

Avec seulement cela, on peut voir au démarrage de Minecraft le mod chargé dans la liste des mods.

Maintenant nous allons ajouter des objets.

On crée la classe ModEventSubscriber dans le même package que le main et on annote de @EventBusSubscriber, celle-ci indique à forge que la classe contient des méthodes qui doivent être subscribed pour gérer des évènements. Il a en paramètre le mod id et le bus : EventBusSubscriber. Finalement on a en paramètre modid = Urcamod2.MODID qui indique à forge que la classe appartient au mod. C’est important car le scan de EventBusSubscriber est fait avant que le mod ne soit chargé or forge ne sais pas encore à quel mod la classe qu’il scan appartient. Ensuite le paramètre bus = EventBusSubscriber.Bus.MOD indique à forge que les méthodes @SubscribeEvent doivent recevoir les évènements de MOD du bus.

On crée une méthode onRegisterItems avec comme paramètre RegistryEvent.Register<Item> et on l’annote avec @SubscribeEvent. Ce dernier indique que la méthode veut subscribe un évènement et le paramètre que la méthode est appelée quand le mod doit enregistrer l’objet.

Ensuite on ajoute les méthodes setup pour allouer le nom des entries à la fin de la classe ModEventSubscriber.

On va créer et enregistrer l’objet, dans la méthode onRegisterItem on fait un appelle à la fonction registerAll et de setup dans lequel on instancie le nouveau bloc.

# La concrétisation des idées et des mods

## URCAmod1

### Les tanières des loups

### La crypte

### Le monde de Mr Rabat, RabatWorld

## URCAmod2

Nous pouvons voir que lorsque l'on charge le jeu, le mod se charge bien avec, et nous pouvons obtenir le block construit avec la commande "/give".

Pour créer l’event le plus important - eventVirus - nous avons besoin d’une nouvelle variable associé aux joueurs. C’est pourquoi on va créer la classe PlayerData qui contient les variables entityPlayer et la nouvelle variable booléenne que l’on appelle simplement virus. On y retrouve un constructeur pour ces deux variables et le getter et setter pour virus. Retour dans la classe ModEventSubscriber, on ajoute la méthode eventVirus(AttackEntityEvent event) et à l'intérieur il suffit de mettre la condition d’obtenir le statut contaminé. Si je suis sain et que l’entité adverse est contaminée. Alors l’effet maudit s’applique et la variable virus passe à true, l’ajout du print est utile pour le log.

Effectivement cette partie était la plus compliqué car il faut à la fois trouver un mode de contamination efficace, non contraignant pour les joueurs, et qui fonctionne. On a eu différentes pistes comme utiliser la variable virus sous la forme d’une compétence qui augmenterait avec de l'expérience qui s'acquiert avec l’exposition aux autres joueurs contaminé.

En conclusion pour être infecté du virus il faut attaquer un joueur infecté ou inversement ce qui rend l'accessibilité pas optimale étant donné la nature du jeu, un contacte devrait pouvoir suffire est serait utilisable dans différents modes de jeu

# Conclusion :

Pour conclure, nous avons réussi à créer un MOD Minecraft qui allie bien l'exploration avec le biome et le monde ajouté, que le défis avec les loups garous, les vampires, ainsi que la tanière des loups ou le cimetière. Nous avons tous appris avec ce projet, que ce soit de la manipulation de langage Java avec une API, ou du scratch avec les différentes procédures que nous avons créé, certains on même appris à faire des modèles et textures 3D. Cependant, nous avons renoncé à quelques idées que nous avons eu de base, comme la forge ou le prêtre villageois dans les chapelles.