GAME OVERVIEW DOCUMENT

2GD - 1er semestre

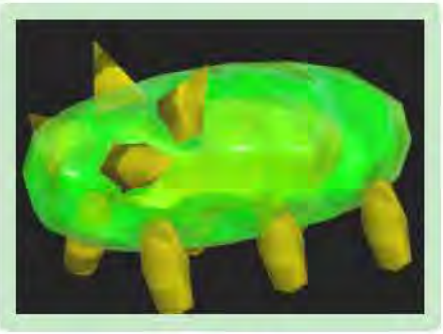
Matthieu COLIN - Barthélemy ISLIWA - Ethan LANCELLE - Nathan RAUH

Equipe

COLIN
Matthieu
Lead Game programmer
Game designer
Sound designer

ISLIWA
Barthélemy
Lead Game designer
Game programmer

LANCELLE
Ethan
Co-Lead Game artist
Game programmer



RAUH
Nathan
Co-Lead Game artist
Game designer

TABLE DES MATIERES

Introduction

Pitch

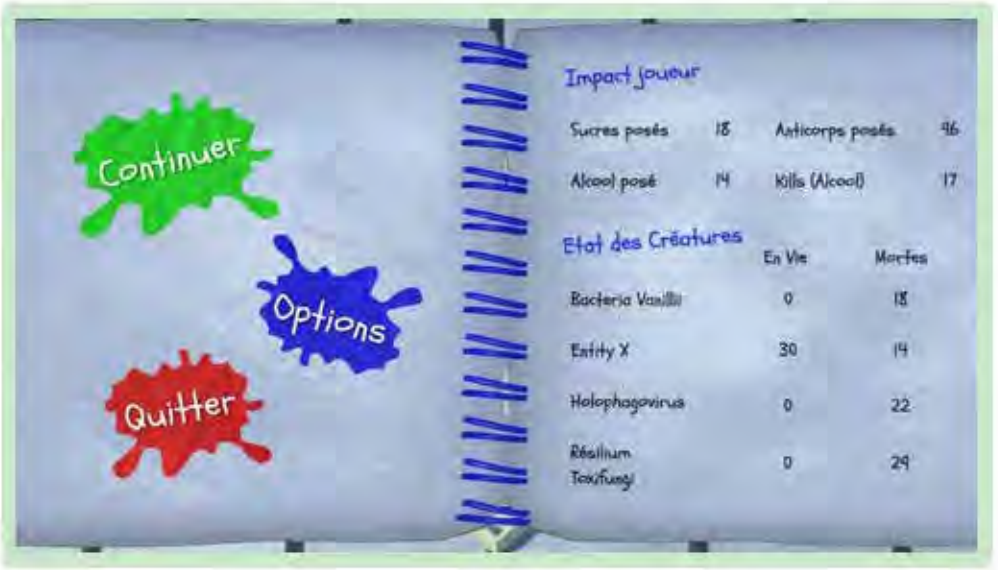
Dans ce "god game", vous êtes un scientifique, observez une boîte de Petri remplie de multiples espèces hostiles entre elles et dotées chacune de leurs propres comportements. Essayez au mieux de les comprendre, pour y parvenir vous avez plusieurs outils vous permettant d'influencer, favoriser, détruire et créer... Peu importe vos décisions, ces petites créatures sont à votre merci et vous êtes maître de leur destin.

Univers

Protocol PRV se déroule dans un monde en pâte à modeler très colorée. L'univers est loufoque.

Cible :

Casual,10-26 ans~



Game Design

Intention

Notre intention de design est de donner aux joueurs l'impression d'endosser le rôle de scientifiques.

Nous cherchons à nourrir la fantaisie de l'expérimentation en leur donnant la possibilité d'observer et d'agir afin de tirer des conclusions sur le comportement des créatures, conclusions qui forgeront les futures expériences du joueur. Nous prenons également en compte l'aspect posé et calme de notre toy.

Tous les outils mis à disposition du joueur ont été conçus spécifiquement pour être imprécis afin d'accentuer l'aspect expérimental, où il est souvent compliqué de tout maîtriser avec précision.

3Cs

Camera

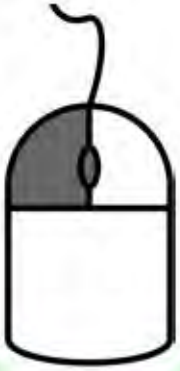
La caméra est en topdown vue perspective avec une illusion d'orthographique grâce à une technique qui consite à éloigner grandement la caméra des objets importants mais d'à la fois avoir un champ de vision très bas (FOV : 20).

La caméra est fixe au-dessus du terrain de jeu.

Controller

Les contrôles sont très simples.
Seulement la souris est utilisée :

- **Clic gauche** - sélectionner/utiliser
- **Molette** - sélection alternative



Character

Le personnage joué est un personnage de jeu « god game ». Il est omnipotent par rapport à l'espace de jeu.

Il peut influencer les créatures en utilisant les outils à sa disposition.

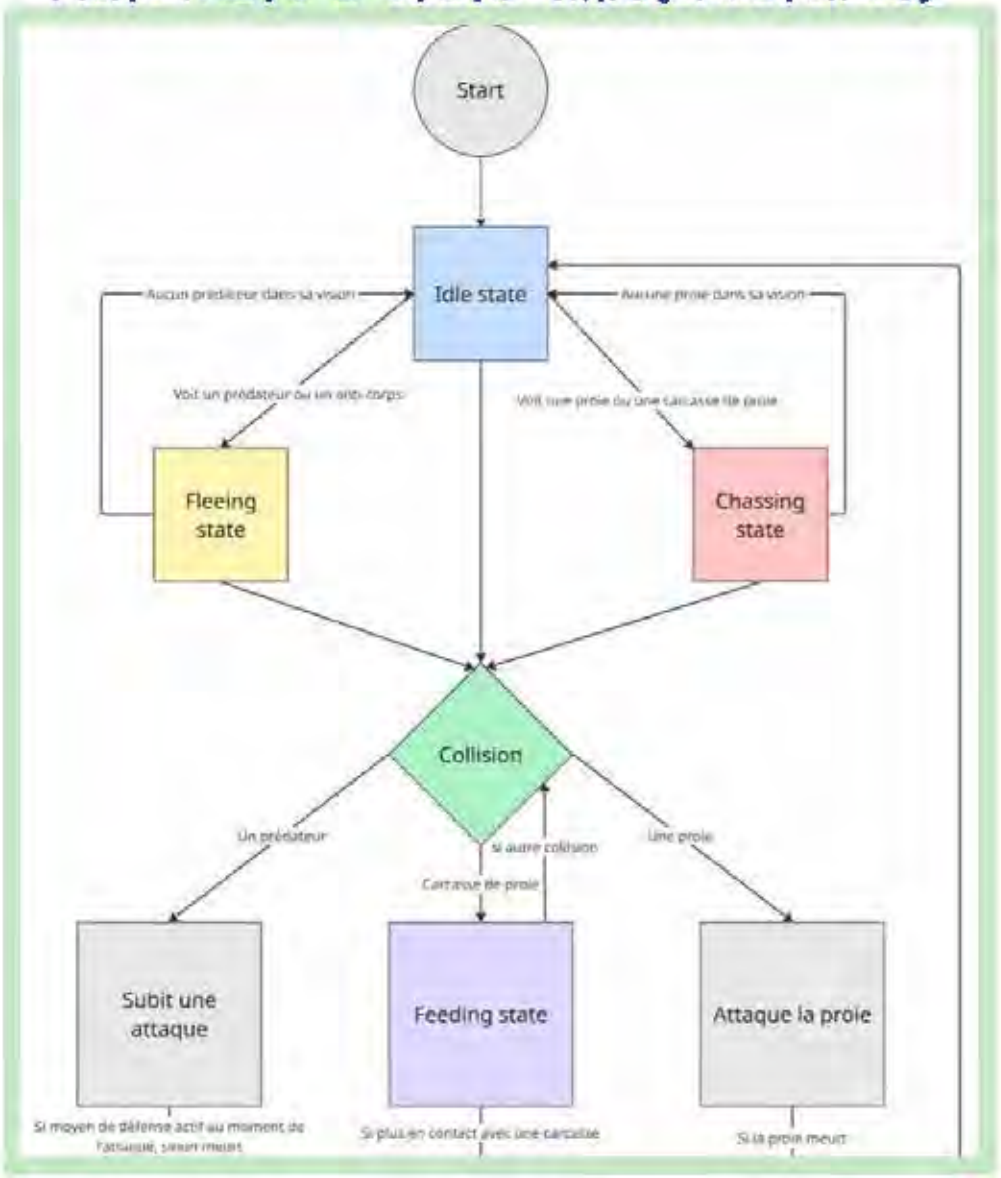
Mécanique et systèmes

Créatures et comportements

Le jeu est composé d'un terrain, de quatre espèces différentes de créatures et de plusieurs outils à la disposition du joueur (cf. Outils).
Chaque créature agit différemment, mais plusieurs comportements sont partagés entre les espèces.

Idle state	= Se déplace en avant et change de direction de manière aléatoire dans son FOV.
Chassing state	= Gagne un boost de vitesse et se déplace en direction de la proie
Fleeing state	= Gagne un boost de vitesse et se déplace dans la direction opposée du prédateur
Feeding state	= Ne se déplace plus et mange, une fois assez nourrit la créature fait naître une autre créature de son espèce

Automate à états finis(créatures)



Système de cadavre :

Une créature qui se fait chasser par un prédateur laisse un cadavre à sa mort.
Chaque créature peut se nourrir des cadavres des espèces qu'elle chasse. Une fois suffisamment nourrie, la créature donne naissance à une autre créature de son espèce.

1. Avant impact

2. Vanillii devenu un cadavre

3. Holo consomme le cadavre

4. Holo fait naître un congénère !

Les quatres espèces

Bacteria Vanillii

Spécificité

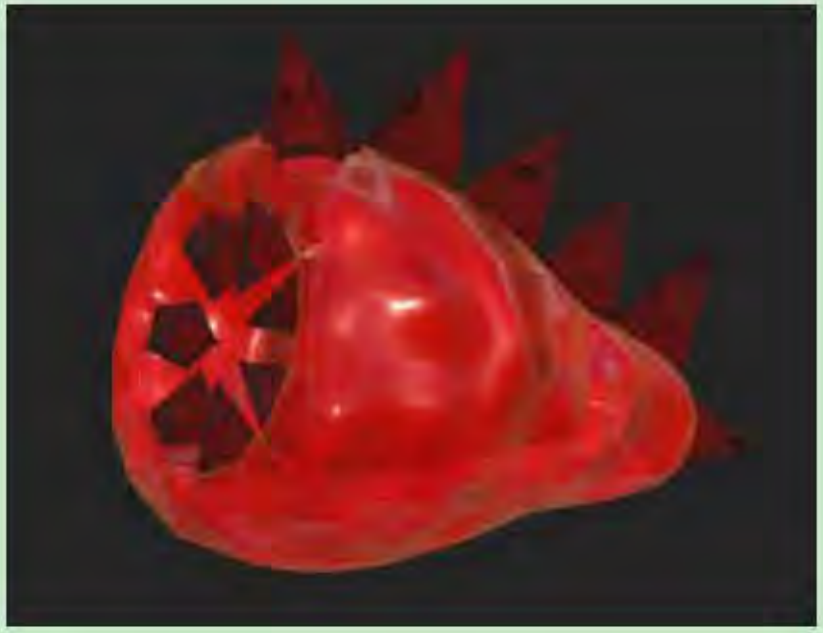
Chasse deux espèces différentes, à l'instar des autres espèces qui n'en chassent qu'une.

Proies

Entity X, Resilium Toxifongi

Prédateur

Holophagovirus



Les quatres espèces

Entity X

Spécificité

Cette créature possède trois états physiques différents. Au contact d'une créature de son espèce et du même état, elles s'absorbent et forment une seule entité plus grande et plus lente.

Différents états

- **Niv.1** L'état de base, vitesse normale, taille normale. Meurt au contact de son prédateur.
- **Niv.2** L'état atteint lorsque deux Niv.1 entrent en contact, vitesse faible, grande taille. Se divise en deux créatures de Niv.1 au contact de son prédateur.
- **Niv.3** L'état atteint lorsque deux Niv.2 entrent en contact, vitesse très faible, très grande taille. Se divise en quatre créatures de Niv.1 au contact de son prédateur.

Proies

Resilium Toxifongi

Prédateur

Bacteria Vanillii

Les quatres espèces

Holophagovirus

Spécificité

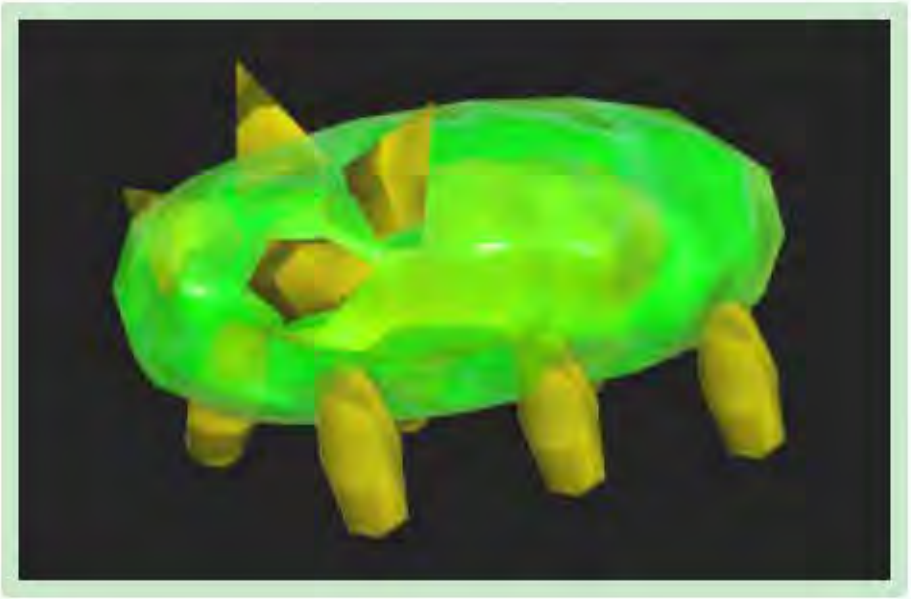
Cette créature peut manger toutes les carcasses, sauf celles de son espèce. À sa mort, sa carcasse contient plus de nutriments.

Proies

Bacteria Vanillii

Prédateur

Resilium Toxifongi



Les quatres espèces

Resilium Toxifongi

Spécificité

Cette créature, au contact de son prédateur, gagne en vitesse et libère un nuage de toxines empoisonnant toutes les créatures des autres espèces à proximité. Une créature empoisonnée est plus lente. Si elle est touchée par son prédateur avant le retour de sa poche toxique, elle meurt.

Récupération du nuage toxic

Récupération après s'être suffisamment nourrie (cadavres).

Proies

Holophagovirus

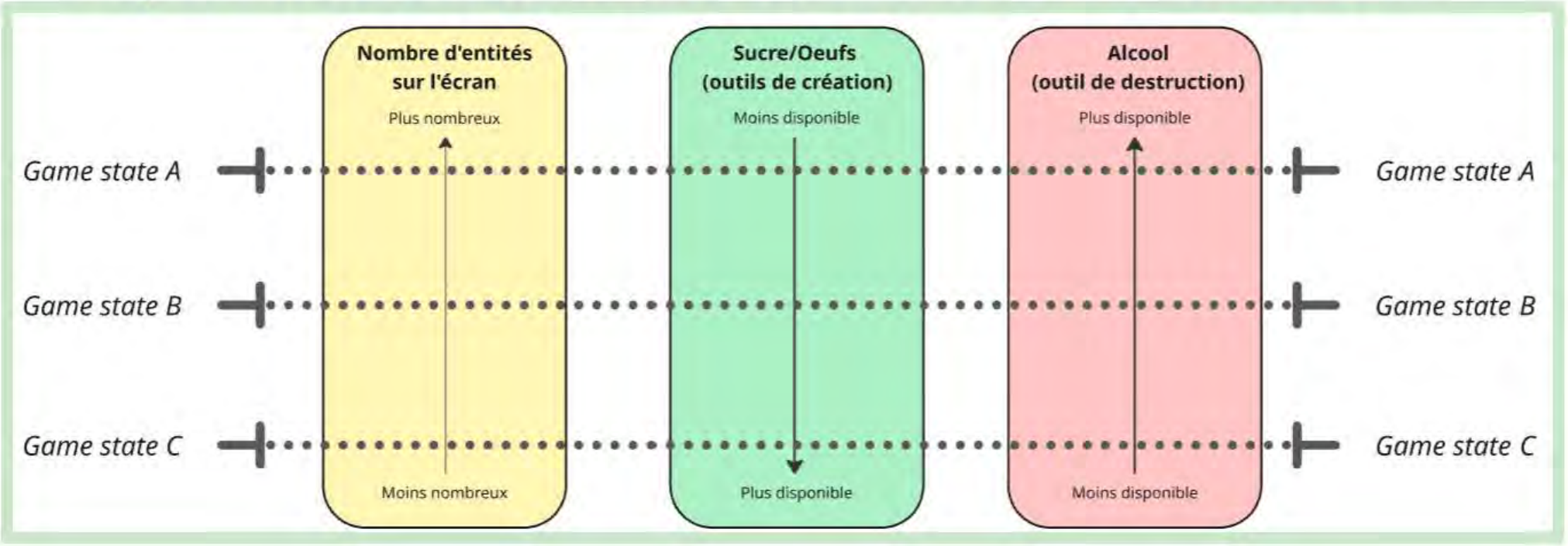
Prédateur

Bacteria Vanillii

Outils

Les joueurs possèdent plusieurs outils à leur disposition. La disponibilité de ces outils varie en fonction du game state actuel et, plus spécifiquement, du nombre de créatures.

Disponibilité des outils en fonction du nombres de créatures à l'écran



Descriptions des outils

Les joueurs peuvent sélectionner n'importe quel outil. Cliquer sur le terrain permet d'utiliser l'outil sélectionné.

Morceau de sucre

Description

L'utiliser dépose un morceau de sucre sur le terrain et attire l'attention des créatures. À son contact, les entités le consomment et se multiplient instantanément.

Durée de vie

Infinie. Disparaît si consommé.

Anticorps factice

Description

L'utiliser dépose un anticorps factice sur le terrain et effraie toute créature qui le voit.

Durée de vie

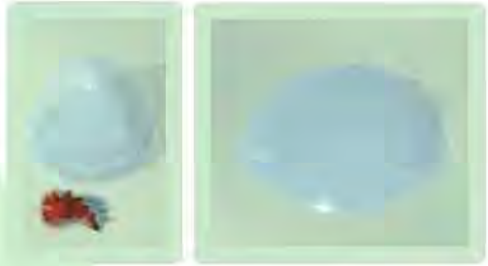
40 secondes.

Gel hydroalcoolique

Description

L'utiliser fait tomber une goutte de gel hydroalcoolique sur le terrain. Au contact, les créatures meurent sans laisser de carcasses.

Durée de vie : 0.5s de latence puis 0,5s actif.



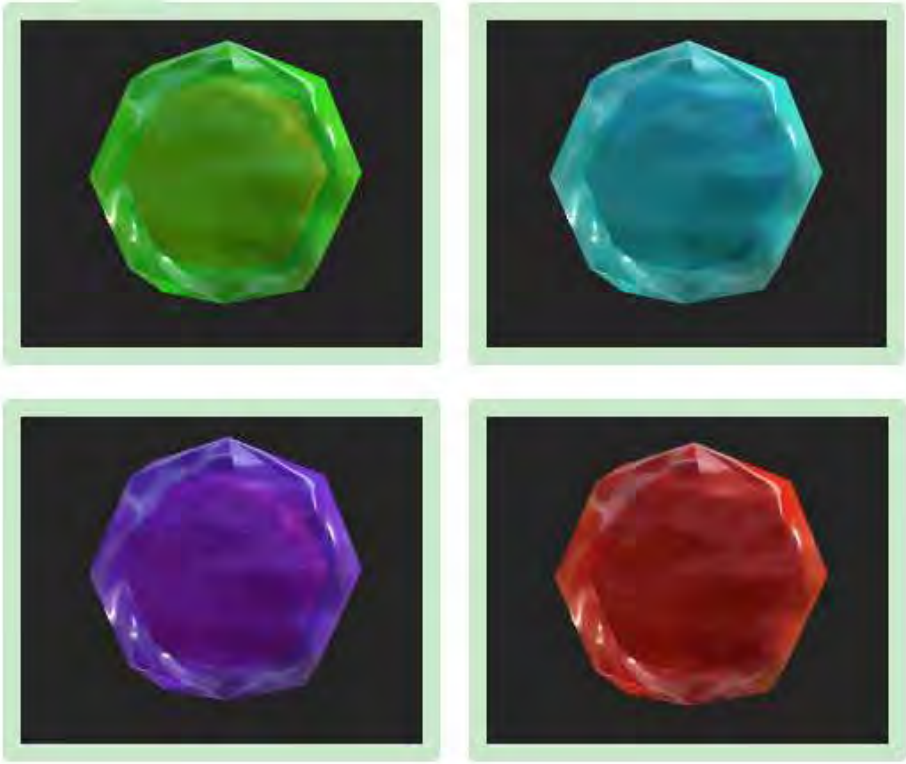
Souches

Description

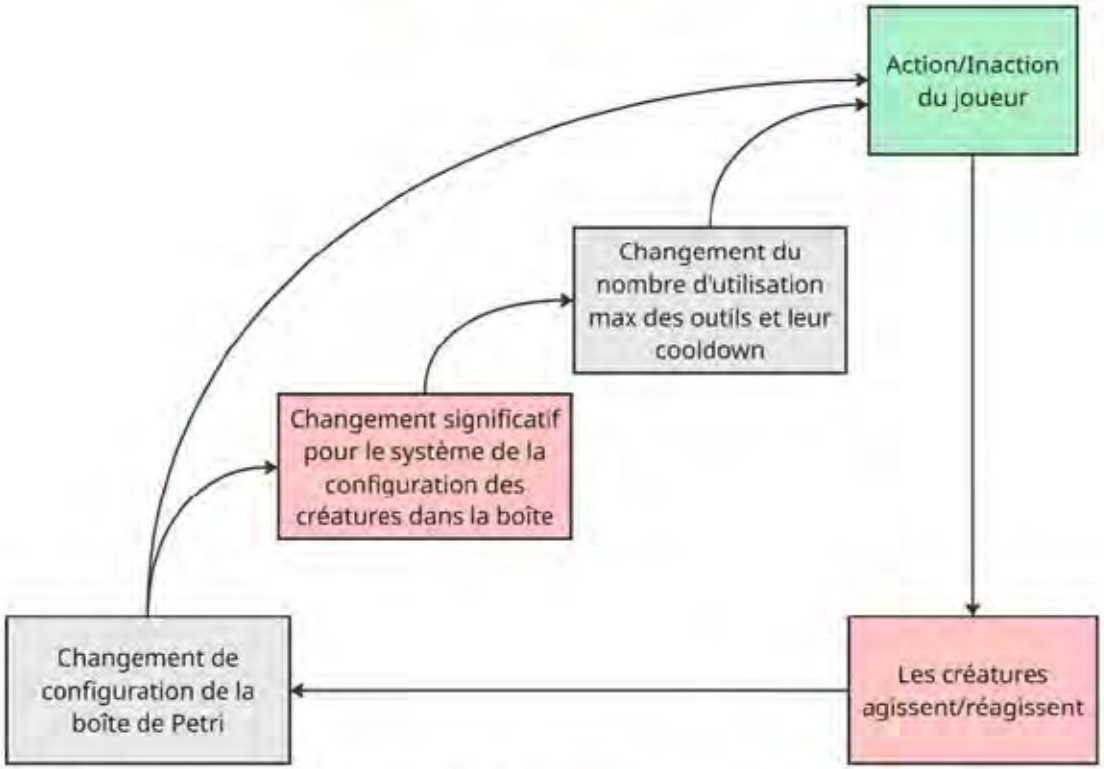
Il existe une variante pour chaque espèce. L'utiliser dépose trois souches qui, après un certain temps, donnent chacune naissance à un individu de l'espèce.

Durée de vie

Chacune disparaît disparaît lorsqu'elle donne naissance à un individu.



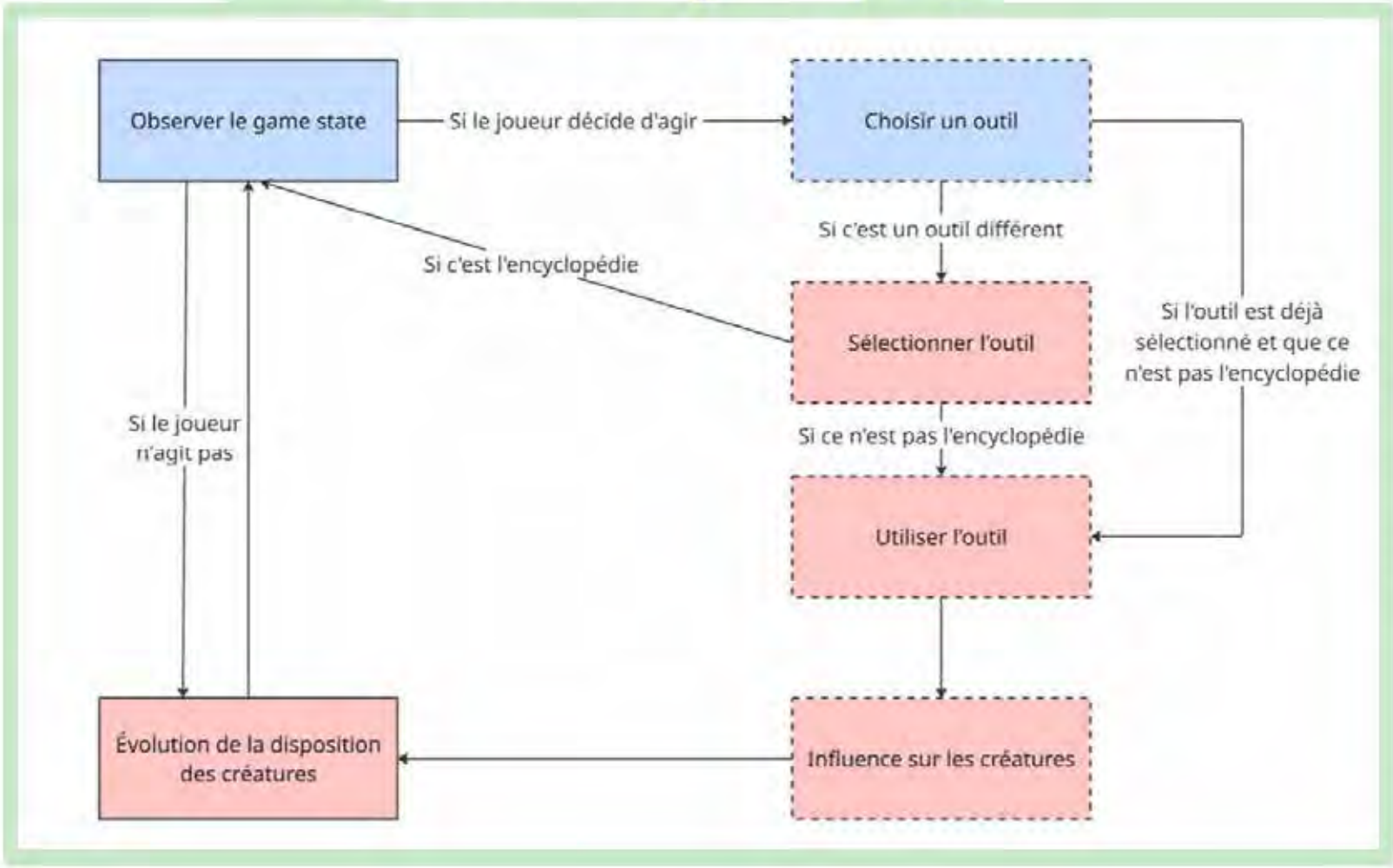
Métaboucle



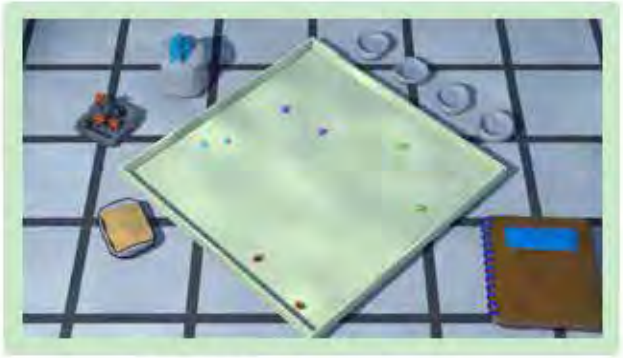
La configuration de la boîte de Petri change constamment. En revanche, si ces changements ne sont pas significatifs pour le système (ex : il ne reste plus qu'une seule espèce dans la boîte), la boucle ne contient que 3 étapes.

Lors de forts changements, la boucle contient 5 étapes : le système réagit à ce changement et régule en conséquence la disponibilité des outils.

Boucle de gameplay

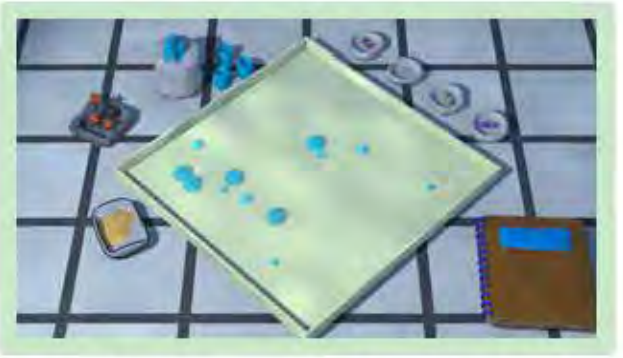


Evolution d'une partie type en plusieurs game state



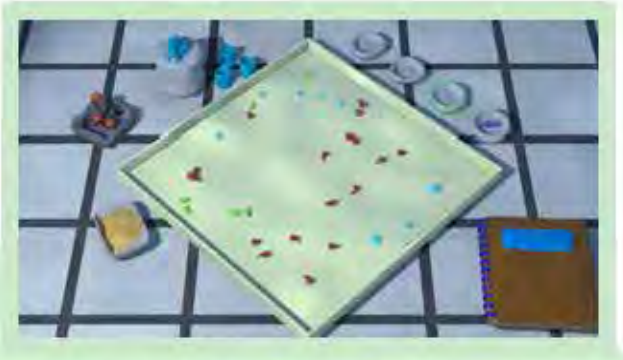
1. Début de partie

2. Environnement favorable pour les Entity X

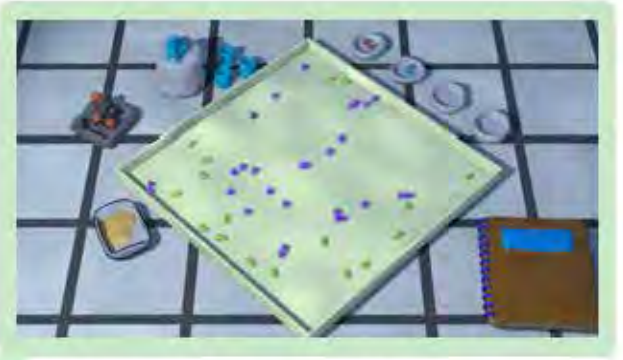


3. Domination Entity X

4. Carnet (statistiques)



5. Réintroduction des Vanillii et des Holo



6. Réintroduction des Toxi

- Extinction Vanillii et Entity X
- Domination Toxi imminente

Tableau de signes et feedbacks 1/2

Action in game	Parties interagissantes	Signes	Actions du joueur	Feedbacks visuels			Feedbacks sonores	Priorité
				Couleur	Forme	Technique		
Souris cliquée	Joueur	/	Clic Gauche	/	/	/	Son de clic	Basse
Hovering d'un bouton du menu / du carnet	Joueur/Bouton	Le bouton en question est blanc, la souris n'est pas dessus	Passer la souris sur un bouton	Noir	Forme du texte du bouton	/	/	Basse
Apparition de créature	Créature	Les conditions d'apparition d'une créature sont réunies (souche, contact avec sucre/cadavre)	/	Couleur de la créature	Forme de la créature	/	Son de "Pop" en 8 variantes	Haute
Changement de state de créature	Créature/Environnement	Une créature a dans son champ de vision des proies/prédateurs/anticorps/sucres	/	/	/	Animation Rig / Transform / SoftBody	/	Haute
Disparition de créature/ Apparition de cadavre	Créature/Créature	Une créature se fait toucher par son prédateur	/	Couleur de la créature morte	Forme de la créature morte	Simulation de "Cloth"; Animation par Shape Keys	Son d'objet spongieux écrasé en 4 variantes	Haute
Consommation de cadavre	Créature/Cadavre	Une créature est en contact avec un cadavre d'une de ses proies	/	Couleur du cadavre	Forme du cadavre	Diminution de la scale par code	Son de mâchouillage en 4 variantes	Moyenne
Apparition de sucre	Joueur/Sucre	Du sucre est disponible, le bouton "Sucre" est sélectionné	Clic gauche sur le terrain alors que le bouton "Sucre" est sélectionné	Marron	Petit tas de sucre	Simulation de "Cloth" avec rotation aléatoire sur Y	Son de "Pop" + Son de sucre versé en 4 variantes	Haute
Apparition d'anticorps	Joueur/Anticorps	Le bouton "Anticorps" est sélectionné	Clic gauche sur le terrain alors que le bouton "Anticorps" est sélectionné	Orange/Noir	Tour/totem avec base en forme de Y	Style pâte à modeler	Son de "Pop" + Son de bois frappé en 4 variantes	Haute
Disparition d'anticorps	Anticorps	L'anticorps est déjà resté sur le terrain pendant 40 secondes	/	Orange/Noir	Tour/totem avec base en forme de Y	Diminution de la scale par code	/	Moyenne
Apparition d'une goutte d'alcool	Joueur/Alcool	Le bouton "Alcool" est sélectionné, de l'alcool est disponible	Clic gauche sur le terrain alors que le bouton "Alcool" est sélectionné	Blanc translucide	Goutte	Simulation de "SoftBody"; Animation par Shape Keys	Son de "Pop" + Son de goutte	Haute
Action de l'alcool sur les créatures	Alcool/Creatures	Des créatures se retrouvent dans le champ d'action de l'alcool 0,5 secondes après son instanciation	/	Blanc translucide	Flaque	Simulation de "SoftBody"; Animation par Shape Keys	Son de bacon qui grille, volume module selon le nb d'entités touchées	Moyenne
Disparition de la goutte d'alcool	Alcool	L'anticorps est déjà resté sur le terrain pendant 0,5 seconde.	/	Blanc translucide	Flaque	Diminution de la scale par code	/	Moyenne
Apparition de souches	Joueur/Souches	Une souche est disponible	Clic gauche sur le terrain alors qu'un bouton "Souche" est sélectionné	Couleur de la souche liée à la créature	Trois sphères	/	Son d'œufs craqués	Haute
Tentative infructueuse d'instanciation	Joueur	L'objet dont le bouton est sélectionné n'est pas disponible	Clic gauche sur le terrain alors qu'un bouton au contenu limité est	/	/	/	Son de buzzer joué deux fois	Moyenne

Tableau de signes et feedbacks 2/2

Action in game	Parties interagissantes	Signes	Actions du joueur	Feedbacks visuels			Feedbacks sonores	Priorité
				Couleur	Forme	Technique		
Augmentation de la quantité dispo de:								
Sucre	Sucre	Le nb de sucre disponible est plus petit que sa valeur maximale, et un temps t dépendant du nb de créatures vivantes est passé depuis la dernière recharge	/	Marron	Tas de sucre	Déplacement du tas en Y	Son de glaçon tapant sur du verre, raccourci	Haute
Alcool	Alcool	Le nb d'alcool disponible est plus petit que sa valeur maximale, et un temps t dépendant du nb de créatures vivantes est passé depuis la dernière recharge	/	Bleu	Verre à shot	/	Son de bouteille qui verse du liquide	Haute
Souches	Souche	La créature liée à la souche en question n'a plus d'individu en vie.	/	Couleur de la souche liée à la créature	Trois sphères	/	Son de micro-ondes qui sonne	Haute
Hovering d'un bouton in game	Joueur/Bouton	Le bouton en question n'a pas d'outline, la souris n'est pas dessus	Passer la souris sur un bouton	Blanc	Forme du bouton	Shader Outline	/	Basse
Sélection d'un bouton in game	Joueur/Bouton	Le bouton en question a une outline, la souris est dessus	Clic gauche sur un bouton / Scroll de la molette	Noir	Forme du bouton	Shader Outline	Son différent selon le bouton	Haute
Ouverture du carnet	Joueur/Carnet	Le carnet a une outline, la souris est dessus	Clic gauche sur le carnet	Blanc/Gris	Livre ouvert	Style pâte à modeler	Son de stylo écrivant sur du papier	Moyenne
Fermeture du carnet	Joueur/Carnet	Le bouton "Retour" est noir avec la souris dessus, ou la souris est hors des limites du livre	Clic gauche hors du carnet /sur le bouton "Retour"	/	/	/	Son de page qui se tourne	Basse

RGD

Protocole PRV, par sa nature de toy, n'a pas d'objectif prédéterminé.

Chaque joueur peut se donner ses propres objectifs. Vous trouverez ci-dessous plusieurs boucles OCR qu'un joueur type pourrait rencontrer.

Ces situations ont été constatées lors des playtests.

Objectif	Challenge	Paramètres Atomiques	Reward	Intérêt
Garder un équilibre dans la biodiversité	Décision, Timing, Précision	Quantité de sucre/alcool disponible Temps de recharge du sucre/alcool Vitesse des créatures Taux de rafraîchissement de la direction des créatures	L'équilibre des espèces est momentanément sauvegardé, permettant d'avoir à l'écran des visuels de chaque espèce et comportement	Haute
Accélérer la suprématie d'une des créatures	Décision, Timing, Précision	Quantité de sucre/alcool disponible Temps de recharge du sucre/alcool Vitesse des créatures Taux de rafraîchissement de la direction des créatures	La créature choisie par le joueur est la seule présente dans la Boîte	Moyenne
Créer une zone de "quarantaine" pour contenir quelques individus	Précision, Reflexe	Taille des anticorps Temps de vie des anticorps Vitesse des créatures Taux de rafraîchissement de la direction des créatures	Un ou plusieurs individus se retrouvent bloqués, entourés par des anticorps	Basse
Maximiser une/plusieurs des statistiques du carnet	Tactique, Timing et Précision pour les kills	Quantité de sucre/alcool disponible Taille de la zone d'effet de l'alcool Temps d'activation de la zone d'alcool Temps de recharge du sucre/alcool Nombre de souches disponibles par recharge Vitesse des créatures Taux de rafraîchissement de la direction des créatures	La statistique recherchée voit son nombre augmenter.	Moyenne

Références de game design

Les outils imprécis

L'aspect imprécis des outils à disposition du joueur nous vient de jeux comme *September 12th* ou *We Become What We Behold*, où chaque action du joueur peut entraîner des conséquences involontaires, en raison de la difficulté d'utilisation et l'imprévisibilité des entités simulées.

September 12th - 2015

We become what we behold - 2016

Simulation d'entités hostiles

Une des parties les plus importantes de notre jeu est l'aspect chasse/fuite de nos entités à la rencontre de prédateurs/proies. La première inspiration est le jeu de groupe Poule Renard Vipère (jeu qui donnera aussi à *Protocole PRV* son nom). *Universal Paperclips* propose, comme mini-jeu, un système de combat entre des entités de deux factions qui fut l'une des inspirations. *Agar.io* repose essentiellement sur l'interaction et la relation de chasse/fuite entre les joueurs. Bien que ce ne soit pas des IA, ce jeu fut une des premières inspirations.

Universal Paperclips - 2017

Agar.io - 2015

Gestion d'entités variées et tension

Dans *Bad North*, le joueur contrôle et gère des entités au combat. Cette dynamique crée une tension et un challenge pour le joueur, il faut être à la fois rapide et prendre les bonnes décisions. Il était aussi important pour nous d'avoir une asymétrie entre les espèces, dans *Rimworld* chaque PNJ a des habitudes et comportements différents. Les joueurs peuvent plus ou moins gérer les PNJ mais leurs différents comportements créent des situations imprévisibles.

Rimworld - 2013

Bad North - 2018

Direction Artistique

Intentions

Vision globale

Nous voulions établir une ambiance pseudo-scientifique volontairement niaise, faisant écho aux expériences que nous réalisions petits, chez nous ou en cours de découverte des sciences, à la fin de la primaire.

La direction artistique cherche à évoquer la vision du monde scientifique à cet âge, où l'amusement et l'expérimentation priment sur la rigueur et la réalité scientifique.

Le joueur n'est pas face à une simulation réaliste, mais face à un terrain de jeu qu'il se serait construit lui-même enfant, ou du moins le souvenir qu'il en aurait aujourd'hui, en jouant au jeu.

Positionnement scientifique

Scientifiquement, le jeu est volontairement éloigné de la réalité. Ce n'est pas un serious game, ni un jeu réaliste ou à but éducatif. La science est ici un support visuel pour aider à la compréhension, mais pas une finalité scientifique actée.

La direction artistique respecte son rôle de vecteur de compréhension du système, tout en renforçant l'immersion et l'ambiance générale du jeu.

Points clés de la DA

La direction artistique se devait de respecter plusieurs choses, dont voici une liste.

- Elle doit évoquer de la curiosité face à un écosystème vivant et inconnu, donnant envie au joueur d'observer.
- Elle doit donner envie de s'amuser et de tout essayer, grâce à des formes simples, des couleurs vives et des animations rigolotes.
- Elle se doit aussi d'avoir une grande lisibilité, ainsi qu'une simplicité de compréhension car le toy a un grand potentiel chaotique.
- Pour ce qui est de l'UI, elle doit être intradiégétique, afin de ne pas sortir le joueur du monde dans lequel il se plonge.

Matériaux utilisés

Nous avons fait le choix d'utiliser des matériaux identifiables afin de donner une identité visuelle forte au jeu.

Deux matières principales structurent l'ensemble de la direction artistique : la gelée, réservée aux créatures, et la pâte à modeler, pour l'UI.

Tous les matériaux utilisés pour le projet on été faits à la main, avec du node scripting sur Substance Designer.

Gelée

Pâte à modeler

Cristal Sucré

Gelée

La gelée est le matériau principal des créatures.

Ce matériau s'inscrit dans un imaginaire enfantin et rappelle les bonbons, type Jelly Bears.

Elle évoque à la fois :

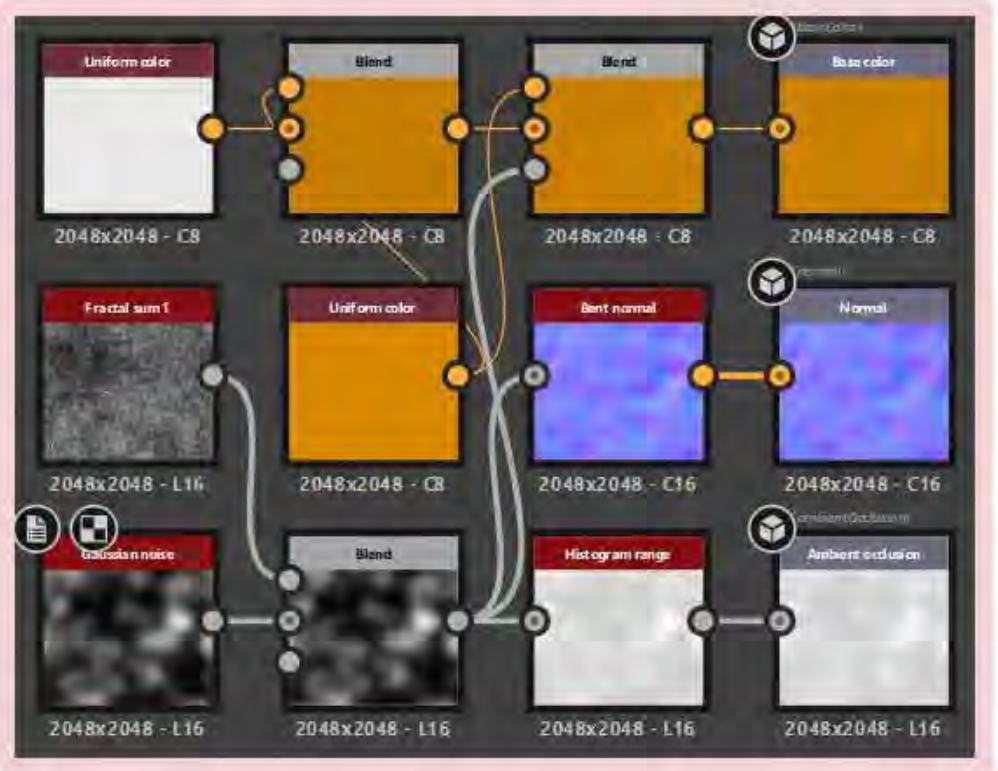
- Le cytoplasme des cellules
- Le mou, le sucré
- La ludicité et l'enfance

Sa transparence renforce son identité et facilite grandement la lecture des interactions entre les créatures, permettant des les voir les unes aux travers des autres.

Elle a été réalisée sur une base de bruit Gaussien, deux nodes de couleur et un d'opacité. Elle a également une roughness faible, pour lui donner de la brillance.

Gaussian Noise

Résultat Final



Le procédé nodal de la Jelly :

Deux noises sont envoyés dans les channels Color, Normals et Occlusion, clampés avant ce dernier pour éviter d'être trop sombre. Le second (Fractal sum) est désactivé de base, mais est présent sur la texture du sucre

Les Roughness, Metalness, Height et Opacity sont gérés de manière uniforme par un slider.

Coeur de la gelée

C'est la texture qui sert de coeur aux créatures, un équivalent du noyau d'une cellule.

Elle représente un cristal de sucre, comme les gommes à mâcher au centre de certains bonbons. Elle est moins visible en jeu, mais elle n'en reste pas moins importante.

Elle à été réalisée sur une base de noise "Crystal" et deux couleurs.

Totalement opaque et très peu brillante, les couleurs utilisées sont des dérivés des couleurs de la gelée pour chaque créature.

Cette texture est utilisée sur les éléments distinctifs de chacune des espèces, afin de contraster la créature et de renforcer son identité.

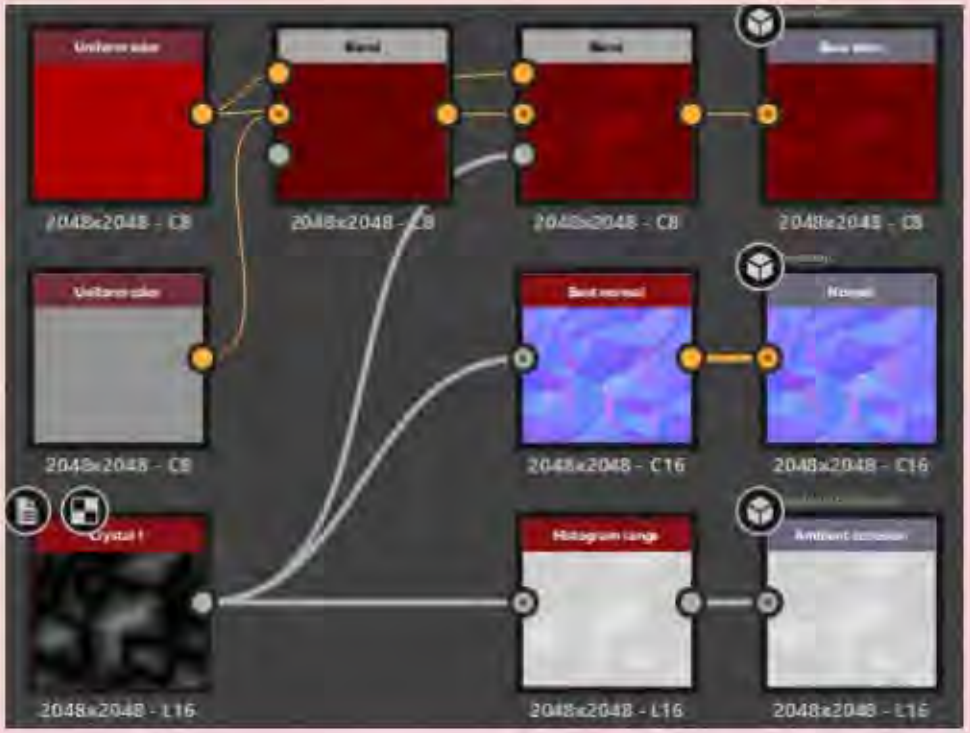
Crystal Noise

Résultat Final

Le procédé nodal du Coeur Crystal:

Un noise est envoyé dans les channels Color, Normals et Occlusion, clampé avant ce dernier pour éviter d'être trop sombre.

Les Roughness, Metalness et Height sont gérées de manière uniforme par un slider.



Pâte à modeler

La pâte à modeler est principalement utilisée pour l'UI et certains éléments manipulés par le joueur.

Elle rappelle le fait main, l'enfance et le bricolage, et donne cette impression que le décor et les outils ont été faits par l'avatar qu'il incarne dans cet univers et non du matériel professionnel.

Ce choix renforce le détachement vis-à-vis de toute cohérence scientifique stricte et assume un ton ludique et accessible.

Elle a également été réalisée sur une base de bruit gaussien, avec un noise "Clouds" pour ajouter les détails de rugosité.

Elle est beaucoup moins brillante que la gelée, et les couleurs utilisées sont très saturées.

Cloud Noise

Résultat Final

Le procédé nodal de la texture de pâte à modeler :

Trois noises différents sont superposés de différentes manières. Ils sont ensuite envoyés dans les channels de Color, Normals, Occlusion et Height.

Les Roughness et Metalness sont gérés de manière uniforme par un slider.

Créatures

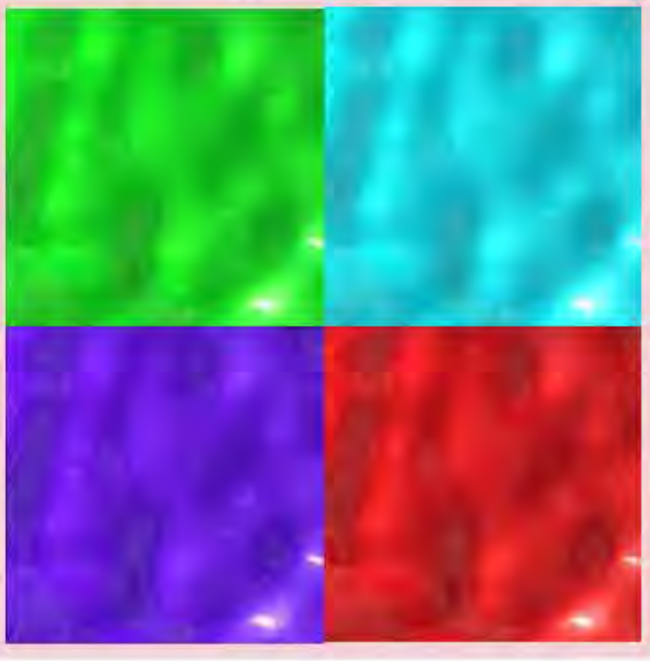
Formes

Chaque créature possède une forme simple mais évocatrice de son comportement. Les couleurs sont également réfléchies pour correspondre avec leurs comportements

Matériaux

Comme expliqué précédemment, ces créatures sont en gelée, en utilisant des couleurs basiques et saturées pour simplifier l'assimilation de leur comportement et de leur identité par le joueur.

La gelée rappelle également les gummy bears des petits bonbons mignons. Le coeur de cristal sucré est aussi utilisé pour renforcer leur identité.



Les Jellies

Les Sugar Crystals

Bacteria Vanillii :

Forme

Forme de cône couché avec des pics le long de son dos. Elle possède aussi des dents placées de manière circulaire et un coeur.

Ces éléments visuels traduisent son rôle de prédateur, qui découle de sa compétence unique à pouvoir attaquer deux créatures différentes.

Couleurs

Corps : Rouge pur

Core, Dents, Pics : Rouge foncé

L'utilisation du rouge est pensée pour démontrer l'agressivité de cette créature, son comportement de prédation.

Animations

Sa queue oscille de gauche à droite par à-coups, accentuant son comportement menaçant et le danger que cette espèce représente pour les autres.

Inspiration

Cette créature est inspirée de la sangsue, un animal qui même pour l'homme inspire une certaine crainte. En plus de cela, cet animal s'attaque à quasiment tout autre animal en aspirant son sang, ce qui permet d'appuyer l'utilisation du rouge pour ***Bacteria Vanilli***.

Entity X :

Forme

Forme de slime basique, qui possède un ou plusieurs coeurs en fonction de son stade d'assemblement.

Ces éléments symbolisent sa capacité à s'assembler avec ses congénères et à agrandir sa masse progressivement. Sa silhouette rend immédiatement lisible son comportement d'agglutination.

Couleurs

Corps : Cyan

Cores : Bleu pur

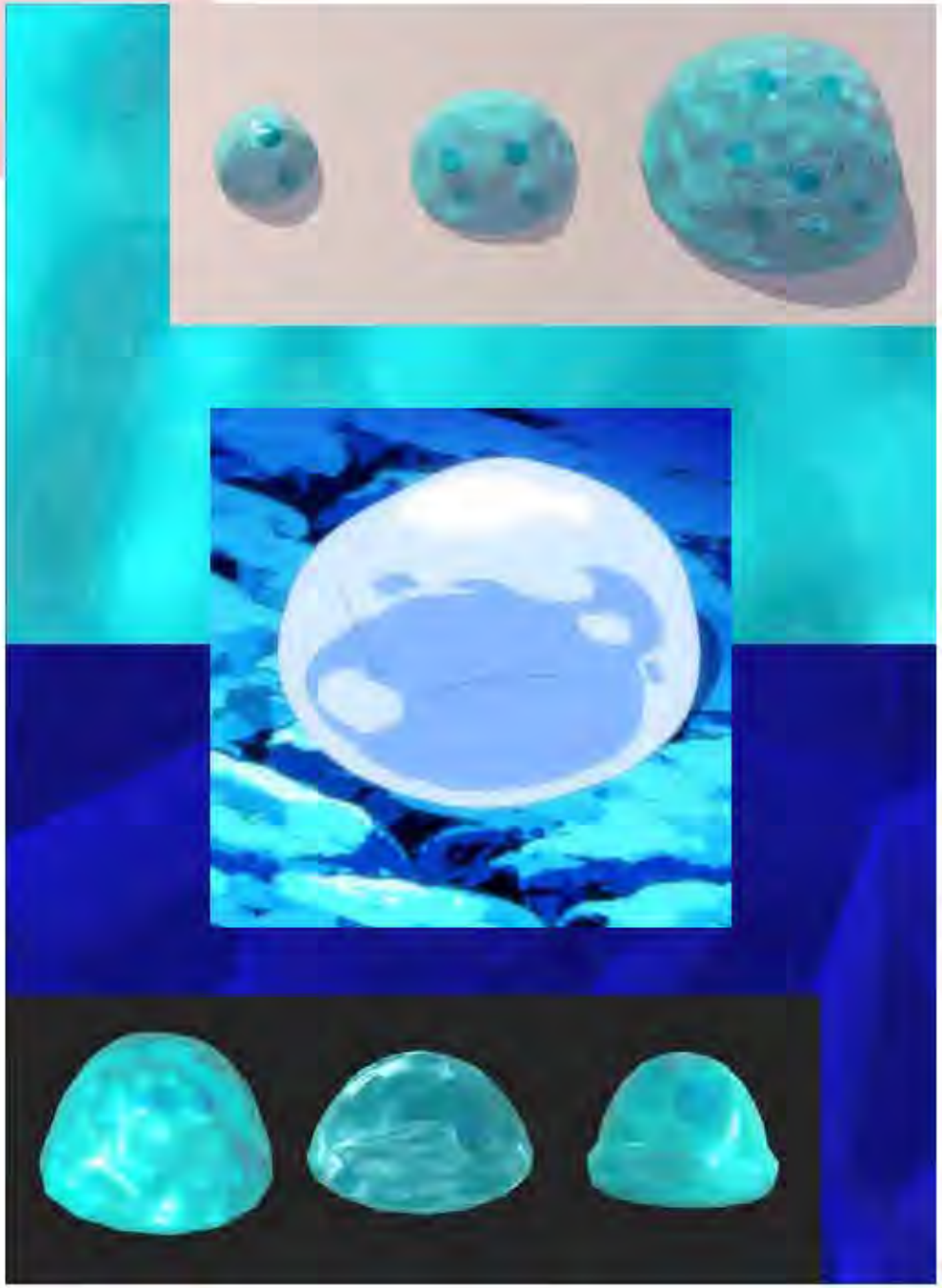
Le bleu est ici utilisé par référence, la plupart des slimes sont bleus, et il est en général utilisé pour les capacités de défense dans nombre de références diverses.

Animations

La créature effectue de légers rebondissements, et son ou ses noyaux se déplacent à l'intérieur, mettant en avant son aspect gélatineux et sa capacité d'accumulation.

Inspiration

Cette créature est inspirée du slime, une créature imaginaire souvent capable de se décomposer et de se reconstituer, faisaint souvent preuve de longévité.



Holophagovirus :

Forme

Forme cylindrique sur pattes, accompagnée de pics frontaux et d'un coeur.

Cette configuration traduit son rôle de créature grouillante et opportuniste, capable de consommer la majorité des carcasses et de générer beaucoup de ressources à sa mort.

Couleurs

Corps : Vert pur

Core, pics : Jaune

Ici, le vert permet d'appuyer le rôle de nettoyeur de cette créature, le symbole de vie prospère qu'elle représente. Le jaune rappelle sa vivacité et sert à lui donner un aspect dérangeant.

Animations

Ses pattes bougent rapidement à la manière des pattes des fourmis, avec un pattern bien précis. Son corps fait un mouvement de vague pour renforcer l'aspect grouillant.

Inspiration

Les chenilles/mille-pattes sont les inspirations majeures de cette créature, ayant un rôle important dans l'écosytème, tout en n'étant jamais très agréable à voir, à moins de s'y intéresser de près.

Resilium Toxifungi :

Forme

Forme ronde, accentuée par un grand coeur visible au centre et dépassant au sommet, ainsi que des petites pattes rondes à la base.

Le coeur est une poche de spores, indiquant sa capacité d'autodéfense, compréhensible après sa première activation.

Couleurs

Corps : Indigo

Core : Magenta

La base violette de ces couleurs suggère le mystère et la toxicité, mettant en avant sa capacité unique d'autodéfense au sein de l'écosystème.

Animations

Les petites pattes rondes tournent pour permettre le déplacement de la créature, simulant une lévitation. Pour illustrer son nuage de gaz, de simples particules roses sont relâchées sur le terrain.

Inspiration

Dans l'idée du monde microscopique, cette créature se base sur les champignons, mais une plus grande liberté d'interprétation a ici été prise.

Carcasses

Forme

Les carcasses des créatures perdent leur volume, comme une sorte de liquéfaction, conséquence directe de la nature gélatineuse des créatures.

Cela permet de conserver une continuité visuelle entre la créature vivante et son cadavre.

Animations

À la mort d'une créature, son corps se tasse et ses attributs tombent au sol, comme si la force vitale des créatures cessait de les tenir en place.

Les carcasses rapetissent progressivement au fur et à mesure qu'elles sont consommées par les autres créatures, offrant un feedback clair de leur utilité.

Environnement - UI

L'UI est entièrement intradiégétique afin de ne jamais sortir le joueur de son immersion et de son expérience.

Chaque élément de l'interface est directement intégré au monde, même les quantités de chaque outil.

Le matériau principal de l'UI est la pâte à modeler, avec des couleurs vives et des topologies déformées pour renforcer cet aspect d'outils faits à la main par l'avatar de cet univers.

L'utilisation de ce matériau permet également d'assurer le détachement de la rigueur scientifique.

Afin de bien voir quel élément est actif, une outline blanche est présente quand le joueur survole un outil, et elle devient noire quand l'outil est sélectionné, à l'exception des souches.

Voici ensuite en détail chaque élément de l'interface.

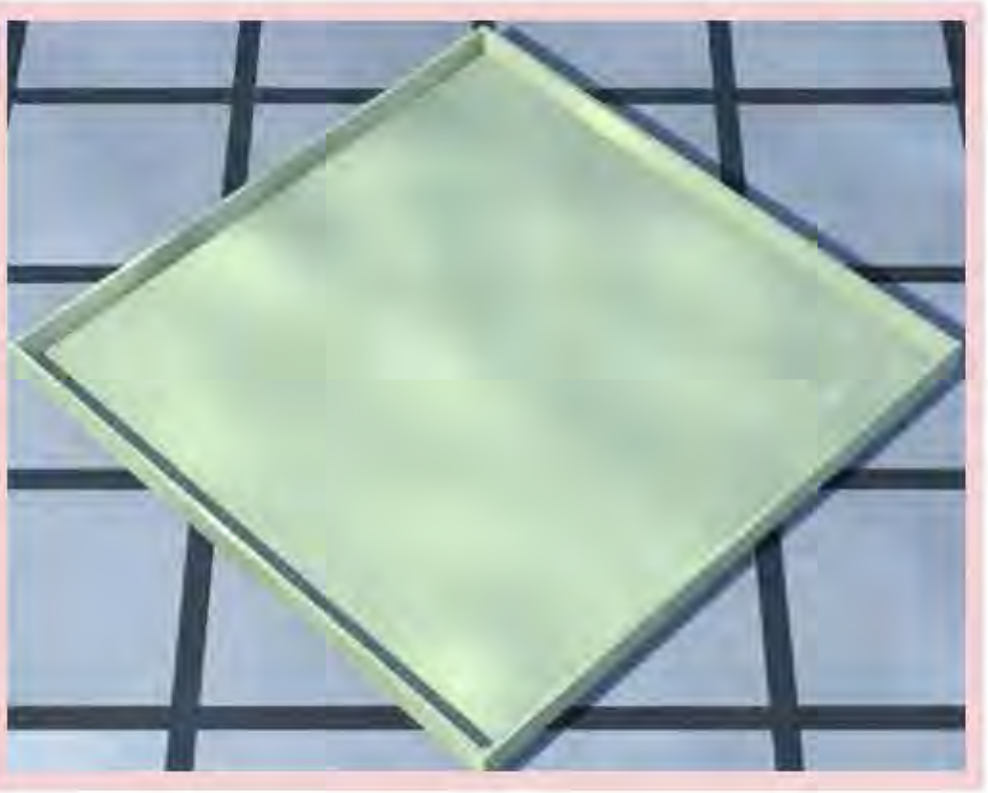
La boîte de pétri

L'environnement des créatures est représenté par une boîte de Pétri géante.

Elle est légèrement colorée et brillante, afin de représenter un matériau céramique.

C'est l'un des seuls éléments non vivants utilisant le matériau de la gelée, à la seule différence qu'il est opaque et beaucoup plus lisse afin d'offrir une lisibilité optimale du terrain et de ne pas occuper la vision du joueur.

C'est un élément important, centre clos des expérimentations du joueur, renforçant sa position d'observateur et d'acteur sur ce petit écosystème.



Le sucre

La nourriture que le joueur peut donner aux créatures a un effet instantané. Elle agit rapidement et attire leur attention.

Elle se devait donc d'être représentée dans ce sens. C'est pour cela que le sucre se prêtait parfaitement à ce rôle.

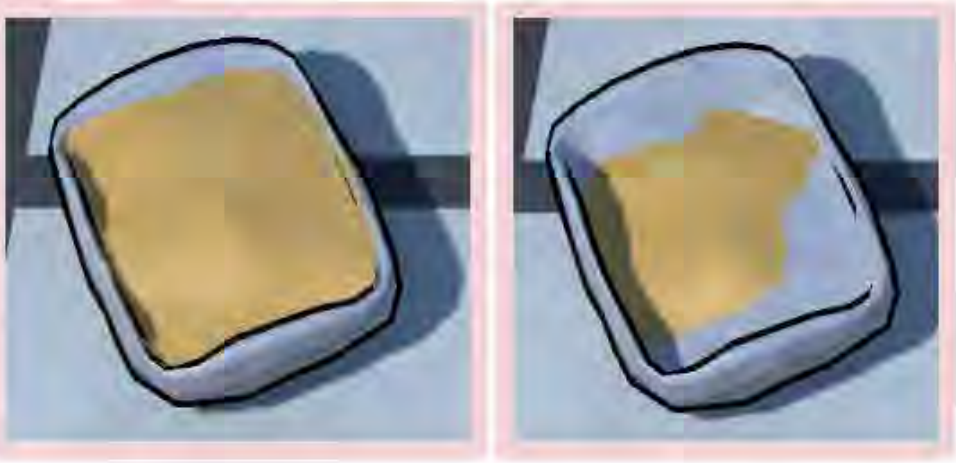
Dans les outils du joueur, c'est une petite boîte en pâte à modeler blanche, avec un tas de sucre en poudre dedans.

En fonction du sucre disponible, le tas est plus ou moins grand, indiquant la quantité disponible pour le joueur.

Sur le terrain, il est représenté par des petits tas de sucre sur le sol.

Pour casser la répétitivité, à leur instanciation leur est donné une rotation aléatoire sur l'axe vertical.

La texture du sucre se base également sur celle de la gelée. Le matériau est moins brillant, opaque, et un paramètre supplémentaire a été ajouté sur les normales, un bruit blanc, donnant un aspect granuleux.



Les Anticorps

Leur but est d'effrayer les créatures, comme des épouvantails, ces dernières ayant une forte ressemblance avec des bactéries, des virus etc...

Quoi de mieux pour effrayer des microbes que des anticorps ?

Cependant, les créatures ont aussi l'air douces et mignonnes, il fallait donc donner un signal au joueur pour lui montrer que cet outil faisait peur aux créatures.

Le modèle se base donc sur celui d'un anticorps, à l'échelle macroscopique.

L'objet en lui-même est en pâte à modeler, contrastant avec la gelée des créatures, posant un premier élément d'opposition.

Le plus important reste la tête des anticorps, une étoile piquante, une forme non lisse, contrastant encore avec la forme globale des créatures.

Dernier élément d'opposition, leur couleur. Ils sont gris foncé, donc très peu colorés et saturés. Leur tête reste cependant orange vif, indiquant tout de même un danger.

La boîte dans laquelle ils se trouvent est également gris foncé, afin d'indiquer dès le lancement du toy qu'ils ont une utilité particulière.

Une autre référence importante est l'oeil de Sauron, figure effrayante se dressant au milieu d'un champ de bataille.

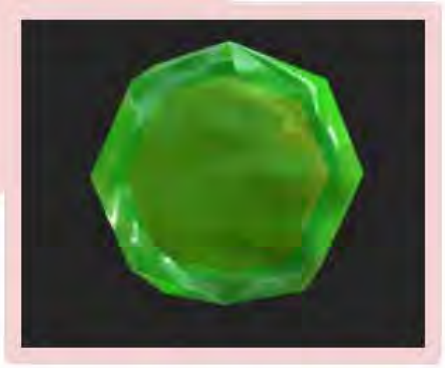
Les boîtes d'souches

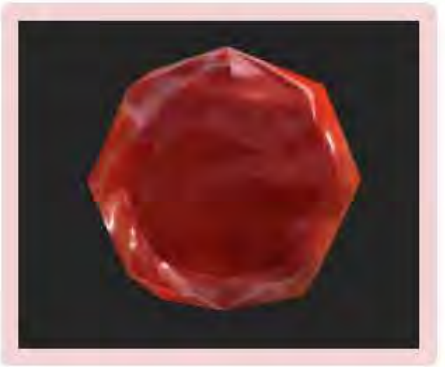
Les boîtes sont rondes, en pâte à modeler blanche.

Elles contiennent chacune deux groupes de souches, dont autant sont affichés que ceux qui sont à disposition d'un joueur, afin de lui indiquer clairement ce qu'il a à sa disposition à un instant T.

Etant relativement petites, il était important d'adopter une forme basique, elles sont donc de simples boules, avec un noyeau et un "corps", beaucoup plus proches d'une cellule, et inspirés visuellement par les oeufs de grenouille.

Quand cet élément est sélectionné, son outline n'est pas noir, mais de la couleur de la gelée de la souche en question, afin d'indiquer plus clairement la créature sélectionnée, quelque chose qui peut être compliqué dû à la petite taille des souches.



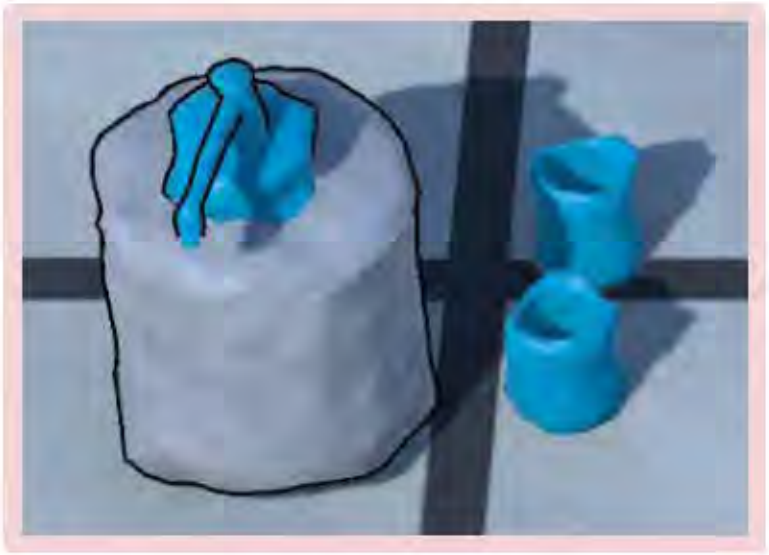


L'alcool

Cet élément est représenté par une grosse bouteille de gel hydroalcoolique. Elle est en pâte à modeler, et respecte les couleurs classiques de cet objet.

A côté de cette bouteille se trouvent cinq petits gobelets maximum, représentant chacune des charges disponibles pour le joueur, de la même manière que le sucre.

Sur le terrain, il est représenté par une goutte de gel transparent légèrement bleuté. Cette goutte est animée et s'étale progressivement sur le terrain.



Le carnet

Le carnet est important. Il sert à cacher les seuls éléments extradiégétiques de l'UI, comme l'option pour quitter le jeu, ou des statistiques.

C'est un carnet de notes, tout ce qu'il y a de plus simple, mais le format de livre est important, puisqu'il permet de garder l'immersion, tout en conservant certaines informations importantes qui peuvent difficilement être intradiégétiques.

En termes de couleurs, l'étiquette bleue rappelle celles que l'on met sur les cahiers de dessin, ou les livres d'école, et le marron permet d'évoquer clairement que c'est un livre.

Lumières et Post-Processing

Un travail important a été réalisé sur la lumière et le post processing.

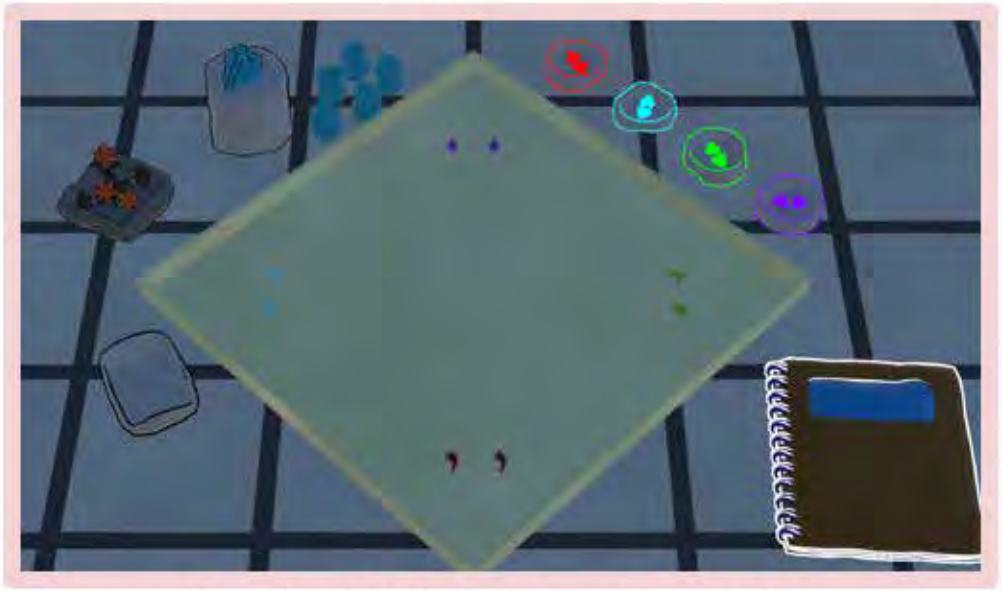
La scène est éclairée par deux lumières, une très forte et l'autre très douce, pour illuminer très légèrement la totalité de la scène. La lumière principale éclaire depuis la gauche, et la secondaire depuis la droite, à même distance.

En termes de post processing, la couleur globale de la scène a été refroidie avec un filtre de température.

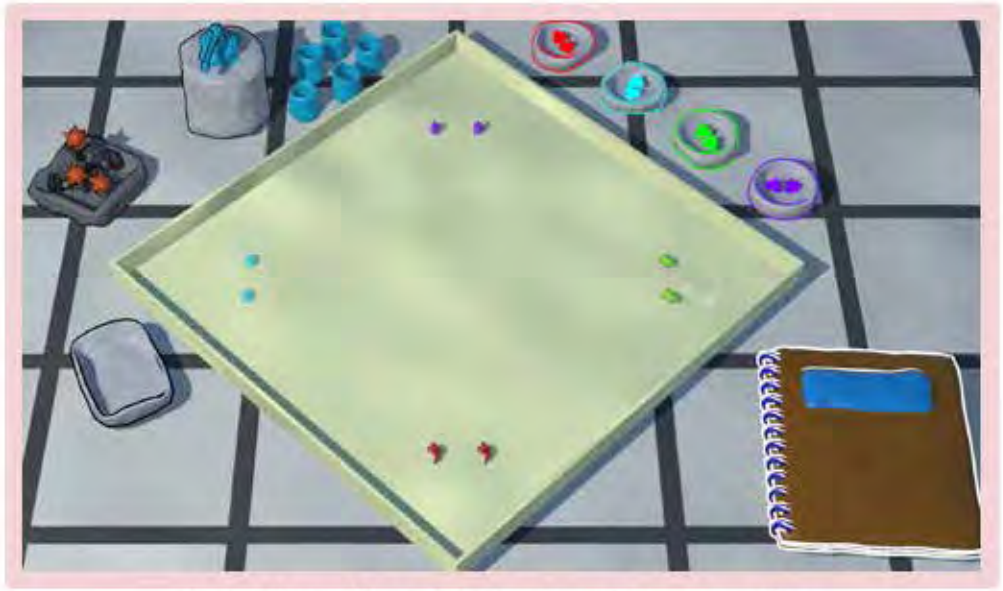
Les zones d'ombre sont également accentuées avec un filtre léger d'ambient occlusion grisâtre, afin de ne pas trop foncer ces zones.

Pour finir, un léger effet de bloom à été ajouté. De couleur jaune, il permet surtout de déblanchir les reflets de la boîte de pétri, ne les effaçant pas, au contraire, les adoucissant.

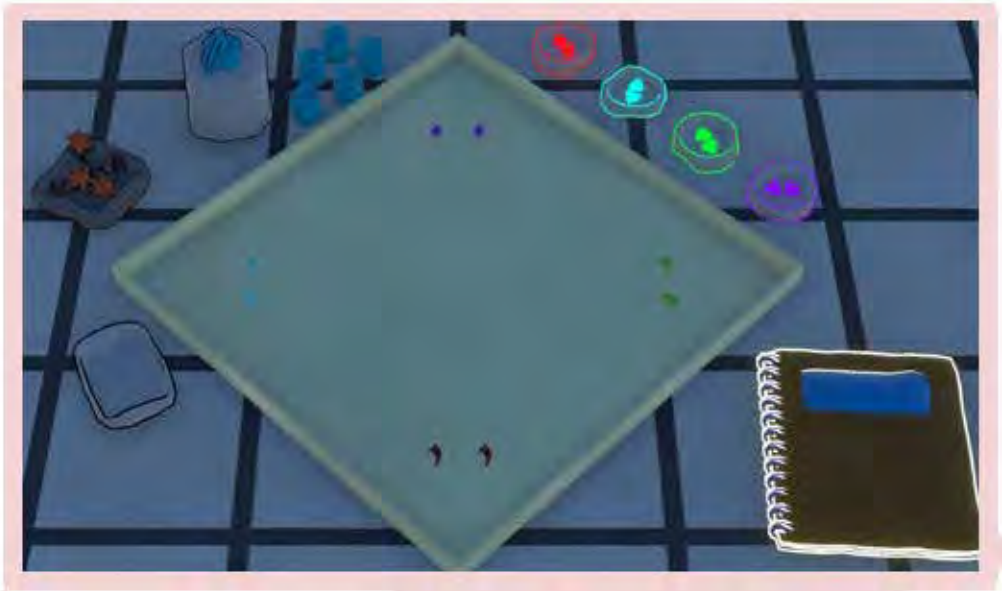
Sans Lumière & Sans Post Processing



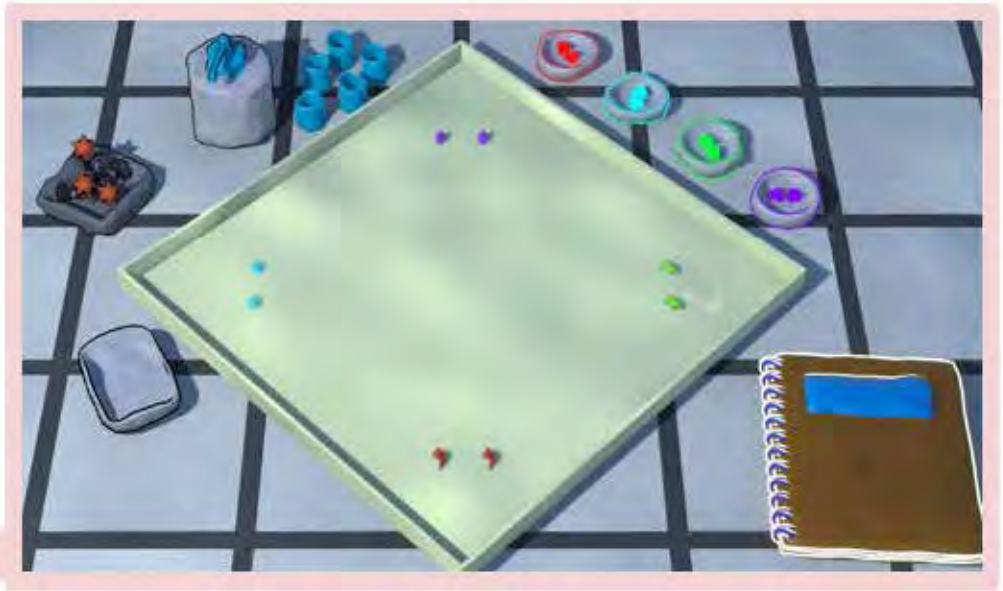
Avec Lumière & Sans Post Processing



Sans Lumière & Avec Post Processing



Avec Lumière & Avec Post Processing



Vision globale de la scène

Menu Principal

Intentions

Nous avons souhaité rester dans une logique d'expériences d'école/collège, en cohérence avec l'esprit du jeu.

Pour cela, le menu principal prend la forme d'une salle de SVT, afin de plonger immédiatement le joueur dans l'ambiance expérimentale et enfantine du projet.

Principe général

Le joueur contrôle un curseur classique lui permettant d'interagir avec différents boutons.

Les boutons sont représentés sous forme de splashes de pâte à modeler colorée, même pâte que les éléments d'UI.

Ces splashes contiennent du texte blanc afin de garantir une lisibilité immédiate des actions possibles. Quand on survole le bouton, une légère outline apparaît en dessous.

Salle de classe

Dans la salle se trouvent des paillasses, des tabourets et une étagère sur le côté.

Tous ces éléments sont placés de manière à exprimer au mieux l'ambiance qui se dégage de ces salles où les premières expériences "scientifiques" sont vécues.

Couleurs

Les couleurs du menu sont tirées des couleurs que l'on retrouve dans les salles de classe à l'école. Elles rappellent encore une fois cet univers de l'enfance.

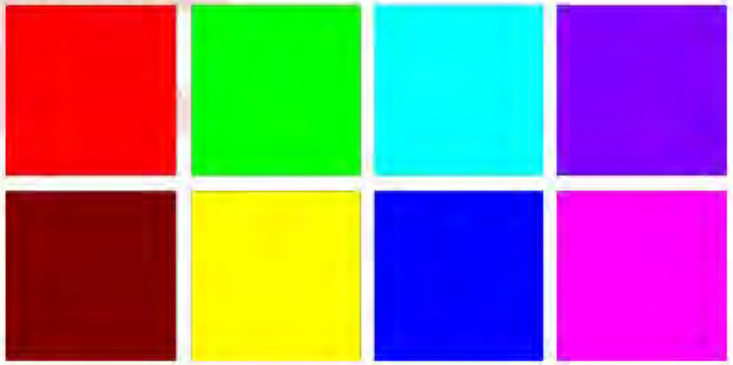
Les splashes de pâte à modeler utilisent des couleurs vives et contrastées afin d'identifier clairement les éléments interactifs et d'assurer une bonne lisibilité.

Palettes de Couleurs

La palette de couleurs du jeu est majoritairement saturée, en cohérence avec les matériaux utilisés : la pâte à modeler pour l'UI et les éléments manipulables, et la gelée pour les entités vivantes. Cette saturation renforce le côté ludique de l'univers, tout en facilitant la lisibilité et la hiérarchie visuelle.

Voici un résumé global de toutes les couleurs utilisées.

Créatures



UI

Main Menu

Logo

Le logo à été designé par Barthélemy, et reprend les silouhettes des entités, ainsi que leurs matériaux. Pour le texte, il est fait en pâte à modeler blanche. Le logo est simple, coloré, évocalteur de l'ambiance globale du jeu.

Références de direction artistique

Pâte à modeler

Pour la pâte à modeler, la première référence à laquelle nous nous sommes référés est la pâte à modeler ***Play-Doh***.

Pour partir sur une bonne base, nous avons découvert et appris comment faire différents types de pâte à modeler, ce qui nous a permis d'ouvrir nos horizons et de ne pas nous fermer de portes. C'est l'aspect saturé des textures que nous avons choisi de conserver dans cette référence.

C'est pourquoi nous avons pris comme référence ***Wallace & Gromit*** qui possède une esthétique en pâte à modeler faite à la main, avec des formes imparfaites, créant un univers visuel unique, reconnaissable et aujourd'hui encore établi, et ce depuis l'enfance de nombreuses personnes.

Une autre de nos références est ***The Never Hood***, un jeu possédant une ésthétique qui pousse plus loin l'aspect pâte à modeler, mettant en avant la matière, les traces de modelage et l'aspect manuel.

Le mélange de ces trois styles nous a permis de partir sur une base visuelle marquée, déjà installée depuis plusieurs années dans l'imaginaire collectif, tout en nous démarquant, afin d'affirmer l'identité de ***Protocole PRV***.

Play-Doh

Wallace & Gromit - 1989

The Never Hood - 1996

Gelée et créatures

Pour la gelée, et plus généralement les créatures, la référence principale est *Spore,* notamment la phase cellulaire en début de jeu. C'est une référence importante, car bien connue du grand public, ayant son charme, et ayant des similarités avec notre Toy, même en termes de Game Design.

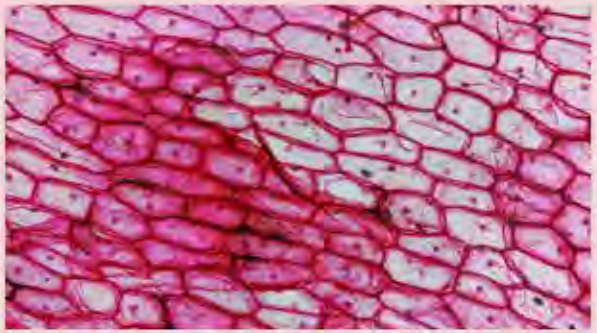
La superposition de plusieurs éléments visuels était un challenge, et l'adoption de ce style nous a permis de le dépasser.

Les deux autres références majeures de nos entités étaient évidemment les bonbons Jelly Bears et... Les cellules des organismes vivants.

Les cellules nous ont donné une bonne base de design et de fonctionnement globale de la DA, alors que les bonbons sont venus donner une identité visuelle et un aspect enfantin.

Grâce à toutes ces références, nous avons pu donner une esthétique cohérente et agréable, presque naïve au Toy.

Spore - 2008



Cellules vivantes

Phase cellulaire

Jelly Bears

Sound Design

Introduction

Bien que cette partie soit plus courte que les autres, le design sonore reste un aspect important de notre jouet, surtout en termes de feedbacks. Nous y parlerons d'abord des références, puis nous discuterons de chaque son et sa signification, et enfin nous terminerons en exposant l'implémentation de ces derniers.

I - Références

Pour créer nos sons, nous nous sommes surtout basés sur deux références: la première est Stacklands, petit jeu développé par Sokpok Collective et sorti en 2022. Le design sonore y est assez simple quoiqu'efficace, composé en grande majorité de variantes de sons de "pops" courts et satisfaisants. Il a donc naturellement été notre inspiration pour la consistance et la texture de nos sons, qui vont eux aussi contenir différentes variantes de "pops".

La seconde inspiration est Sid Meier's Civilization (5 et 6), développés par Firaxis Games et sortis respectivement en 2012 et 2016. On est là sur une production à gros budget, bien sûr, mais c'est surtout l'identité donnée à chaque son qui nous a intéressé. En effet, chaque unité, si on la sélectionne, joue un court son représentatif de ce qu'elle est: un spadassin va jouer le son d'une épée sortant d'un fourreau, l'on va entendre les chenilles et le moteur d'un char, etc. Ses sons, de plus, sont joués avec une attaque assez franche, mais ont une release assez rapide, pour ne pas trop encombrer l'espace sonore du joueur. Nous en avons donc pris inspiration, et avons conçu des sons spécifiques à chaque objet, qui suivent au mieux la philosophie de ces jeux, bien que nous ne soyons évidemment pas à un niveau équivalent.

II - Sons créés

Nous allons à présent nous pencher sur les origines des différents sons créés pour le jouet. A part pour le premier, le matériau primaire à la création des sons a été trouvé sur FreeSound.

a) Créatures

Plusieurs sons ont été assignés à nos créatures:

Son d'apparition : Seul son du projet enregistré par nos soins, à l'aide d'un Zoom H6 prêté par l'école. Il s'agit simplement de plusieurs "pops" sortis de ma bouche, puis édités par Esteban TERRIEN dans le contexte d'un devoir de sound design en début d'année. Il y en a 8 variants différents, jouant sur le pitch et le volume pour se différencier.

Son de state Consuming : Son d'une feuille un peu mouillée que l'on a froissée. Existe en 4 variantes qui sont des moments différents du son découpé. Le volume a ensuite été largement diminué, ainsi que les hautes fréquences dans une moindre mesure.

Son de mort : Son d'un objet spongieux plein d'eau écrasé. Existe en 4 variantes qui diffèrent par leurs pitchs. Là, les hautes et les basses fréquences ont été diminuées, laissant la place aux fréquences intermédiaires.

Nous avons choisi de ne pas ajouter de sons de déplacement ou d'autres states comme Chasing ou Fleeing, car ces cas arrivant trop souvent, l'espace sonore risquerait de s'en retrouver saturé, allant à l'encontre de notre philosophie expliquée précédemment.

b) Sucre

Trois sons ont été nécessaires à propos du sucre:

Son de sélection : Quand on clique ou que la molette s'arrête sur le sucre. Son d'un paquet de sucre agité 3 fois. Les basses ont été boostées pour lui donner plus de texture.

Son de placement dans la Boîte : Extrait du son utilisé précédemment, découpé à un autre endroit, et combiné à un son d'un objet frappant un drap boosté dans les aigus, créant une alternative aux "pops" qui diversifie un peu les identités sonores. Existe en 4 variantes qui diffèrent par leur pitch.

Son de recharge : Quand une unité de sucre apparaît dans son contenant. Son de glaçon tombant dans un verre, extrêmement raccourci, et boosté dans les aigus pour donner un effet "grattant".

c) Anticorps

Les anticorps n'ont pas besoin de son de recharge; il n'y a donc que deux sons liés à eux :

Son de sélection : Son de deux morceaux de bois frappant l'un contre l'autre, boosté dans les basses, puis répété avec un pitch différent qui donne un effet un peu dissonant, pointant l'aspect "antagoniste" des anticorps pour nos créatures.

Son de placement dans la Boîte : Autre extrait du sample précédent, accompagné d'un petit "pop" discret boosté lui aussi dans les basses. Existe en 3 variantes qui jouent sur les pitch des deux pistes pour se différencier.

d) Alcool

Passons maintenant aux différents sons liés à l'alcool :

Son de sélection : Son d'un poussoir de bouteille de gel hydroalcoolique, accéléré et diminué dans les basses.

Son de placement dans la Boîte : Combinaison d'un son de goutte d'eau qui tombe, boosté dans les basses, et d'un son de "pop" pour rester dans notre vision. Le pop, cette fois-ci, est un peu plus aigu. L'objectif étant de laisser la place à la goutte dans les tonalités basses.

Son de brûlure : Se déclenche quand des créatures ont été touchées par la zone d'effet de l'alcool. Son de bacon qui grille, sans quasiment aucune basse et avec un volume très faible, mais augmente son volume si plusieurs créatures ont été touchées.

Son de recharge : Son de bouteille d'alcool versée, un peu boosté dans les basses.

e) Souches

Pour les souches, certains sons sont communs, d'autres dépendent de l'espèce concernée :

Son de sélection Bacteria Vanillii : Son de tremblement de mâchoire et de claquement de dents très accéléré, puis boosté dans les aigus.

Son de sélection Entity X : Son de bulles sortant d'un corps visqueux, avec légèrement plus de basses.

Son de sélection Holophagovirus : Le bruit le plus "diabolique" selon Barthélemy. Son de pas sur un sol boueux, très accéléré comme plus haut, et réduit dans les basses.

Son de sélection Résilium Toxifungi : Son de pschitt de pulvérisateur, cette fois-ci bien ralenti, et découpé de manière à donner un petit effet de répétition sur la fin.

Son de placement dans la Boîte (commun) : Son d'œufs cassés, bien boostés dans les basses et diminué dans les aigus.

Son de recharge (commun) : Son de micro-ondes qui sonne (honnêtement on a gardé cette proposition surtout parce que c'était marrant). Peu de changement y sont à signaler, à part peut-être un tout petit peu de basses diminuées.

f) Carnet

Deux sons sont liés au carnet:

Son d'ouverture : Son de stylo à bille écrivant sur du papier. Les aigus y ont été diminués.

Son de fermeture : Son de page qui se tourne. Bien diminué en volume, aussi avec peu d'aigus.

g) Clics généraux

Des sons ont étés ajoutés pour ajouter de la juiciness à la simple action de cliquer:

Son de clic : Littéralement, son de clic. Nous avons choisi cependant de le positionner dans une tonalité plutôt aiguë par rapport au sample d'origine, et avons diminué son volume assez drastiquement pour ne pas qu'il prenne le pas sur le reste des sons.

Son de signalement que l'objet sélectionné est vide : Son de buzzer qui joue 2 fois, au contraire diminué dans les aigus mais également fortement baissé en volume.

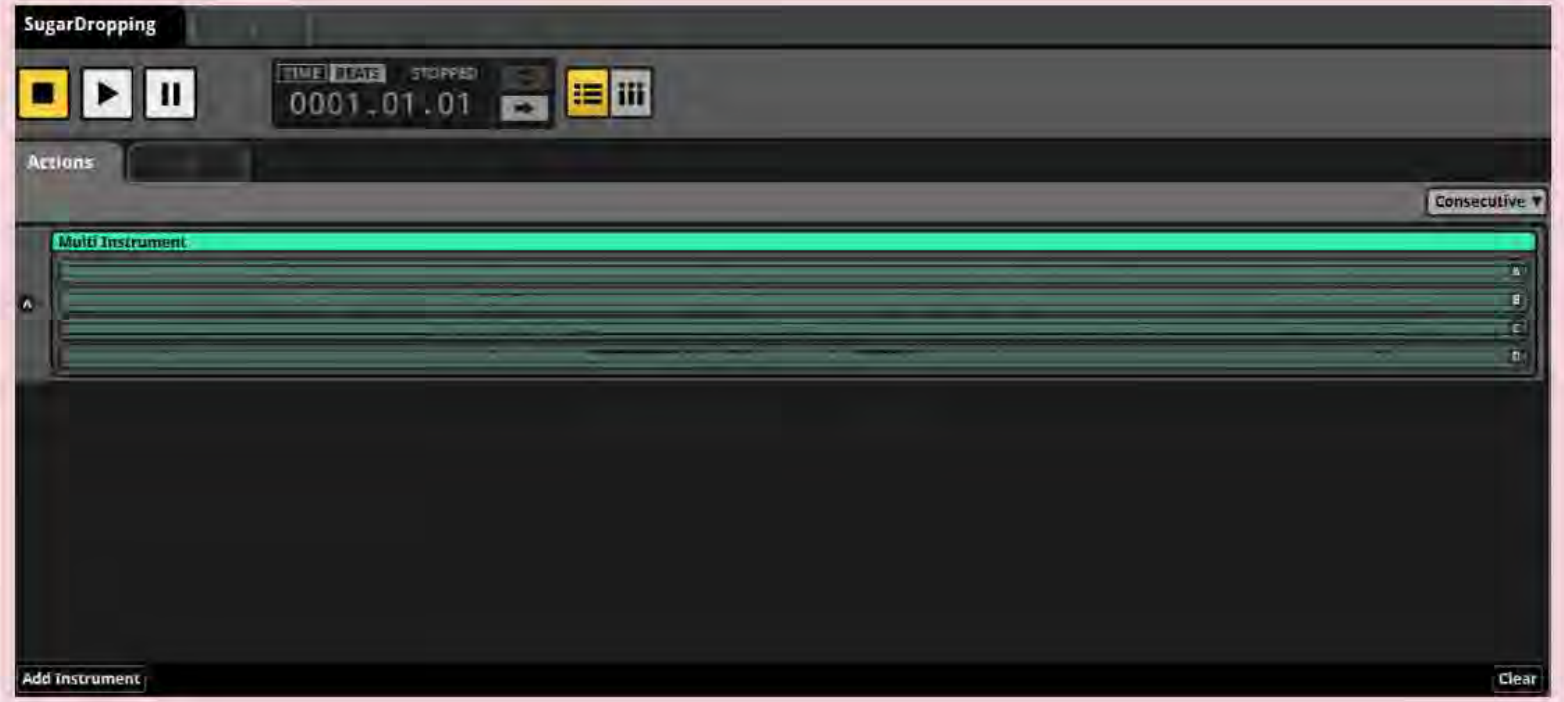
III - Implémentation

Finalement, après avoir hésité un certain temps sur la charge de travail qu'il nous restait, nous avons décidé d'essayer de passer par FMOD pour implémenter nos sons, rendant plus simple la gestion des random containers et les listes de sons conditionnels, au prix d'un effort d'apprentissage qui fut relativement rapide. Encore une fois, ce n'est pas parfait, il y a probablement des techniques d'utilisation d'FMOD plus efficaces, mais tant que ça fonctionne, c'est le principal.

Déjà, une grande partie des sons sont des oneshots s'activant à l'apparition ou la disparition d'un objet, ce qui est très simple à implémenter puisqu'il suffit de placer un Event Emitter sur le prefab de l'objet, de lui attribuer l'évènement FMOD correspondant, et de choisir "Object Start" ou "Object Destroy" dans le menu déroulant prévu à cet effet. On parle ici:

- De tous les sons de placement d'objets dans la Boîte;
- Des sons d'apparition et de mort des créatures.

Exemple de l'Action sheet de placement du sucre; il s'agit d'un Multi Instrument, simulant un random container:



Pour le reste, c'est un peu plus compliqué. J'ai eu à créer des Parameter Sheet, permettant de jouer des sons différents simplement en changeant la valeur d'un paramètre donné. J'ai même découvert une méthode peu orthodoxe pour que ces sons se jouent dès que la valeur cible du paramètre est atteinte, sans avoir besoin de rajouter une ligne de code pour lui demander de se jouer: il suffit de rajouter à la sheet une piste "blanche", contenant un son silencieux, que l'on lance à l'"Object Start" et qui tourne en boucle. On parle ici:

- Des sons de sélection. Les différents boutons sélectionnables sont placés dans un tableau, et on peut appliquer à notre variable la valeur de l'index du bouton pour jouer le son correspondant.
- Des sons de recharge. La variable correspondante passe de 0 à 1 dès que le contenant se remplit, puis repasse à 0 juste après.
- Le son de brûlure. Après sa zone de mort effective, l'alcool attend 0.1 secondes, puis compte le nombre de créatures touchées. Ce nombre est ensuite cappé à 4, puis la variable est set au nombre donné, changeant le volume du son, avant que la zone ne se détruise.
- Les sons du carnet. Selon l'état ouvert ou fermé, la variable est set à 1 ou 2, avant de repasser à 0.

Exemple ici de la Parameter Sheet de sélection d'objet: on voit bien la piste audio 2 contenant le son vide.

Il ne nous reste plus qu'à parler de quelques sons. Le son de clic s'active dès que Input.GetMouseButtonDown est true, celui de signalement d'un manque d'objet est activé sur un else if lié au son qui devrait se jouer si l'objet était disponible, et celui du state Consuming s'active dès que la créature rentre dans ledit state, puis se répète 4 fois aléatoirement pour simuler les bruits de "déchiquetage" du cadavre.

Documentation technique

Introduction

Dans cette partie, nous allons nous pencher spécialement sur le fonctionnement concret de notre système, et les challenges que nous avons eu à traverser pour proposer la meilleure expérience de jeu possible.

Nous commencerons par parler des enjeux techniques, puis des éléments centraux de notre jeu, les créatures, en décortiquant leur fonctionnement de base et en mettant la lumière sur leurs spécificités, et enfin nous nous intéresserons aux actions du joueur, que ce soit directement dans le gameplay mais aussi dans l'interface.

I - Enjeux Techniques

Notre jeu étant un jeu de simulation pseudo-scientifique, notre objectif principal est de rendre le comportement des créatures et du joueur « crédibles ».

Pour cela, il nous a paru nécessaire de proposer un jeu fluide, mais qui ait l'air d'un mécanisme inlassablement en mouvement, et qui existe au-delà de notre présence en tant que joueur.

Ainsi, les plus grandes difficultés reposent sur le déplacement des créatures, qui nous a poussés à tester différentes méthodes (NavMesh, Rigidbody) avant d'être satisfaits.

Finalement, nous avons préféré le Rigidbody, plus pratique pour contrôler précisément chaque frame de déplacement, grâce à son lien étroit avec le moteur physique.

Une autre difficulté s'est cependant présentée à nous à ce moment :

Les collisions entre les créatures couplées à la quantité de créatures possibles sur le terrain peuvent casser cette sensation de mécanisme imperturbable et de fluidité ; nous avons donc opté pour des créatures uniquement Trigger, qui peuvent passer les unes « à travers » les autres, ne saccadant donc pas leur mouvement sans devoir gérer des Raycasts en quantité.

Car la quantité, elle, est le dernier gros enjeu de notre projet : l'on doit pouvoir proposer un système qui accepte un très grand nombre d'objets à la fois sur le terrain, et interagissant en même temps, tout en restant lisible et pas prise de tête pour le joueur. Pour cet enjeu, ce seront nos choix de direction artistique et sonore qui primeront.

II - Créatures

Les créatures et leurs interactions sont le point central de notre jeu. Voici comment notre proposition technique leur à permis de répondre aux enjeux.

1) Fonctionnement de base

Pour simuler des petits êtres vivants, rien de mieux que de s'inspirer de la vraie vie pour les rendre « crédibles ».

Ainsi, la structure et la relation entre les différents scripts se base sur le concept d'« organes », ayant chacun un domaine d'expertise et communiquant activement entre eux.

Ce n'est bien sûr qu'une image, les micro-organismes ne possédant pas forcément d'organes à proprement parler.

Ce que l'on peut se demander pour commencer, c'est quelles sont les informations auxquelles doivent accéder les créatures, quels comportements elles doivent adopter en fonction de ces informations, et comment cela impacte les autres créatures et l'état système.

L'on peut alors déjà déterminer une liste d'« organes » permettant de donner une réponse partielle à ces problèmes.

A tout cela l'on peut rajouter un comportement post-mortem (lorsque la créature est détruite), ainsi qu'un script « ADN », qui centralise les variables d'identification de la créature, comme sa vitesse, ou son angle de vision.
Plongeons maintenant un peu plus en profondeur dans le fonctionnement de chaque « organe ».

Liste d'« organes » :

- **Des « yeux »**

Permettent de détecter la présence d'objets dans un champ de vision.

- **Un « cerveau »**

Prend en compte les informations des yeux pour choisir l'état dans lequel entrer.

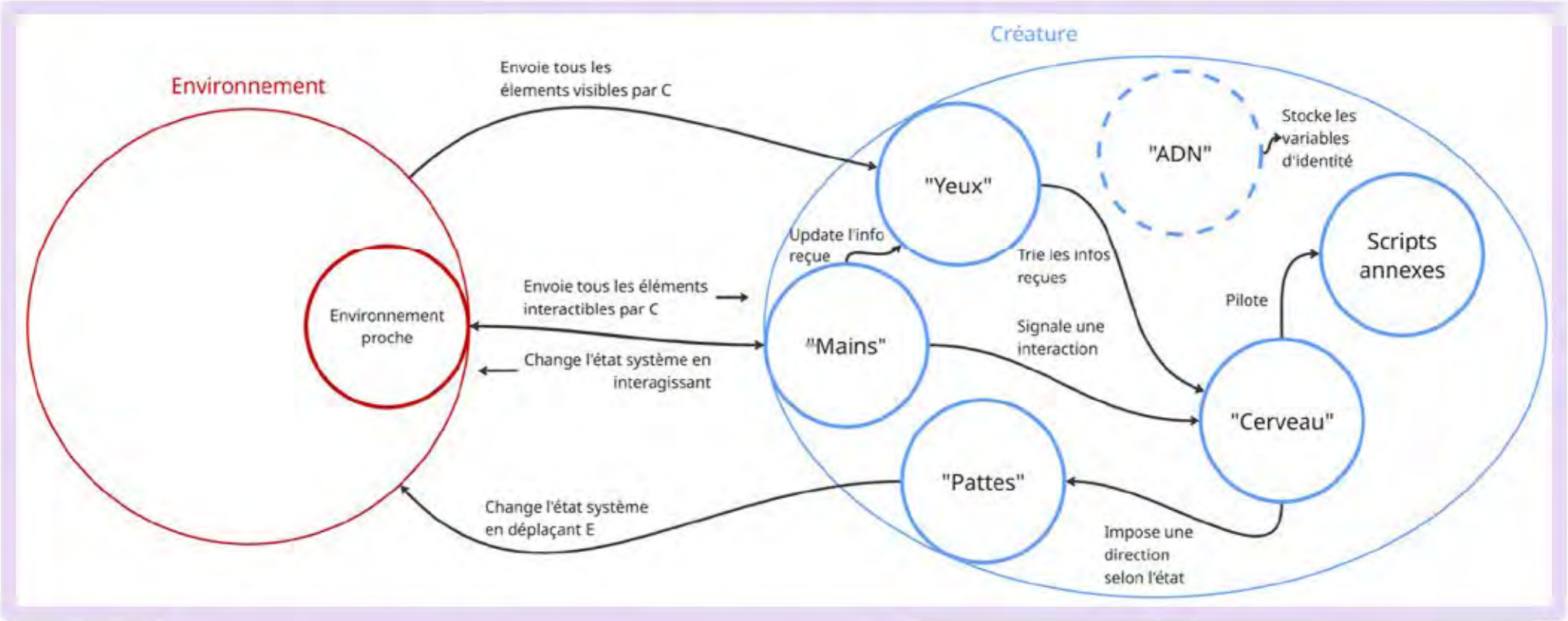
- **Des « pattes »**

Déplacent la créature en fonction de l'état choisi par le cerveau.

- **Des « mains »**

Interagissent à proximité d'autres objets.

Schéma du fonctionnement de base d'une créature



a. ADN

Le script « ***ADN*** » de nos créatures se nomme « ***EntityIdentity*** ». Comme expliqué en préambule, il centralise toutes les variables d'identification de notre créature, afin de simplifier leur utilisation au sein d'autres scripts. Nous allons ici dresser une liste exhaustive des variables qu'il contient, en prenant l'exemple d'une créature C :

- ***species (enum « Species »)*** :
  L'identificateur de l'espèce de C.
- ***state (enum « State »)*** :
  l'état de C (idle, fuite, chasse...).
- ***strength/weaknessAgainst (arrays GameObjects)*** :
  respectivement l'ensemble des espèces sur lesquelles C domine, et celui par lesquelles elle se fait dominer.
- ***nativeSpeed (float)*** :
  la vitesse de base de C.
- ***rotationSpeed (float)*** :
  la vitesse de rotation de C.
- ***refreshPathRate (float)*** :
  le temps nécessaire à C pour le recalcul d'une nouvelle destination.
- ***speedModifierWhenX (float)*** :
  un multiplicateur applicable à la vitesse selon l'état de C. Il en existe 3 : Fleeing, Chasing et Fatigued.
- ***enduranceWhenX (float)*** :
  le temps maximal que C peut passer dans un certain état. Deux versions : Fleeing et Chasing.
- ***recoveryTime (float)*** :
  le temps nécessaire à C pour sortir de l'état Fatigued.
- ***fovRadius (float)*** :
  a distance maximale de détection d'objets de C.
- ***fovAngle (float)*** :
  l'angle total (en degrés) de détection d'objets de C.
- ***interactingRadius (float)*** :
  la distance maximale d'interaction de C.

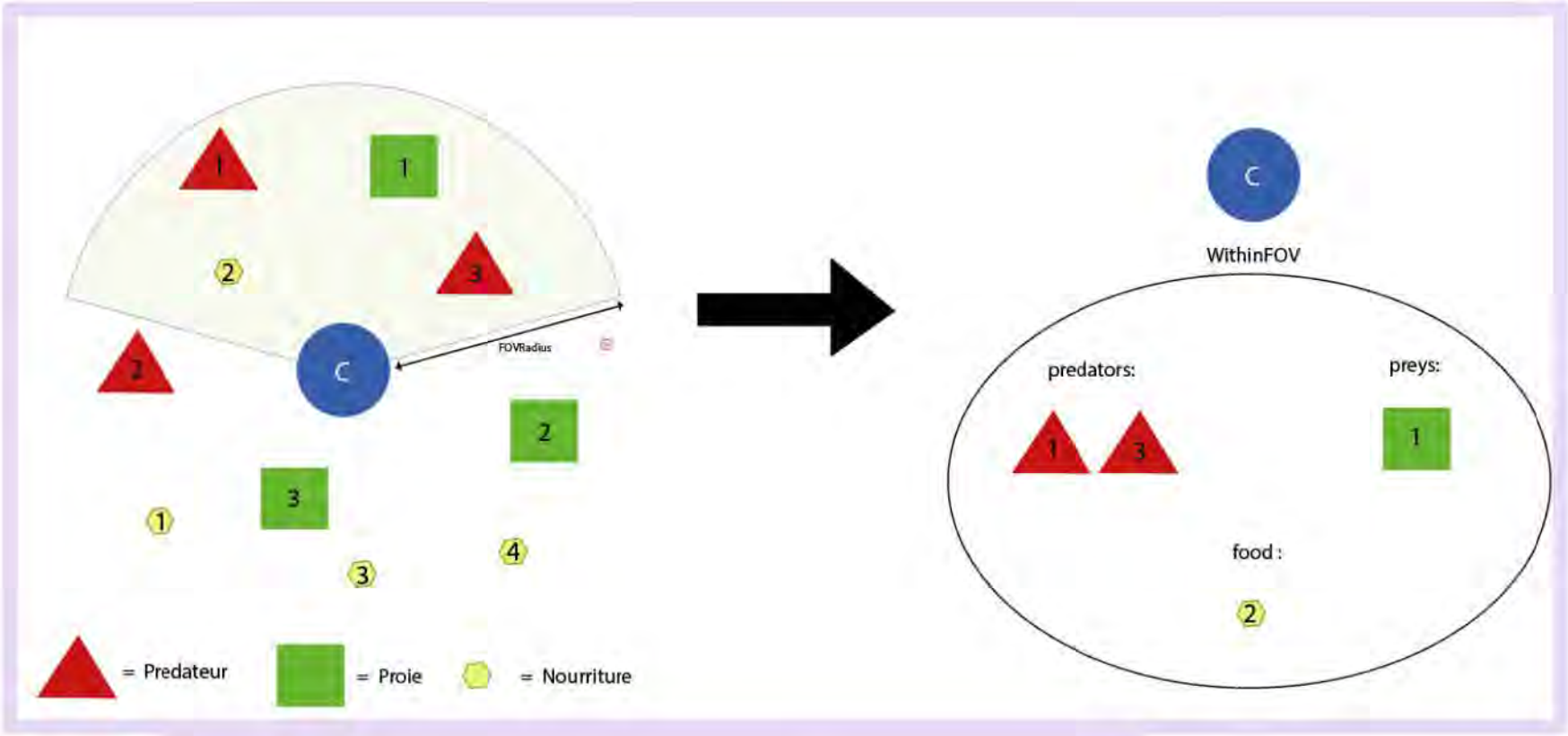
b. Détection des entourages

La détection des entourages est gérée par le script « ***EntityFOV*** », notre script « ***yeux*** ».

Celui-ci récupère les informations de ***fovRadius*** pour générer une OverlapSphere autour de notre créature C à intervalles réguliers (0.1sec) ; puis, il limite la prise en compte des objets à un angle de vue (***fovAngle***).

Tous les objets ainsi détectés sont alors stockés dans une liste, pour être ensuite triés à l'aide d'un foreach dans diverses sous-listes : ***predatorsWithinFOV*** s'il s'agit d'un membre d'une espèce faisant partie des ***weaknessAgainst***, ***preyWithinFOV*** si ***strengthAgainst***, ***matesWithinFOV*** si c'est un membre de la même espèce et ***foodWithinFOV*** s'il s'agit de nourriture.

A noter que le self collider et le ground sont donc ignorés. Voici un schéma exemple avec C :



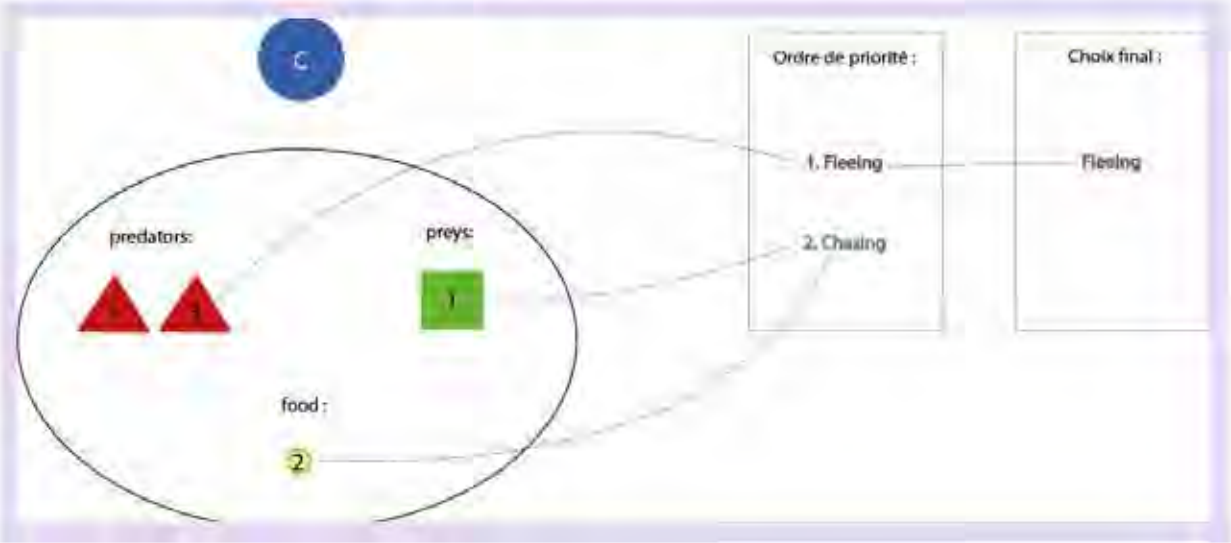
c. Interprétation de l'information reçue

Les informations contenues dans ces sous-listes sont ensuite transmises au script « ***cerveau*** », que l'on a appelé « ***StateChecker*** ». Ce dernier sert globalement de state machine, qui opère avec un ordre de priorité défini :

- Nous avons choisi la survie comme valeur la plus importante pour nos créatures. Ainsi, dès que la sous-liste ***predatorsWithinFOV*** se remplit, E change de ***state Idle*** vers ***Fleeing***.
- Si il n'y a pas de danger, et qu'au moins l'une des deux sous-listes ***preyWithinFOV*** et ***foodWithinFOV*** contient au moins un objet, E passe au ***state Chasing***, toujours depuis ***Idle***.
- Si E est restée dans un état X plus longtemps que ***enduranceWhenX***, elle passe en ***state Fatigued*** ; si elle est restée en ***Fatigued*** plus de ***recoveryTime***, elle repasse en ***Idle***.

Le changement d'état n'est cependant pas la seule fonction de « ***StateChecker*** ». Selon l'état choisi, il va également calculer la trajectoire et les changements de vitesse que devrait prendre E en conséquence. Ainsi :

- Si l'on reste en ***Idle***, rien à dire, il laisse le script de déplacement se débrouiller seul.
- En ***Fleeing***, il cherche à s'éloigner le plus possible de ses prédateurs, en vérifiant leurs positions et en cherchant un vecteur allant dans la direction inverse, et augmente momentanément ***fovAngle*** pour simuler un comportement « aux aguets ».
- En ***Chasing***, il bloque sa position cible à la position de la proie la plus proche.
- En ***Fatigued***, c'est la même chose qu'en ***Idle***.
- Voici un schéma d'exemple toujours avec C:



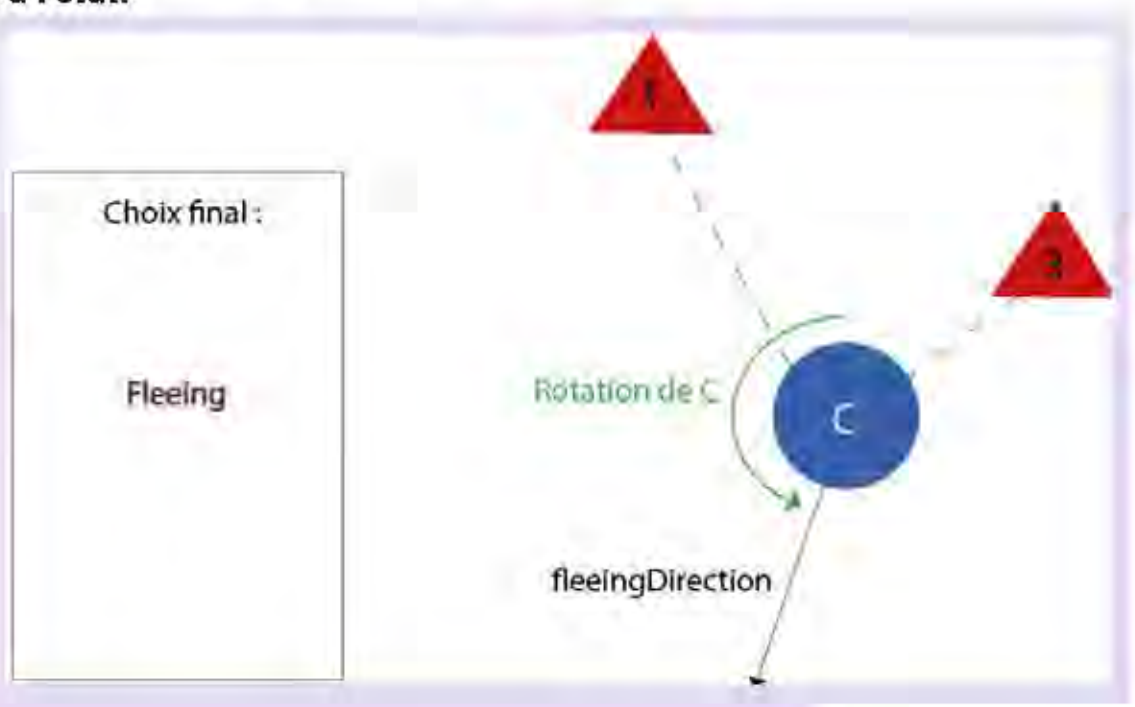
d. Déplacements

Il est à présent temps pour le script « ***pattes*** », nommé « ***EntityMovementRandomInFOV*** », de s'activer. Il est le seul script à interagir avec le moteur physique, en déplaçant C avec ***Rigidbody.linearVelocity***.

Celle-ci va en permanence être dirigée vers le ***transform.forward*** de C ; ainsi, les changements de direction auront lieu sous la forme d'une simple rotation de C. Cette rotation est définie par une direction cible, soit choisie aléatoirement dans le champ de vision tous les ***refreshPathRate*** secondes, soit prenant en compte les commandes de ***StateChecker***. En fonction du state de C, il applique de plus à la vitesse (basée sur ***nativeSpeed***) un ***speedModifierWhenX***, X correspondant à l'état.

C'est également le script qui va faire attention à ne pas sortir des limites du terrain. Il vérifie à chaque frame si une position à l'avant de C sort des ***Bounds*** du ***ground***, et si c'est le cas, il interrompt la mise à jour de ***refreshPathRate*** pour forcer une direction encline à remettre C sur de bons rails. Cette solution n'est pas parfaite, mais un autre petit script, « ***ForceClamping*** », gère les cas extrêmes en bloquant totalement E de sortir des limites des ***Bounds*** du ***ground***.

Gardons le même exemple de C pour expliquer en schéma.



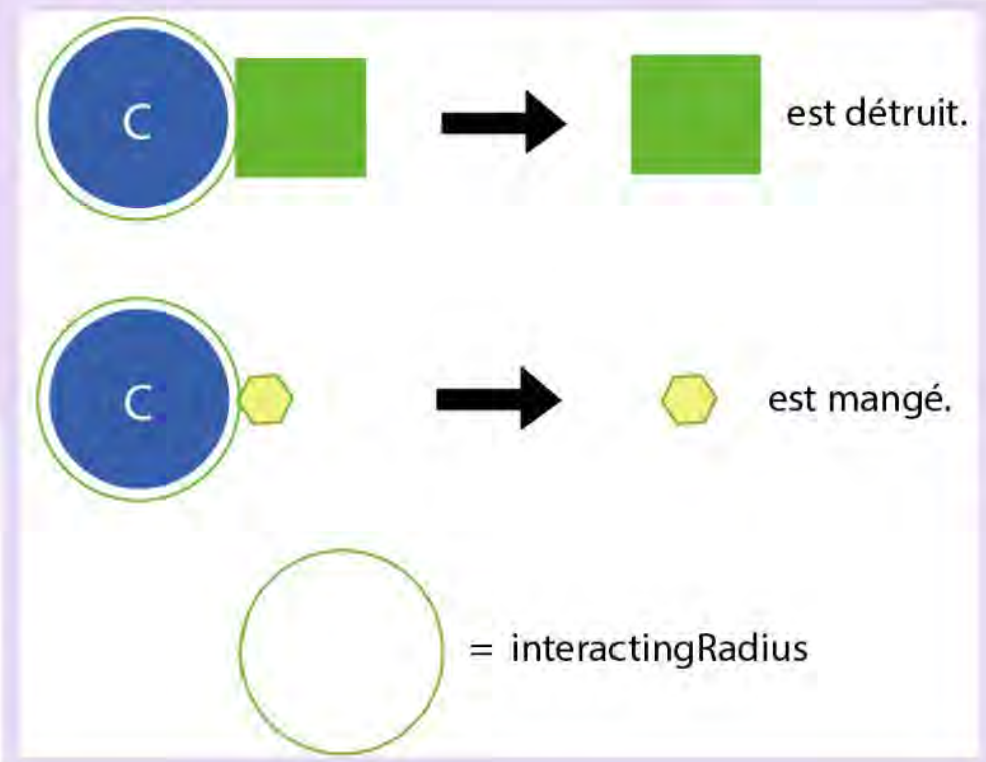
e. Interactions

Notre script « ***mains*** », nommé simplement « ***Interact*** », à un fonctionnement bien plus proche de celui de « ***Entity-FOV*** » que de celui de « ***EntityMovementRandomInFOV*** », le plaçant plutôt comme le « sens du toucher », mais influant tout de même de manière automatique sur l'environnement.

Une nouvelle ***OverlapSphere***, de rayon ***interactingRadius***, bien plus petit que ***fovRadius***, récupère les informations des objets très proches de C.

Elle va ensuite accomplir des actions selon la teneur de ses objets :

- S'il s'agit d'une proie listée dans ***strengthAgainst***, elle la tue.
- S'il s'agit de nourriture, elle communique avec un script « ***FoodEater*** » dont nous parlerons dans l'axe III.
- S'il s'agit d'autre chose, rien ne se passe.
- D'autres actions peuvent être déclenchées, mais nous en parlerons dans le 2).

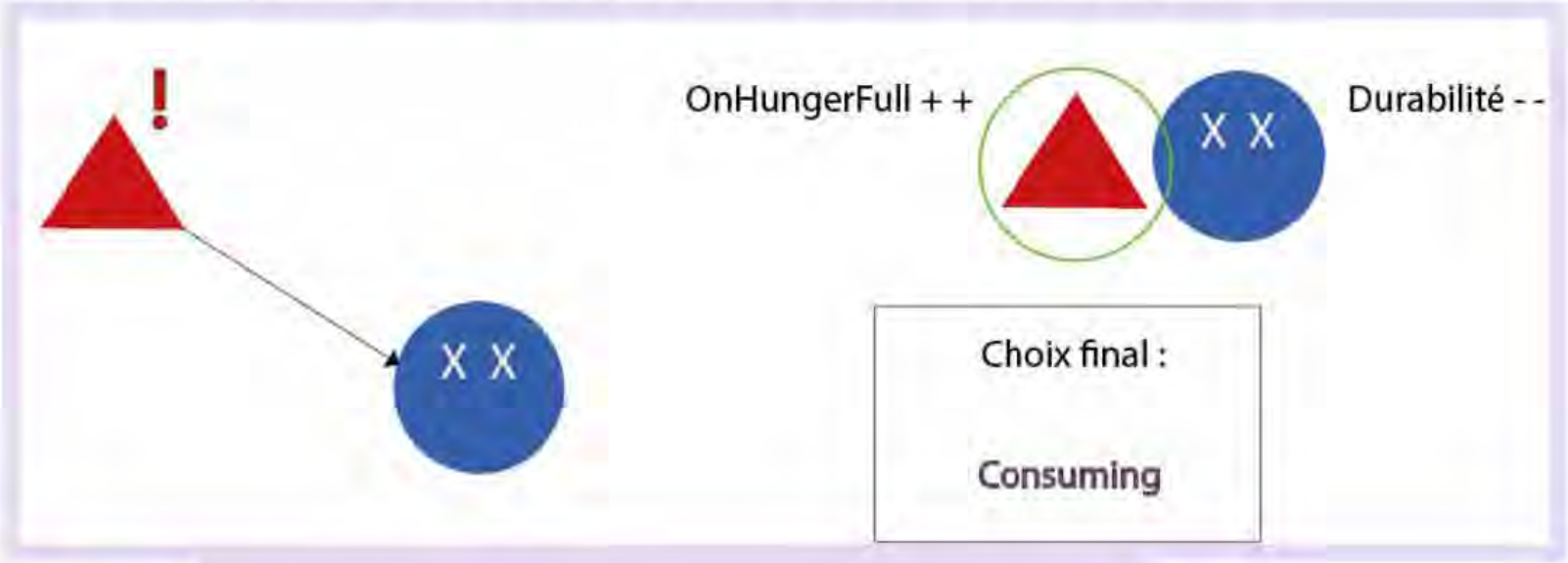


f. Mort

Lorsqu'elle meurt, C, à l'aide du script « ***CarcassSpawn*** », laisse derrière elle un cadavre. Pour les autres créatures encore en vie et qui voient C comme une proie, ce cadavre est perçu comme une proie par « ***EntityFOV*** », les attirant donc.

Au contact dudit cadavre, les créatures intéressées sont placées dans le ***state Consuming***, qui les immobilise le temps qu'elles se « sustentent ». En effet, elles possèdent également un script « ***OnHungerFull*** » qui vérifie le temps passé à manger et fait spawner une nouvelle instance d'elle-même s'il atteint son objectif, ce qui amène d'ailleurs son ***StateChecker*** à le faire repasser en ***Idle***.

Le cadavre, lui, perd de la durabilité à mesure qu'il est consommé, via le script « ***CarcassState*** », et disparaît soit parce qu'elle est à 0, soit parce qu'un temps défini est passé.



2) Spécificités des créatures

En plus de leurs fonctionnalités de base, chaque espèce de créature possède ses propres spécificités, dont nous avons parlé dans la partie Game Design de ce document. Voici alors comment ces spécificités ont été intégrées.

a. Bacteria Vanilla

C'est l'espèce la plus simple, « vanilla ». La seule chose qui la distingue vraiment est sa capacité à attaquer deux autres espèces au lieu d'une, ce qui se traduit simplement en une ligne de plus dans la liste ***strengthAgainst***.

b. Holophagovirus

Il a deux spécificités à signaler : premièrement, il peut consommer les cadavres de n'importe quelle espèce (sauf son espèce), même ses propres prédateurs. Cela est réglé par un tweak des ***scripts EntityFOV*** et ***Interact*** qui vérifient la ***species*** de E et passent outre les conditions d'attirance pour les objets cadavres dans ce cas.

Deuxièmement, les cadavres qu'elle laisse derrière elle sont plus nutritifs que les autres ; il suffit donc, dans le script ***CarcassState***, d'appliquer un multiplicateur au Start si c'est un membre de cette espèce qui l'a fait apparaître.

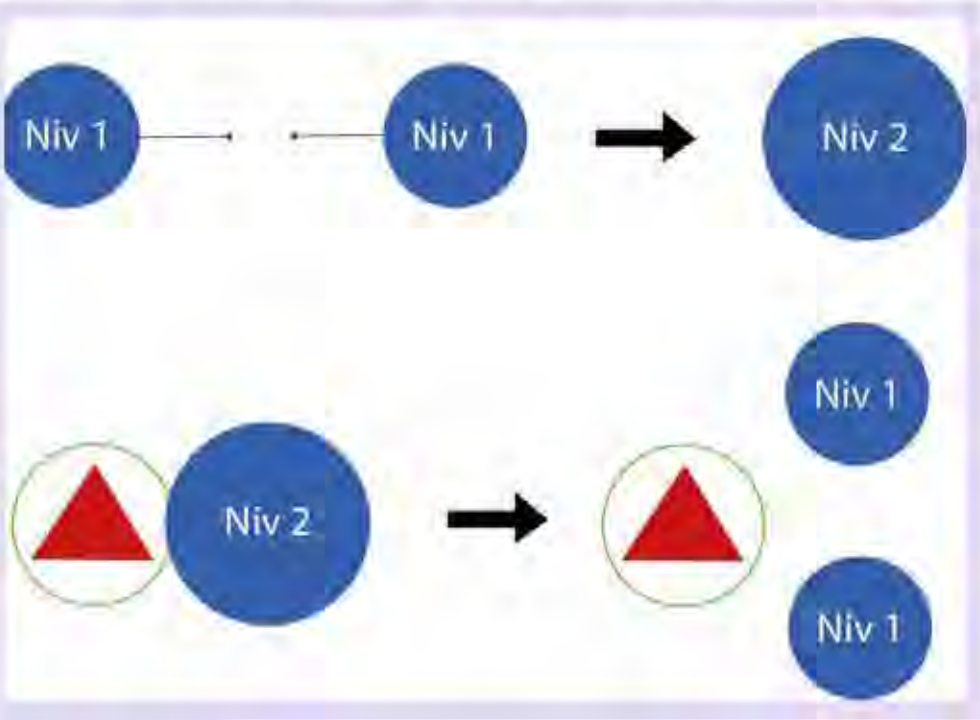
c. Entity X

On arrive maintenant dans des comportements plus complexes.

Pour commencer, un individu de cette espèce peut fusionner avec l'un de ses congénères pour former une créature plus grande, qui elle-même pourra se fusionner à une créature de même taille qu'elle-même.

Pour ce faire, un script « ***MergeOnTrigger*** » a été conçu spécialement pour elle : celui-ci propose qu'au contact entre deux individus de même « niveau » (un ***enum Level*** est créé pour l'occasion), un individu de niveau supérieur, plus gros et plus lent, est instancié, ce qui cause la destruction des individus originels. Ce processus est cependant limité par un timer qui s'active à l'instanciation, empêchant l'individu de se refusionner immédiatement, mais lui octroyant en contrepartie une invincibilité temporaire.

Ces fusions ont ensuite un comportement particulier lorsqu'elles se font chasser : au lieu d'être tuée par ***l'Interact*** de l'adversaire, une condition y est rajoutée, qui la divise en autant d'individus de premier niveau nécessaires à sa création (2 pour le niveau 2, 4 pour le niveau 3). Le mécanisme est similaire à celui de fusion (instanciation d'individus de niveau 1 puis autodestruction). Elle ne laisse alors pas de cadavre derrière elle, sauf si tuée au niveau 1.



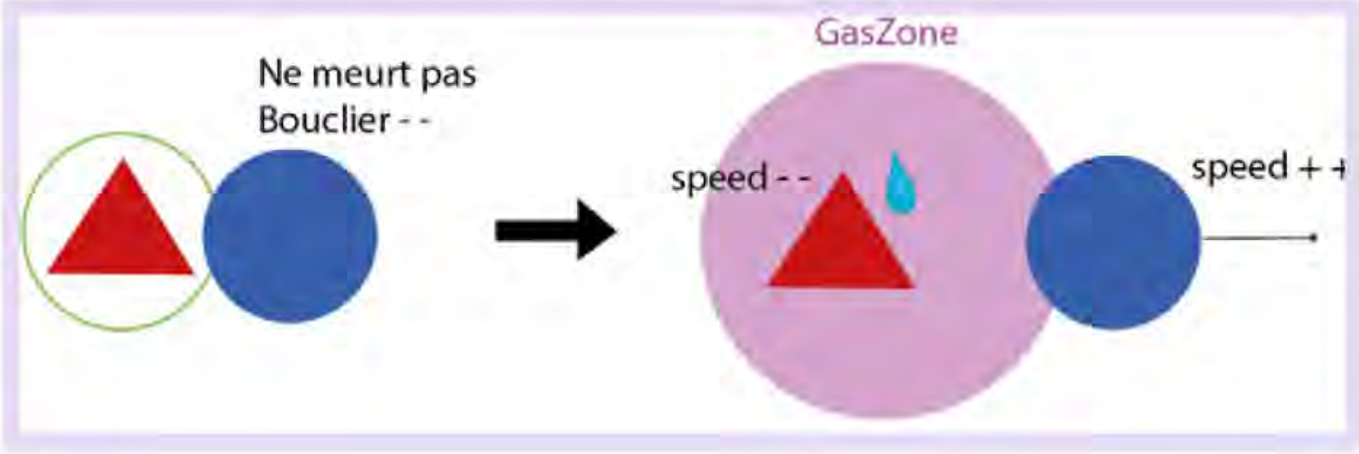
d. Resilium Toxifungi

Notre dernière espèce est la plus complexe, celle qui implique le plus d'interactions entre différents scripts.

Sa première capacité est son bouclier, lui permettant de résister à la première attaque qu'elle subit. Finalement, l'embranchement a aussi lieu dans le script ***Interact***, et cette fois-ci c'est un ***booléen isShieldActivated*** qui est toggled à ***false***, dans un nouveau script spécialisé : « ***ShieldAndToxicGas*** ». *(Cette capacité se régénère si la créature se nourrit d'assez de cadavres).*

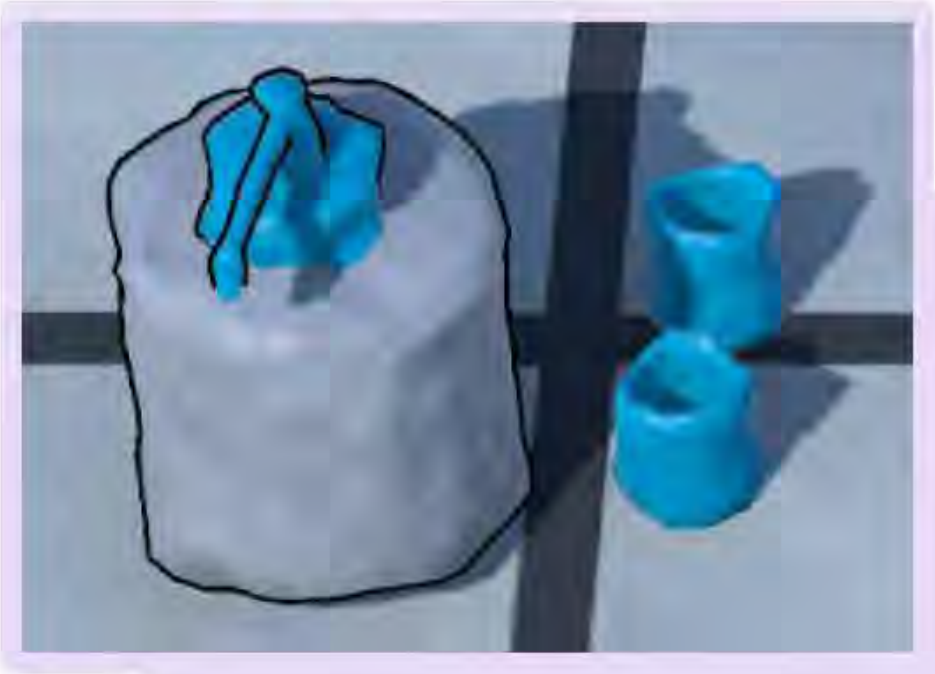
Cette information va pousser ledit script à déclencher une période d'invincibilité (via un ***booléen isInvincibilityStillRunning*** et un timer) où notre individu voit sa vitesse augmenter d'un nouveau multiplicateur : ***speedMultWhenQuickEscaping***, transmis de ***ShieldAndToxicGas*** à ***EntityMovementRandomInFOV***.

La deuxième capacité de notre espèce est de faire apparaître une zone d'effet à l'emplacement où son bouclier se brise. C'est aussi le script ***ShieldAndToxicGas*** qui gère son instanciation au même moment que les actions du paragraphe précédent. Cette zone ralentit les individus d'autres espèces ; cela est géré par un nouveau script attaché à la zone elle-même, « ***GasZoneBehavior*** », qui inflige un ***TriggerStay*** aux créatures d'espèces différentes en son sein, lançant un timer qui va être en lien avec ***EntityMovementRandomInFOV***. Tant que ce timer est lancé, ***EntityMovementRandomInFOV*** va récupérer un autre modificateur de vitesse, ***speedMultWhenPoisoned***, et le multiplier à la vitesse de déplacement. Enfin, un dernier timer régit la disparition de la zone.



III - Interactions joueur

Maintenant que le plus gros morceau à été abordé, nous pouvons passer aux éléments liés au gameplay en lui-même. Notre philosophie a été de faire en sorte que le joueur soit dans une posture de « créateur », apportant à l'espace de jeu plus qu'il ne retire. Ce qu'il apporte peut cependant avoir un comportement destructeur, ce n'est pas incompatible.



I) Eléments plaçables

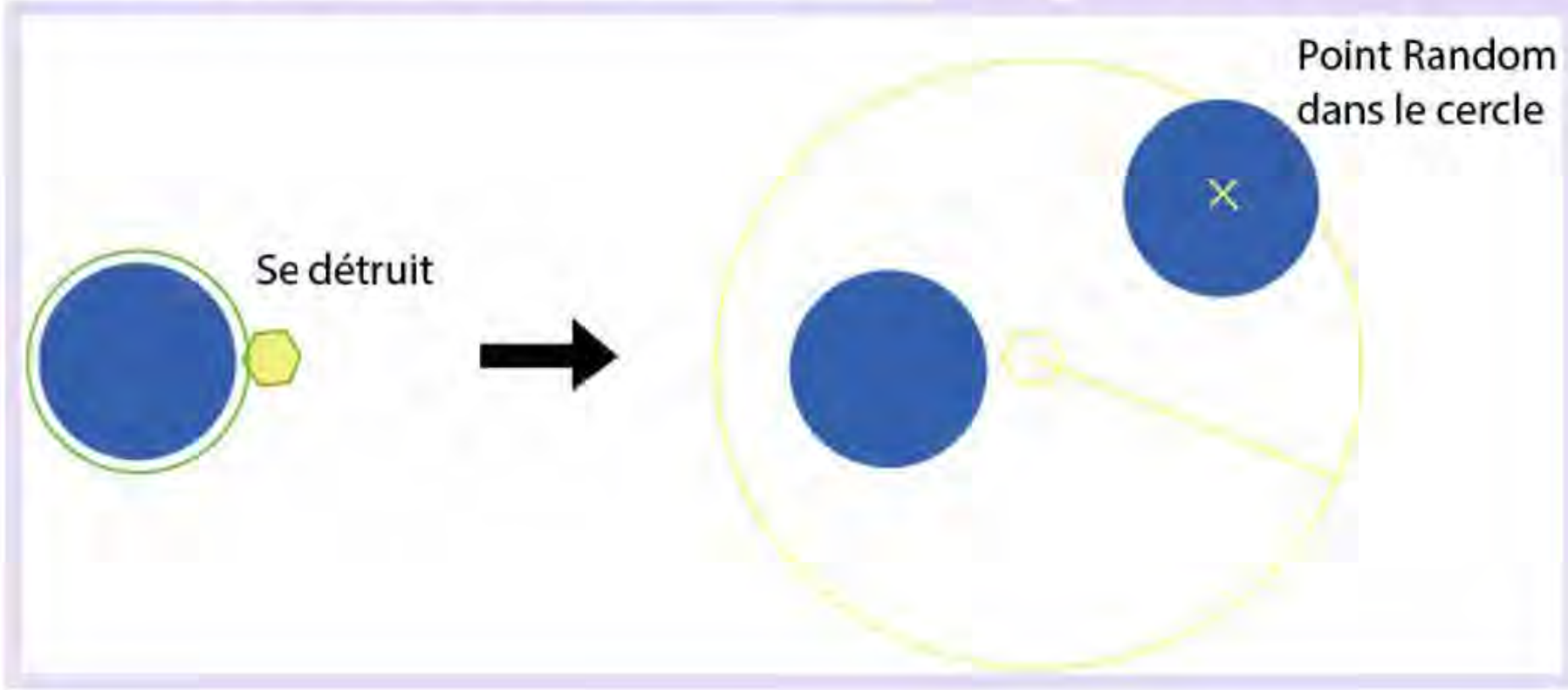
Nous allons ici parler de tous les éléments que le joueur peut faire apparaître sur le terrain. De manière générale, un script va se charger de l'instanciation de ces éléments, ***SpawnByPlayer***. Ce dernier va récupérer l'information de quel objet doit être instancié, d'après une liste de choix choices dans laquelle les inputs du joueur vont permettre de naviguer (nous le verrons en 2), puis vérifier si le clic gauche est appuyé pour ensuite instancier l'objet là où touche le raycast envoyé par le pointeur de la souris. Chaque objet, cependant, à ses propres spécificités et conditions de spawn.

a. Nourriture

La nourriture est la toute première idée que nous avons eu à propos d'interaction.

Elle permet à la créature qui entre en contact avec d'instantanément se dupliquer. Pour ce faire, elle est repérée par ***EntityFOV*** comme une proie, et déclenche donc une réaction ***Chasing*** chez toutes les créatures. Ensuite, elle porte en elle-même le script ***FoodEating***, qui récupère l'identité de la créature rentrée en contact avec, et instancie un individu de la même espèce aléatoirement dans un rayon donné, à l'aide de ***Random.insideUnitCircle***. Enfin, elle s'autodétruit une fois sa besogne terminée.

A noter également que nous avions codé la possibilité qu'une nourriture puisse faire apparaître plusieurs créatures, avec une variable ***spawningNumber*** injectée dans une boucle for, mais aussi que les enfants de nos créatures ne soient pas forcément des individus de leur propre espèce, géré par une autre variable de variance aléatoire. Cependant, ces ajouts n'ont pas été gardés dans le jeu final, un bon équilibre n'ayant pu être trouvé.



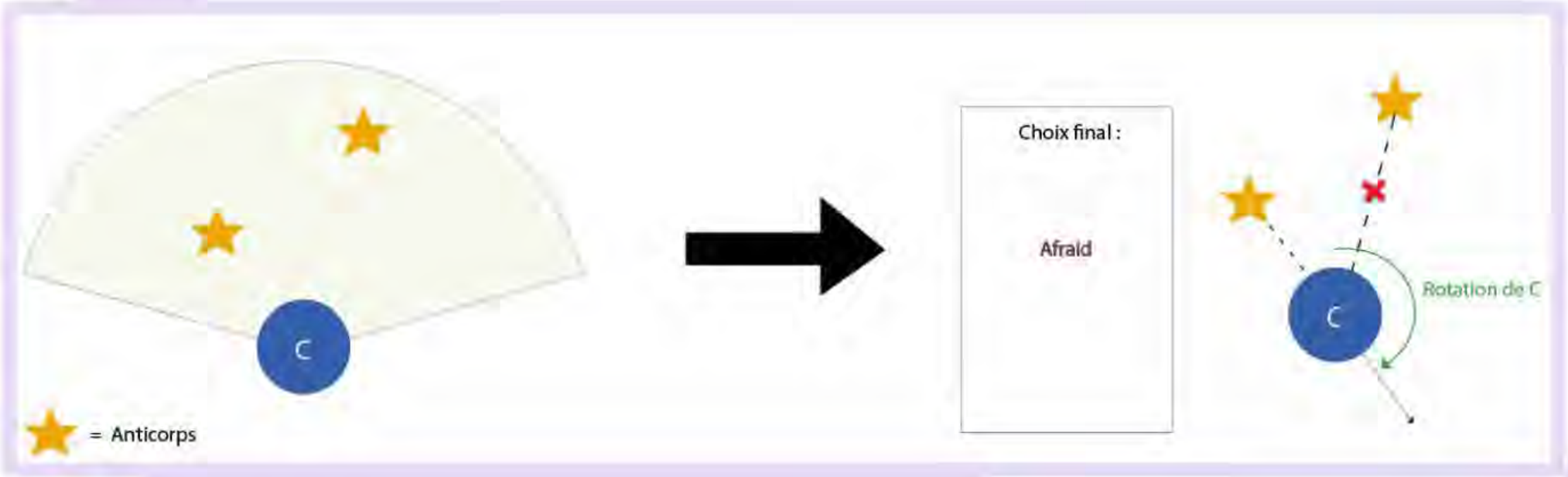
b. Anticorps

Les anticorps (à l'origine scarecrows lorsque nous n'étions pas sûrs du thème du jeu) ont été pensés comme devant faire prendre aux créatures un comportement inverse à celui de la nourriture. Là où la nourriture attire, les anticorps devaient repousser.

Ainsi, une nouvelle liste a été ajoutée aux objets détectés par ***EntityFOV*** : ***scarecrowsWithinFOV***. Si ***StateChecker*** comprend que cette liste contient des éléments alors que les autres listes sont vides (les anticorps sont derniers dans l'ordre de priorité, car ils n'ont aucun impact sur les créatures hormis changer leur direction), il fait entrer la créature dans un nouveau ***state***, ***Afraid***.

En se voyant dans ce state, ***EntityMovementRandomInFOV*** utilise le même script que quand il est dans le ***state Chasing***, mais inverse simplement le vecteur direction qui en ressort ; ainsi, notre créature vérifie la position de l'anticorps le plus proche, et adapte sa direction pour aller à l'opposé. Contrairement aux ***states Fleeing*** et ***Chasing***, il n'y a pas de notion d'endurance ou de passage dans le ***state Fatigued***, l'objectif était vraiment juste d'influencer la direction de nos créatures sans complètement changer leur comportement.

Ces anticorps n'ont aucun cooldown, aucune limite, et peuvent donc être spammés à volonté par le joueur.

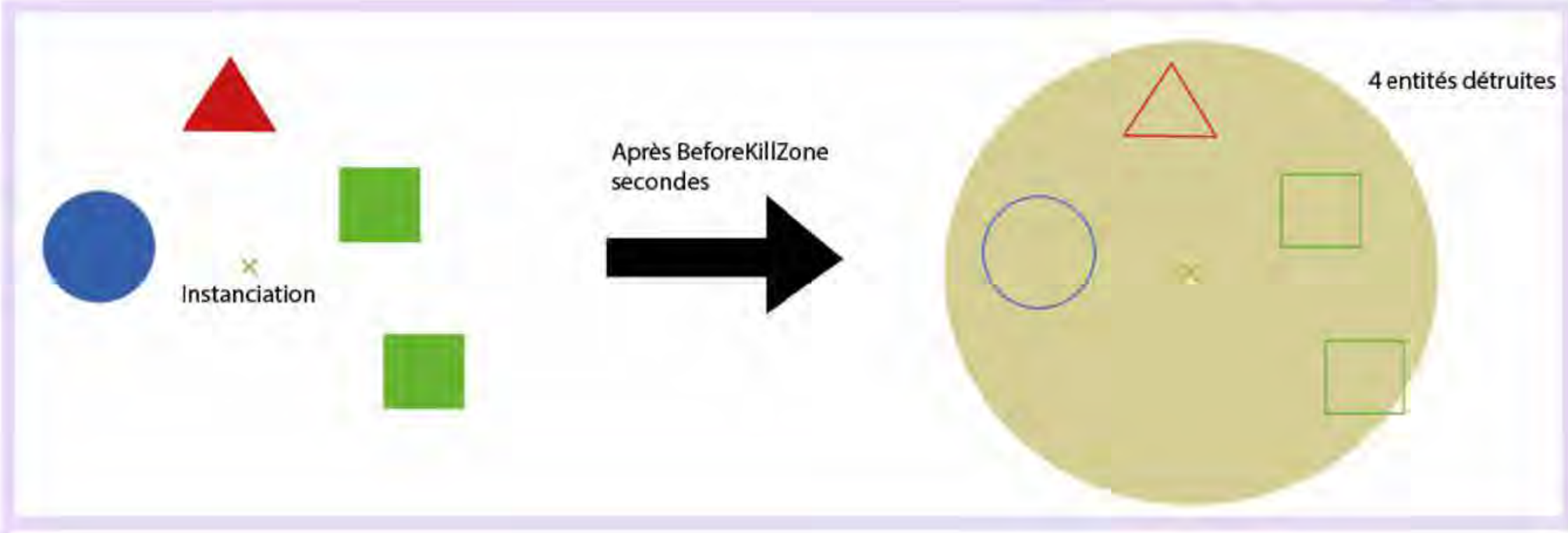


c. Gel hydroalcoolique

Dans notre jeu, l'alcool fait office d'arme de destruction massive (un peu comme dans la vraie vie, finalement).

Lorsque le joueur va l'instancier, il va d'abord apparaître sans son ***collider Trigger***, puis lance un timer dans un script dédié (***timerBeforeDestroy***) avant d'activer son ***collider***, l'objectif étant de simuler une latence entre le clic et l'action inspirée de ***September 12th***. Cette zone a une durée de vie de ***timerBeforeDestroy*** secondes, et durant ce temps, toutes les créatures qui entrent dans son ***trigger***, couvrant une surface de diamètre d'environ 3 créatures mises les unes à la suite des autres, sont instantanément détruites sans laisser de cadavre.

Tout comme la nourriture, l'utilisation de cette arme est dépendante du nombre de créatures sur le terrain. Cependant, sa relation à ce nombre est inverse ; plus le nombre de créatures est grand, plus le temps de recharge de la réserve est court, et plus le nombre d'utilisations maximale est grand, pouvant monter jusqu'à 5 si ***maxEntityCount*** est atteint.

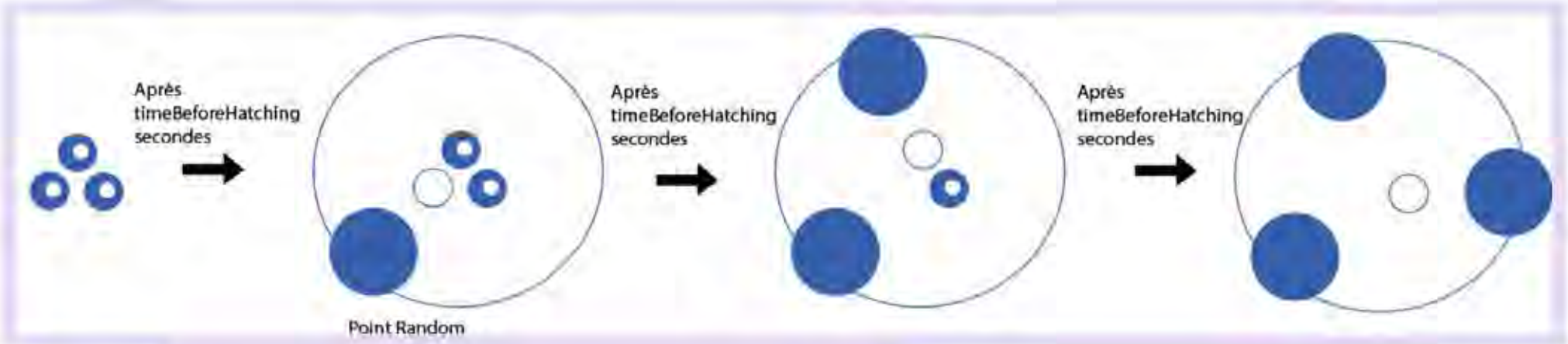


d. Souches

Le dernier élément instanciable par le joueur est la souche. Il en existe 4, une pour chaque species de créatures.

Elles fonctionnent comme des spawners automatiques, se basant sur leur script dédié, ***EggsBehaviour*** (encore une fois, nous n'étions pas sûrs de la thématique à ce moment-là) : ce dernier attend ***timeBeforeHatching*** secondes, puis instancie un individu de la ***species*** qu'il représente, et répète ce processus ***eggs.Count*** fois, ce nombre représentant le nombre d'individus par souche (3 dans notre cas). Tout au long de ce processus, leur modèle/préfab se met à jour, se désagrégeant petit à petit, pour au final se détruire après le dernier spawn.

Le pouvoir des souches étant extrêmement puissant, puisqu'il permet de faire spawner la créature exacte que l'on souhaite de manière sûre, son utilisation devait s'en trouver limitée. Pour le coup, nous avons choisi de ne rendre possible leur instanciation que lorsque la ***species*** qu'elles représente est complètement éteinte. Pour ce faire, le script ***EntityCount*** a une section classant les créatures par ***species***, et ainsi ***SpawnByPlayer*** vérifie si le compte de l'espèce demandée est à 0 pour autoriser l'instanciation.



2) Interactions dans l'UI

La dernière section qu'il nous semble intéressant de disséquer dans ce document est la manière dont l'UI est gérée. Cette partie sera plus courte que les autres, mais est d'après nous une bonne manière de clôturer cette partie.

a. Menu principal

Le menu principal n'a pas grand chose de particulier. Il est composé de trois boutons (***Components Button***) avec lesquels interagir : le bouton "Jouer" charge la scène principale, le bouton "Quitter" ferme le programme, et le bouton "Options" fait apparaître l'écran des options. L'écran des options, de son côté, possède un bouton "Retour" qui fait réapparaître le menu principal, mais aussi un slider (***Component Slider***) qui influe directement sur le volume global, en appliquant un ***SetVolume(valeur du slider)*** à la VCA venant de FMOD. Le volume peut ainsi osciller entre 0 et 2 (2 fois le volume de base).

b. Changement de type d'objet à placer

Une fois la partie lancée, le joueur doit pouvoir choisir l'élément qu'il souhaite instancier. Pour ce faire, nous avons placé des objets faisant office de boutons autour de la boîte de Pétri, chaque objet étant lié à un élément à instancier (à l'exception de l'encyclopédie, que nous verrons en dernière partie). Un premier script, nommé ***UIActions*** et placé sur chaque objet-bouton, vérifie en premier lieu par un raycast si le pointeur de la souris passe au-dessus de l'objet, lui donne le comportement de surbrillance si c'est le cas, puis vérifie si le clic gauche est activé. Si les deux conditions sont réunies, alors un ***bool isSelected*** est activé.

L'information est ensuite transmise au script ***UIManager***, situé sur le game manager. Ce dernier permet d'éviter de potentiels « clashs » de sélection, ne permettant qu'au dernier bouton cliqué d'avoir son ***bool isSelected*** à ***true***. Il permet aussi au joueur d'avoir une manière autre que le clic pour changer d'objet ; en faisant rouler la molette, le joueur parcourt une liste des objets sélectionnables via une fonction ***ChangeSelectionByWheel***, qui passe à ***false*** le ***isSelected*** précédemment activé, et passe à ***true*** celui suivant dans la liste. Enfin, après toutes ses actions, ***UIManager*** transmet l'information duquel bouton est sélectionné à ***SpawnByPlayer***, qui adapte en conséquence l'objet qui sera instancié au clic.

c. Carnet

Le dernier point de cette partie technique concerne le carnet. L'on peut y accéder via un script très similaire dans son fonctionnement à ***UIActions***, ***EncyclopedieSelecter*** (encore une fois, nous n'étions pas forcément sûrs du nom à ce moment). Ce script permet, lorsque l'on clique sur le carnet, de passer un ***bool isOpened*** à true, communiquant avec ***UIManager*** ainsi que ***UIActions*** et ***SpawnByPlayer*** pour les empêcher d'agir durant ce temps, afin d'éviter que le joueur ne puisse naviguer dans les objets alors qu'il lit le carnet. Les boutons du menu principal sont de retour sur sa page de gauche, avec pour petite différence que le bouton "Jouer" est remplacé par un bouton "Retour" qui repasse le ***bool isOpened*** à false, permettant à nouveau d'influer sur la Boîte, et le bouton "Quitter" renvoie au menu principal.

Sur la page de droite, différents compteurs en temps réel sont présents. Pour les compteurs de la section "Etat des Créatures", les textes sont mis à jour par le script ***EntityCount***, qui transforme simplement ses différentes variables avec la méthode ***ToString()***. Pour ceux de la section "Impact joueur", les trois en lien avec un objet placé sont mis à jour par ***SpawnByPlayer*** qui incrémente leur valeur affichée à chaque instance invoquée. Enfin, pour le compteur de kills par alcool, un nouveau script a été créé, ***PlayerKillCount***, qui reçoit des informations sur le nombre de créatures touchées par chaque goutte d'alcool posée, et met à jour le texte en conséquence.

Pistes d'améliorations :

- Approfondir le système de cadavre.
- Améliorer le carnet (ex : prendre un screenshot pour l'afficher dans le carnet, prendre des notes sous les screenshots etc).
- Améliorer l'interaction des créatures avec les obstacles.
- Améliorer le système pour automatiquement défaire les extrêmes.
- Améliorer les shaders de transparence.

Remerciements

Nous tenons tout d'abord à remercier l'ICAN et l'ensemble de nos professeurs pour la qualité de leur enseignement et de leur accompagnement, ainsi que pour la définition du cadre qui nous a permis de réaliser ce projet.

Nous remercions ensuite tous nos playtesteurs : nos familles, nos amis, nos camarades, et toutes les personnes ayant pu apporter un avis constructif sur notre jouet, et sans qui ce dernier ne serait pas ce à quoi il ressemble aujourd'hui.

Nous adressons également des remerciements spéciaux à Esteban TERRIEN, pour sa contribution au design sonore, et à tous ceux qui ont participé de près ou de loin à la réalisation du projet, ne serait-ce que par un conseil, une correction, etc.

Enfin, nous vous remercions vous, cher lecteur, pour l'attention que vous portez à notre jouet, en espérant que votre expérience en jouant à ce dernier soit la meilleure possible !

Bonne continuation, et amusez-vous bien !

Matthieu COLIN, Barthélemy ISLIWA, Ethan LANCELLE et Nathan RAUH.