



---

# CRÉATION DE URCAMOD

---

Avec MCreator et Forge



## Équipe :

TONNELLE Nathan, THEODORE Jennyfer, FOUQUE Andy,  
LAROCHÉ Tristan, MACZYTA Nicolas, OURDOUILLE Renaud

## Encadrant :

Mr Cyril RABAT

## Date :

2019 - 2020

# Table des matières

Remerciements .....	3
Résumé .....	4
Introduction .....	5
Présentation du contexte .....	6
Présentation du besoin .....	6
Le logiciel utilisé .....	6
L'évaluation de ses utilités .....	7
Organisation du travail.....	7
Étude de mod existant .....	8
Étude générale .....	8
Fonctionnalités .....	8
Structures .....	8
Les monstres.....	8
La nourriture .....	8
Étude des solutions disponibles .....	9
La génération de monstres .....	9
La création de structures.....	9
La création de nouveaux objets .....	9
Développement de la solution choisie .....	11
Conception .....	12
Architecture de conception .....	12
Des structures.....	12
Des monstres.....	13
Des objets .....	15
URCAmod2.....	16
Le fonctionnement logique .....	17
Des monstres.....	17
Objets .....	17
Blocks .....	18
Structures .....	19
Du biome .....	20
Un fluide .....	21
URCAmod2.....	21
Bilan .....	23
État du développement .....	23
URCAmod1 .....	23

URCAmod2.....	23
Difficultés rencontrées.....	23
URCAmod1 .....	23
URCAmod2.....	23
Pistes d'amélioration .....	23
URCAmod1 .....	23
URCAmod2.....	23
Conclusion .....	24
Le petit message de chacun sur le projet .....	25
Andy .....	25
Jennyfer.....	25
Tristan .....	25
Nicolas .....	25
Renaud .....	25
Nathan.....	25
Glossaire.....	27

## Remerciements

Nous remercions, Mr Rabat de nous avoir accompagné dans ce projet, nous le remercions également de nous avoir donné l'opportunité de le faire, c'était une idée remarquable et d'autant plus de nous la proposer en tant que projet.

De la part du chef de projet : «je voudrais remercier toute l'équipe, même si un participait peu. Vous avez fait du bon travail, tous ensemble. Nous avons également réussi, avec de l'organisation, à faire ce que nous voulions dans ce mod, c'est pour tout cela que je vous remercie.»

## Résumé

Dans ce projet, nous devons construire, pièce par pièce un mod Minecraft. Nous avons réussi cela avec le logiciel MCreator, qui regroupe tous les éléments nécessaires pour le faire, création d'éléments graphiques, créations de processus sous forme de blocs avec Scratch, et la création de contenus dans le jeu. Notre mod porte sur les thèmes de l'exploration avec une touche de défis, accompagné de 2 nouveaux boss.

## Introduction

Minecraft est un jeu multiplateforme se développant depuis fin 2011, sur les plateformes Windows, Windows Phone, Linux, Nintendo Switch, FireOs, FireTv, Gear TV, Xbox, PlayStation, Mac, Android, iOS ou encore Raspberry Pi. Touchant un grand nombre de joueurs du monde entier, il est conçu en 96 langues, il touche autant les petits que les grands. C'est un jeu de création qui propose beaucoup de contenus, de quoi tenir quelques longues heures de jeu ! 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 ou programmer des activités basées sur les TP de Licence 1. Mr Rabat nous avait donné une piste pour la création d'un mod : l'incarnation d'un médecin tuant des zombies atteints du COVID avant que l'infection ne se propage. Après réflexion et recherches, nous sommes partis sur la base d'un mod alliant l'exploration aux défis, que nous allons découvrir maintenant.

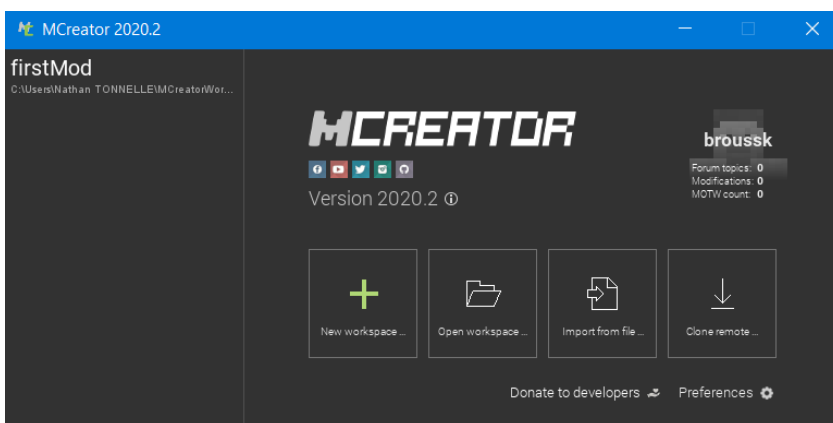
## Présentation du contexte

Minecraft propose déjà beaucoup de contenus à lui tout seul, mais il nous semblait évident que l'aventure était encore trop courte pour nous, qui connaissons beaucoup le jeu.

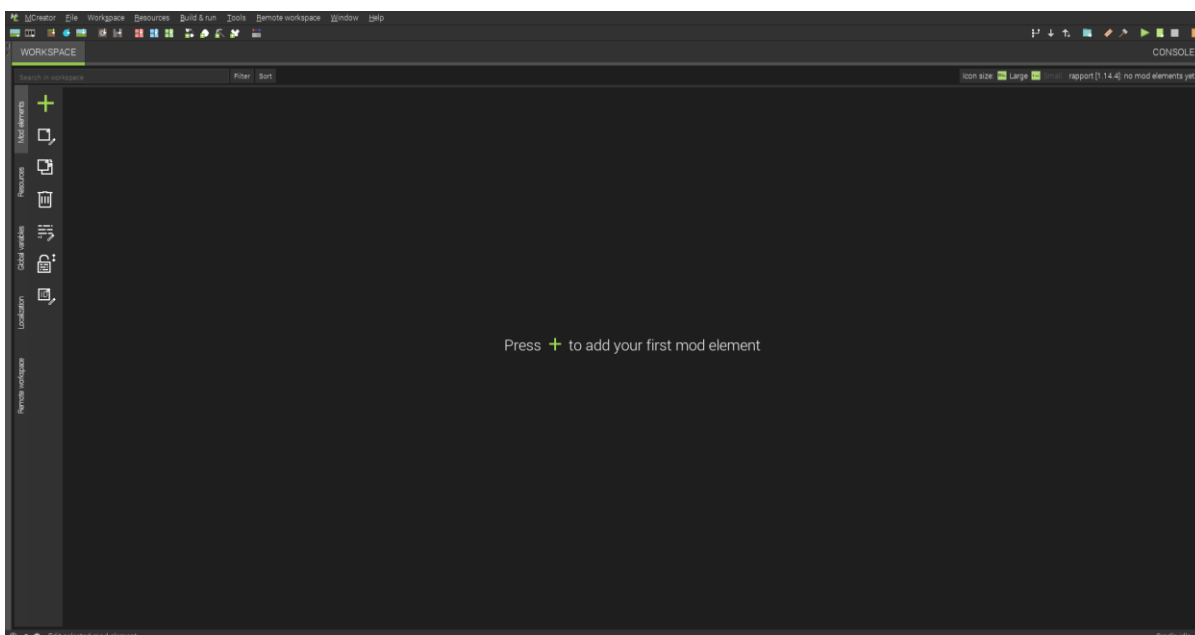
## Présentation du besoin

### Le logiciel utilisé

Pour concevoir ce mod, nous avons utilisé MCreator, qui nous a apporté tout ce dont nous avions besoin pour faire ce que nous devons faire, à savoir construire un mod du début à la fin. Il existe plusieurs versions de ce logiciel, comme de versions de Minecraft, nous avons choisi la dernière version qui nous était proposée, la version 2020.2 valable pour faire un mod en version 1.14.4 de Minecraft. Ce qui nous arrangeait beaucoup. L'interface de cette version du logiciel étant plus facile à comprendre et à utiliser, d'autant plus pour nous qui n'avions jamais essayé de créer un mod. Son interface se présente comme cela, dans un premier temps, lors du choix de création de projets, appelés « Workspace », ou de la possibilité de faire un projet avec GitHub, ce qui nous a beaucoup servi pour nous partager les tâches et travailler tous ensemble, nous pouvons également voir les projets où nous avons récemment travaillé.



Puis dans un second temps, lorsque l'on est sur un projet, nous avons le Workspace et tout un tas de boutons avec des fonctionnalités différentes et notamment le « + » à gauche nous permettant d'ajouter, via un menu, des éléments dans le mod.



## L'évaluation de ses utilités

Ce logiciel nous a permis de faire beaucoup de choses dans le mod, cependant, il possède quelques manques, notamment dans la création de texture et de modèles 3D, nous ne pouvons voir le rendu directement sur le modèle 3D sur lequel nous voulons le mettre. Nous avons donc fait appel à plusieurs autres logiciels tels que BlockBench, NovaSkin, ou même Paint pour tout ce qui est textures, ces logiciels, sauf Paint, permettent de voir en temps réel ce que cela allait donner avec une certaine structure 3D. Techne, utilisé pour la création d'un nouveau type de monstre, en modèle 3D. Et IntelliJ IDEA Edu pour faire le second mod, avec l'API Forge.

Ce logiciel nous appris beaucoup de choses notamment à nous servir de scratch, qui est un langage de programme auquel nous n'avions utilisé.

Il a fallu faire des tests au début pour savoir comment faire, mais nous avons réussi à nous organiser, notamment avec sa fonctionnalité de partage de projet en relation avec GitHub.

## Organisation du travail

L'organisation du travail a été facilitée grâce à l'élection d'un chef de projet, Nathan, et à l'organisation de réunions chaque début de semaines, pour savoir ce qu'il fallait faire, ce qui a été fait et s'il y avait de nouvelles idées. En fin de semaines, Nathan nous demandait à chacun de lui faire un mini rapport sur où il en était, comment il avait travaillé, ces difficultés également. Si nous ne pouvions participer à une réunion, il nous prenait à part quand on pouvait, mais toujours en début de semaine, il nous faisait un résumé sur ce qui a été dit durant la réunion. Nous avons fait un serveur discord pour que cela soit plus facile pour chacun et que Nathan nous rappelle lorsque les réunions allaient commencer, le travail qui était à faire. Lorsque Mr Rabat nous a demandé de lui rendre un « feedback » sur ce qui a été fait chaque semaine, pendant la réunion, Nathan nous prenait un par un et parlait avec chacun de ce que nous avions fait, ce que nous avions à faire, de comment nous nous sentions dans le groupe et avec le travail qui était demandé. Si nous avions besoin d'aide sur quelque chose où on avait du mal.

Nous avons eu un bon chef de projet qui nous encadrait et qui était là à tous moments de chaque journée pour nous aider de son maximum sur tous les sujets du projet. Il était compréhensif quand nous ne pouvions participer aux réunions ou au travail pendant la semaine, comme Tristan qui, avec sa famille ne pouvait être tout le temps présent, ni travailler quand il le voulait, mais seulement quand il le pouvait, avec son fils et le travail à la maison, sa femme enceinte, et pour d'autres, comme Renaud et ses problèmes personnels.



## Étude de mod existant

Dans cette partie, nous allons voir de quoi sont constitués principalement les mods et donc de quoi nous avons besoin de mettre dans le nôtre pour les thèmes que nous voulons.

### Étude générale

Nous allons voir de quoi est composé généralement un mod.

#### Fonctionnalités

Dans un mod, il existe plusieurs fonctionnalités ajoutées des interfaces, des items (outils, arme, nourriture), des blocs, des structures, des mobs (amicaux ou non), quêtes, biomes, des commandes, des dimensions, des fluides, des touches, des musiques, de nouvelles tables de génération d'item dans les coffres, des plantes, des potions, des recettes, des projectiles.

Chaque mod apporte son lot d'ajouts, de modifications, c'est avec cela que l'on spécifie le mod.

#### Structures

Minecraft comporte des structures comme le bastion (StrongHold), les mines abandonnées, les donjons, les monuments sous-marins et bien d'autres. Ces structures sont des épreuves du jeu, nous permettant d'avoir accès à des ressources rares, un «spawner» de monstres, ou une nouvelle dimension avec le bastion, et un mini boss avec les monuments sous-marins et boss avec la dimension de l'ender.



#### Les monstres

Le jeu comporte 24 mobs passifs, 9 mobs neutres, 29 mobs agressifs et 2 boss. Par exemple :

- Mobs passifs : le poulet, le mouton, le perroquet, le villageois, le cheval
- Mobs neutres : le panda, le dauphin, l'abeille, le loup, l'Enderman
- Mobs agressifs : l'araignée, le Blaze, le Creeper, le Pillager, le Wither squelette
- Boss : Ender dragon et Wither

Dont chacun apporte une difficulté et des caractéristiques propres à son type.

#### La nourriture

Dans le jeu il y a 39 nourritures différentes, comme les monstres, ils ont des caractéristiques propres à leurs types, comme la culture possible, la cuisson, les effets.

Nous pouvons trier ces nourritures en fonction de leurs valeurs nutritionnelles (en fonction de leurs saturations) :

Alimentation	Valeur	Nourriture
<b>Surnaturelle</b>	2,4	
<b>Bonne</b>	1,6	
<b>Normale</b>	1,2	
<b>Faible</b>	0,6	
<b>Pauvre</b>	0,2	

## Étude des solutions disponibles

Nous allons voir maintenant ce que nous pouvons faire à partir de MCreator, ce que nous pouvons créer, également à partir de l'API Forge.

### La génération de monstres

Minimal number of entities in a spawn group: 5

Maximal number of entities in a spawn group: 10

Biomes where the entity spawns (empty for no restriction): firstmod:terredesole

Does this entity spawn in dungeons? ☐ Spawn in dungeons

La génération peut être gérée au niveau de MCreator pour chaque monstre, suivant le nombre minimum et maximum dans un groupe, ainsi qu'où ils peuvent être générés.

### La création de structures

Pour la création de structures, il faut tout d'abord construire ladite structure, et la sauvegarder grâce à des blocs spéciaux, les structures blocs. Ensuite il faut créer un objet dans MCreator avec la structure que nous avons sauvegardée.

Structure to spawn: brokenvillage

### La création de nouveaux objets

Avec MCreator nous pouvons créer plusieurs types d'objets : des items, des comestibles (boisson ou nourriture), des blocs. Les possibilités sont multiples.

#### Les blocs

Pour faire un bloc, il nous faut sa texture, on peut avoir toutes les faces du bloc différentes, et nous pouvons gérer beaucoup de paramètres, à savoir, s'il est capable de se générer tout seul, s'il est solide (comme la pierre), ou si on peut marcher dessus (comme la culture de blé, par exemple). Nous pouvons également régler sa « hit box », c'est-à-dire l'endroit qu'il faut cliquer sur le bloc pour le toucher, et plein d'autres réglages, mais un des plus importants est celui du modèle 3D que nous pouvons choisir.

Block textures: Top, Left, Front, Right, Back, Bottom / main

Block base (model and behaviour): Default basic block

Special information about the block:

Render type and rotation: Block model: Normal, Block rotation mode: No rotation

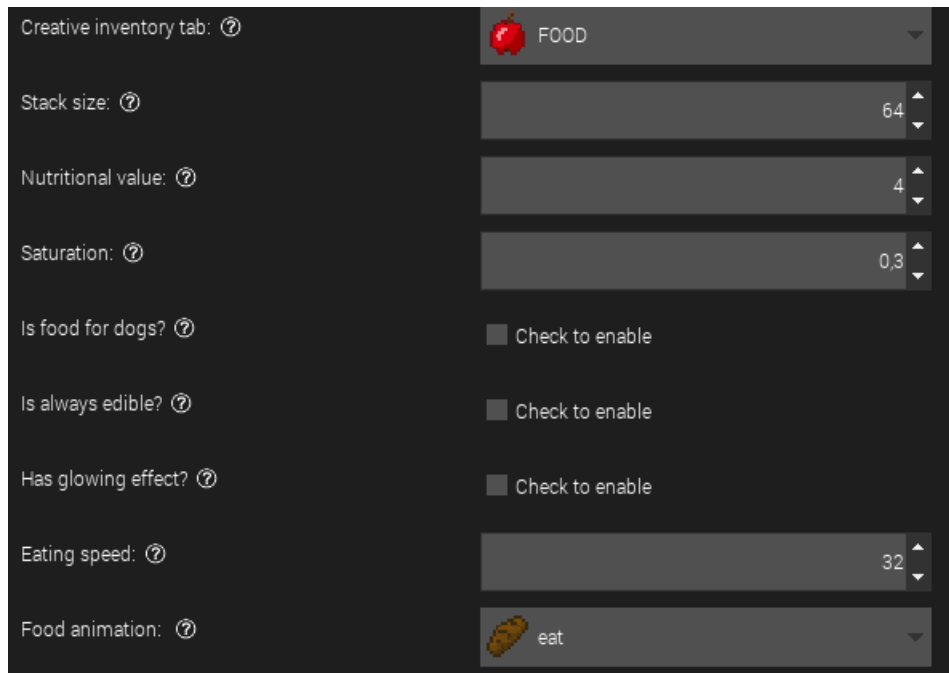
Transparency: Transparency type: SOLID

Block dimensions & bounding box: Block face shape: SOLID, Min. X coord: 0, Min. Y coord: 0, Min. Z coord: 0, Max. X coord: 1, Max. Y coord: 1, Max. Z coord: 1

Le développement de blocs avec Forge est plus laborieux, car il faut créer et enregistrer l'objet, dans la méthode `onRegisterItem` on fait un appel à la fonction `registerAll` et de `setup` dans lequel on instancie le nouveau bloc.

### La nourriture

Pour la nourriture, nous ne devons choisir que la texture, et ensuite faire les réglages sur le nombre de gigots qu'elle va apporter, la saturation qu'elle donne, si elle se boit ou se mange, ou encore si nous pouvons la donner à manger à un chien. Comme les blocs nous pouvons lui donner une structure 3D spéciale.

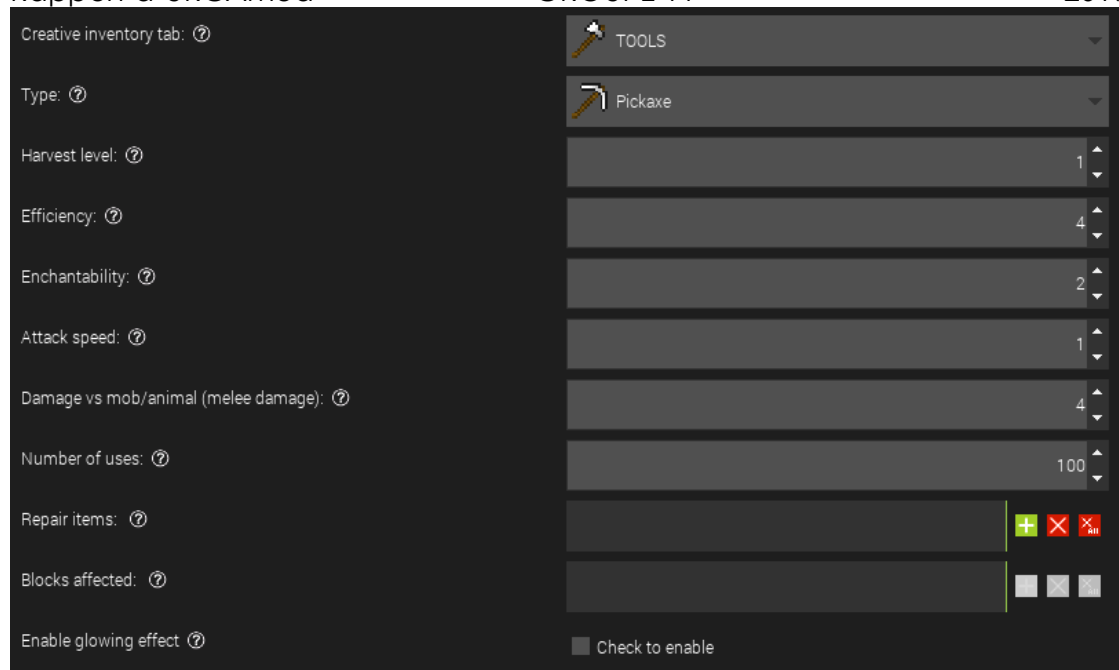


The image shows a screenshot of the Creative inventory tab for a food item in Minecraft. The interface is dark-themed. At the top, it says 'Creative inventory tab: ?'. Below this, there is a dropdown menu for 'FOOD' with a red apple icon. The following settings are listed:

- Stack size: ? (64)
- Nutritional value: ? (4)
- Saturation: ? (0.3)
- Is food for dogs? ? (Check to enable)
- Is always edible? ? (Check to enable)
- Has glowing effect? ? (Check to enable)
- Eating speed: ? (32)
- Food animation: ? (eat)

### Les outils

Pour les outils cela est la même chose que pour la nourriture sur le premier panel, cependant nous ne réglons pas le nombre de gigots que cela peut nous donner, mais le nombre d'utilisations que nous pouvons faire avec cet outil, à quelle vitesse il peut aller, quel est son niveau de minage, ou encore sa puissance d'attaque, avec quoi nous pouvons réparer l'outil également. Et le plus important à quelle catégorie appartient cet outil, s'il s'agit d'une pelle, d'une pioche, d'une épée, d'une houe, d'une hache ou d'une paire de ciseaux, mais nous pouvons également dire que c'est un outil spécial ou qu'il appartient à plusieurs catégories.



## Développement de la solution choisie

La solution la plus utilisée est celle de MCreator, car c'est la solution que nous avons utilisée dès le début, donc nous nous sommes familiarisés avec le logiciel, cependant, Jennyfer avait des problèmes avec ce logiciel, le jeu et son ordinateur. Il a fallu faire autrement pour qu'elle puisse participer et montrer sa détermination dans le projet, nous avons donc créé un second mod, URCAmod2 qui est fait avec l'API Forge et qu'elle a faite toute seule.

Ce qui nous a permis de faire ce comparatif entre les 2 méthodes, entre le temps de développement, la simplicité de la méthode et la plus fonctionnelle.

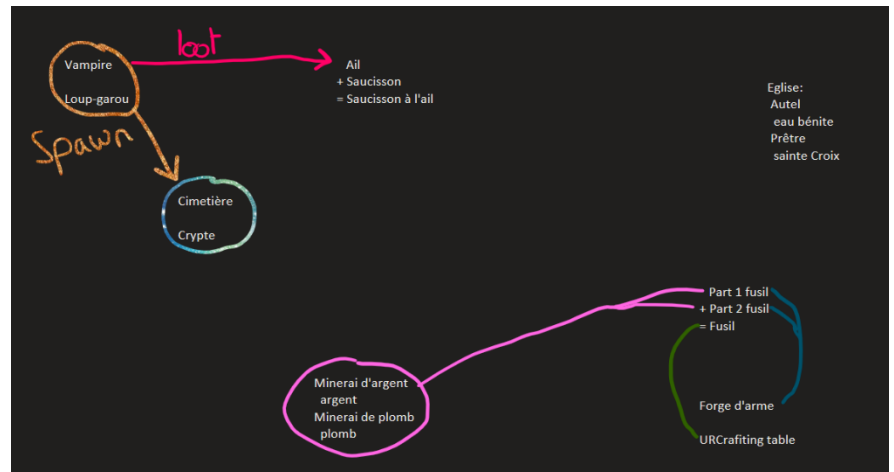
## Conception

Pour la conception du mod, nous avons fait étape par étape. C'est-à-dire que nous avons commencé par ce qui était nécessaire pour créer le reste : les minerais, les premières créatures auxquelles nous avons pensé lors de la première réunion, ainsi que le biome. Et pour URCAmod2 la conception était de faire la base du mod avant de créer quoi que ce soit, puis de créer un nouveau bloc et enfin d'attacher une nouvelle variable aux joueurs.

## Architecture de conception

L'architecture de conception pour URCAmod1 fut plutôt simple, car c'est MCreator qui fait toute la base du mod à notre place, nous avons juste à rajouter ce que nous voulions, avec des procédures si nous voulions créer un comportement spécifique.

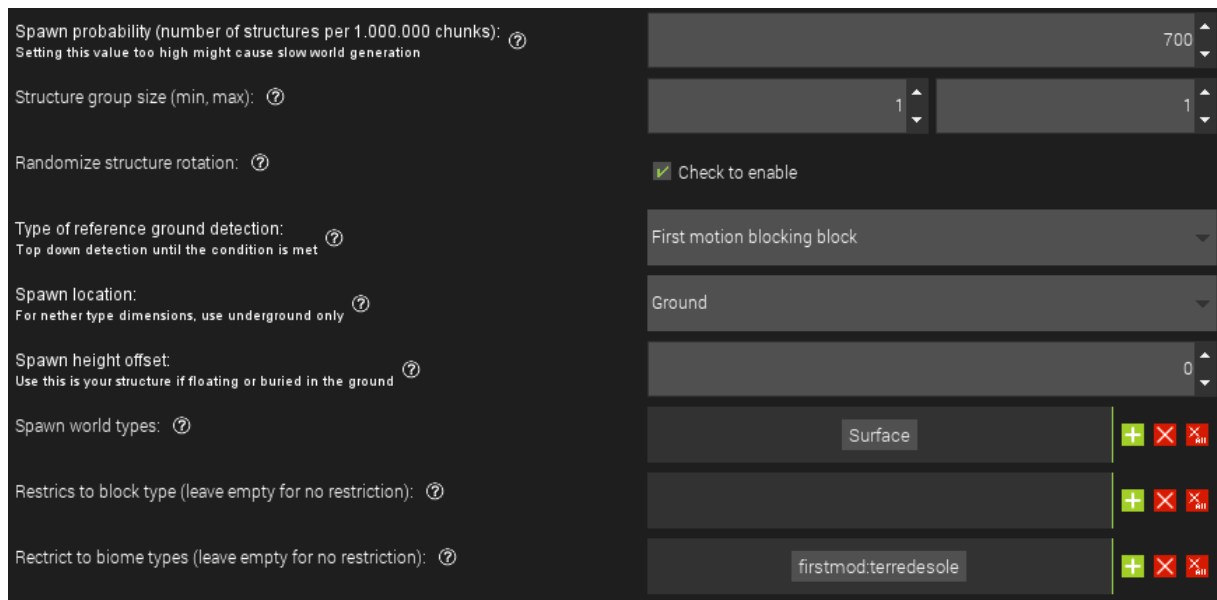
Du côté des idées, nous sommes partis avec ce simple schéma :



Comme nous pouvons le voir, nous avons déjà des idées sur les relations que pouvaient avoir certaines fonctionnalités.

## Des structures

Pour les structures, nous avons utilisé des structures blocs qui sont dans le jeu depuis plusieurs versions, ce qui nous a permis de faire énormément de choses. Nous avons réussi à faire en sorte que chacune d'elles soit trouvable, mais rare, voire très rare pour certaines. Et tout cela a été fait depuis ce panel :



Grâce à ce panneau, nous pouvons régler le taux de génération de structure, appliquer des restrictions au niveau des dimensions, et du biome dans lequel elles génèrent. Également, le nombre de structures par groupes, et si elles suivent le même axe de rotation ou laisser la rotation se faire au hasard.

## Des monstres

Pour les monstres, passifs ou agressifs, cela se déroule de la même façon sauf à un choix, où nous avons choisi de faire nos monstres uniquement agressifs.

Pour bien les régler, nous avons 6 panneaux :

The screenshot shows the configuration interface for a mob named "Loup Garou". The interface is divided into several sections:

- Name of entity:** Loup Garou
- Entity model:** Biped
- Texture file of entity:** oup\_arou-64x32.png
- Entity model bounding box:** Width/Depth: 0.6, Height: 1.8, Shadow Size: 0.5. There is a checkbox for "Disable collision box".
- Spawn egg options:** Enable, base color, dot color, creative tab. The base color is set to 102,102,102. There are buttons for "PINK" and "PROGRESS".
- Equipment (optional):** Main hand, off hand, helmet, body, leggings, boots. Only works for Biped and Zombie models.
- Entity label:** (leave blank to hide it)
- Living sound:** entity.generic.hurt
- Hurt sound:** entity.generic.hurt
- Death sound:** entity.generic.death

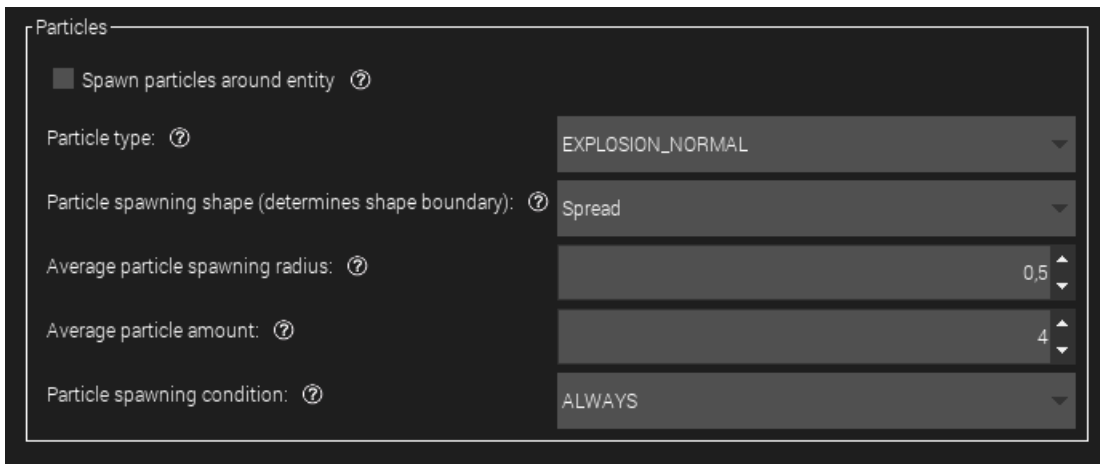
Le premier où nous pouvons régler le modèle, le nom, la texture, la hitbox, la texture de l'œuf, si ces monstres qui possède une barre de boss ou non ainsi que les sons.

Les modèles de monstres sont durs à développer, tellement que le modèle que Tristan a voulu développer pour le boss des loups-garous a moyennement marché, pour cause, il fallait augmenter la taille de chaque section du modèle et changer les textures, car elles n'allaient plus du tout avec.

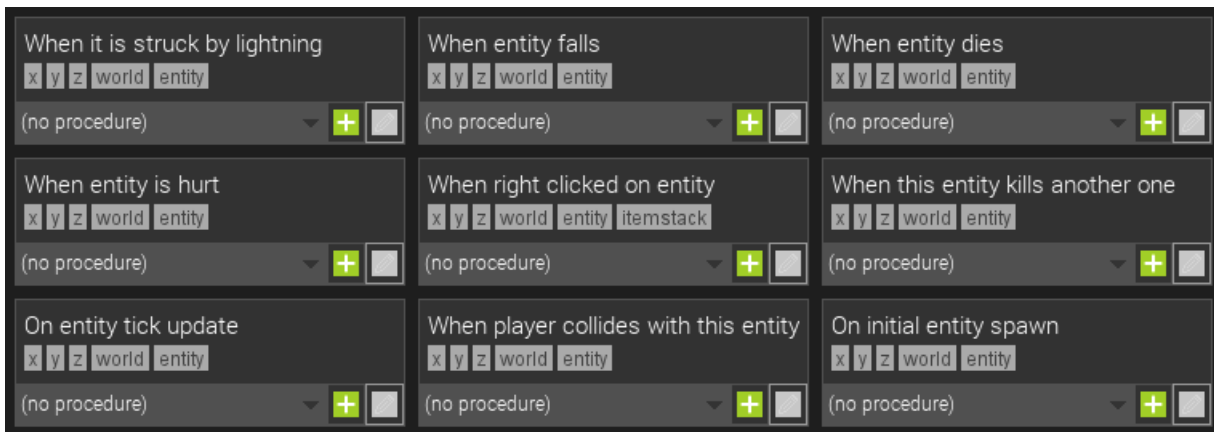
The screenshot shows the configuration interface for a mob named "Mob". The interface is divided into several sections:

- Behavioural characteristics:** Mob is aggressive, Creature is passive.
- Creature type:** UNDEFINED
- Entity health value, experience amount:** 25
- Movement speed, tracking range ("render distance"):** 0.3
- Mob default drop:** Drop is optional, leave blank for no drop
- Attack strength, armor protection base value:** 3
- Entity immune to:** Fire, Arrows, Lightning, Fall damage, Cactus, Drowning, Potions, Direct player attack.
- Check to make this entity rideable by player:** You can optionally turn on living entity controls too. Options: Rideable, Forward movement control, Strafe movement control.
- Check if this entity is water entity:** NOTE: You still need AI tasks to make mob use these properties. Option: Is water entity (breathes underwater, is not pushed by water flow).
- Check if this entity is flying entity:** NOTE: You still need AI tasks to make mob use these properties. Option: Is flying entity (gravity does not pull it down, does not take fall damage).

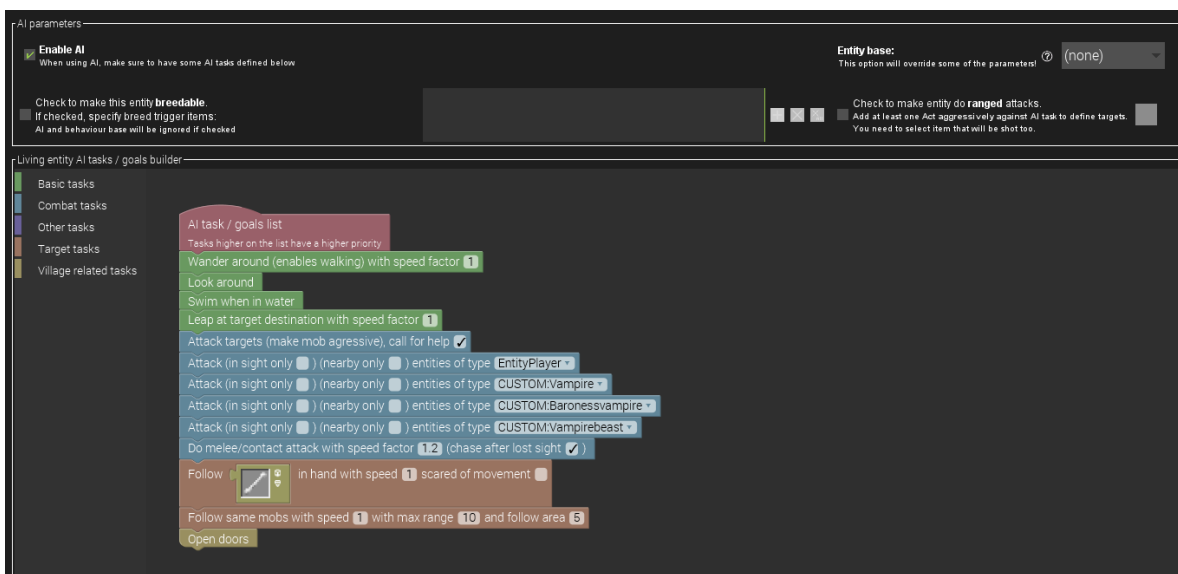
Dans ce deuxième panneau, nous devons choisir entre faire une créature agressive ou non, avec le premier choix. Et nous pouvons définir un drop personnalisé, la vie, l'expérience qu'il apporte une fois tués, la vitesse de ses mouvements, à quoi il est vulnérable, si nous pouvons monter la créature, si elle peut nager ou voler.



Dans ce troisième panneau, nous réglons les particules que peut émettre la créature créée. Donc le type, le rayon, le nombre et quand.



Sur ce quatrième panneau, nous pouvons donner des procédures, des codes en scrachs, aux créatures, à chaque ticks (secondes dans le jeu), quand il tombe, quand il meurt et autres.



Le cinquième panneau est très important, car c'est lui qui décidera de comment agira la créature en fonction de ce qu'elle a autour d'elle. Ici on peut voir que cette créature, le loup-garou, est attirée par les os, mais qu'elle attaque également les joueurs et tous les vampires.

Entity spawning properties

- Enable entity spawning: ☒
- Despawn when idle: ☒
- Spawn weight: 20
- Mob spawning type (affects the spawning location and time range): monster
- Minimal number of entities in a spawn group: 5
- Maximal number of entities in a spawn group: 10
- Biomes where the entity spawns (empty for no restriction): firstmod:terredesole
- Does this entity spawn in dungeons? ☐ Spawn in dungeons

Sur ce sixième et dernier panneau, nous pouvons régler si la créature se génère naturellement dans le monde, par groupe ou encore où elle peut spawner.

## Des objets

Il y a différents types d'objets, les nourritures, les blocs, les outils, les plantes, etc.

## Les nourritures

Pour les nourritures, comme dit dans la précédente partie, nous devons spécifier leurs saturations, et leurs valeurs nutritionnelles. Mais il faut commencer, comme un près tout dans MCreator, il faut donner une texture et un modèle 3D à cette nourriture :

Food 3D model

Item model:  
Select the item model to be used. Supported: JSON, OBJ

BUILT IN Normal

Food texture

Puis nous pouvons lui donner des caractéristiques propres à elle seule :

Name in GUI: Toast Food

Creative inventory tab: FOOD

Stack size: 64

Nutritional value: 8

Saturation: 9

Is food for dogs? ☐ Check to enable

Is always edible? ☐ Check to enable

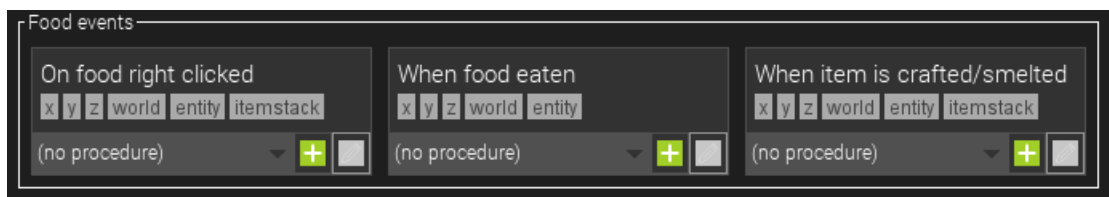
Has glowing effect? ☐ Check to enable

Eating speed: 32

Food animation: eat



Et si nous le voulions, rajouter des procédures, quand elle est cliquée avec le bouton droit de la souris, quand elle est mangée, ou quand elle est cuite ou fabriquée :



Pour toutes les nourritures, nous n'avons utilisé que la texture et le deuxième panneau.

## URCAmod2

L'architecture de conception pour URCAmod2 est la suivante :

Package : com.github.broussk.URCAmod2;

Urcamod2
String MODID : urcamod2
Logger LOGGER :
LogManager.getLogger(MODID)
Urcamod2()

ModEventSubscriber
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

## Le fonctionnement logique

### Des monstres

Pour les monstres nous pouvons leur faire ressembler à ce que nous voulions, seule leur forme est difficile à faire, c'est ce que nous avons essayé de faire avec le monstre « Wolf » qui est le boss des loups dans les tanières. Nous pouvons également leur faire avoir le comportement que nous voulions qu'ils aient, comme montrés dans la partie précédente avec le 5<sup>e</sup> panneau.

Chaque monstre est unique, car ils ne possèdent pas tous le même comportement ni la même texture.

Pour le boss Mr Rabat, nous avons fait le choix de pouvoir le faire apparaître grâce à un bloc, le Rabat Spawn Bloc, qui coûte chère au joueur et qui ne peut être invoqué que dans son monde, permis toutes les créatures qu'il a pu assouvir pendant son règne.

### Objets

Les objets sont assez faciles à faire, sauf certains, par exemple si nous voulions faire une culture de quelque chose, il fallait faire une procédure pour chaque étape, par exemple la culture de l'ail :

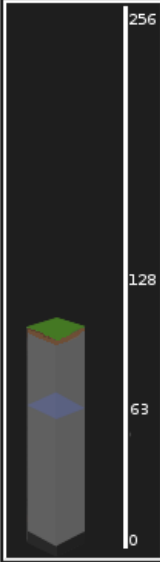


Voici une partie de la procédure visant à faire pousser la culture de l'ail, nous avons donc utilisé des variables aléatoires, ainsi que la vérification du niveau de luminosité sur la culture, et affecté une chance de pousse pour chacune des étapes. Mais avant de savoir si la culture peut pousser, il faut savoir si elle est sur le bon bloc, que nous avons décidé, sinon la culture va arrêter de grandir et disparaître du bloc sur lequel elle était posée.

## Blocks

Les blocs ont leurs propres fonctionnements, par exemple nous pouvons leur dire de se générer à la place de pierre, si c'est des minerais, entre telle et telle couche, ainsi que la dimension dans laquelle ils peuvent générer :

Block spawning/generation properties



Dimensions to generate in (leave empty to disable spawning): ?	Surface	CUSTOM:RabatWorld	+	×	ALL
Blocks this ore can replace: ?	Blocks.STONE		+	×	ALL
Restriction biomes (leave empty for no restriction): ?			+	×	ALL
Average amount of ore groups per chunk: ? <small>Setting this value too high (&gt;32) might crash world generator</small>			4		
Average number of ores in a group: ? <small>Setting this value too high (&gt;32) might crash world generator</small>			9		
Minimal generation height: ?			5		
Maximal generation height: ?			50		

Nous pouvons également leur spécifier le type de bloc qu'ils sont, leurs sons, à quoi ils servent, leurs résistances, ce qui traduit avec quels outils nous pouvons les casser et les récupérer, s'ils ont un drop customisé, ou encore leurs rajouter des particules.

Advanced block properties

Tick rate: ?	10
Tick randomly: <small>Ticks randomly with global tick rate factor</small>	<input type="checkbox"/> Check to enable
Block color on the map: ?	DEFAULT
Can plants sustain on this block? ?	<input type="checkbox"/> Check to enable
Can this block be beacon base? ?	<input type="checkbox"/> Check to enable
Does block act like ladder? ? <small>Block must not be full for this to work</small>	<input type="checkbox"/> Check to enable
Does redstone connect to this block? ?	<input type="checkbox"/> Check to enable
Enchantments power bonus: ?	0
Block flammability: ?	0
Fire spreading speed: <small>Leave 0 for vanilla handling</small>	0
Block slipperiness: ?	0.6
Reaction to being pushed: ?	NORMAL

Particle properties

<input type="checkbox"/> Spawn particles around block ?	
Particle type: ?	EXPLOSION_NORMAL
Particle spawning shape: ? <small>Determines shape boundary</small>	Spread
Average particle spawning radius: ?	0.5
Average particle amount: ?	4
Particle spawning condition: ?	ALWAYS

Où s'ils ont une interface, comme la table de craft du mod, cela est géré ici :

Bind this block to GUI:  Optional: block can have container but no GUI

Open bound GUI on right click: ☐ Enable

☐ Enable tile entity and inventory on this block  
Check this box if you want any of these: block that can contain items, block that can store NBT data, block that can interact with hoppers and/or comparators

Block's tile entity and inventory settings (for chests, machines and blocks with NBT data)

Size of inventory (slot count):  If the block is bound to GUI, set this value to the *biggest slot ID in the GUI* + 1

Max size of stack:

Drop items from inventory when block destroyed: ☒ Enable

Enable block output comparator data: ☒ Enable

Comma separated list of IDs of output slots These slots won't be used for storage by hoppers  ?

Nous pouvons même leurs rajouter des procédures, selon les évènements qui sont résumés ici :

<b>On block right clicked</b> <input type="text" value="x y z world entity direction"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>	<b>When block added</b> <input type="text" value="x y z world"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>	<b>When neighbour block changes</b> <input type="text" value="x y z world"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>	<b>Update tick</b> <input type="text" value="x y z world"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>
<b>When block destroyed by player</b> <input type="text" value="x y z world entity"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>	<b>When block destroyed by explosion</b> <input type="text" value="x y z world"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>	<b>When player starts to destroy</b> <input type="text" value="x y z world entity"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>	<b>When entity collides in the block</b> <input type="text" value="x y z world entity"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>
<b>When entity walks on the block</b> <input type="text" value="x y z world entity"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>	<b>When block is placed by</b> <input type="text" value="x y z world entity itemstack"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>	<b>Redstone on</b> <input type="text" value="x y z world"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>	<b>Redstone off</b> <input type="text" value="x y z world"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>
<b>Client display random tick</b> <input type="text" value="x y z world entity"/> (no procedure) <input type="button" value="+"/> <input type="checkbox"/>			

## Structures

Pour les structures, comme dans dit dans la partie précédente, nous pouvons régler le taux de génération des structures, et même rajouter une procédure par rapport à ceci. Malheureusement nous n'avons pas tout à fait compris comment elles se généraient et il se peut qu'il y ait des structures générées de manières bizarres, comme celle-ci :



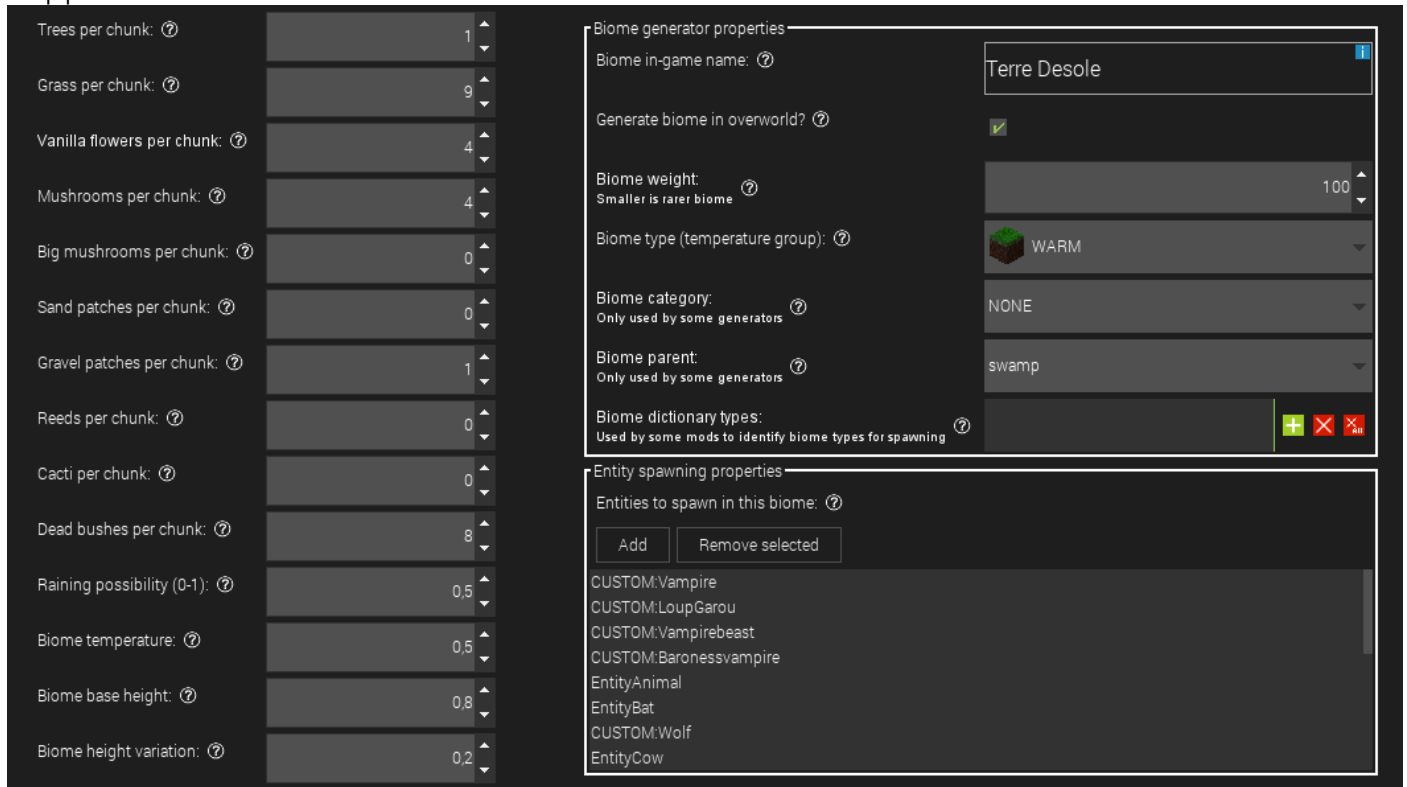
Où il y a un «trou», un vide, sous la structure.

La crypte est composée de plusieurs salles, toutes différentes, avec un boss à l'intérieur, ce qui rend le parcours dans celui-ci plus complexe. D'autant plus que dans chaque salle, il y a au minimum 1 spawner à monstre redoutable.

## Du biome

Le biome est là pour être le centre même du mod, c'est à partir du biome que nous avons construit tout le reste, les monstres, les structures, etc.





Comme nous pouvons le voir, un biome est composé de deux panneaux, chacun gérant une partie de celui-ci. Le premier, permet de mettre tout ce qui est aspect, ambiance, par exemple avec de l'eau couleur rouge, de l'herbe bleu foncé et de l'air légèrement gris, pour donner un aspect brumeux. Le deuxième panneau, a pour but de fixer les spécificités du biome, comme sa hauteur, et des tas de réglages pour chaque chunk.

Ce biome est le lieu où nous pouvons retrouver la crypte qui est un donjon à étage et labyrinthe dans lequel les joueurs peuvent obtenir des équipements enchantés en argent.

## Un fluide

Nous avons créé un fluide spécial, appelé «eau bénite», elle est spéciale car elle tue toute créature, passives ou agressives mais vous soigne, ce qui est un allié parfait dans le mod. Cependant ceci en fait un objet très déséquilibrant le jeu, donc nous avons régler cela en la mettant sous forme de source unique, ce qui veut dire que les joueurs ne peuvent en faire de source d'eau infinie comme avec l'eau normale. Il ne peut se trouver que dans les chapelles, qui sont très rares, ce qui équilibre le jeu.

## URCAmod2

Nous démarrons par la première classe urcamod2 appelée 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 à forge que cette classe est un mod et qu'il doit être chargé au lancement du jeu.

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

Pour ajouter des objets, nous créons la classe ModEventSubscriber dans le même package que le main et nous l'annotons de @EventBusSubscriber, celle-ci indiquant à forge que la classe contient des méthodes qui doivent être subscribed pour gérer des événements. Il a en paramètre le mod id et le bus : EventBusSubscriber. Finalement, nous avons 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 entrees à la fin de la classe ModEventSubscriber.

Comme dit précédemment, pour créer et enregistrer un nouvel objet, dans la méthode onRegisterItem, on fait appel à la fonction registerAll et de setup dans lequel on instancie le nouveau bloc.

Ensuite pour créer l'event le plus important «eventVirus», nous avons eu besoin d'une nouvelle variable associée aux joueurs. C'est pourquoi nous avons créé 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 vraie, l'ajout de la saisie de l'information dans le log est très utile.

Pour être infecté par le virus, il faut donc, attaquer un joueur infecté ou inversement, ce qui rend l'accessibilité, non optimal étant donnée la nature du jeu, un contact devrait pouvoir suffire est serait utilisable dans différents modes de jeux.

## Bilan

### État du développement

#### URCAmod1

Le mod est opérationnel, nous avons tout fait pour ce qui ne marchait pas ne puisse pas ruiner le jeu du joueur. La dimension fonctionne parfaitement, les monstres créés agissent bien comme nous le voulions, nous avons réussi à bien équilibrer le jeu, à rajouter de l'exploration et des défis à surmonter.

Par rapport aux idées de bases, nous avons abandonné l'idée de faire fabriquer les armes par le biais d'une forge et en plusieurs morceaux que l'on aurait assemblés. Nous avons également abandonné l'idée de mettre un villageois prêtre avec lequel nous pouvons faire un échange contre une croix bénite, nous avons remplacé cela par une croix bénite dans un cadre sur un des murs de la chapelle.

#### URCAmod2

On lance le jeu et on peut avoir les blocs créés avec la commande /give. Les joueurs peuvent tomber malades, grâce à la variable qui leur est rajoutée, avec le contact à un autre joueur atteint par la maladie.

### Difficultés rencontrées

#### URCAmod1

La difficulté principale rencontrée par tous était le logiciel, car personne ne le connaissait et avait déjà fait un mod. La secondaire, plus relative au projet, était d'équilibrer le jeu entre ce que tout le monde faisait. Les armes ne devaient ni être trop puissantes, ni trop faciles à manipuler. Les nouveaux boss devaient être forts pour le défi du mod et obtenir l'objet ultime : la tête d'un Mr Rabat. Les donjons devaient eux aussi donner beaucoup de défis pour permettre aux joueurs de s'aventurer correctement dans l'aventure avec de bons équipements.

#### URCAmod2

La partie la plus compliquée était celle pour attacher une nouvelle variable aux joueurs, 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 al forme d'une compétence qui augmenterait avec de l'expérience qui s'acquiert avec l'exposition aux autres joueurs contaminés.

### Pistes d'amélioration

#### URCAmod1

Nous pouvons améliorer le mod en rajoutant du contenu, notamment au niveau de la dimension Rabat, qui en manque.

#### URCAmod2

Nous pourrions par exemple faire en sorte que les animaux peuvent avoir le virus et le transmettre aux joueurs, ou encore afficher avec une petite barre de chargement où en est la maladie sur le joueur, au niveau de son interface principale.



## Conclusion

Nous sommes tous heureux de voir le résultat de tant d'heures de travail et de test, surtout avec les conditions sanitaires, qui ont pu ne pas nous permettre de travailler dans de bonnes conditions. Mais nous avons tous appris durant ce projet que ce soit au niveau de la gestion d'équipe, de langage de programmation, d'utilisation d'API, ou de la création graphique.

Nous pouvons dire que nous sommes arrivés à notre but premier, et d'assouvir les demandes du mod quant aux thèmes que nous avions fixés. Il resterait encore du chemin pour faire le mod parfait et découvrir, maîtriser toutes les facettes de MCreator et de Forge, mais nous en avons vu une bonne partie pour que nous puissions prendre du recul sur ceci.

## Le petit message de chacun sur le projet

### Andy

« Étant le premier à avoir choisi dès que le sujet était disponible, je suis très content de l'équipe qu'il m'a été donné d'avoir, tout travailleur et très enthousiaste sur ce projet original et très agréable à mettre en œuvre »

### Jennyfer

« J'ai trouvé le projet intéressant et assez différent de ce que j'ai pu faire jusqu'à maintenant. Tout était bien organisé grâce au chef de projet. Les membres ont eu une très bonne idée de mod et le résultat est en accord avec ce que l'on souhaitait. »

### Tristan

« Bien que je ne sois pas d'accord avec Andy, puisque c'est moi le premier à avoir choisi, je suis plus que satisfait de l'équipe. À dire vrai, ayant choisi un projet lié à Minecraft je m'attendais à ce que les membres de l'équipe soient moins travailleurs. Et pourtant, alors même que le confinement battait son plein tous sans exception ont fourni des efforts. Je ne suis pas celui qui en aura fourni le plus, cet honneur revient à Nathan, ayant pour ma part mes propres occupations pendant le confinement (m'occuper de l'école à domicile pour mon fils et de ma femme enceinte). Les membres de l'équipe ont aussi bien pris ce besoin et je ne me suis jamais senti forcé ou oppressé en travaillant sur le projet. J'ai orienté ma partie de travail sur la construction de structure, m'orientant dans une partie Game Design que j'apprécie particulièrement et je suis plus que satisfait du travail fourni. Je pensais que ce projet serait à la base un véritable calvaire à cause des conditions spéciales (crise sanitaire) et que je ne serais pas avec des personnes que je connaissais ou/et qui accepteraient de travailler à fond sur le projet. Il n'en était rien et je suis particulièrement fier du travail fourni et de notre idée de mod. »

### Nicolas

« Il m'a été agréable de travailler avec le reste de l'équipe, et le fait d'avoir un chef de projet aide beaucoup à l'organisation, malgré le fait que l'on a dû travailler à distance, ce qui ne rend pas la tâche plus facile. »

### Renaud

« Je ne pouvais pas espérer meilleur sujet que "créer un mod Minecraft" pour projet d'informatique. Ayant prévu de me réorienter en science pour l'ingénieur, je portais à la base peu d'intérêt au projet et je m'attendais à ce que personne ne travaille, je n'avais pas l'intention de travailler non plu pour être honnête, mais quand on m'a dit qu'il y avait un projet sur Minecraft, je m'y suis aussitôt inscrit. Coup de chance il restait une place ! Contrairement à l'idée que je m'en faisais, le projet a été pour moi une réussite et un enrichissement personnel inattendu. Cela a été possible grâce à une équipe motivée et organisé notamment grâce à Nathan notre chef de projet qui a fourni avec passion un travail admirable, je juge ne pas avoir fourni un gros travail par rapport à lui. Je me suis personnellement occupé de toute la partie modélisation 3D, j'ai trouvé nos idées de mod très passionnantes, et la modélisation 3D est quelque chose de familier pour moi. Je me suis senti très inspiré et dans mon univers. Je suis fière d'avoir eu un projet aussi bien abouti et plaisant comme dernière expérience de licence informatique. »

### Nathan

« Le projet m'a apporté beaucoup en gestion d'équipe, en création de contenus tant graphiques que numériques avec les procédures dans MCreator et les rapports. J'ai adoré travailler avec toute l'équipe, même si je déplore le peu de travail de Renaud qui s'est mis à travailler de temps en

temps à la fin. Cela fut une très bonne expérience. La partie qui m'a le plus retenu c'est les cultures, à penser comment imbriquer les petites briques de scratch pour qu'aux finales tout marche comme nous le voulions, je trouve cela génial. Tout le monde à toucher à un peu tout dans le mod, je suis notamment très fier du travail réalisé par Jennyfer, qui a travaillé toute seule sur URCAmod2, car nous avons eu des problèmes pour que je puisse travailler avec elle, malgré nos tentatives pendant 1 semaine de tout bien mettre en place, cela ne marchait toujours pas, donc nous avons abandonné cette idée et je me suis concentré à 100 % sur URCAmod1. Comme l'a dit Tristan, il a travaillé comme il a pu avec son fils et sa femme enceinte, je ne lui donnais pas énormément de tâches à accomplir, cependant il les a toutes réussies comme nous voulions. Le travail de Andy et Nicolas est parfait, ils se sont impliqués à fond dans le projet, comme je l'attendais d'eux. Tout cela pour dire que nous formions une bonne équipe.»

## Glossaire

Chunk : section du monde faisant 16\*16\*256.

Feedback : retour sur le travail fourni.

Mob : créature qui peut être passive, agressive ou neutre.

Biome : grande section d'un monde, faisant une taille aléatoire.

Hitbox : endroit où il faut appuyer sur le bloc, ou le mob pour le toucher.