Haida (Arthur) GUERRY , Aymeric LAVAL, Elliott SAUVENAY - 2GD - ICAN 2024-2025

Équipe

Arthur «Haida» Guerry

Game design
Programmation
Direction artistique visuelle
UX / UI

Aymeric Laval

Direction artistique sonore
Game design
Documentation

Eliott Sauvenay

Level design
Game Design
Documentation

Table des matières

Documentation technique 59

Additionnel 84

Game Design

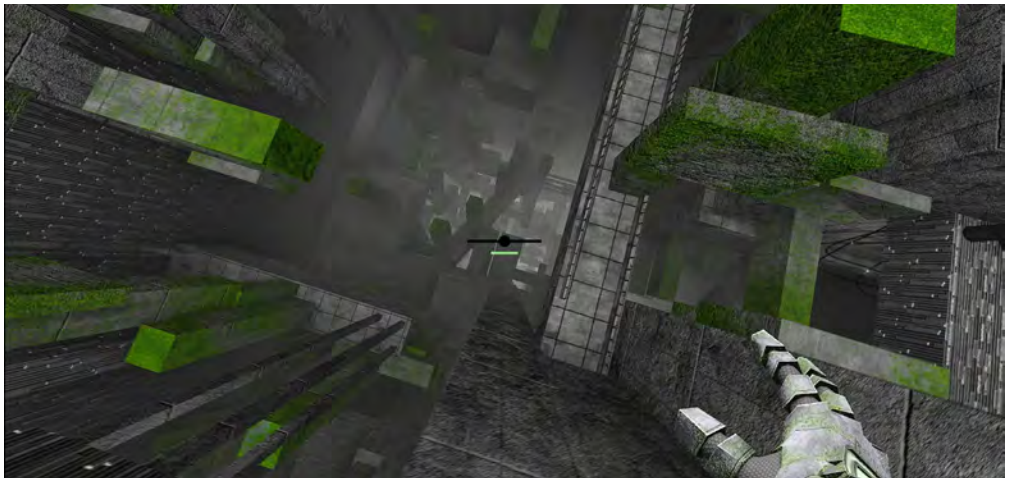
1. Introduction

Pitch

Dans Bios ex Machina, un Toy de plateforme-parkour à la première personne,utilisez vos pouvoirs pour extruder le terrain autour de vous et créer un tout nouvel environnement ou bouger constamment est la clé pour garder votre flow. Wallrun, escalade, glissade, projections, toute interaction avec les extrusions vous fera vous déplacer encore plus loin tout en créant un écosystème de déplacement avec vos extrusions.

Univers

Bios ex Machina se passe dans un univers brutaliste mystérieux dans lequel rien ne fait sens, un environnement architectural écrasant et dénué de couleur. La seule couleur de cet environnement est le vert, couleur de la végétation qui pousse sur les extrusions que le joueur crée.



Public cible

Hardcore/speedrunners.

2. Intentions

Avec ce toy, nous voulions proposer une expérience de jeu axée sur la maîtrise et les sensations de flow et de vitesse.

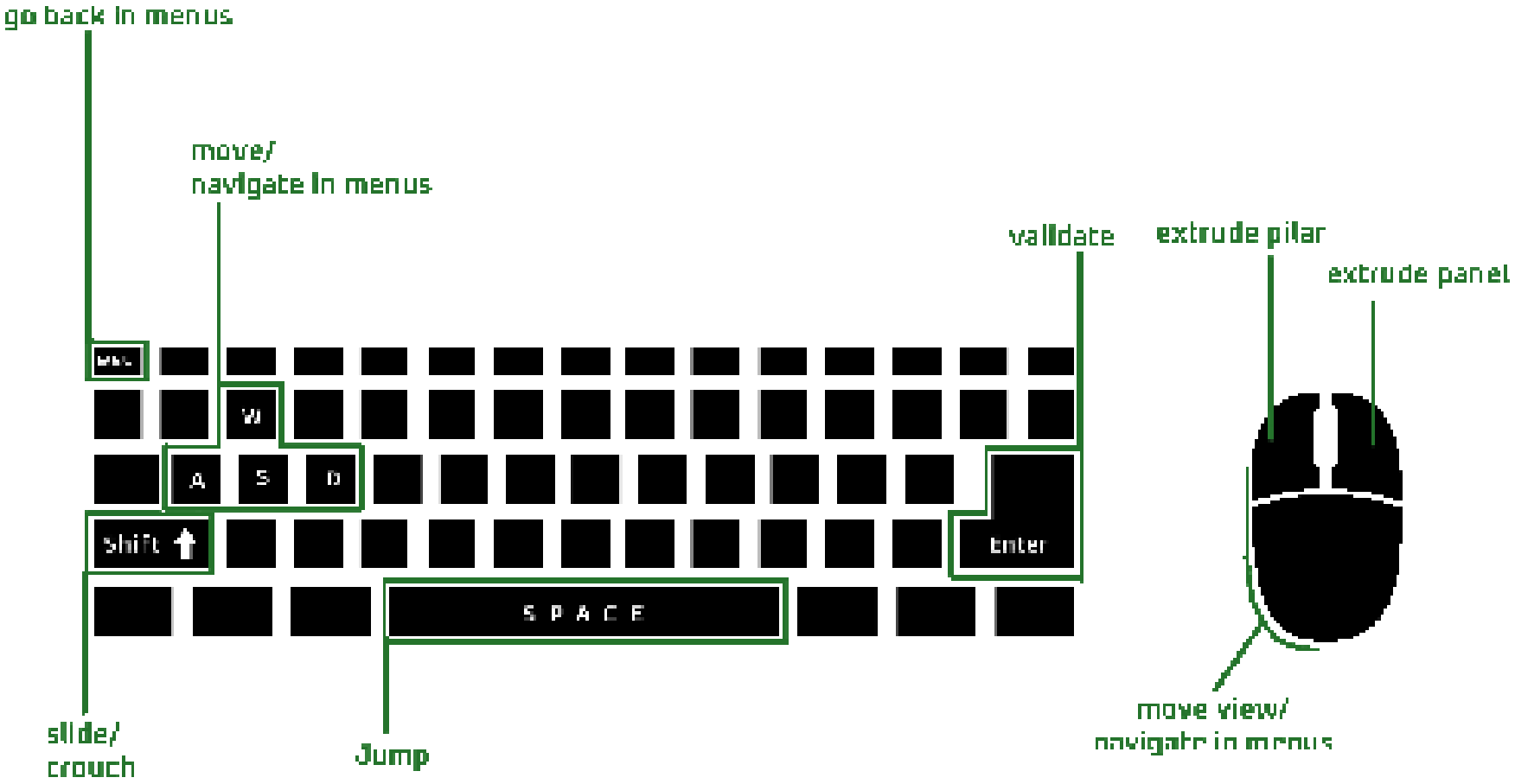
Notre but était principalement de faire un toy dans lequel le joueur utilise toutes les mécaniques pour maximiser sa maîtrise et développer son propre style de jeu.

En plus de créer un système de déplacement dépendant de la maîtrise et du flow, nous voulions, grâce à la mécanique d'extrusion, laisser le joueur créer un systeme évoluant de facon organique pour donner au joueur un écosysteme de déplacement où toutes les parties induisent des dynamiques de déplacement differentes et variées avec beaucoup de manières d'arpenter l'écosysteme créé par les piliers.

Un autre de nos souhaits était de créer une expérience qui évolue en permanence. Le level design change selon les actions du joueur pour que toutes les parties soient différentes et que l'état du système dépende entièrement des choix faits lors de la partie.

3. 3C

3.1 Controls



3. 3C

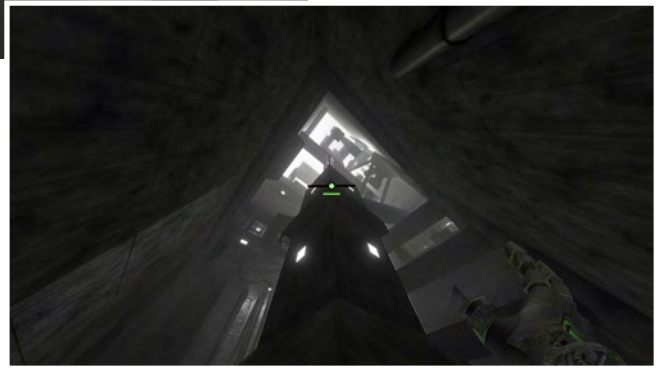
3.2 Camera

Caméra de jeu

La caméra est à la premiere personne, elle suit donc le joueur et celui-ci se déplace selon l'orientation de cette dernière.
Elle subit des changements de FOV importants lors des changements d'états de Flow (voir 5. Mécaniques), et d'autres plus petits, ressentis comme des boosts lors d'actions comme la glissade ou le wallrun.
La caméra subit également des effets de tilt lorsque le joueur se déplace à gauche ou à droite, mais également des effets de tilt plus élevés, par exemple lors du wallrun.
La caméra subit aussi des effets d'impact, comme lors de l'atterissage. Cet effet d'impact est un léger tilt très rapide ressenti comme un déséquilibre.

Caméra de menus

Dans les menus de pause et d'options, la caméra est fixe.
Dans le Main menu, une autre caméra est présente et filme un niveau, constituant l'arrière-plan du menu (voir Documentation technique, 4. Caméra du Main menu).



3. 3C

3.3 Character

Spécificités du personnage

- Il est soumis à la gravité.

- Il a une hitbox.

- Il se déplace selon le forward de la caméra.

- Il se déplace suivant la pente sur des pentes d'angle limité (max 75°).

- Il peut sauter.

- Il peut glisser.

- Il peut sauter en glissant.

- A la fin de la glissade, s'il ne peut pas se relever, il s'accroupit.

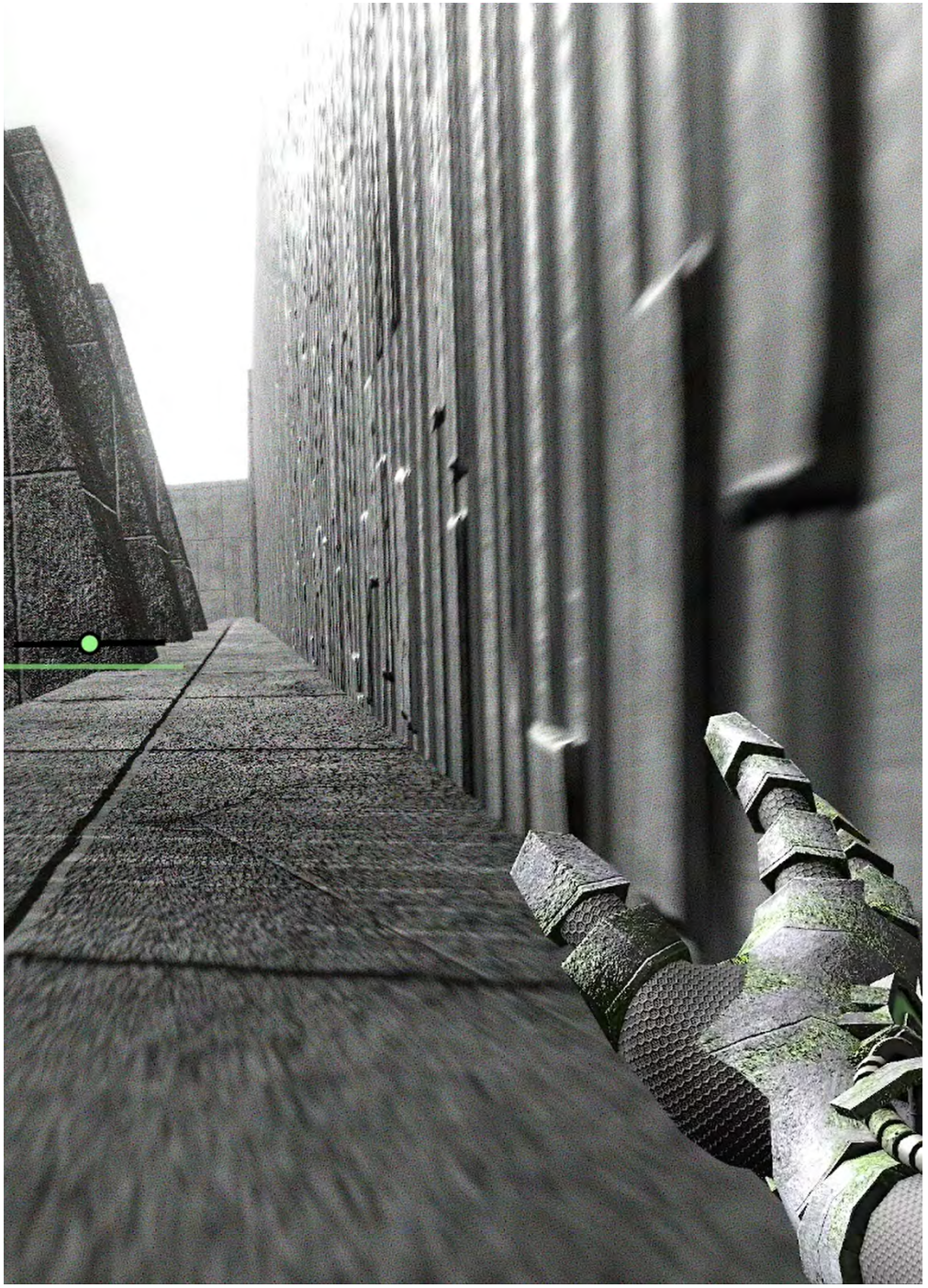
- Il avance plus vite en pente.

- Il peut extruder des formes sur certains murs et sols.

- Il peut rebondir sur les pilliers qui grandissent s'il atterrit sur le dessus du pilier.

- Il peut wallrun sur les extrusions.

- Il peut grimper sur les extrusions.



4. Références de gameplay

Déplacement

Pour le déplacement, nos références principales ont été ghostRunner,TitanFall2 et Mirror's edge pour leur déplacement : First person fluides, rapides, dynamiques et intuitifs, qui reposent sur la mobilité et la maîtrise.

Flow

Pour le concept de Flow, central au jeu, nous avons comme références les sensations de Tony Hawk pro skater 2 et Sonic rush, où le système de déplacement, quand il est maximisé par la maîtrise et l'utilisation du pan complet de mécaniques, apporte un mouvement riche, fluide et sans interruption.

Musique et système de combo

Metal Hellsinger nous a inspiré pour sa musique dynamique construite en couches de son et l'utilisation de celle-ci dans un système de combo.
Le joueur doit en permanence réaliser des actions dans un temps imparti afin de faire monter le combo et passer au palier supérieur. S'il échoue, il retourne au palier précédent.
Ce système de combo dépend de la mécanique de rythme de Metal hellsinger mais son fonctionnement axé sur la maîtrise et la régularité a inspiré notre système de Flow.

5. Mécaniques

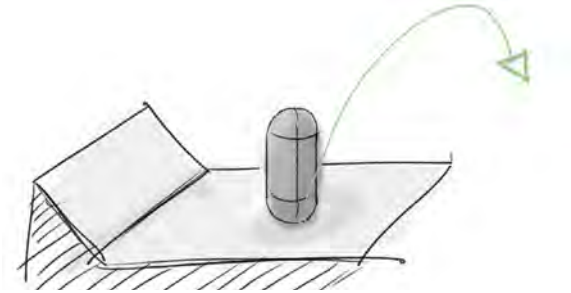
Déplacement

Le joueur se déplace à une vitesse définie sur un plan horizontal.

En pente, il se déplace légèrement plus vite, sur un plan horizontal ajusté a l'angle de la pente.

En l'air, il a moins de contrôle sur son déplacement.

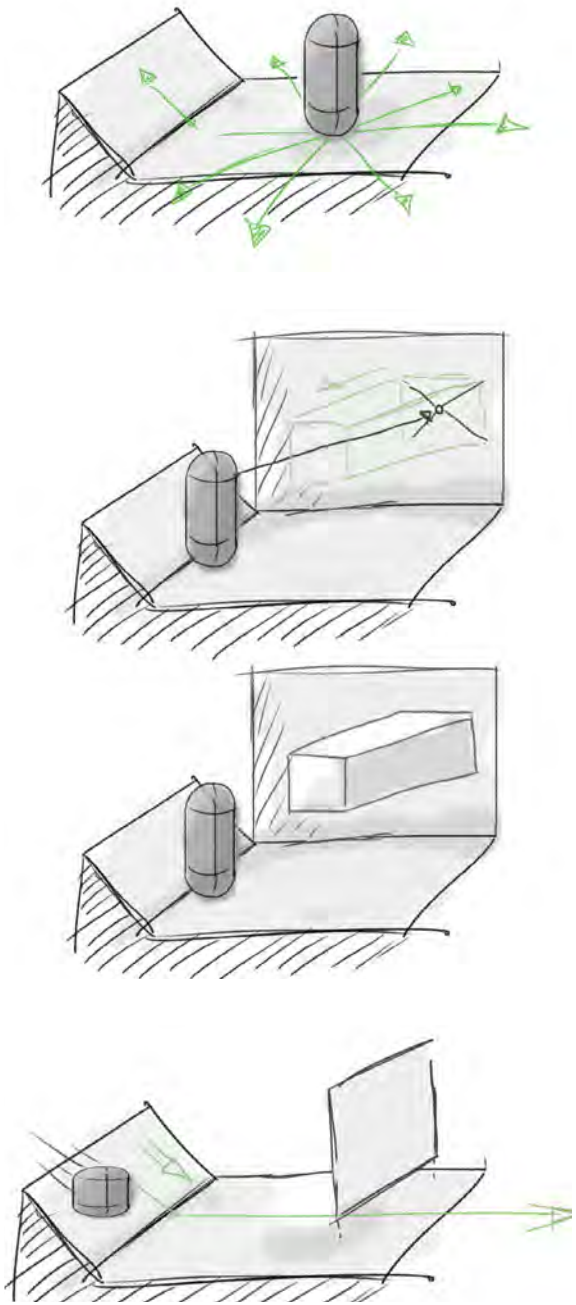
Lorsqu'il arrête de se déplacer au sol, il ne conserve pas d'inertie.



Saut

Une force est appliquée vers le haut au joueur.
Pour sauter il faut que ce dernier soit au sol, en wallrun, ou en climb.

Lorsqu'il saute en glissant, il saute plus haut et plus loin devant lui.
Lorsqu'il saute en wallrun, il saute vers le haut et hors du mur, tout comme pour l'escalade, avec des valeurs différentes.



Extrusion

Lorsque le joueur appuie sur un clic souris, si son curseur pointe vers une surface extrudable et dans la distance d'extrusion, alors, il fait apparaître un élément (pilier ou panneau) qui sort de la surface à une vitesse définie, puis lorsque l'extrusion a atteint sa taille finale ou que sa pousse est bloquée par une surface, l'extrusion s'arrête de grandir.
La mécanique d'extrusion comporte un coolDown.

Glissade

Lorsque le joueur appuie sur Maj, il se baisse et glisse sur le sol. Au bout de 1.5 secondes il arrête de glisser ; s'il ne peut pas se relever, il passe en état accroupi et il se déplace plus lentement.
Il perd en vitesse et doit appuyer sur Maj s'il veut glisser à nouveau.

La glissade n'a pas de fin lorsque le joueur glisse en pente.
En pente, la vitesse du joueur est augmentée.

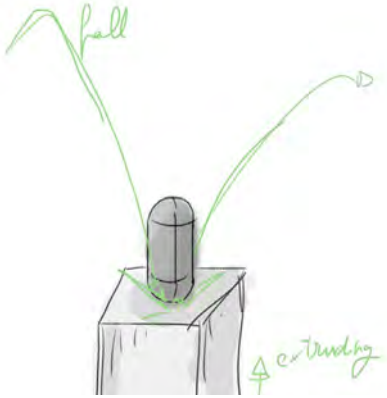
Lorsque le joueur saute en glissant, il saute un petit peu plus haut et beaucoup plus loin vers l'avant.
Quand le joueur atterrit, s'il maintient Maj, il commence une glissade dès l'atterrissage.

5. Mécaniques

Rebond

Lorsque le joueur atterrit ou wallrun sur la face supérieur d'un pilier en train de s'extruder, alors il a un temps très court pour sauter. S'il saute dans ce laps de temps, il rebondit.

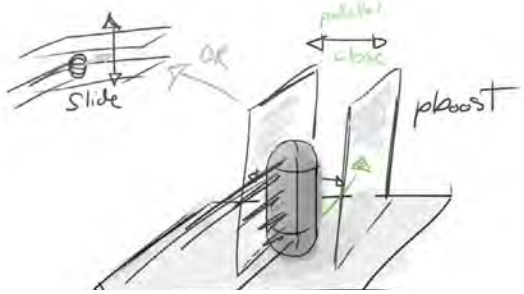
Le rebond est un saut plus puissant : plus fort verticalement, et avec une force avant également dans la direction du joueur.

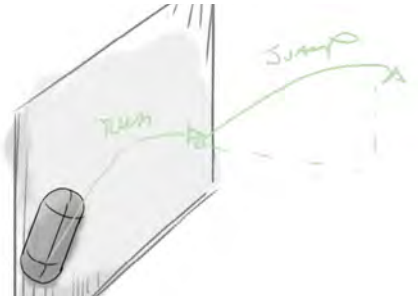


Climb

Lorsque le joueur est en l'air, s'il fait face à une extrusion et maintient Z et Space, il escalade. Après un certain temps, il s'arrête de grimper et tombe ; il peut également sauter du mur au cours de l'escalade.

Lorsqu'il cesse de grimper, et tombe d'un mur, le joueur ne peut pas immédiatement se rattraper à ce mur, mais s'il atterrit au sol, grimpe un autre mur ou effectue un wallrun, il peut à nouveau escalader ce mur.





Wallrun

Lorsque le joueur est en l'air, s'il longe une extrusion tout en avançant, il commence à wallrun le long de ce mur.

Pendant le wallrun, le joueur monte puis descend jusqu'à ce qu'il saute du mur ou tombe.

Additionnellement le joueur peut maintenir Maj pendant le wallrun pour descendre manuellement.

ParallelBoost

Si le joueur avance ou glisse entre deux surfaces parallèles et assez rapprochées, il gagne en Flow et a une impression d'accélération. S'il glisse, son timer de glissade est alors reset et il peut glisser plus longtemps.

6. Extrusions

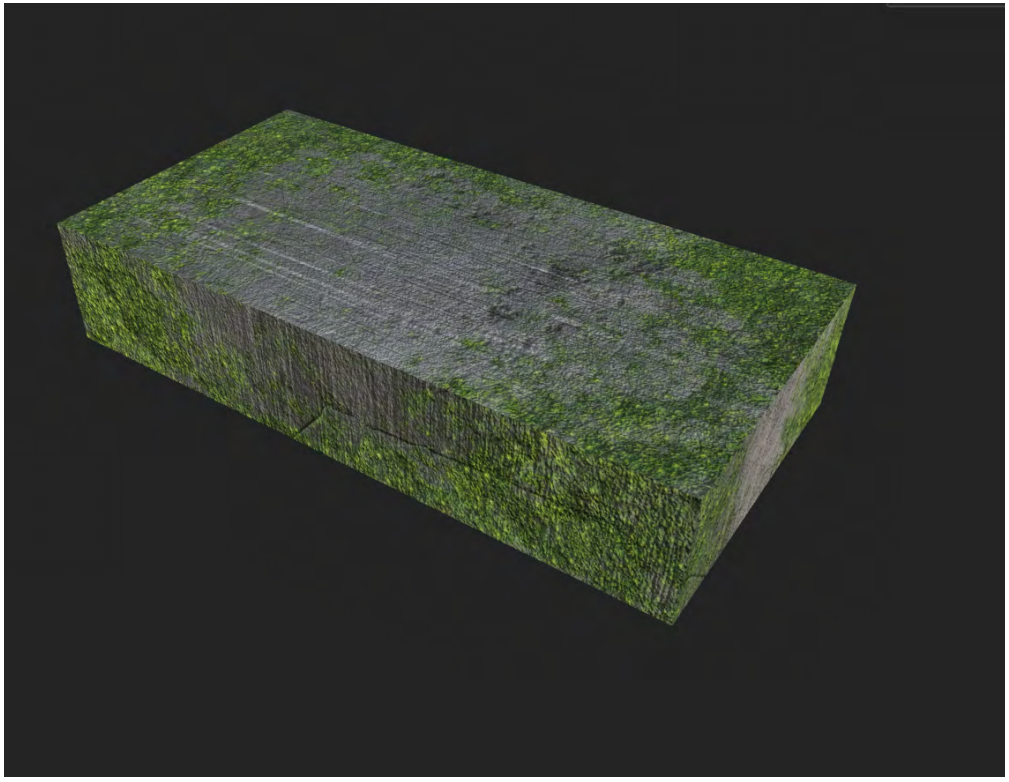
Il y a deux formes d'extrusions, le pilier et le panneau.

Pilier

Taille finale : 4 x 4 x 16 mètres

Vitesse d'extrusion : 5.3 mètres par seconde

Panneau



Taille finale : 10 x 5 x 1.7 mètres

Vitesse d'extrusion : 2.3 mètres par seconde

7. Mécanique de Flow

Flow

Notre vision du flow dans le game design de Bios ex Machina est un état dans lequel le joueur se déplace dans s'arrêter tout en serpentant à travers le niveau en utilisant diverses mécaniques de déplacement dans un mouvement parfaitement fluide et continu.

Cet état est maximisé :
- quand le joueur maîtrise le système au maximum,
- quand le joueur utilise les mécaniques de déplacement liées aux extrusions (wallrun, climb, parallel boost, rebond) alternés avec la glissade,
- quand les distances de LD sont assez rapprochées pour que le joueur puisse le parcourir sans tomber.

Lorsque le joueur « extrude », ses extrusions permettent donc de raccourcir les distances, de créer des espaces de parallel boost, d'étendre les surfaces de glissade et de créer des surfaces propices aux wallrun, climb et rebond.
Ainsi, les extrusions participent activement à créer cet écosystème de déplacement fluide et ininterrompu.

Système de Flow

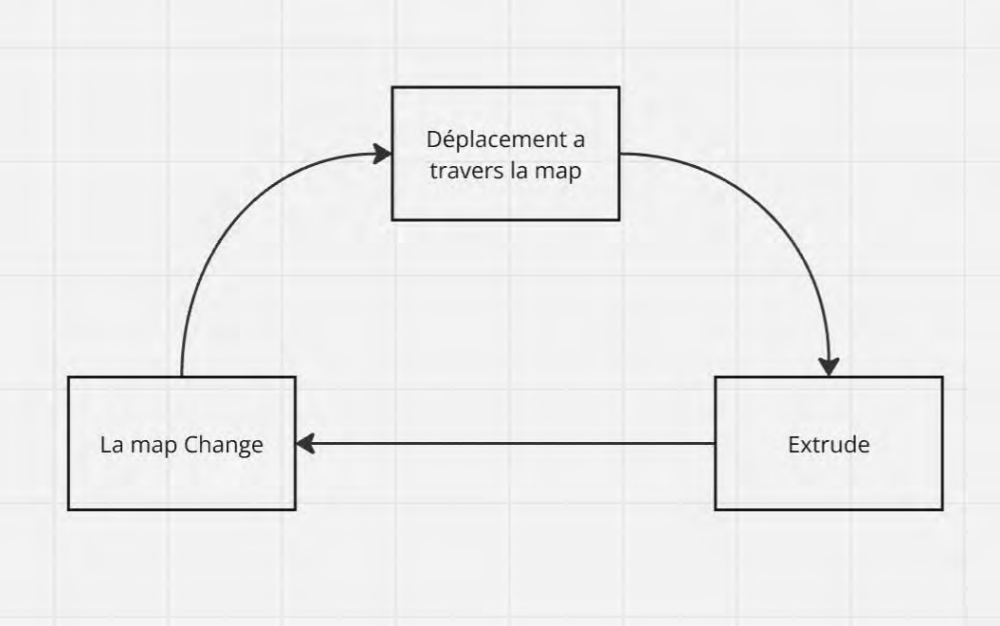
Pour donner du feedback sur le flow du joueur, nous avons créé le système de Flow. Dans ce système, le joueur doit effectuer une action dite « de Flow » toutes les x secondes ; s'il maintient son état pendant y secondes, alors il passe au palier supérieur de Flow ou le timer x est plus court.
S'il ne réussit pas à maintenir son flow, alors il retourne à l'état antérieur.

Ce système permet de garantir que le joueur maximise son flow en faisant régulièrement des actions liées aux extrusions.
Ainsi, plus les extrusions sont nombreuses le plus le joueur peut maintenir son flow facilement.

Cette mécanique explicite donc une sensation de flow implicite du joueur pour y lier du feedback et donc de la gratification et de la motivation, tout en lui faisant maximiser son écosystème de déplacement. Elle fait le palier entre flow ressenti et flow réel.

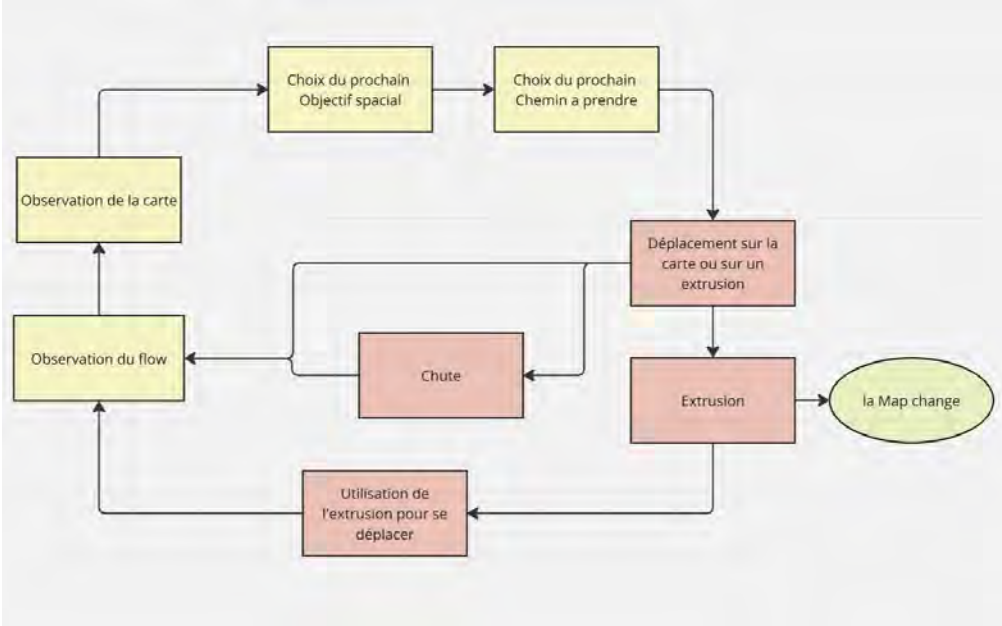
8. Métaboucle

Métaboucle



Le jeu se compose de 3 phases. La boucle peut durer quelques secondes à peine : il s'agit du déplacement, de l'extrusion qui a comme conséquence un changement de la map, puis retour au déplacement.
Dans ces étapes se placent les différents choix du joueur, les modulations actives et passives de la carte, ainsi que la retraversée à travers le nouveau level design créé.
Lors de l'étape déplacement, le joueur se déplace à travers la carte et souvent en utilisant le panel de mouvements lié aux extrusions.

Boucle de Gameplay



Dans la boucle de gameplay, le joueur ne fais pas attention à l'état général du niveau après qu'il ait effectué une extrusion. Il fait une extrusion dans un but précis de mouvement et le changement de la map est davantage une conséquence de l'extrusion. Ainsi, lors du prochain loop de la boucle, lors de l'analyse de son environnement, il prendra en compte la nouvelle géométrie.
Le joueur place donc ses extrusions activement, mais leur incidence sur le long terme est passive.

9. Système

Dans son état initial, le système tend vers l'immobilité.

Pour contrer cet état, le joueur peut se déplacer, mais tout est trop loin et inaccessible, le joueur extrude donc des surfaces pour se créer des passages.

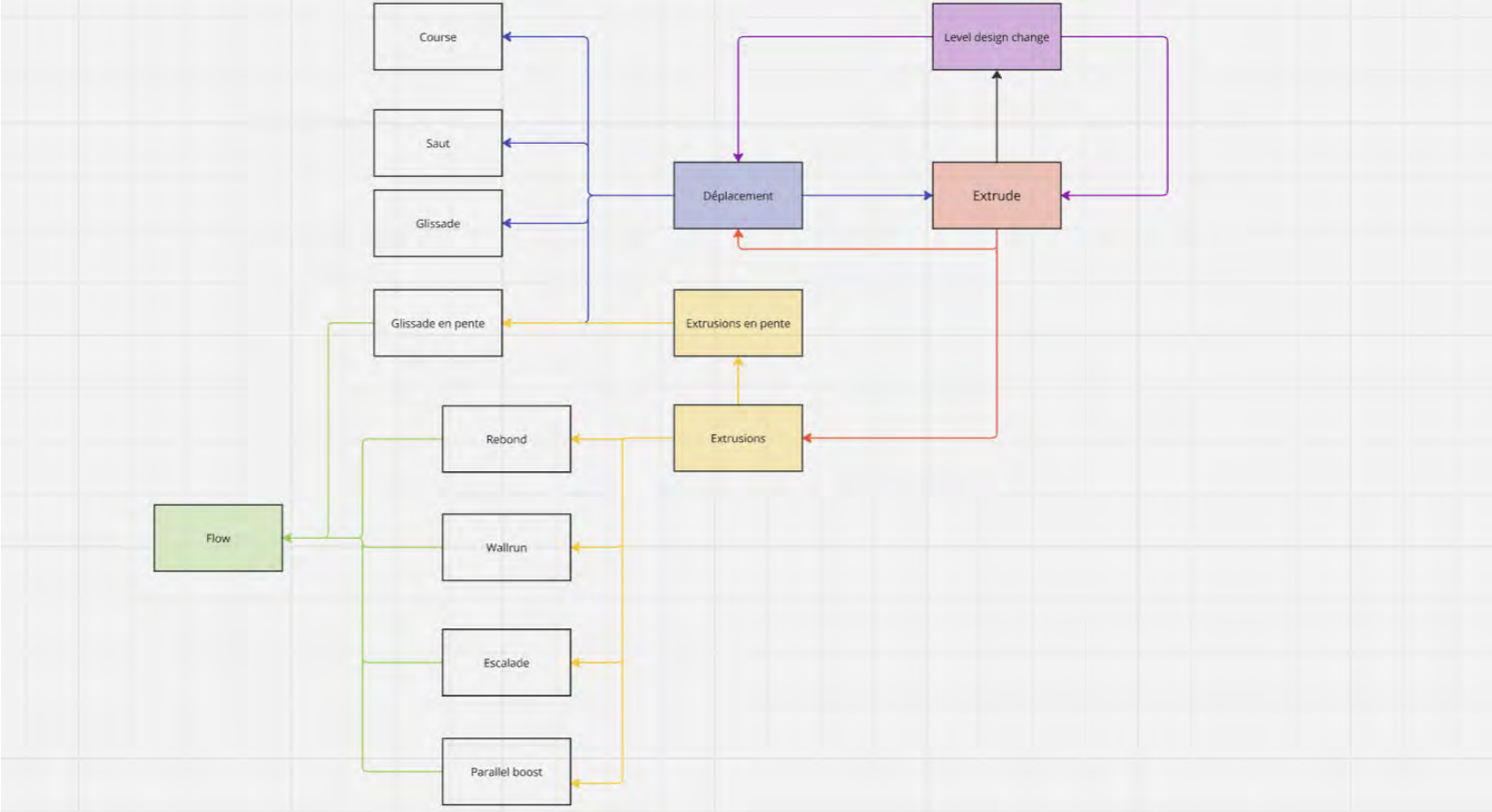
De cette manière, le système tend vers un nouvel état au fil des extrusions : l'état saturé.

Le joueur peut contrer cet état en changeant d'endroit, ou en faisant plus attention à ses extrusions, mais lors des longues parties cet état est inéluctable.

Lors de la partie, le joueur monte, que ce soit pour éviter la saturation ou simplement pour bouger vers la position inverse de sa position initiale. En effet, lors du démarrage du système le joueur se trouve au sol ; on lui montre une lumière au plafond, au loin. Il veut donc s'y rendre et contrer le système pour y aller. Le joueur tend à tomber, et donc retraverser ses extrusions, et composer avec les conséquences, qu'elles soient perçues comme positives ou négatives.

10. Interactivité entre les mécaniques

Schéma d'interaction entre les mécaniques



Ce schéma montre les liens entre les différentes mécaniques de jeu. On y voit que le déplacement influence les extrusions, qui influencent la carte, qui influence elle-même le déplacement. On y voit également les influences liées au déplacement : le déplacement de base est lié à des mécaniques comme la course, le saut et la glissade, tandis que les mécaniques liées aux extrusions sont le wallrun, la glissade en pente, le parallel boost et l'escalade.
On y voit surtout que toutes les mécaniques liées aux extrusions sont liées au Flow, et donc que le flow en est dépendant.
Ainsi, les extrusions sont centrales et nécessaires pour maintenir cet état.

11. Retraversée

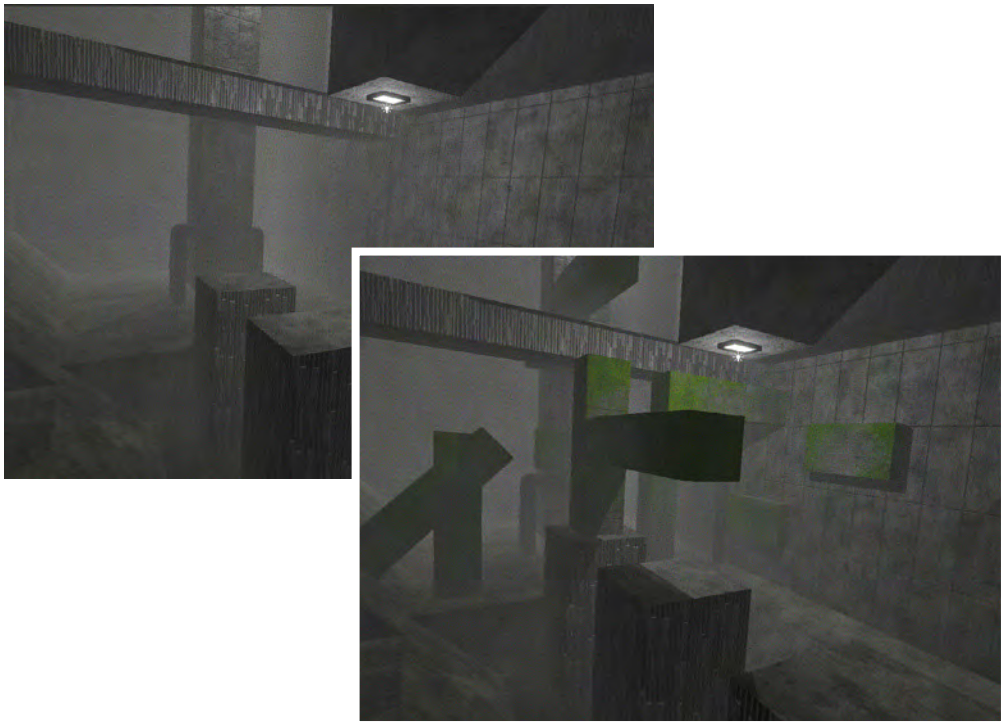
La retraversée est un élément extrêmement important de l'évolution du joueur à travers le level design et donc du gameplay.

En effet, la map étant verticale, le joueur extrude pour monter, mais ce level design est de plus en plus compliqué et pousse le joueur à tomber, et pour remonter il peut utiliser les extrusions placées précédemment sur son chemin.

Le joueur extrude donc pour une raison primaire de déplacement, puis retraverse le terrain et réutilise ses extrusions.

Les extrusions ont donc été pensées pour avoir 2 utilisations : l'utilisation primaire et l'utilisation passive.
Les utilisations primaires sont des interactions seulement disponibles lors de l'extrusion. Les utilisations passives sont celles que le joueur va mettre à profit lors de sa retraversée.

Les piliers sont pensés comme plus polyvalents et plus verticaux tandis que les panneaux sont plus horizontaux, plus axés sur le maintien de flow ou encore pour supporter les utilisations de piliers, par exemple en plaçant une plateforme pour élargir la géométrie d'un pilier.



Panneau

Primaire : wallrun, plateforme
Passive : plateforme, nouvelle géométrie, wallrun, parallel boost, escalade

Pillier

Primaire : propulsion, pente, escalade
Passive : nouvelle géométrie, pente, escalade, wallrun, parallel boost

(les utilisations sont classées par importance/fréquence d'utilisation chez le joueur)

12. Piliers de GD

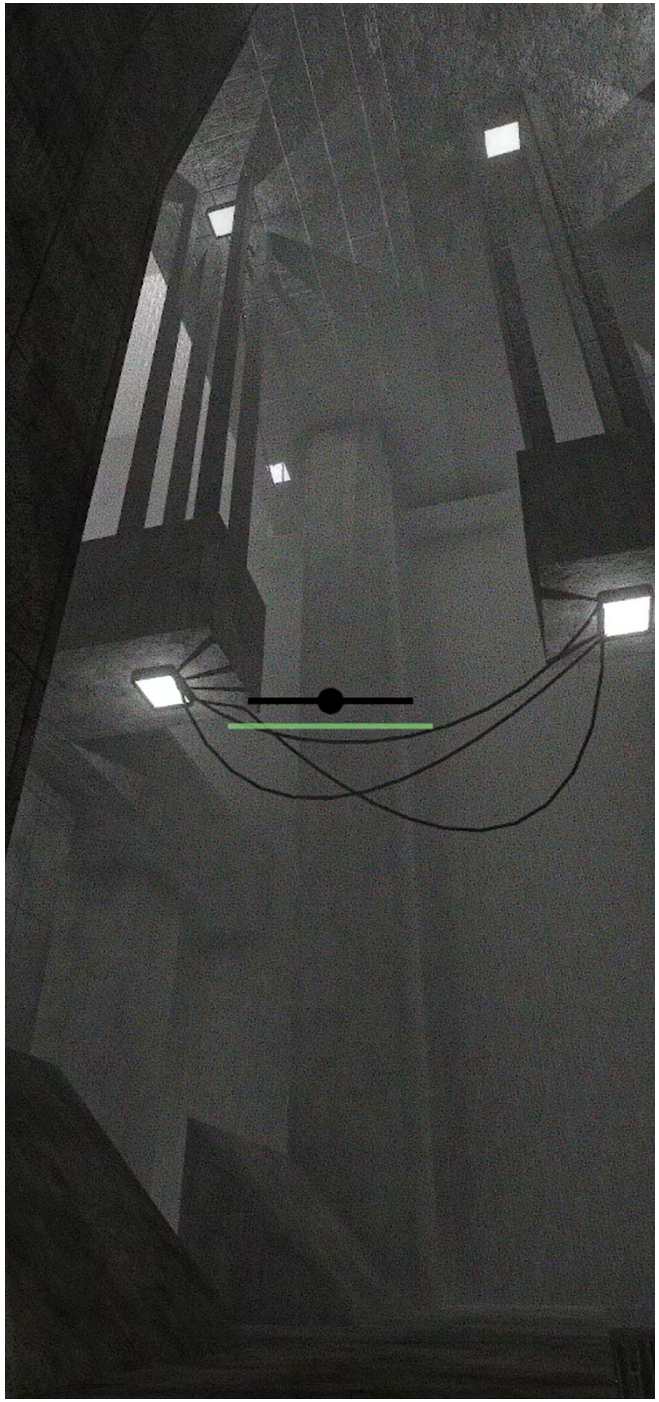
Bios ex Machina fonctionne sur 3 piliers de design :
-La création d'un écosystème
-L'exploration et réexploration
-Le Flow
Ces 3 notions sont interconnectés et constituent la majorité de nos choix de design et de gameplay.

Flow

Comme dit précédemment, chaque extrusion participe à la capacité du joueur à maintenir son flow. Les interactions primaires lui permettent de gagner du Flow en premier lieu tandis que les passives lui permettent de le conserver lorsqu'il retourne dans dans parties de la carte qu'il a déjà visité.
Les choix concernant le Gamefeel comme les contrôles, les Feedbacks constants de la caméra, le travail sur les sensations de glissade, de propulsion, de physique, d'accélération vont dans le sens de la sensation de flow.

Exploration et réexploration

Les extrusions servent à explorer la carte, à se déplacer à travers elle, à serpenter à travers le niveau. Le joueur n'est pas motivé par l'exploration uniquement grâce à l'architecture et son envie de découvrir la carte, il est surtout motivé par son envie de se déplacer au travers, de découvrir les interactions avec l'architecture de la carte.
Or, lorsque le joueur se déplace au travers de la carte, il change sa géométrie, et est encouragé lorsqu'il retraverse cet endroit à découvrir cette nouvelle géométrie.
Ainsi l'exploration d'Bios ex Machina est une découverte constante dépendante uniquement des choix et des conséquences des extrusions du joueur.
Le LD incite également le joueur à monter, mais celui ci s'avère de plus en plus difficile, le faisant redescendre et le forçant à retraverser son ancienne géométrie changée.



12. Piliers de GD

Création d'un écosystème

Lors de son déplacement, le joueur est amené à placer, de façon réfléchie ou non, des extrusions sur la carte. Ces changements offrent un nouveau level design où les metrics sont plus rapprochées et plus propices au mouvement.
Les extrusions, comme vu précédemment ont également des utilisations passives.
Ainsi, lorsque la carte se remplit d'extrusions, le joueur gagne plus de moyens de déplacement, de géométrie extrudable et de géométrie traversable.

Le parallel boost est une mécanique faite pour maximiser cet écosystème de déplacement où toutes les parties amènent une nouvelle manière de se déplacer et de gagner du Flow. Le parallel boost est une mécanique passive qui permet au joueur de regagner du Flow en plus d'une impression de vitesse.
La rotation des extrusions est toujours des multiples de 90 degrés, sauf en pente. Ainsi, ils sont presque toujours parallèles les uns aux autres, et quand deux extrusions sont parallèles et proches, le joueur peut se faufiler entre elles ou glisser dessous afin de profiter du parallel boost.
Et comme vu précédemment, au fil de la saturation du système en piliers, la géométrie se resserre et devient remplie d'espaces négatifs parallèles par la même occasion.
L'environnement lors de la saturation devient de plus en plus propice au parallel boost, en plus de devenir plus traversable, créant ainsi un écosystème de déplacement qui évolue organiquement au fil des actions du joueur.

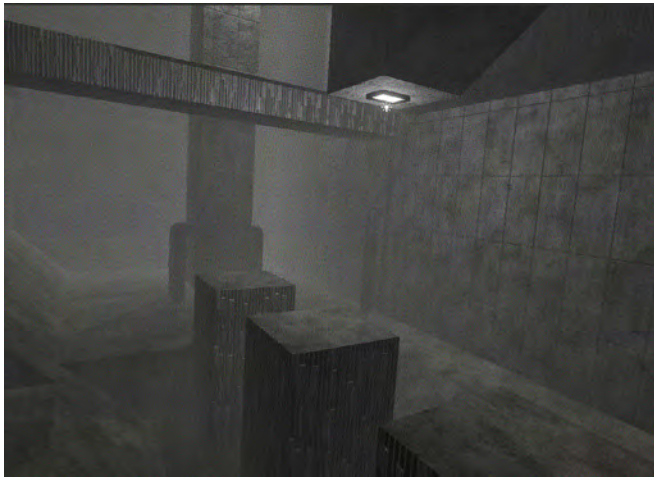
Conséquence du système : la saturation

Le système, avec les actions du joueur, tend vers un niveau de saturation où il est de moins en moins traversable car trop fourni, jusqu'à devenir entièrement inaccessible. C'est un moment où le joueur change de zone ou arrête la partie.

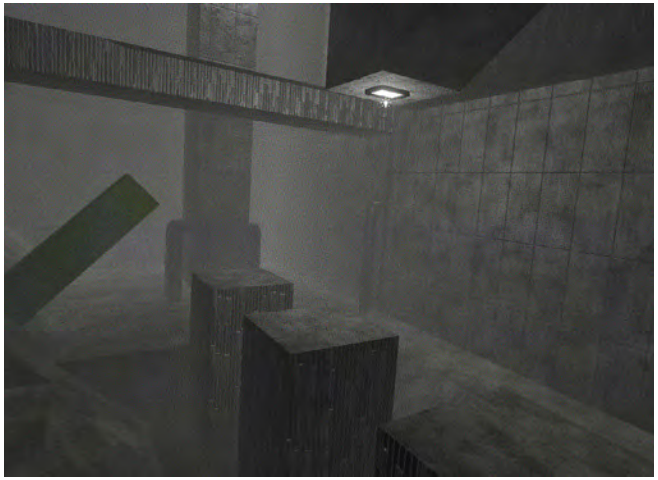
Cette saturation est assumée par l'impossibilité de détruire les extrusions, ainsi tout changement est définitif : le joueur doit réfléchir à ses extrusions, malgré le fait que cette finalité est inéluctable si le joueur joue sans s'arrêter.

13. Evolution du LD

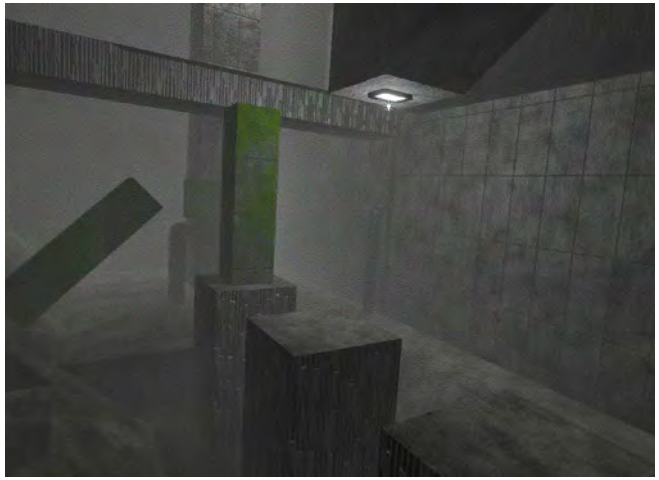
LD1



LD2



LD3



LD4

LD5

LD6

14. Gamefeel

Pour travailler sur le GameFeel, nous avions plusieurs axes que nous devions maîtriser dans le projet : l'intuitivité, la réactivité et le « Juice ».

Intuitivité

Afin que le contrôle du personnage soit intuitif, les contrôles tiennent en 6 boutons : ZQSD pour le déplacement, Space pour le saut et l'escalade, Shift pour la glissade.
Le wallrun et l'escalade se font entièrement selon l'angle du personnage par rapport au mur ; la condition repose entièrement sur l'intuitivité et les prénotions du joueur dépendant des contrôles d'autres jeux.

Pour aller dans le sens de l'intuitivité et des attentes du joueur vis-à-vis du mouvement, le character controller fonctionne selon une architecture de forces plutôt que de vélocité. Ainsi, le joueur ressent les accélérations du character controller, comme il pourrait s'y attendre pour les jeux du genre parkour à la première personne. Ces forces sont couplées à des codes de transition entre les différentes vitesses de déplacement, le character controller est donc parfaitement fluide, ne casse pas l'immersion du joueur, et donc participe à son gamefeel.

Additionnellement, des comportements physiques attendus par le joueur sont exagérés et codés, comme le rebond sur les piliers ou la glissade. Ces comportements sont intuitifs et attendus par les joueurs mais leur fonctionnement est un mélange entre l'intuitivité, pour aller dans le sens du gameFeel, et l'exagération pour aller dans le sens du Flow et de notre système de jeu.

14. Gamefeel

Réactivité

Pour ne pas sortir le joueur de son état de flow et favoriser le gameFeel, les contrôles sont réactifs. Peu importe le mouvement que le joueur fait, il n'est jamais bloqué dans un mouvement, il a le contrôle total et immédiat de l'état de son personnage. Par exemple, à tout moment durant le wallrun il peut décider de sauter pour en sortir et le résultat de ce saut est immédiat.
Chaque action que le joueur effectue est immédiate pour aller dans le sens du speedrun et du gameFeel extrêmement réactif et rapide que nous voulons proposer au joueur.

Juice

Le Juice de Bios ex Machina est principalement lié au tilt de caméra et au changement de FOV.
le camerawork comprend des tilt lors du wallrun, de la glissade, du sway lors de la course (réduit en l'air), un headBob, ainsi que des boost légers de fov durant le slide, le wallrun et plus grands durant le ParallelBoost.
Il y a également un impact d'atterrissage à la caméra,
Les changements de FOV liés au flow sont également un effet de Juice principal.
Nous pouvons aussi ajouter les animations des mains.
Ainsi la caméra et la main du personnage répondent à chaque action apportant du Juice au toy et favorisant le gameflow du joueur.

15. HUD

Le HUD doit respecter certaines contraintes :
Il doit être sobre et facilement lisible pour ne pas détourner l'attention du joueur, et il doit se trouver au centre de l'écran. De cette manière, toute l'attention du joueur y est fixée en permanence, il ne regarde presque jamais les cotés de l'écran.

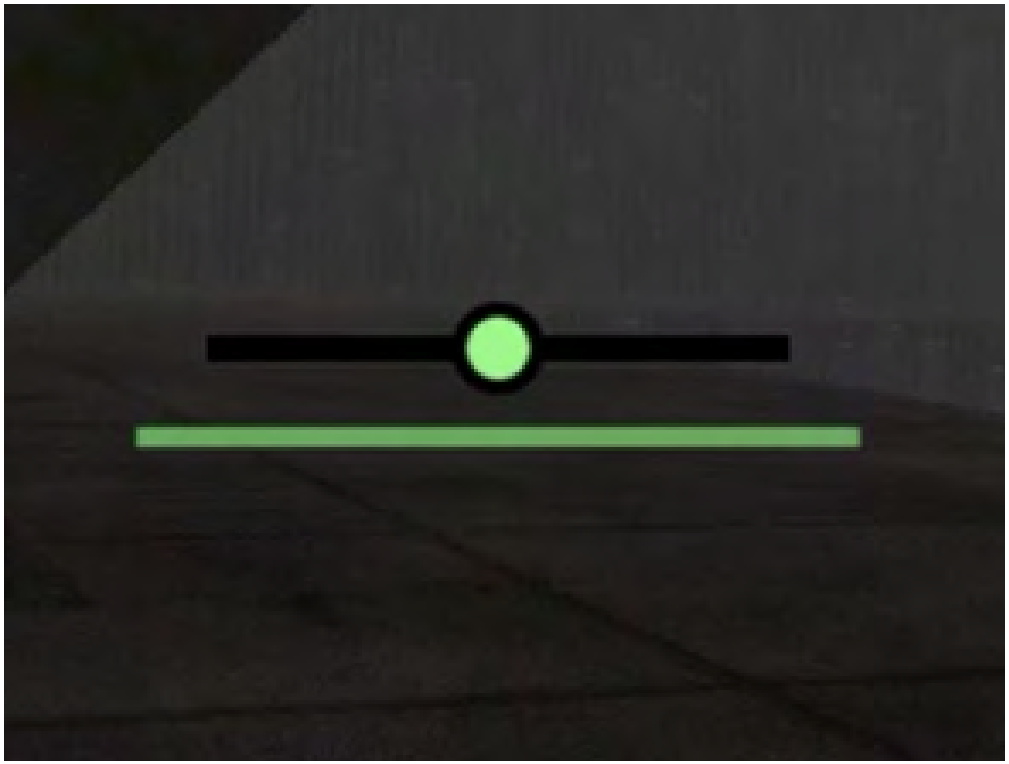
Le hud est donc composé de trois éléments principaux :

- Le curseur, qui indique au joueur où il va extruder. La couleur du curseur indique s'il peut extruder.

- La barre de timer diminue avec le timer du système de Flow. A chaque action de Flow, il revient à sa longueur maximale.

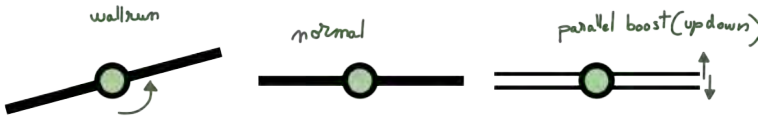
- La flowbar réagit à chaque action de flow, et donc à chaque interaction avec les extrusions.

Le HUD est purement informatif, à l'exception de la flowbar qui, en plus d'être informative, ajoute un élément esthétique par ses mouvements.



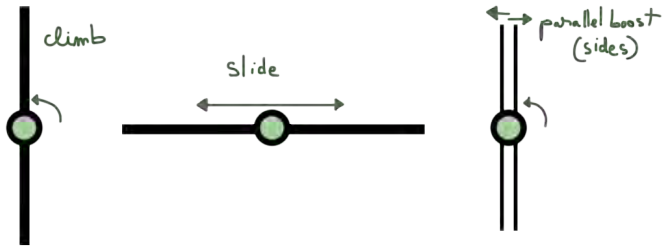
16.Tableau de Feedbacks

Event	Camera	Animation	Son	Flowbar
Mouvement avant	HeadBob	mouvement des bras	bruit de pas	
Mouvement sur le coté	Tilt sur le coté	mouvement des bras	bruit de pas	
Saut			son de saut	
Glissade	Tilt leger a droite,FOV+	changement d'idle de la main	Son continu de gfrottement	scale x ++
Wallrun	Tilt coté inverse du mur, FOV+	anim main contre le mur	Pas + frottement	Tilt opposé a l la cam (meme valeur)
Escalade		anim mains escalade	pas + son des mains	position verticale
Chute			son de vent croissant	
Atterissage	Tilt extremement leger: impact		son d'impact	
Rebond			son de saut	Scale y + (impact)
Extrude		main extrusion	son d'extrusion	
L'extrusion grandit			grondement spatialisé	
Wallrun saut / Escalade saut			son de saut	Scale y + (impact)
Flow palier +	FOV +		musique +	
Flow palier -	FOV -		musique -	
Movement air coté	Tilt leger	Idle main air		
mouvement air avant		Idle main air		
Parallel boost	FOV+			tilt selon l angle des murs + dédoublement



Tous les événements apportant du Flow ont un feedback lié à la Flowbar, qui notifie le joueur dès qu'il gagne du Flow en se déplaçant.

La Flowbar est un élément de feedback et esthétique.



17. RGD

Bios Ex Machina étant un toy, le joueur n'a pas d'objectif fixé par le système, mais les objectifs qu'il se fixe lui-même peuvent se rapprocher de paramètres et challenges de RGD.

Dans Bios Ex Machina , la plupart de ces challenges relèvent de la précision, la prédiction et le Timing.

L'état système actuel (nombre d'extrusions) modifie certains paramètres de ces challenges mais d'autres peuvent être modifiés lors du Game design et level design.

Les paramètres principaux que nous pouvons modifier relèvent de la géométrie du LD, de la taille des extrusions, des force et vitesse du personnage, des cooldowns, de la durée maximale de certaines actions comme le wallrun ou l'escalade ou encore de la fenêtre d'opportunité comme par exemple pour le rebond.

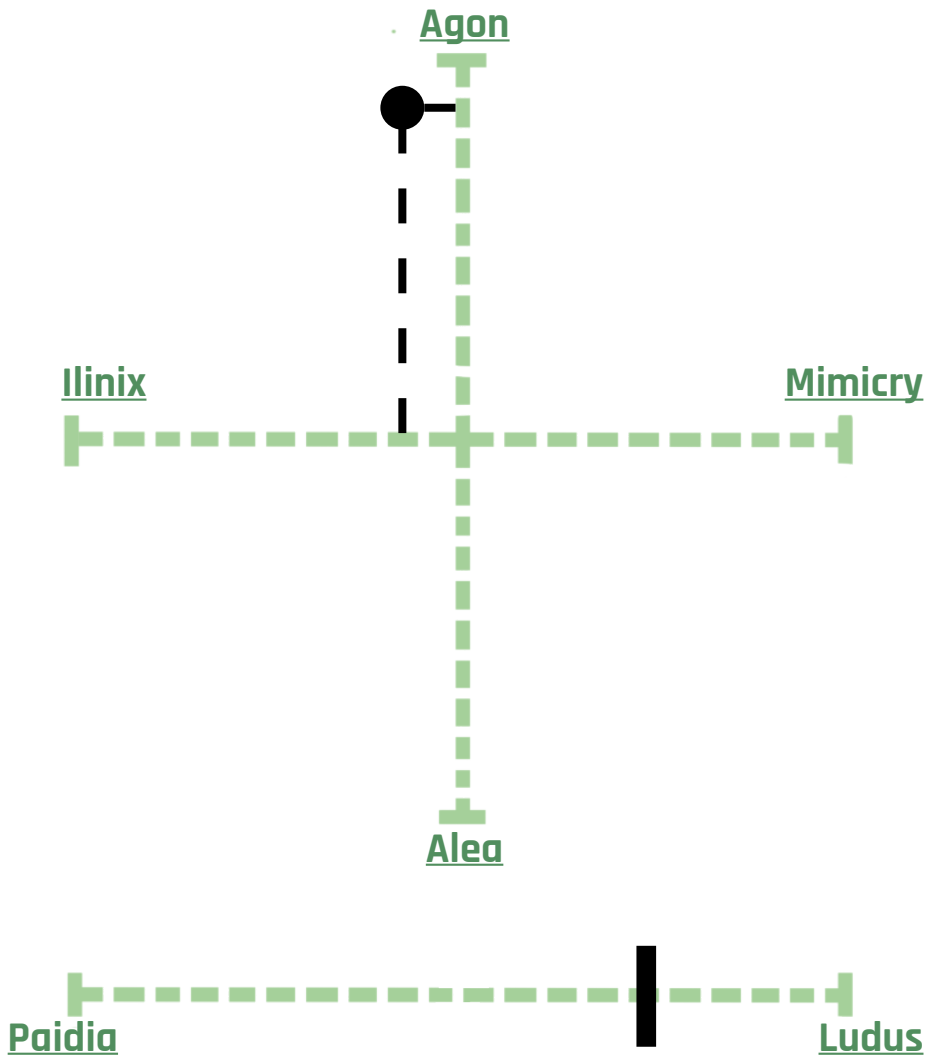
Objectif	Challenge	parametres atomiques	Reward
atterrir quelque part	prédiction, précision	taille de la surface, vitesse du personnage en l'air,contrôle du personnage en l'air	le joueur réussit l'objectif qu'il s'est fixé, il acquiere de la satisfaction personnelle et peut continuer. De plus il perds moins de verticalité et donc en Flow.
Atteindre un objectif spatial	prédiction(utilisation de la bonne mécaique, extrusion si besoin,prédiction distances,Level design, état du LD (Extrusions)	distance de l'objectif, taille de l' espace négatif , distance verticale, nombre et placement des surfaces extrudables, extrusions déjà présentes	le joueur réussit l'objectif qu'il s'est fixé, il acquiere de la satisfaction personnelle et peut continuer
rebond	timing, prédiction (atterrir sur l'extrusion)	Taille de l'extrusion, Fenetre d'opportunité	Le joueur gagne en flow et en verticalité
maintenir son flow	Tilt leger a droite,FOV+	level design, taille des espaces négatifs, nombre d'extrusions et placement des extrusions, nombre et placement des pentes	Feedback positif, plus de gameflow
ne pas tomber	observation,précision (adresse)	taille de la surface sur lequel le personnage marche, ground drag, vitesse du personnage	Le joeur en chute pas, il ne perd donc pas son flow
Parallel Boost	prédiction	nombre et rapprochement des extrusions	Gameflow amplifié (sensation de boost), gain en flow
Placer une extrusion	observation,précision	Distance du mur, nombre de murs extrudables	Le joueur gagne en mobilité et en flow grace a l'extrusion, le système évolue

18. Analyse

Selon le modèle de Roger Caillois, Bios ex Machina est un toy axé sur l'Agon, c'est à dire la compétition : le joueur est en permanence en compétition avec lui-même pour mieux performer ; c'est un jeu d' Explorers/Achievers dans le modèle de Bartle.

Pour ce qui est de l'aléa, il n y en a pas : tout est fixé et clair du côté du système. Il n'y a également pas de Mimicry, le toy ne simule rien de réel.
En revanche, Bios ex Machina s'inscrit légèrement dans l'Ilinix : lorsque la saturation de la carte est au maximum, le joueur se sent déstabilisé car bloqué et en perte de contrôle.

Bios ex Machina S'approche plus du Ludus que du Paidia, car son système et ses règles sont fixes. Cependant le Paidia reste présent : le joueur a la liberté de son style de jeu et surtout de comment il va moduler la carte, modulant au passage son expérience de jeu complète.



19. Eléments de jeu

Sol Extrudable

Le sol extrudable en béton est un élément de level design déjà placé sur lequel le joueur peut marcher, mais également extruder.

Sol non Extrudable

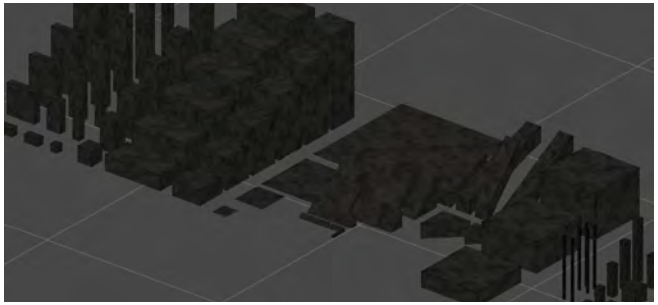
Le sol non extrudable, une surface métallisée, est un élément de level design déjà placé sur lequel le joueur peut marcher mais pas extruder.

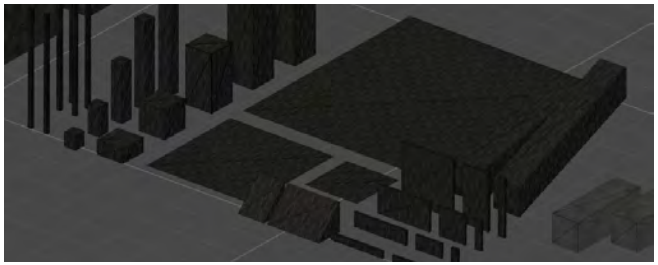
Surface lumineuse

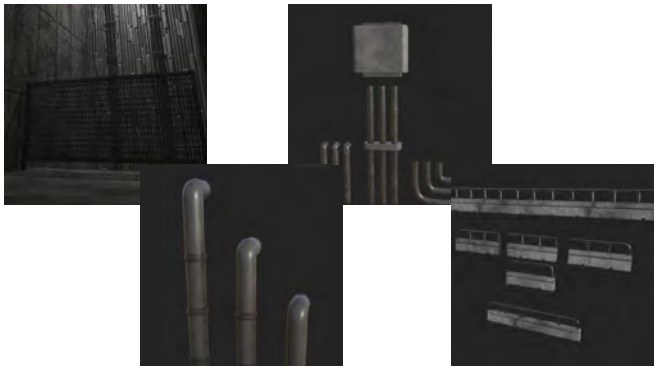
La surface lumineuse est utilisée dans le toy seulement au plafond et pour les lumières. Le joueur ne peut pas l'extruder et peut rentrer en collision avec cette dernière ; elle est blanche et composée d'un matériau émissif qui, couplée a l'effet de Bloom, donne un effet de lumière.

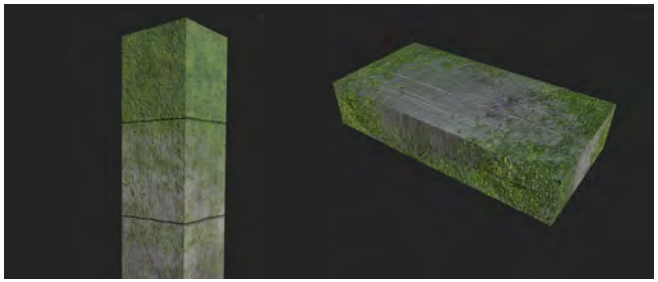
Obstacles

Les obstacles comme les barrières ou les tuyaux sont des éléments fonctionnant comme le sol non extrudable. Ils habillent le niveau et créent des obstacles pour le joueur qui ne sont pas forcément des grandes surfaces.









Pentes

Les pentes sont des formes d'éléments plutôt que des éléments à part. Dans le Level design, elles ne sont que des surfaces extrudable, ainsi le joueur peut glisser dessus pour gagner du Flow, mais surtout les extruder pour créer plus de pentes et ainsi gagner plus de Flow.

Extrusions

Les extrusions sont créées par le joueur. Quand elles apparaissent, elles grandissent à une vitesse régulière jusqu'à ce qu'elles aient atteint leur taille finale ou que leur pousse soit bloquée par un élément. Si c'est le cas, elles s'arrêtent à 1 mètre de cet élément, assez pour que le joueur puisse se faufiler ou glisser dans cette petite ouverture, et également assez pour ne pas pousser le joueur hors de la carte s'il est pris en étau par une extrusion en poussée.

Les extrusions, comme vu précédemment, permettent au joueur d'étendre son panel de mouvements et de gagner en Flow.

20. Level design

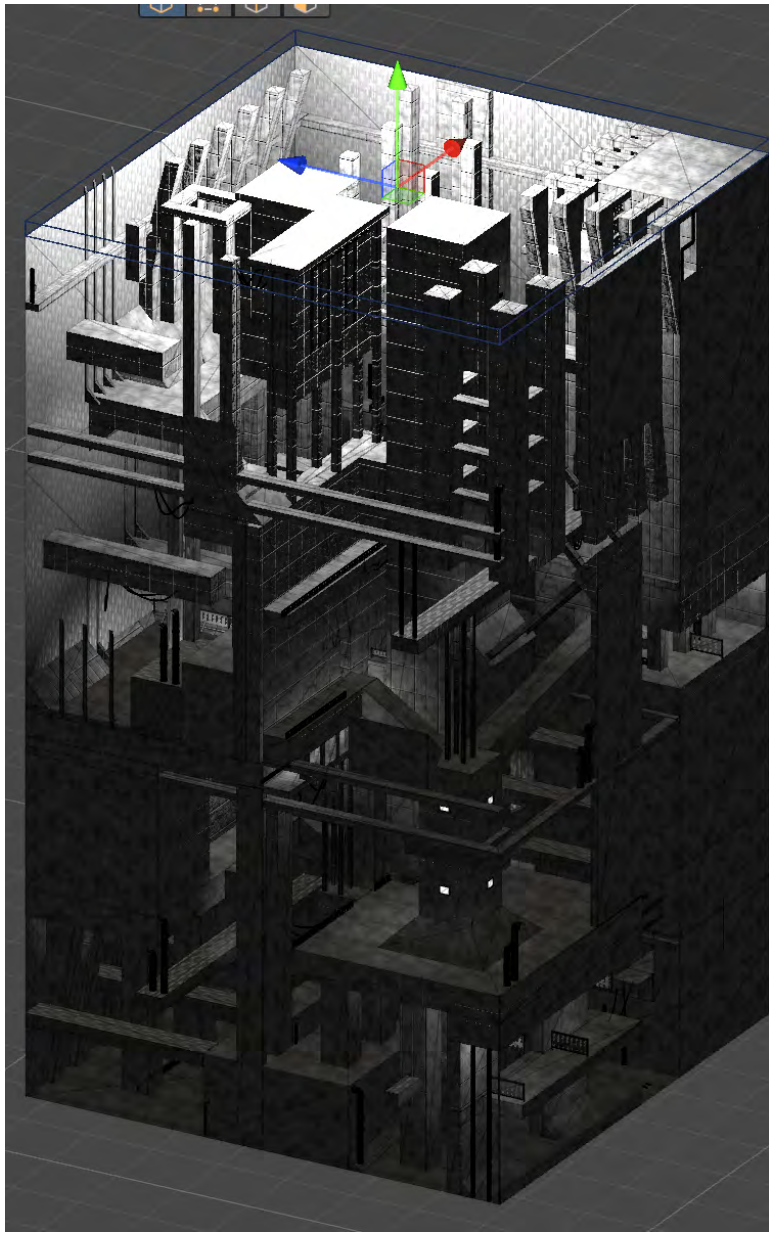
Le level design de notre toy a été pensé comme une grande tour verticale, qui permettrait d'exploiter pleinement le character controller ainsi que les différentes mécaniques.
La tour est divisée en plusieurs étages ou paliers, chacun introduisant une difficulté croissante dans l'escalade. Ces étages sont structurés pour exiger une utilisation millimétrée des mécaniques de jeu. La mécanique principale d'extrusion permet au joueur de créer des points d'appuis qui facilitent à la fois l'ascension et les phases de retraversée après une potentielle chute.

Le joueur apparait tout d'abord dans un couloir qui l'oblige à utiliser les mécaniques principales et les premiers étages offrent une grande liberté d'exploration permettant de les exploiter. Mais au fur et à mesure de l'ascension, le nombre d'éléments exploitables pour grimper diminue, obligeant à une planification plus rigoureuse. Des murs non extrudables apparaissent également pour restreindre les mouvements et obliger le joueur à repenser son positionnement.

Un élément crucial de notre toy est la réinterprétation constante des espaces déjà explorés. La retraversée est au cœur de l'expérience de jeu, quand le joueur tombe, les modifications qu'il a apporté à l'environnement grâce aux extrusions deviennent une part même du level design, qui permet au joueur d'avoir une nouvelle vision du terrain de jeu. Ces transformations simplifient les passages où l'on aurait échoué précédemment, la tour devient alors un espace vivant, où chaque tentative permet d'enrichir ou totalement modifier la suivante.

Les niveaux sont construits sur des grilles ou tous les elements principaux de blocking sont perpendiculaires ou paraleles les uns aux autres, ainsi les extrusions le seront également et quand le joueur aura maximisé son espace de jeu, il sera propice a la mécanique de parallel boost.

Le niveau "Flat" est un niveau alternatif entièrement plat et permet de maximiser l'impact de la mécanique d'extrusion, le système reposant entièrement sur elle, le level design se construit alors exclusivement autour des actions du joueur, enlevant toute contrainte imposée par un environnement préconstruit et laissant la place au joueur de s'exprimer librement.



Direction Artistique visuelle

1. Intentions et références

1.1 Recherche de la thématique

Pour trouver la thématique de notre toy, nous sommes partis du système et de ses évolutions au cours de la partie.
Le système à son état initial est immobile, inerte et le level design est grand, assez carré et inaccessible, peu propice au déplacement.
Ainsi lors du début de la partie nous avons un système froid, inerte et immobile, aux angles marqués.

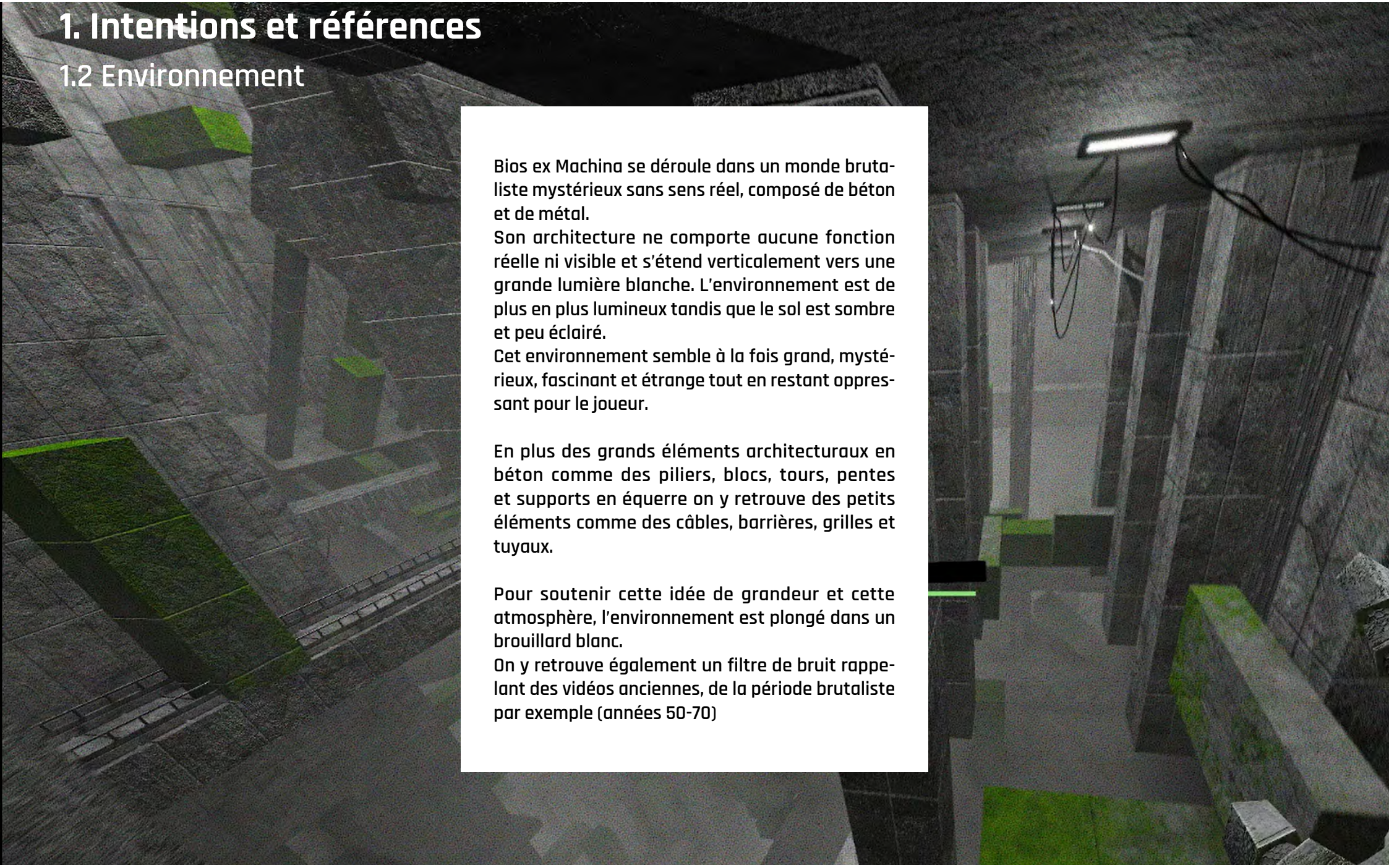
Nous avons donc associé à ce système et à cet environnement le brutalisme : froid, grand, inerte, carré, massif.

Puis viennent les extrusions. Elles apportent du mouvement, du Flow en créant un écosystème propice au déplacement. Elles « poussent » des surfaces et créent un environnement désordonné où les piliers et panneaux vont « dans tous les sens », rendant le terrain de départ moins droit et carré.
Ces extrusions doivent également contraster visuellement avec le terrain, tout en étant du matériau du terrain extrudé pour que le système reste logique.

Ces notions de mouvement et d'écosystème nous ont mené vers la végétation, nous avons donc choisi de créer un environnement brutaliste, bétonné duquel on extrude des piliers et panneaux bétonnés mais recouverts de mousse verte.
Ainsi le joueur au fil de sa partie crée une sorte de jardin de panneaux et de piliers végétalisés qui viennent colorer la carte en la désordonnant au passage, cassant l'aspect droit et ordonné du niveau pour y apporter de la vie, de l'organique, et un écosystème dans son fonctionnement comme dans son évolution et sa forme.

1. Intentions et références

1.2 Environnement



Bios ex Machina se déroule dans un monde brutaliste mystérieux sans sens réel, composé de béton et de métal.
Son architecture ne comporte aucune fonction réelle ni visible et s'étend verticalement vers une grande lumière blanche. L'environnement est de plus en plus lumineux tandis que le sol est sombre et peu éclairé.
Cet environnement semble à la fois grand, mystérieux, fascinant et étrange tout en restant oppressant pour le joueur.

En plus des grands éléments architecturaux en béton comme des piliers, blocs, tours, pentes et supports en équerre on y retrouve des petits éléments comme des câbles, barrières, grilles et tuyaux.

Pour soutenir cette idée de grandeur et cette atmosphère, l'environnement est plongé dans un brouillard blanc.
On y retrouve également un filtre de bruit rappelant des vidéos anciennes, de la période brutaliste par exemple (années 50-70)

1. Intentions et références

1.3 Références architecturales

Logique

Afin de garder l'aspect sans but de l'architecture, nous avons fait attention à ce que les différentes parties du level design ne se connectent jamais logiquement. Il n'y a pas d'escaliers, les tuyaux ne mènent à rien, on ne trouve ni portes ni fenêtres, et aucun aménagement urbain logique.
Nous avons donc pu nous concentrer sur l'architecture et le level design sans trop se soucier de la logique diététique.

Inspirations architecturales

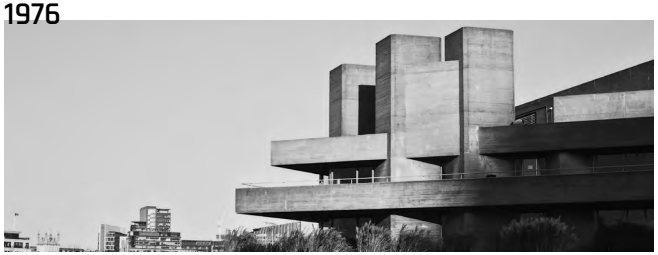
Au niveau de l'architecture en elle-même, nos inspirations principales étaient :

-HDV Boston 1963 Kellan McKinnel (la partie basse)
-Halo forerunner architecture
-Blame!
-Brutalist Quake Game Jam
-UCSD Library San Diego 1970 William Pereira
-Mc esher stairs painting
-Rotyal National Theatre London 1976 Denys Lasdun
-Films de Denis Villeneuve (Dune Arrakeen city, Arrival, Blade Runner 2049)

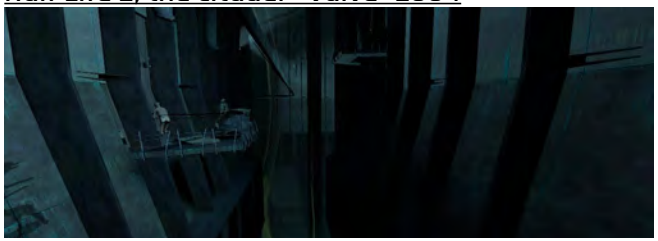
Blade Runner 2049 - Denis Villeneuve -2017

UCSD Library,San Diego - William Pereirrra - 1970

Royal National Theatre,London - Denys Lasdun - 1976



Half Life 2, the citadel - Valve -2004



Relativity - M.C. Escher - 1953

Boston city Hall - Kallan McKinnel - 1963

Dune, Arrakeen - Denis Villeneuve - 2021

Halo CE forerunner architecture - Bungie - 2002

1. Intentions et références

1.4 Références matériaux

Nos références de couleurs se sont basées sur des échelles de gris.

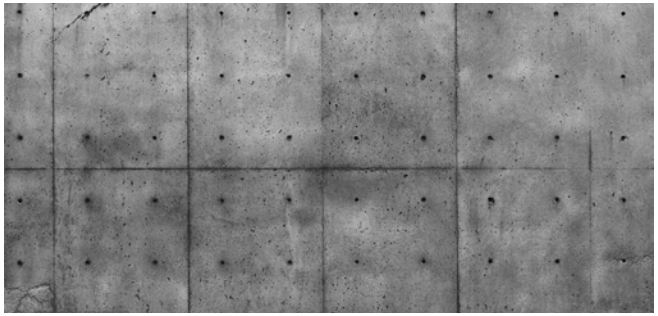
Nous avons choisi d'avoir une palette de noir, blanc gris avec comme seule couleur le vert. Pour cela nous nous sommes inspirés d'œuvres comme Sin city ou encore Mad World.

Pour ce qui est des matériaux, nos inspirations principales étaient naturellement les matériaux utilisés dans le brutalisme comme le béton, différentes formes géométriques métalliques pour nos surfaces non extrudables et pour finir nous nous sommes intéressés à la manière dont la mousse se forme sur la roche, le béton et autres matériaux urbains.

Half life 2 - Valve - 2004 (metal, geometry)

Mad world - SEGA - 2010

Panneaux/blocs de béton



Royal National Theatre,London - Denys Lasdun - 1976 (panneaux/blocs de béton)

Sin city - Franck Miller - 2005

Mousse poussant sur du bitume

Mousse poussant sur du béton

1. Intentions et références

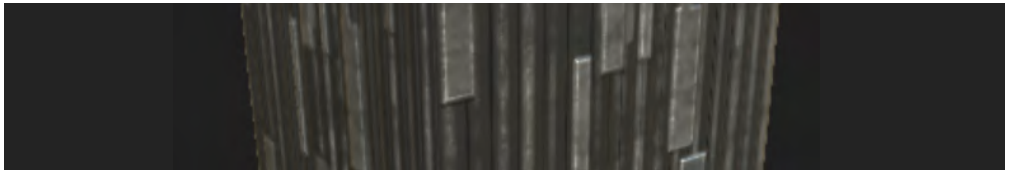
1.5 Couleurs

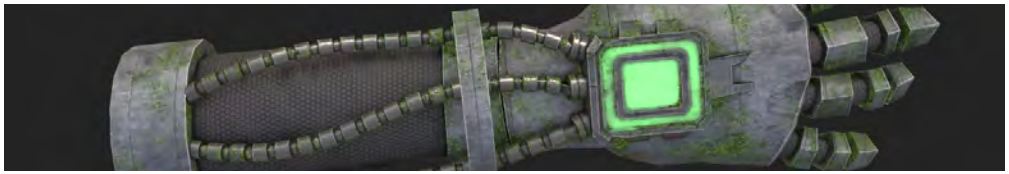
Couleurs

Pour appuyer les contrastes et l'aspect inerte du système initial, nous avons choisi que tout le jeu serait en grayscale, sans couleur, sauf le vert du HUD, du personnage et de la végétation des extrusions.
Les couleurs sont donc basées sur les contrastes.

Répartition des couleurs

Surfaces extrudables	gris clair mat
Surfaces non extrudables	gris foncé métallique
Props	gris foncé + détails clair
Lumières	blanc émissif
Extrusion	gris clair mat + Vert clair
Character	gris foncé + clair + vert





1. Intentions et références

1.6 Références d'ambiance et de lumières

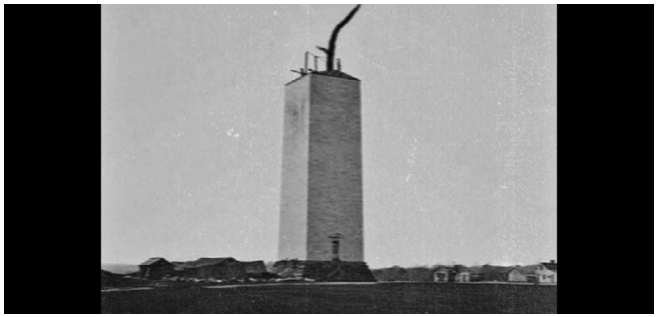
Nos références principales pour l'ambiance et la lumiçre apportées à l'environnement sont principalement liées à la grandeur (Brutalist Quake entry). Cette grandeur est amplifiée par un brouillard blanc mystérieux (The mist, Silent hill) et un ciel très blanc et lumineux (Brutalist Quake entry, Gieidi Prime).

Pour souligner cette ambiance sombre, et rappeler l'époque du brutalisme nous avons pris comme référence le grain des vidéos VHS comme par exemple dans Monument Mythos ou d'autres séries du genre «analog horror».

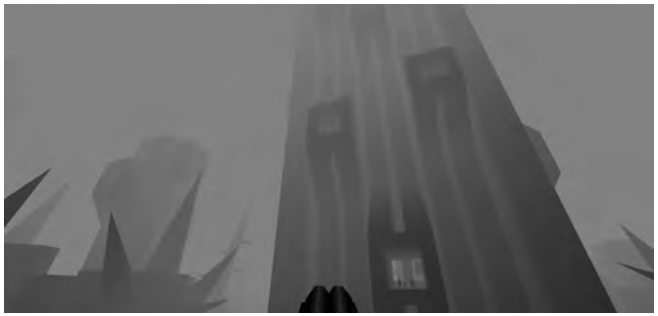
La lumiere venant d'en haut apporte une hiérarchisation de la carte : plus on monte plus la lumiere est forte. Ce concept est inspiré de Métropolis de Fritz Lang, où l'aisance des habitants dépend de leur hauteur dans les niveaux de la ville, et donc du niveau de lumière.

Metropolis - Fritz Lang - 1927

Monument Mythos - Alex Casanas - 2020



Brutalist quake jam entry - Fabio Peireirra - 2022



Silent Hill 2 - Konami - 2001

Brutalist quake jam entry - Fabio Peireirra - 2022

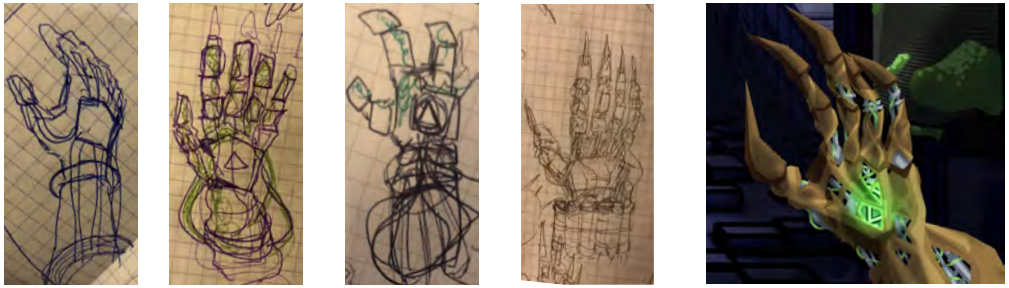
The mist - Stephen King - 1980

Gieidi Prime, Dune 2 - Denis Villeneuve - 2024

1. Intentions et références

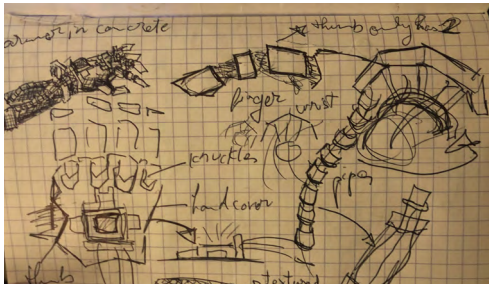
1.7 Character

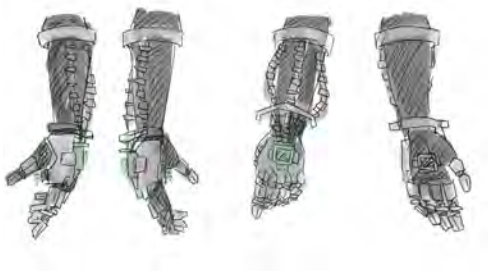
Notre personnage est composé de deux mains, dont l'une des deux est visible à l'écran en permanence.
Ce choix a été fait en prenant comme inspiration les armes de FPS : notre main fonctionne comme une de ces dernières, en permanence à la droite de l'écran, animée et bougeant selon les actions du joueur.
La main gauche ne sort que pour le wallrun gauche et l'extrusion lors du wallrun droit, ainsi que l'escalade.
Le choix d'avoir la main gauche visible seulement par moment a été fait pour ne pas encombrer la vision du joueur plus que nécessaire.



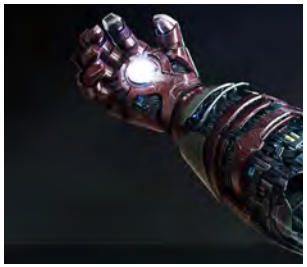
Divers concept arts pour le design des mains du personnage

Quant à nos intentions de design, nous voulions que le joueur comprenne par le design le rôle de cette main. Il comprend par exemple que le personnage peut effectuer une action à distance avec un mouvement de jeté, ici une extrusion. Peu importe l'action, le joueur doit comprendre par le design que cette main est faite pour une action à distance, et qu'elle appartient à un personnage agile.
De plus, nous voulions que son design fasse office de palier entre l'environnement et les extrusions, un pallier entre l'artificiel du béton et le naturel de la mousse.
Ainsi la main est humaine, naturelle, mais comporte une sorte de gant/armure de la couleur du béton et des machines y sont également présentes.
Ce gant de la couleur du béton est légèrement couvert d'une mousse verte qui rappelle les extrusions, et la machine émet une lueur verte, seule couleur du toy et couleur liée aux extrusions.





Nos références principales pour la machine sont les mains d'ironman. Notre référence principale pour la lueur de la machine est la main de Corvo Attano dans Dishonored, qui illustre également parfaitement les pouvoirs du personnage. Pour la forme du gant, nos deux références principales sont les mains de l'armure HEV du personange d'Half-life et certains gants tactiques .



Iron man 3 - John Favreau
2014

Dishonored - Bethesda
2012

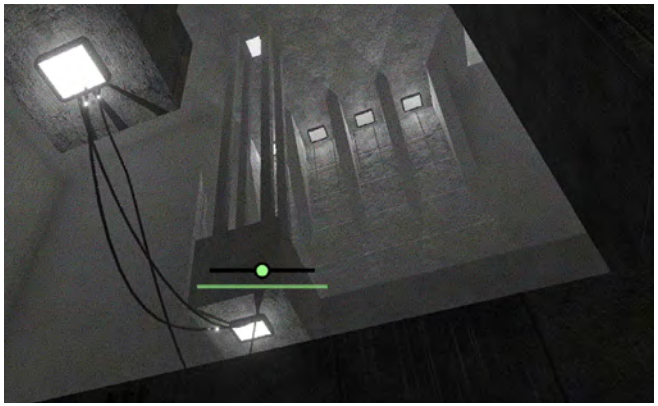
Half-life - Valve
1998

Gant tactique

2. Architecture

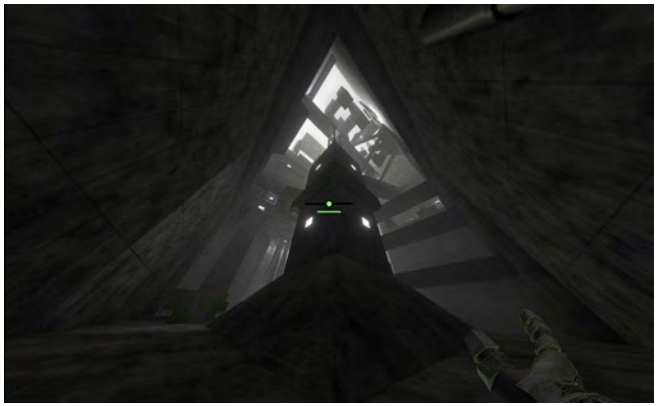
De ces références, nous avons donc extrait des axes majeurs que nous avons placé dans notre niveau en prenant en compte le level design.

Les piliers



Les supports

Les Supports style Pereira



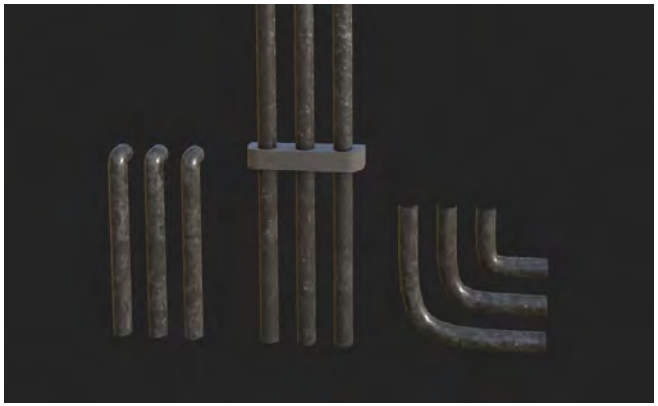
Les blocs

Les plateformes

Les balcons

3. Props

Nous avons également créé des props que nous avons placé à certains endroits du niveau. Comme énoncé précédemment, certaines de ces props servent d'obstacles, d'affordances, de repères ou encore de simples décorations.

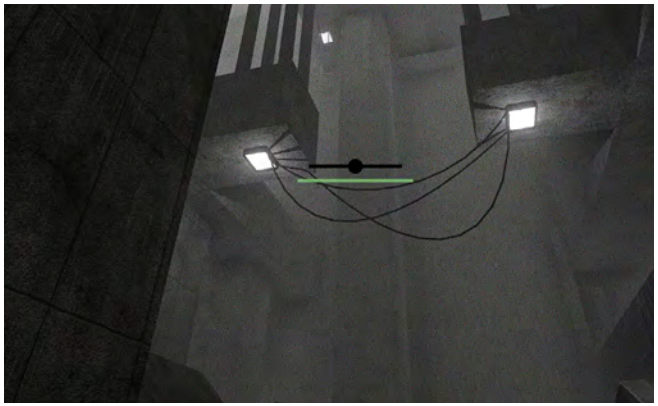


Câbles couverts :
Pas de hitbox, décoration, affordances

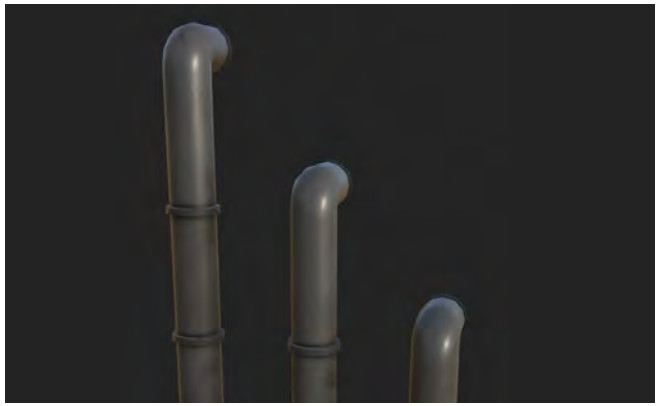
Boitier électriques :
Pas de hitbox, décoration

Barrières :
Level design

Grilles :
Level design, fonctionne bien avec les effets de lumière



Câbles pendants :
Décoration, repères

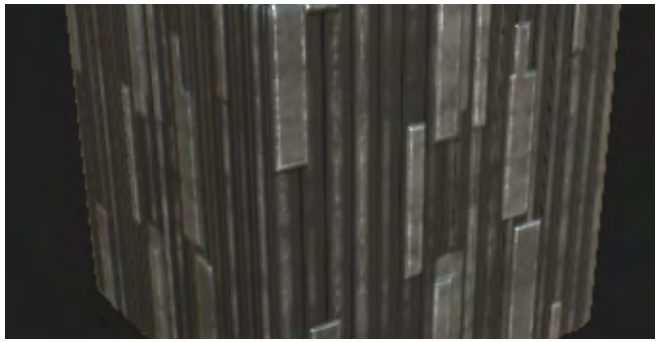


Tuyaux :
Level design, repères

4. Matériaux

Non extrudable

Pour les surfaces non extrudables, nous devions trouver un matériau noir et blanc contrastant tout de même avec le béton gris clair et faisant comprendre qu'on ne peut pas extruder ce dernier : le métal. Il contraste avec le reste de l'environnement :
- Par la luminosité de sa couleur en étant plus sombre,
- Par sa forme différente ainsi que sa réflectivité.
- Par son matériau ; dans l'esprit des joueurs, le béton est rugueux et mat tandis que le métal est lisse et brillant. Ils sont donc opposés tout en partageant une cohérence avec le thème brutaliste et la palette de couleur en nuances de gris.

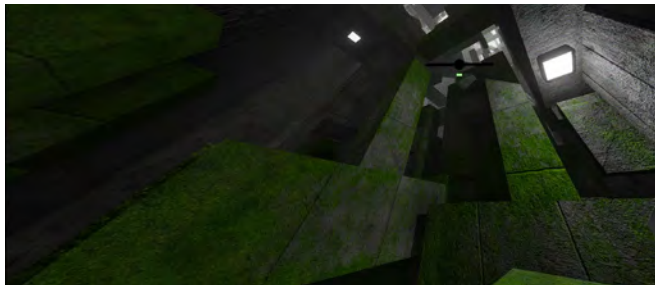


Extrudable

Les surfaces extrudables constituent la majorité de l'environnement et nous avons choisi pour ces dernières le béton. Ce matériau est le principal composant de l'architecture brutaliste qui compose notre environnement, il est donc logique que ce matériau soit celui choisi pour la surface principale de notre environnement.
C'est également un matériel qui se prête aux grandes surfaces planes, composante principale de notre level design et donc de notre environnement.

Extrusions

Les extrusions se font sur du béton, il est donc logique que ces dernières soient bétonnées. Mais comme dit précédemment ces surfaces sont également couvertes de mousse, cette mousse est de plus en plus présente au fil de la longueur de l'extrusion. La mousse est d'une couleur verte saturée contrastant grandement avec l'environnement.
De plus, la végétation appuie le concept d'extrusion et de pousse.



Autres éléments

Les éléments additionnels (tuyaux, grilles, barrières) partagent leurs couleurs et matériaux avec les surfaces non extrudables: du métal lisse,réflectif et sombre.
Ces éléments agissent comme les éléments non extrudables c'est pour cela que leurs matériaux sont si similaires.

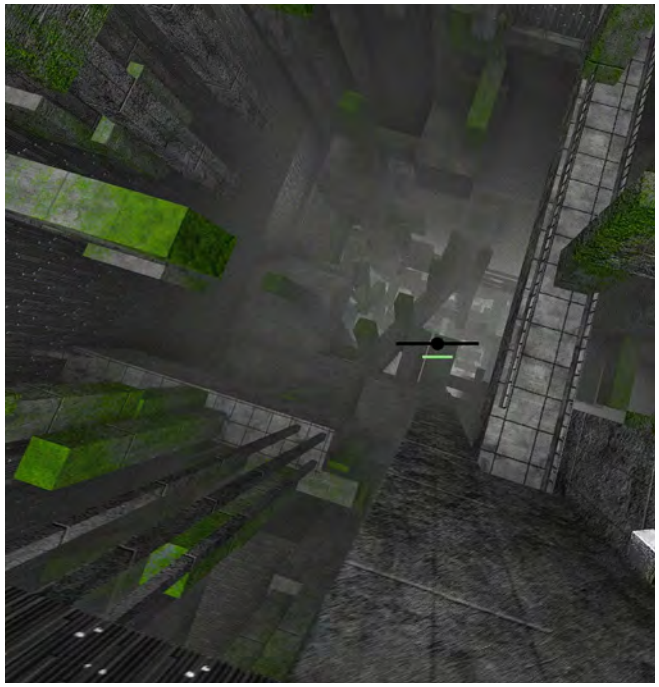
5. Ambiance

Contraste

Notre ambiance est basée sur les contrastes, de lumière comme de couleurs, sur l'oppression de l'environnement sur le joueur et sur la grandeur. Nous avons ajouté des effets de post process détaillés ici pour appuyer l'ambiance et les contrastes.

Fog, ambiant occlusion et profondeur de champ.

Pour donner une impression de lumière, de grandeur et apporter du mystère et une meilleur appréciation des profondeurs, nous avons mis en place un brouillard blanc.
Ce brouillard est couplé à un effet d'occlusion ambiante apportant de l'ombre aux angles concaves des éléments de la scène, ainsi le joueur peut percevoir de la géométrie dans le brouillard sans pour autant le voir clairement. De plus, cet effet accentue les creux et apporte de l'ombre, augmentant l'effet d'oppression désiré.
Finalement, nous avons mis un effet de profondeur de champ en floutant très légèrement la géométrie perçue au loin.



Grain

Nous avons ajouté un effet de grain à l'image, qui rappelle l'époque brutaliste entre les années 50 et 70, particulièrement dans l'Europe de l'est de l'époque soviétique. Cet effet rappelle les vieilles vidéos d'époque les VHS.

Color grading

Les verts ont été rehaussés afin de les faire ressortir davantage, et ainsi de faire ressortir les extrusions.

6. Lumière

L'objectif de la lumière était d'accompagner le level design vertical. La lumière est beaucoup plus élevée en haut du niveau qu'en bas, et incite le jouer à s'en approcher.

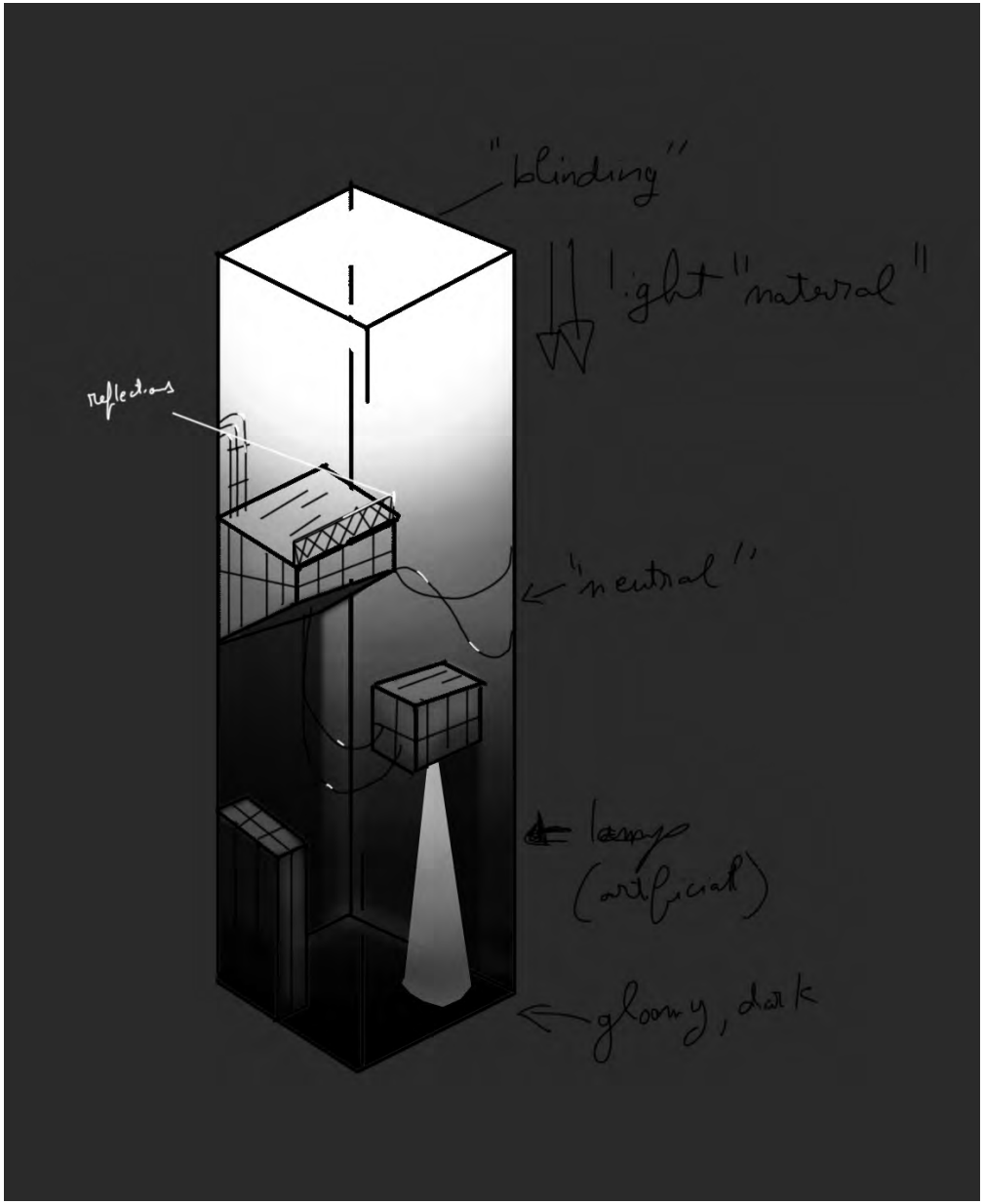
Ainsi, si le joueur tombe, les jeux de lumière lui donnent l'impression de retomber dans les profondeurs de la carte, tandis que quand il monte, il a l'impression de monter vers la lumière. Ce haut de la carte lumineux est également porté par un plafond en matériel émissif blanc, couplé à un effet de bloom donnant l'impression que ce dernier brille.

En bas de la carte il y a donc très peu de cette lumière générale.
Pour que le joueur s'y retrouve quand même, nous avons placé des lampes éclairant artificiellement certaines parties du niveau ; plus on monte moins il y en a et ces lumières sont remplacées par la lumière naturelle.

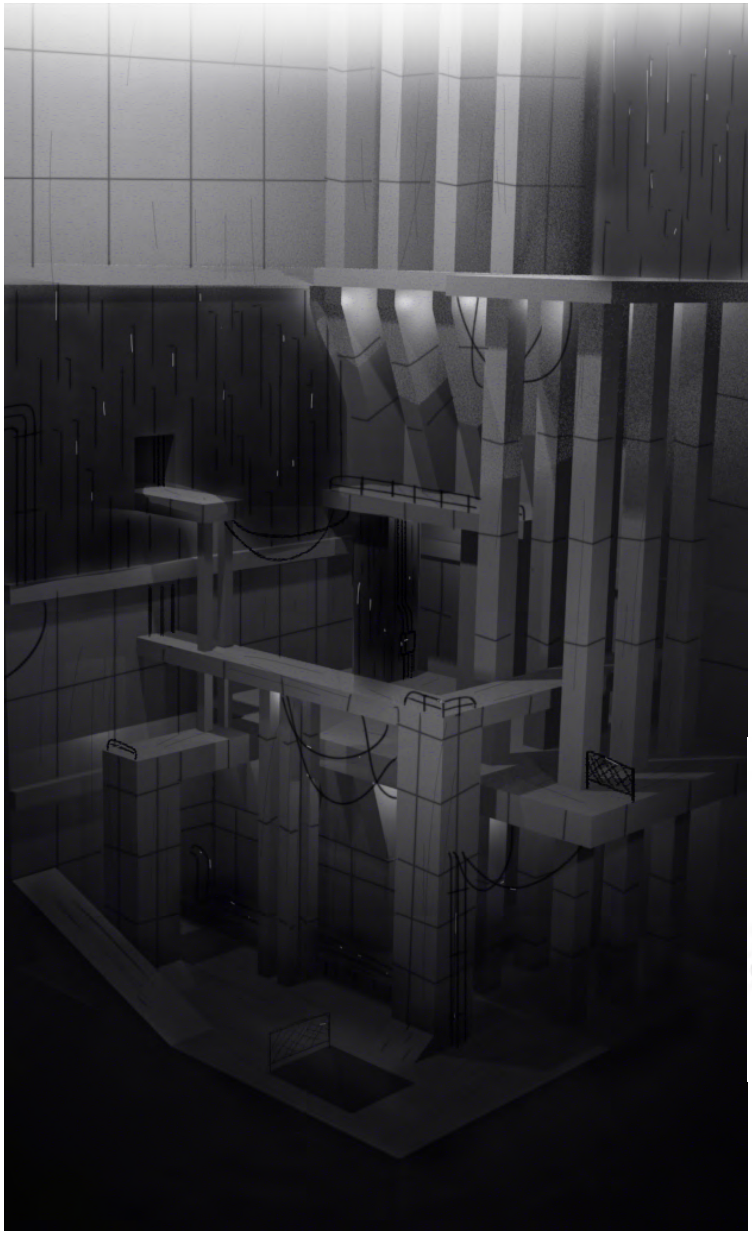
Les lumières apportent également en affordances, éclairant certaines parties stratégiques du level design.

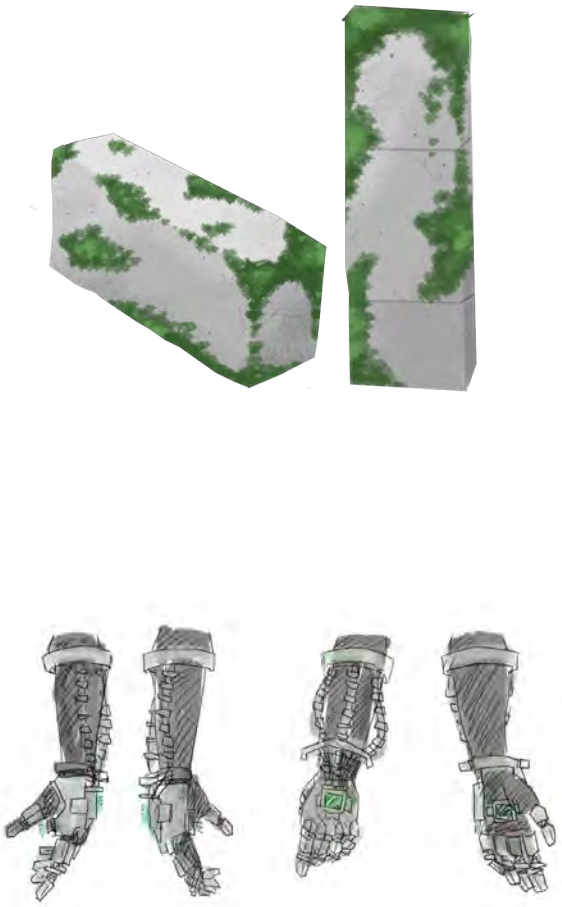
Les lumières en temps réel interagissent avec des éléments de la carte, comme les grilles en passant à travers, et réagissent également aux extrusions. Quand un pilier est extrudé, si celui-ci est proche d'une lumière on peut voir en temps réel les ombres changer, ajoutant donc de l'impact autre que la géométrie et le déplacement à la carte : les extrusions changent également la lumière et apportent du contraste.

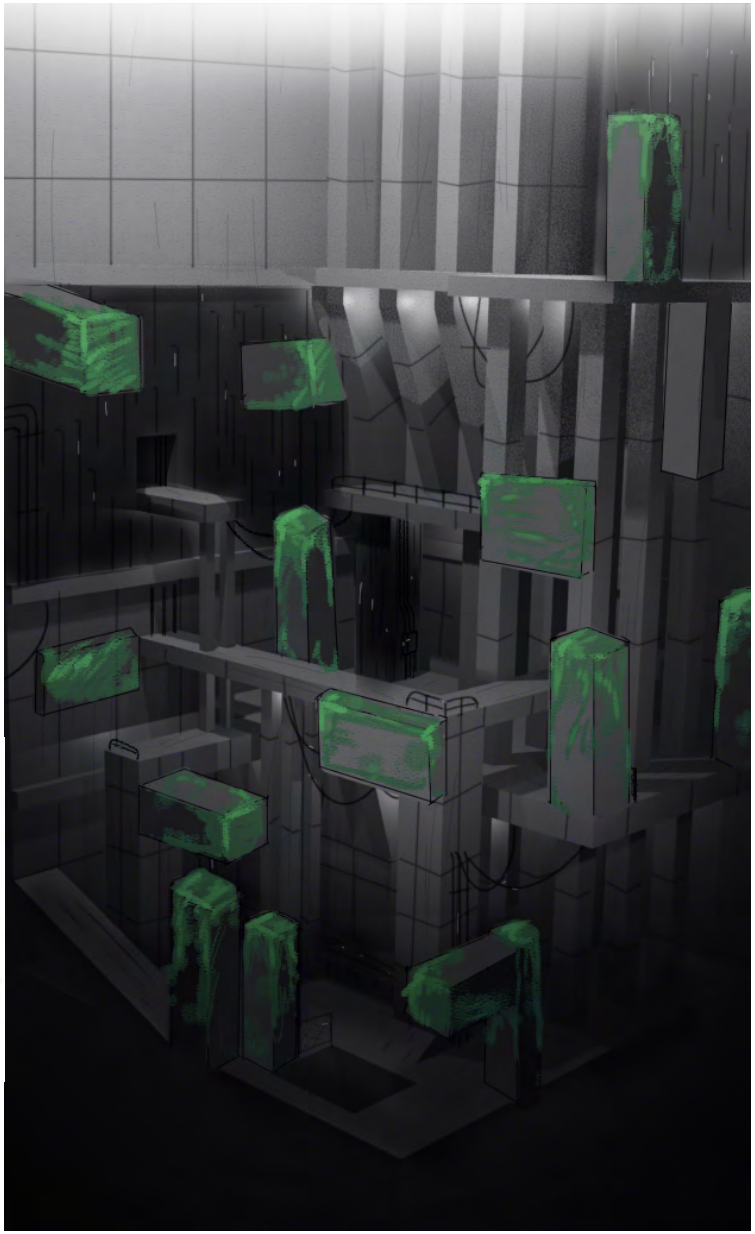
La partie supérieure de la carte surexposée fait ressortir les normal map du béton, le rendant plus rugueux, le joueur remarque donc qu'en montant, la lumiere et les textures changent.
En haut de la carte, le métal reflète aussi beaucoup la lumière, donnant parfois un effet d'éblouissement faisant ressortir les contrastes.



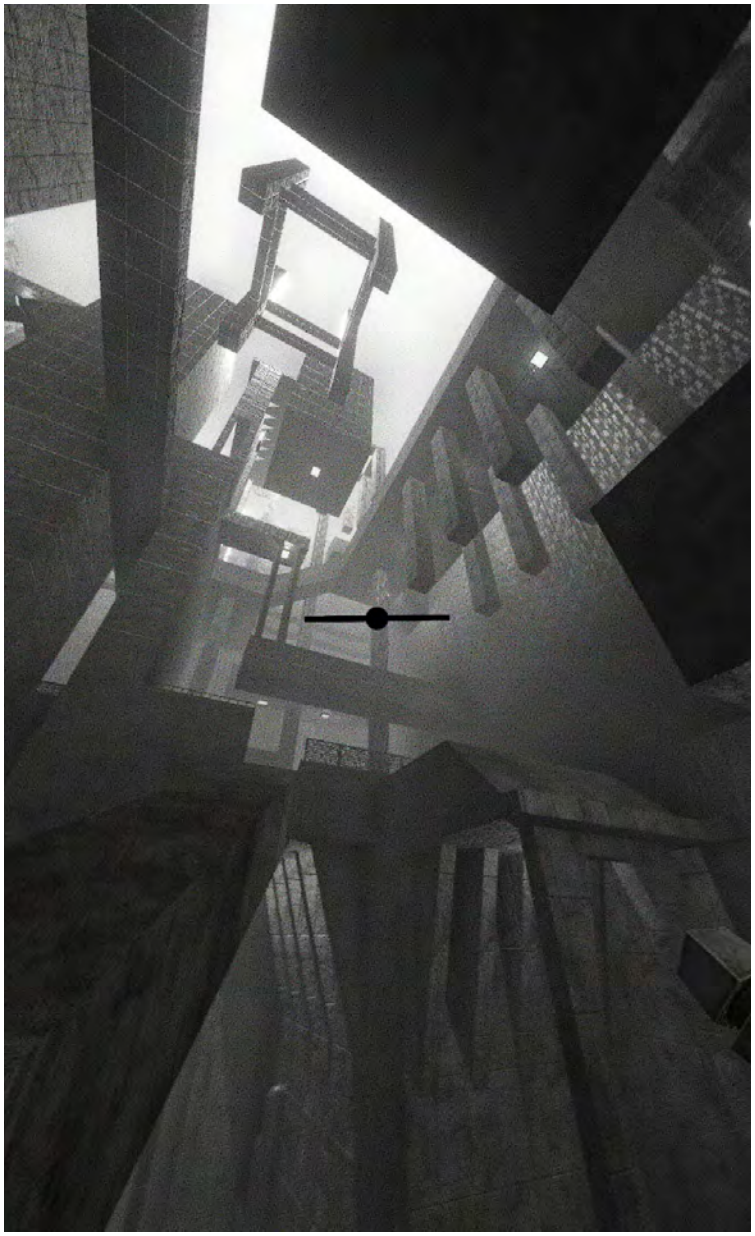
8. Concept arts



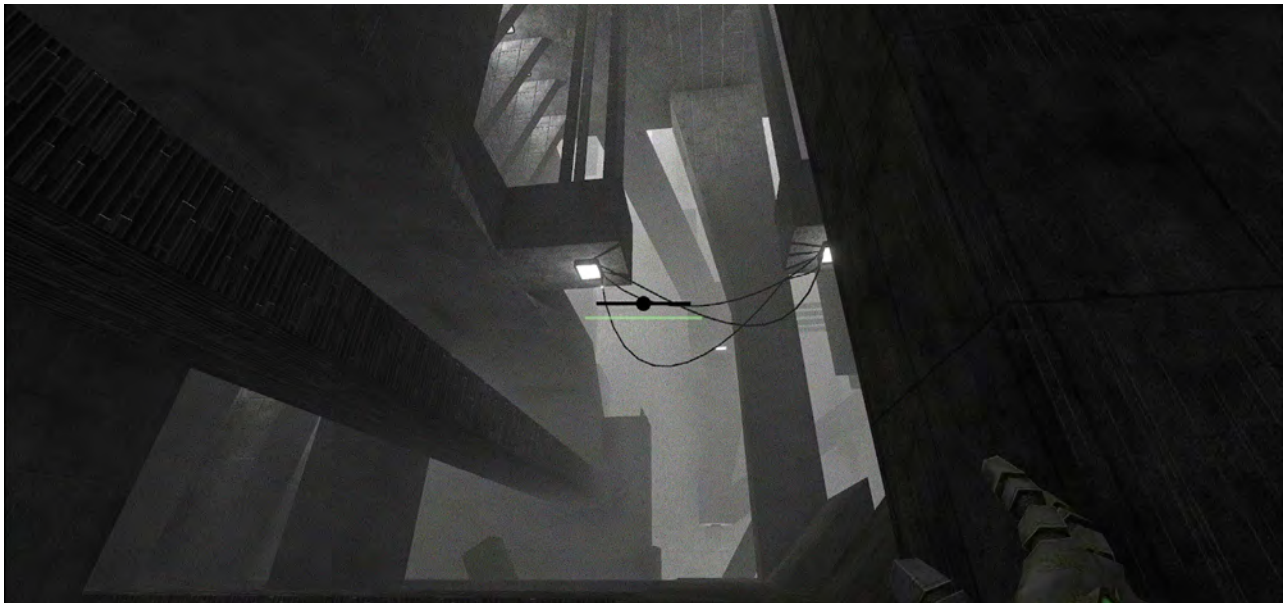


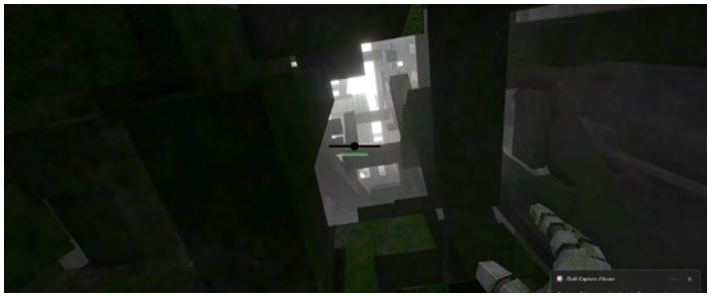


9. Galerie



9. Galerie





Direction artistique sonore

1. Musique

1.1 Intentions

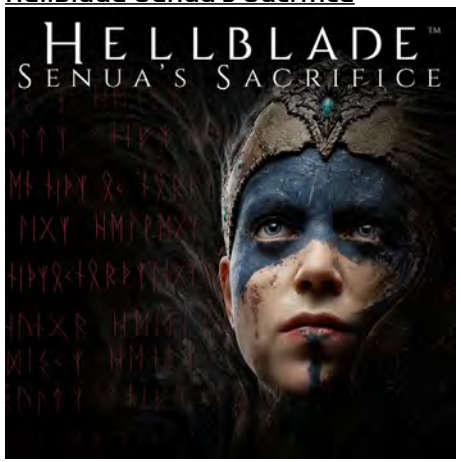
Nous avons envisagé la musique du toy d'après deux critères : son rythme, et son ambiance.

Le rythme devait être rapide, avec un tempo proche de 120 BPM. Les mouvements du jeu sont souvent saccadés, et la musique devait refléter la vitesse du personnage. Pour ceci, nous nous sommes approchés de la musique électro, qui comporte souvent des rythmes marqués grâce à des percussions très présentes.

Pour l'ambiance, nous avons choisi d'approcher un style «folk», avec des instruments traditionnels évoquant à la fois la nature et le mysticismem dans un ton général sombre et inquiétant. Deux éléments nous sont venus rapidement à l'esprit : des voix rauques, semblables au chant guttural mongol, pour souligner l'étrangeté de l'environnement, et la flûte de pan, qui accompagnerait parfaitement la sensation de vitesse du personnage. Nos principales inspirations pour l'ambiance générale ont été la bande-originale des films Dune de Denis Villeneuve, composée par Hans Zimmer, et celle des jeux Hellblade.

Après quelques recherches, nous avons donc décidé de nous approcher du «Folktronica», un genre musical proche de l'électro et utilisant une grande part d'instruments traditionnels dits «ethniques» : flûtes, percussions, et techniques vocales diverses. Nos inspirations de folktronica étaient « Xanadu » de Ummet Oscan, « Hold on » de Belle Sisoski et « Solovey » de Go_A. Nous avons également trouvé de l'inspiration des musiques du groupe de rock mongol The Hu.

Hellblade Senua's Sacrifice



Hold on - Belle Sisoski

Xanadu - Ummet Ozcan

Solovey - So_A

Dune - Hans Zimmer

The HU

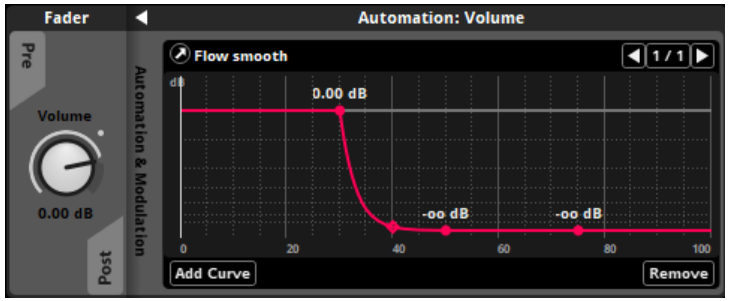
1. Musique

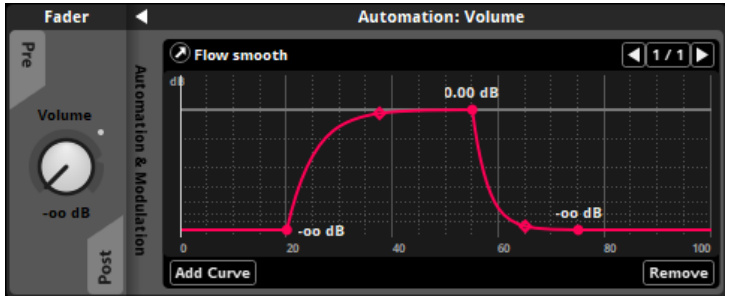
1.2 FMod

La musique est la composante principale de l'environnement sonore de Bios ex Machina. Elle constitue un feedback important pour le joueur car, elle s'adapte au niveau de flow qu'il atteint pendant la partie.

Composée par Nathan Miniere, la musique est en réalité constituée de trois pistes superposées qui sont jouées simultanément. Lorsqu'un changement de niveau de flow a lieu, le volume de la piste correspondante est augmenté, et celui des autres pistes est diminué jusqu'à être inaudible. Les trois pistes bouclent au même moment pour maintenir une superposition parfaite.

Pour permettre des transitions fluides, les trois pistes ont été composées sur la même base : elles ont le même tempo et les mêmes instruments de départ. A chaque niveau de flow, quelques instruments sont ajoutés, qui augmentent la pression du joueur et son sentiment de vitesse.





2. UI

L'UI dBios ex Machina a été pensée pour être minimaliste, et laisser la plus grande place à l'expérience de jeu. Les sons liés à l'UI suivent ce principe, et restent sobres et simples.

Il existe trois sons liés à l'UI, qui sont tous intégrés dans les menus du toy :

Switch, Select, StartGame

Ces deux sons, joués comme des oneshots dans FMod, sont utilisés lorsque le joueur navigue dans les menus :

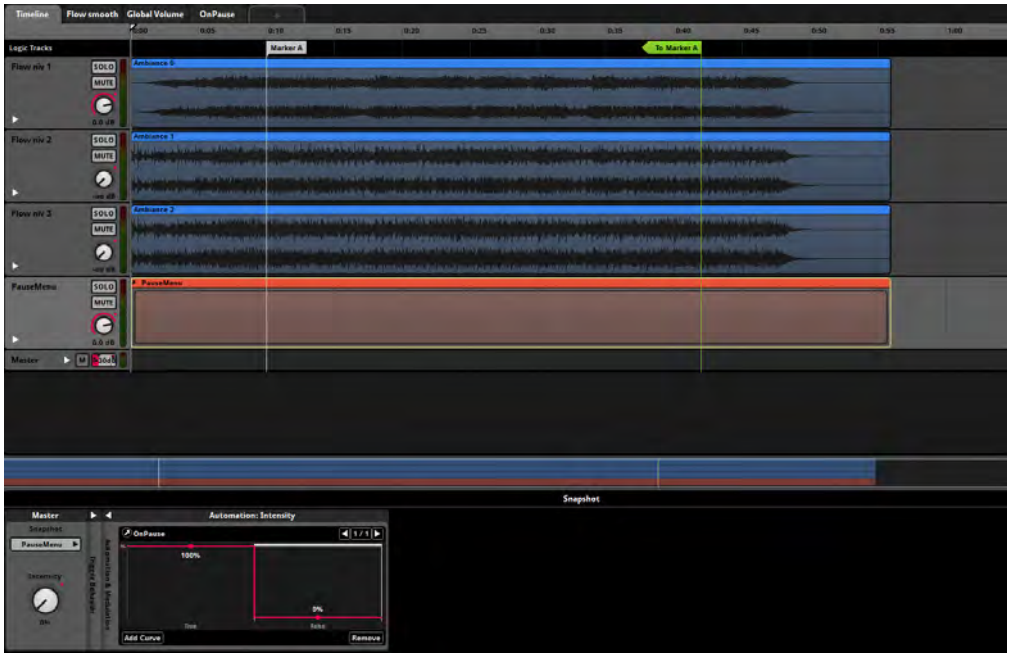
- Switch se joue quand le joueur change de ligne dans un menu ou déplace un curseur sur une barre d'option.
    - Il s'agit d'un simple clic assez aigu, relativement discret auquel nous avons ajouté un léger effet de réverbération.

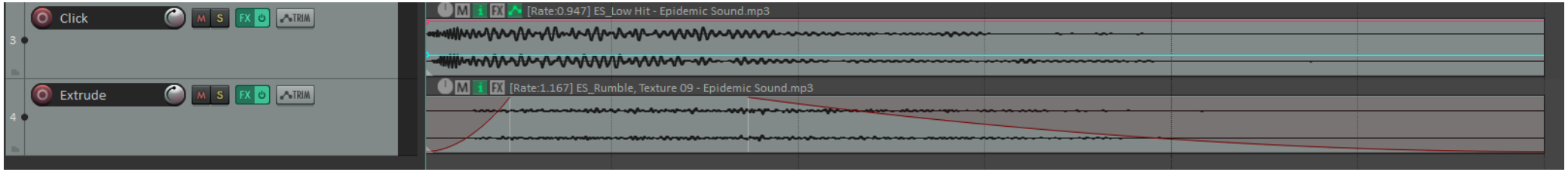
- Select est donc le son de sélection dans les menus.
    - Il est fait du même son que Switch ; il a été pitched down et la réverbération est légèrement augmentée.

- Startgame est une variation de Select, qui n'est lancé qu'au démarrage d'une nouvelle partie.
    - Il reprend le Select, auquel est ajouté le son d'une extrusion modifiée (voir 4. Pilar sounds) avec une forte réverbération et un EQ qui accentue les sons graves.

Musique en menu pause

Un effet d'étouffement a été appliqué à la musique du jeu lorsque le joueur entre dans le menu pause. Cet effet a été réalisé dans FMod grâce à un snapshot «PauseMenu» dont l'intensité est réglée par un paramètre booléen «OnPause».





3. Player sounds

La majorité des sons préparés pour le toy sont des feedbacks pour les actions de déplacement du personnage. Ces sons sont présentés ici : ils ont tous été modifiés dans Reaper puis paramétrés dans FMod.

Footsteps

Les bruits de pas sont intégrés dans un Scatterer instrument de FMod : ils contiennent 25 variations jouées aléatoirement à un rythme qui suit le headbobbing de la caméra. Les sons ont été pitched down pour s'aligner sur l'ambiance mystérieuse et obscure du toy, et une légère réverbération a été ajoutée sur chacune des variations pour accentuer l'effet de grandeur.

Jump/Land

Les sons de saut et d'atterrissage sont des variations des bruits de pas classiques, joués légèrement plus fort pour donner l'impression d'un pas plus puissant.

PilarJump

Le PilarJump est une variation du son « Jump » classique. Pour accentuer la puissance du saut offerte par le pilier en mouvement, un « whoosh » est ajouté au son de pas, sous la forme d'un multi-instrument à quatre variations. Le whoosh est également modifié avec un léger effet de réverbération et un compresseur qui donne l'impression d'une impulsion en deux temps : la première lors du « décollage » du personnage, et la deuxième alors qu'il entre en l'air. Ce décalage appuie sur la puissance du saut.

Slide

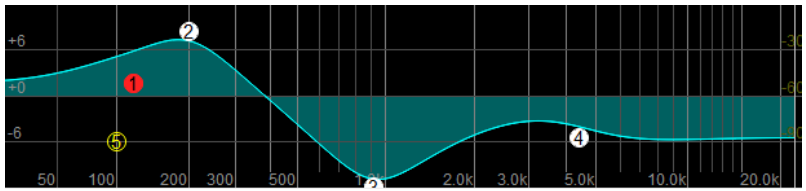
Le Slide est accompagné d'un « swoosh » léger et continu, doublé d'un léger son de frottement sur une surface rocheuse pour donner de la texture au son et illustrer celle du sol sur lequel glisse le personnage.

Wallrun

Le son de wallrun est composé de deux pistes : l'une est un son de frottement, qui illustre le contact de la main sur le mur ou le pilier ; l'autre est un scatterer instrument qui joue des bruits de pas similaires aux « Footsteps » classiques, légèrement pitched up, et dont la cadense est légèrement accélérée. Ces modifications rendent l'effet de pas plus légers et rapides, pour correspondre à des mouvements naturels pour l'équilibre précaire du personnage.

In Air

Le son «In Air» correspond à un bruit joué en boucle lorsque le joueur passe du temps en l'air, généralement en tombant. Le son est un bruit de vent, accéléré pour donner une sensation de vitesse, et auquel nous avons supprimé les fréquences moyennes (autour de 1000 Htz), tout en augmentant légèrement les fréquences basses (200 Htz) et en diminuant légèrement les fréquences hautes (plus de 5000 Htz). Ainsi, le son garde quelques unes des sonorités sifflantes du son original, mais l'emphase est plutôt mise sur la lourdeur du personnage lorsqu'il tombe grâce aux sonorités graves.
Le son est intégré dans FMod avec un fondu d'entrée pour signifier une prise de vitesse progressive du personnage en l'air.



4. Pilar sounds

Il existe deux évènements FMod liés aux sons d'extrusion : un pour chaque type d'extrusion du toy. Les deux évènements comprennent deux pistes : l'une pour un son d'extrusion proche du personnage, et l'autre pour un son d'extrusion lointaine.
Les deux évènements sont en 3d ; la spatialisation permet de localiser le son autour du personnage, mais également d'en calculer la distance. Ainsi, lorsque le personnage s'éloigne le son « proche » est progressivement remplacé par le son « lointain ».
Le son lointain correspond au son proche, duquel ont été supprimé les fréquences hautes, et la réverbération (déjà présente sur le son proche, mais peu appuyée) est augmentée.

Le son du Pilar1 (le pilier haut) est une boucle, à la sortie de laquelle le son se termine par un fade out rapide. Ce système permet d'adapter le son à la durée de l'extrusion ; même si l'extrusion est plus courte prévu (si le pilier ne peut plus « pousser » à cause d'un obstacle par exemple), le son se termine toujours par un fade out fluide.

Le son du Pilar2 (de type « panneau ») est un son «oneshot», car cette extrusion a toujours la même durée. Tout comme pour le son précédent, celui-ci se termine par un fadeout

User Experience
&
User Interface

1. HUD

Comme mentionné précédemment, l'UI d'Bios ex Machina a été pensée comme étant minimaliste. Le toy fonctionne avec un nombre très restreint de mécaniques, et nous n'avons pas ressenti le besoin de multiplier les indicateurs.
De plus, une partie des mécaniques (principalement la mécanique de flow) est déjà représentée et signifiée au joueur par le biais des feedbacks visuels et sonores : musique évolutive, changement du FOV...

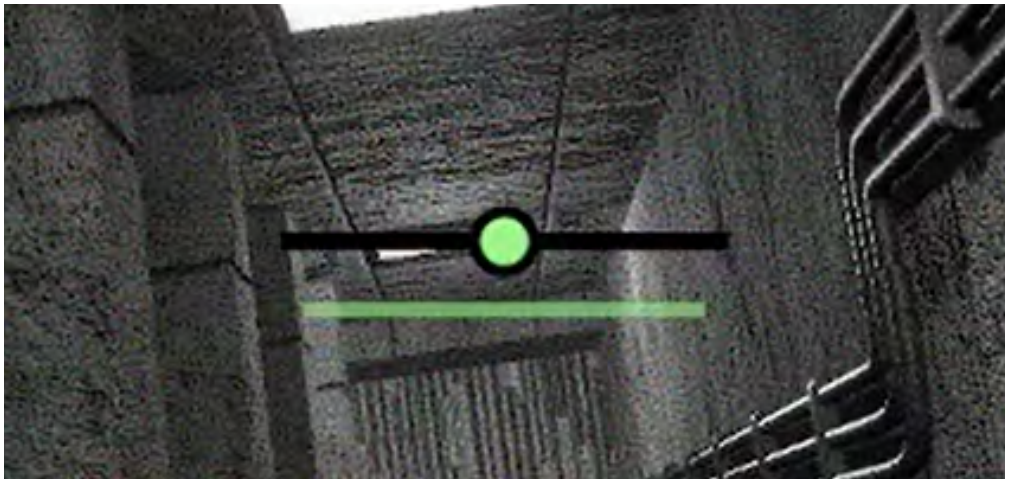
Pour cette raison, le HUD en jeu se résume à deux éléments, la Flowbar, la flowbar et la barre de timer.
Elle sert à la fois de réticule, grâce au point vert central, et d'indicateur d'actions donnant du flow au joueur : Slide, Wallrun ou encore Parallel boost.

Flowbar

- Elle accompagne les mouvements du personnage et en symbolise la vitesse de manière à ce que ses différents états puissent coexister (par exemple, si le joueur effectue un slide et un parallel boost simultanément).

- Elle est suffisamment discrète pour que le joueur puisse aisément se concentrer sur l'action et le gameplay plutôt que sur la Flowbar, mais sa position centrale assure que le joueur reçoive en permanence les informations importantes, sans avoir besoin de détourner le regard.

De plus, le point vert offre des informations supplémentaires relatives à la mécanique d'extrusion : en plus d'infomer le joueur sur la zone visée, le point vert devient noir lorsqu'aucune surface n'est à portée du joueur pour extruder.



Barre de timer

La barre de timer est située sous la Flowbar, et permet au joueur de garder connaissance des timings à respecter pour maintenir son état de Flow.

- Sa longueur totale dépend de l'état de Flow actuel du joueur, et diminue à chaque nouveau palier atteint.

- Sa longueur en jeu change en permanence : elle diminue en même temps que le timer de l'état de Flow.

- Pour le joueur, si la barre se réduit jusqu'à disparaître, il perd un niveau de Flow et la barre de timer s'étend jusqu'à atteindre la longueur total du nouveau palier.

2. Menus

Trois écrans de menu ont été mis en place pour les menus.
Ils sont tous naviguables à la souris ou au clavier, en utilisant les touches Z et S ou les flèches directionnelles pour naviguer et Enter pour sélectionner.

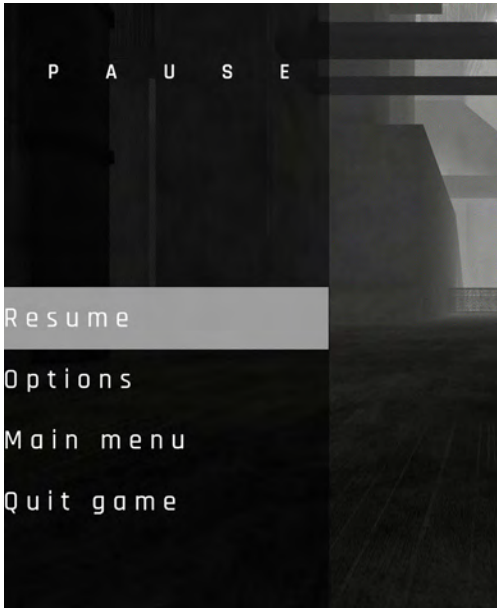
Main menu

Le menu principal s'affiche au lancement du jeu. Il permet de lancer une nouvelle partie sur le niveau choisi, d'accéder au Settings menu via « Options » ou de quitter le jeu.

Sur le main menu, une caméra se déplace sur le niveau selon un trajet préétabli qui permet d'offrir au joueur une première approche du toy avant même qu'il ne lance sa première partie.

Pause menu

Le menu de pause est accessible en partie à tout moment en appuyant sur Esc. Il permet de quitter le jeu, de retourner au Main menu ou d'accéder au Settings menu. Il permet également de revenir dans la partie via « Resume » ou en appuyant de nouveau sur Esc.

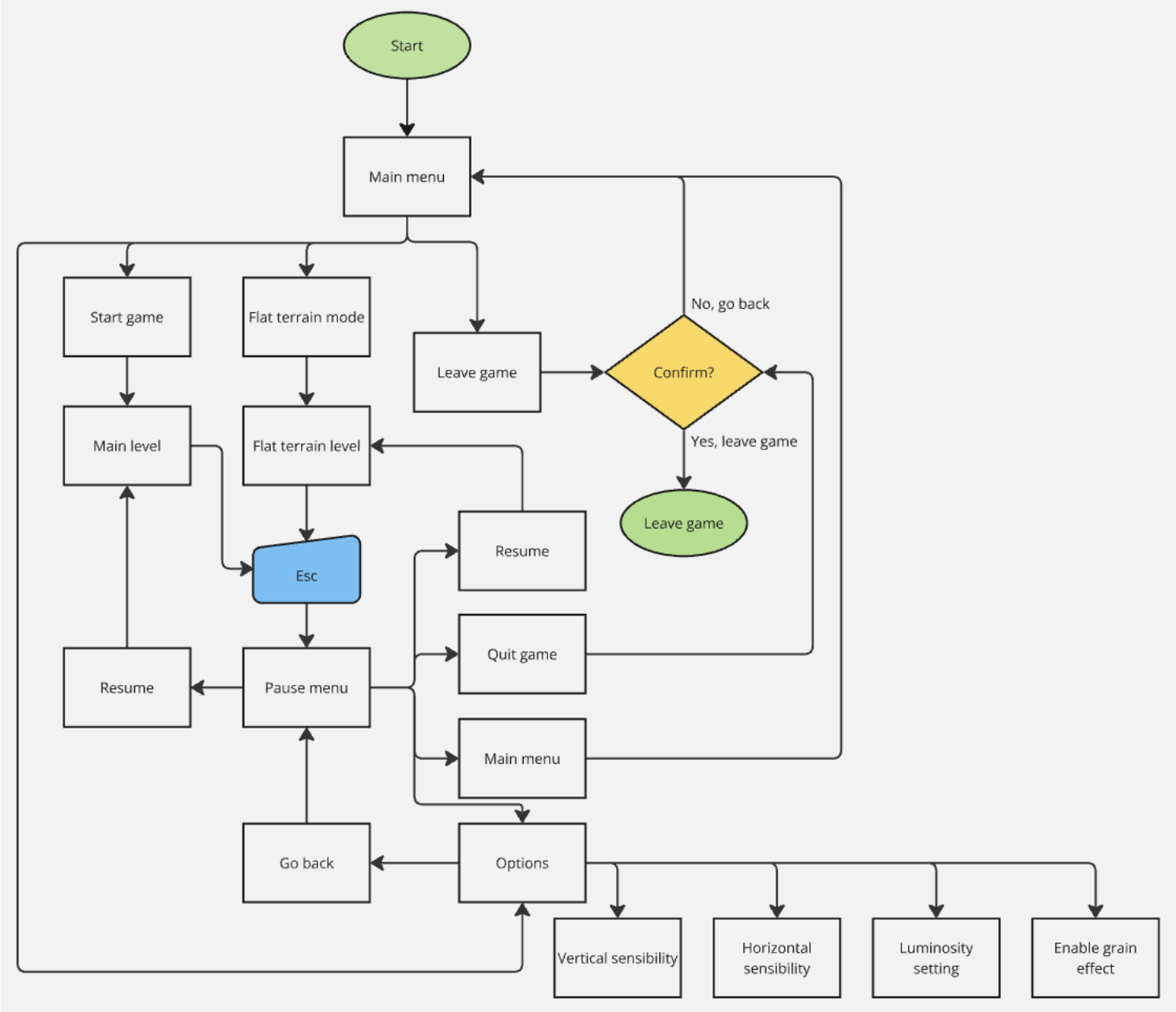


Settings menu

Le Settings menu permet d'accéder au réglage de plusieurs options :

- L'activation ou la désactivation du grain dans les effets visuels du jeu.
- Le réglage de la luminosité.
- Le réglage des sensibilités horizontale et verticale.

3. Flowchart des menus



Documentation technique

1. Enjeux techniques

Bios ex Machina repose principalement sur la mécanique de **Flow**, il était donc très important que le chara controller soit le plus précis et fluide possible, et que tout fonctionne comme le joueur s'y attendait. En bref, que les contrôles de déplacement soit instinctifs.
L'objectif était d'offrir au joueur une bonne maîtrise de ses actions tout en limitant le nombre d'inputs nécessaires. Le tout permettant au joueur de ressentir le flow sans ajouter de surcharge cognitive.

Pour pallier à ces enjeux techniques de fluidité et de réalisme favorisant l'immersion, le chara controller ne modifie pas directement la vélocité du joueur, il fonctionne avec des forces.
Plusieurs mouvements de caméra tels que le HeadBob ou des tilt de strifing, un tilt d'impact d'atterrissage, un tilt de slide et autres ont été utilisés pour favoriser au mieux l'immersion.
Beaucoup de débug et de fonctionnement secondaires ont été ajoutés afin de favoriser l'immersion et le GameFlow du joueur, et ne pas le briser.

La plupart des enjeux techniques du personnage sont donc lié a son fonctionnement, penser et intégrer tous les fonctionnnement additionnels, la physique, les forces et les sensations.
Quand aux piliers, les enjeux techniques étaient liés a leur fonctionnement,la mécanique d'extrusion, leurs textures mais surtout au débug, faire que le joueur ne traverse pas les piliers, que les piliers s'arretent bien, ne poussent pas le joueur a travers la carte...

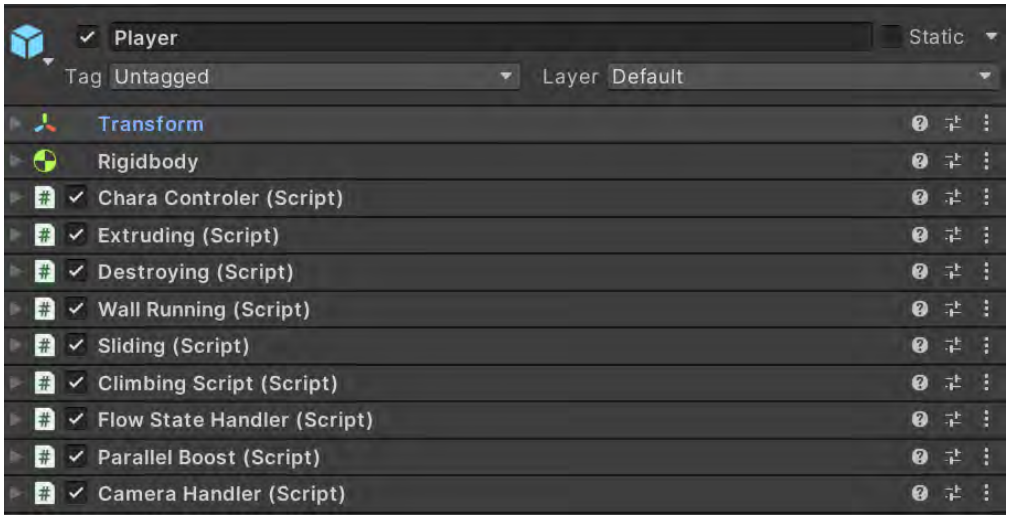
Au niveau direction artistique, l'ambiance trouvée, l'enjeux le plus gradnd était d etout rendre visible, lisible et compréhensible tout en restant en greyscale sauf pour le vert.
La plupart des réalisations de direction artistique ont donc été de l'ordre des matériaux, réalisés entierelment sur substance designer.
L'animation de la main fut le deuxieme grand enjeu de direction artistique.

2. Code

2.1 Player dans l'éditeur

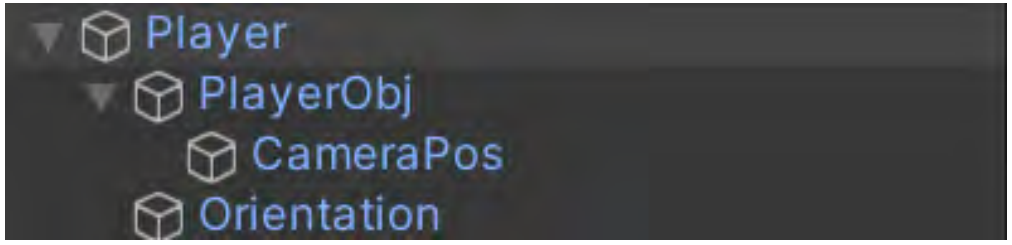
Player :

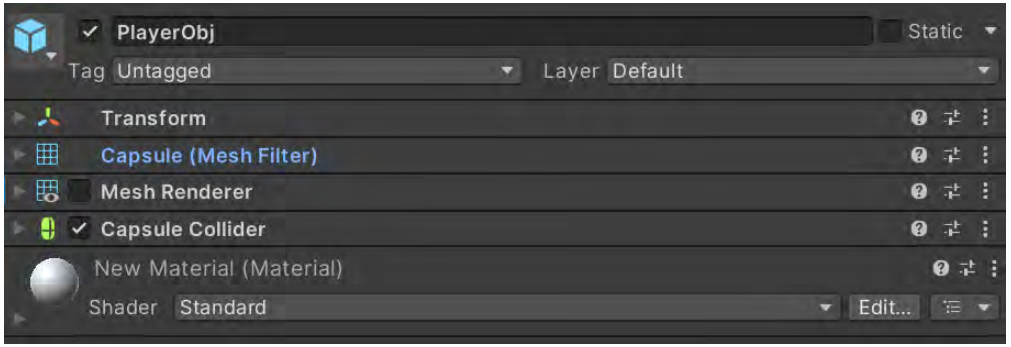
Le personnage est composé principalement d'un GameObject «player» sur lequel le code du chara controller est placé.
Sont placés en enfant de cet objet la hitbox «capsule collider» du joueur sans mesh renderer, un objet orientation gérant le forward du joueur et un objet CamPos sur lequel la camera est placée.

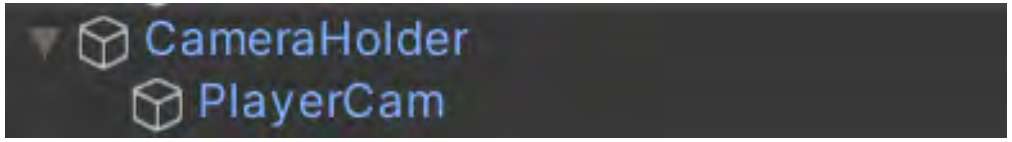


Caméra :

Du côté de la caméra, il y a un objet CamHolder gérant les mouvements de la caméra, et la caméra elle même en enfant.



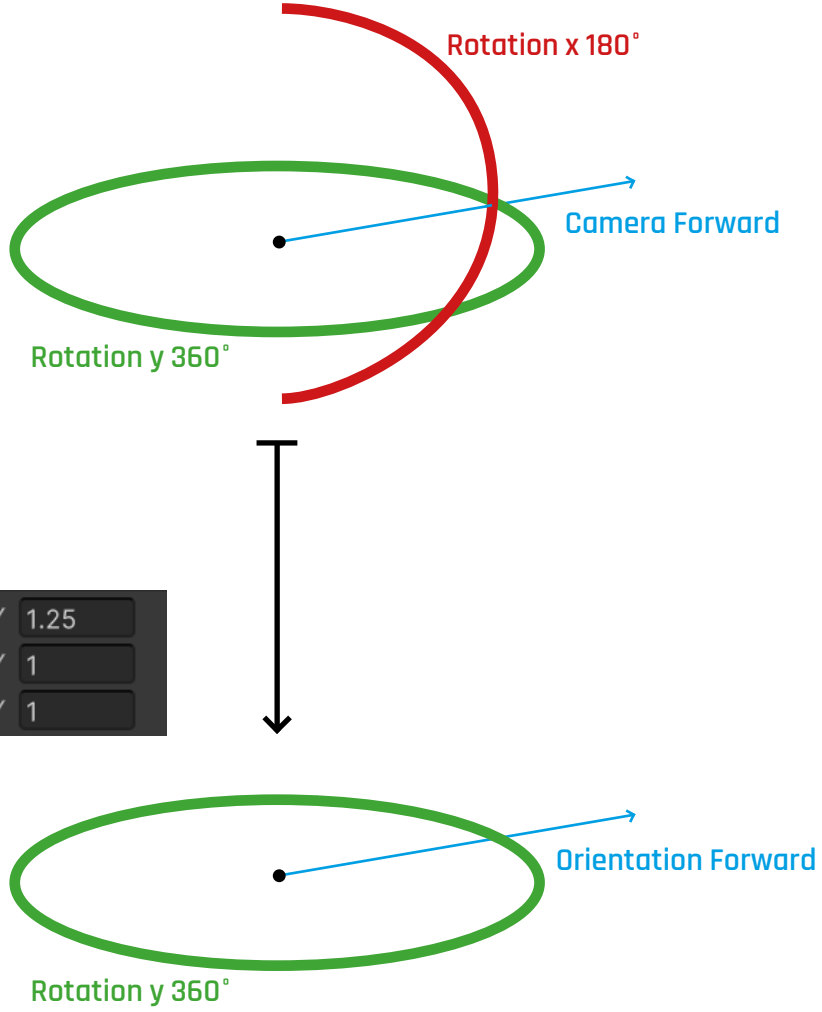




2. Code
2.2 FPS Camera

Caméra FPS :

La caméra fonctionne à la manière de la plupart des jeux en première personne. Elle se contrôle à la souris avec une rotation à 180° sur l'axe y (horizontal) et une rotation Clamp à [90 ; -90] sur l'axe x (vertical).
Les mouvements verticaux et horizontaux de la caméra dépendent de deux variables de sensibilité (x et y) afin que tous les joueurs puissent adapter les contrôles selon leurs préférences.
Ainsi, les mouvements de la souris changent l'angle d'un GameObject CamHolder, dont la caméra est enfant.
La position de cet élément CamHolder est en permanence mise à jour suivant la position d'un GameObject CamPos enfant du joueur. La caméra n'a pas été placée directement en enfant du joueur pour éviter certains bugs.
Un mouvement de Smoothing est appliqué à la caméra ; ainsi, cette dernière n'est pas dépendante du nombre de frames par seconde.



Changements de sensibilité :

Sliding Divider	X	1.5	Y	1.25
Climbing Divider	X	1.75	Y	1
Wallrunning Divider	X	1.25	Y	1

Lorsque le joueur slide, climb ou wallrun, les sensibilités x et/ou y de la camera sont légèrement changées, impactant ainsi la vitesse de rotation de sa vision et donc de son personnage.

Orientation du joueur :

L'objet Orientation est un GameObject enfant du joueur. Il prend l'angle Horizontal Y de la camera, son forward est donc toujours le forward de la caméra, sans prendre en compte la verticalité.
Ce GameObject sera utilisé pendant tout le code du character controller afin d'utiliser son axe forward comme l'axe forward de référence pour le joueur, par exemple pour son mouvement.

2. Code

2.3 Mouvement basique

Le mouvement du personnage se fait en plusieurs étapes:

Chaque Frame :
- -Ground check
- -Input
- -State Machine
- -Speed control

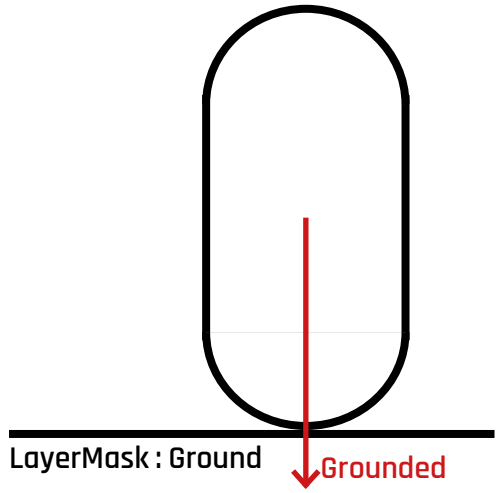
Chaque Fixed update :
- -Move Player

- Le ground check est un bool grounded dépendant d'un Raycast partant du joueur, vers le bas et détectant si le joueur est sur le sol ou non.
Cette méthode de ground check s'accordait parfaitement avec notre projet, y compris pour un système de mouvement en pente, présenté plus bas.

-Lors de la fonction Input, les input ZQSD sont récupérés en «GetAxisRaw». Les axes n'ont donc que 3 valeurs disponibles, -1, 0 et 1.
Les input ZQSD sont ensuite stockés dans deux vecteurs 3 directionnels, Horizontal et Vertical, qui seront utilisés lors de la fonction MovePlayer.

- La state machine gère la vitesse du joueur selon son état. S'il marche, le Float «desiredMoveSpeed» récupère la valeur de la vitesse de marche ; il en va ainsi pour chaque mouvement : wallrun, climb, slide...

- Ensuite, le speed control régule la vitesse (velocity.magnitude) pour qu'elle ne dépasse pas le movespeed de l'état,
Avant de réguler la vitesse, la fonction Speed control vérifie si la «DesiredMoveSpeed» est trop éloignée de la «LastDesiredMoveSpeed». Si c' est le cas, elle fait la liaison entre les deux paliers de vitesse de manière «smooth» en utilisant une corroutine et un Lerp.
Ainsi, les changements de vitesse s'effectuent de manière très fluide.
Ils s'effectuent également plus rapidement lors du slide.



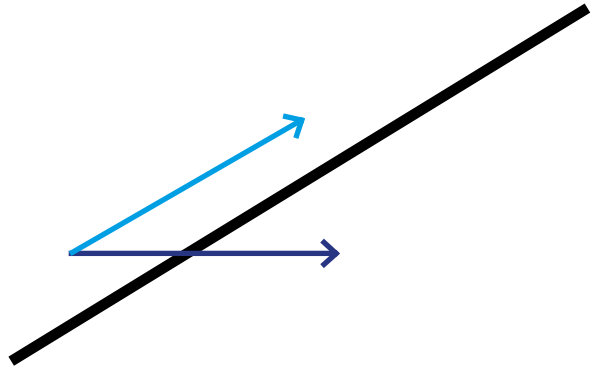
2. Code

2.3-4 Mouvement basique & Jump

Mouvement basique :

Lors de la fonction Move Character, une force s'applique sur le personnage pour le faire se déplacer.
Il se déplace selon un vecteur 3 « MoveDirectionVector » normalisé dépendant des inputs, dans la direction de Orientation.forward pour les inputs verticaux (Z et S) et dans la direction orientation.right pour les inputs horizontaux (Q et D).
La vitesse de mouvement, et donc la force de ce Addforce est dépendante de la vitesse définie par le stateHandler, puis régulée par le speedControl.
En l'air, le mouvement est multiplié par 0.4, réduisant ainsi l'air control.

Déplacement forward sans « project on plane »
Déplacement forward avec « project on plane »



Mouvement en pente :

Bool OnSlope : si le groundcheck renvoie une normale dont l'angle n'est pas 0 (sinon c'est un sol droit), et dont l'angle n'est pas supérieur à l'angle de pente maximal (float), le bool OnSlope est en True, le personnage est donc debout sur une pente.

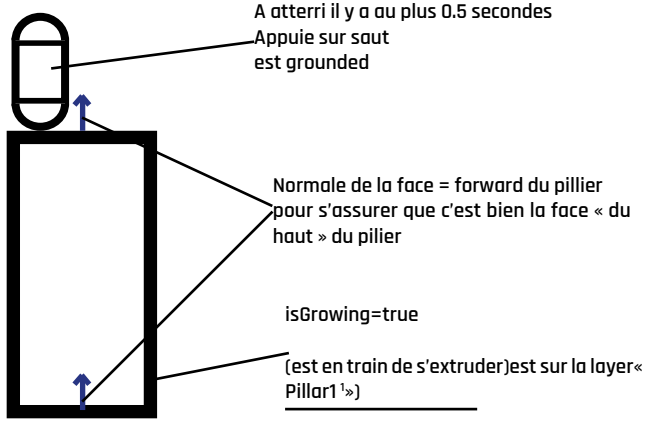
Lorsque le personnage est debout sur une pente, sa vitesse de déplacement est augmentée, et il se déplace de la même manière que sur un sol droit, mais ce déplacement se fait en utilisant un « Project On Plane », ce qui lui permet de se déplacer dans l'angle de la pente. Une force est aussi appliquée au joueur pour le garder en contact avec la pente.

Saut :

Le saut de base est un simple force Impulse en vector3.up, avec comme condition le joueur Grounded et un Cooldown.
Le système comprend également deux autres types de sauts : Le SlideJump, et le PilarJump

Le SlideJump s'effectue lorsque le joueur est en train d'effectuer une glissade et saute. Lors de ce SlideJump, la force up du jump est plus élevée, et une force Forward est également appliquée.
Le PilarJump, lui, s'effectue différemment. Lorsqu'un joueur atterrit sur la face supérieure d'un pilier en train de s'extruder, s'il saute dans les 0.5 secondes suivant son atterissage, il effectue un PilarJump. Tout comme le SlideJump, une force forward en impulse est appliquée et la force de saut est augmentée.

Conditions du pillarJump :



2. Code

2.5 Crouch & Slide

Lorsque le joueur se déplace sur le sol ou atterrit (tout en maintenant une touche de déplacement), si la touche Maj est enfoncée, le joueur commence à slide.

Dans ce cas, son échelle y est divisée par deux, une force en impulse vers le bas est ajoutée en même temps pour que le joueur ne quitte pas le sol pendant quelques instants (comme c'est le cas, par exemple, dans HalfLife 1).

Un timer démarre, et si le joueur saute ou effectue un parallel Boost vertical (nous y reviendrons plus tard), ce timer est redémarré. Si ce timer atteint 0, alors le joueur arrête de glisser et reprend sa taille normale, s'il le peut.
Si il ne le peut pas, alors il passe en « crouch », jusqu'à ce qu'il puisse reprendre sa taille normale.

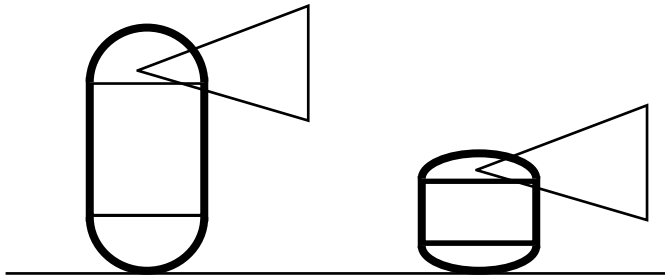
L'état crouch est un état qui comme le slide rend le joueur « plus petit » et plus lent.

Le slide, lui, est rapide, permet d'effectuer des SlideJump, et s'accélère encore en pente.

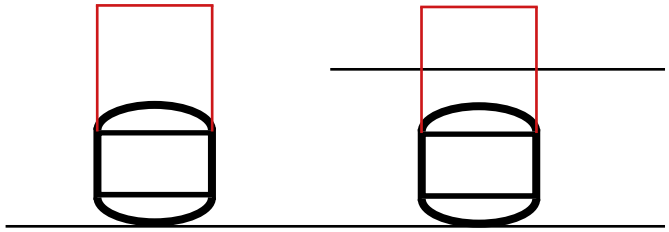
Pour s'assurer que le joueur ait la place de se redresser, un SphereCast est effectué en l'air, qui alimente un Bool «CanStandUp».

Lorsque le joueur se relève, renfoncer la touche Maj lui permet de se remettre à Slide.

Si le joueur slide et arrête de bouger, ou appuie sur Maj en ne bougeant pas, il crouch.



Normal scale et caméra crouch/slide scale et caméra



Can stand up Cannot stand up

2. Code

2.6 Wallrun

Lorsque le joueur est en l'air, deux raycasts sont tirés à droite et à gauche du joueur par rapport à son orientation.

Si un de ces rayons touche un mur, et que cet objet est sur la layer « walls », que le joueur ne wallrun pas déjà et n'est pas en état « exitWall », et que le joueur avance (Z), alors il commence à Wallrun et sa vélocité redescend à 0.

L'orientation avant du mur, et donc la direction vers laquel le joueur wallrun est un vecteur directionnel perpendiculaire à la normale du mur et perpendiculaire à vector3.up. on l'obtient avec le calcul suivant :

```
Vector3 wallForward = Vector3.Cross(wallNormal, transform.up);
```

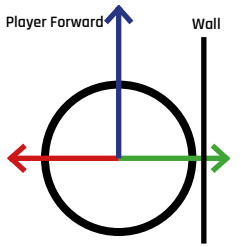
Cependant, si le joueur est orienté vers l'autre côté, alors le forward du mur est inversé, on obtient ceci avec le calcul :

```
if ((orientation.forward - wallForward).magnitude > (orientation.forward - -wallForward).magnitude)
{
    wallForward = -wallForward;
}
```

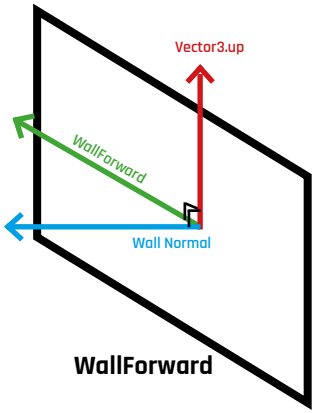
Pendant le wallrun, une force contraire à la gravité est appliquée en permanence afin de réduire celle-ci à une valeur infime proche de 0.

Lorsque le joueur wallrun, un timer démarre. Lorsqu'il se termine, un autre démarre, puis un autre encore afin de segmenter le wallrun en 3.

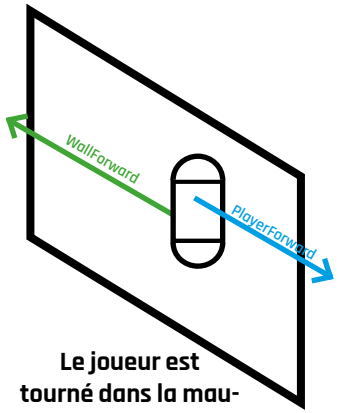
Lorsque le joueur wallrun, une force au forward du mur lui est appliquée. Elle est appliquée differement selon les 3 parties du wallrun : d'abord en diagonale haut-avant, puis juste en avant, puis en avant-bas jusqu'à ce que le joueur s'éloigne du mur.



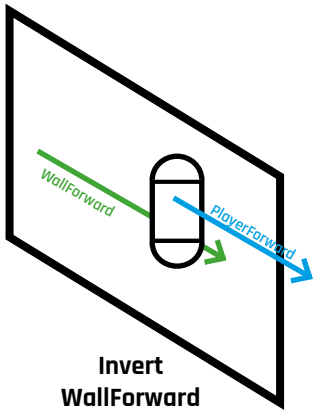
Ici, le joueur peut wallrun sur le mur de droite



WallForward

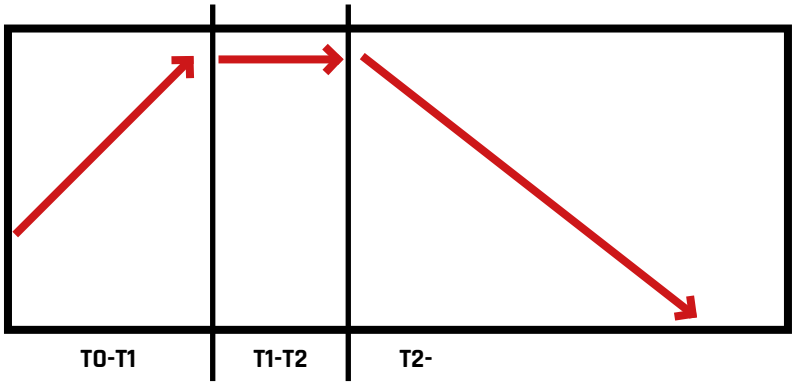


Le joueur est tourné dans la mauvaise direction



Invert WallForward

Déplacement du joueur sur un mur selon les différents Timers



2. Code

2.6 Wallrun

Wallrun down :

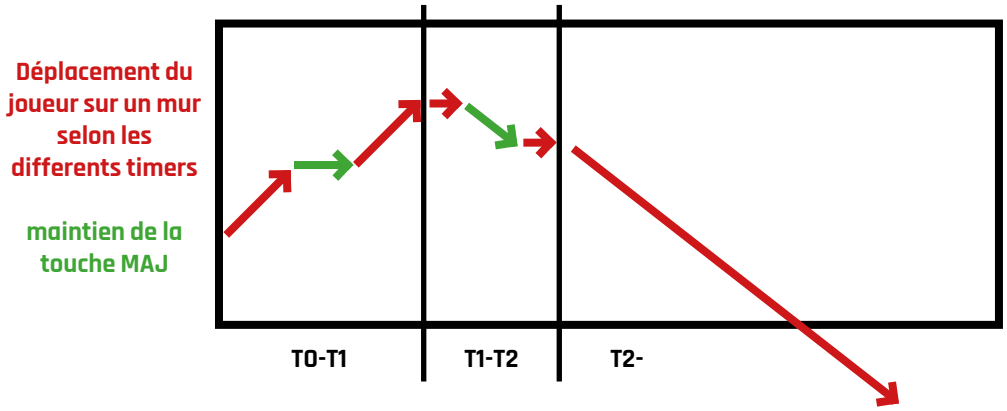
Lorsque le joueur wallrun il peut a tout moment appuyer sur Maj pour temporairement se déplacer en diagonale bas, ainsi il gagne plus de maitrise sur son wallrun.

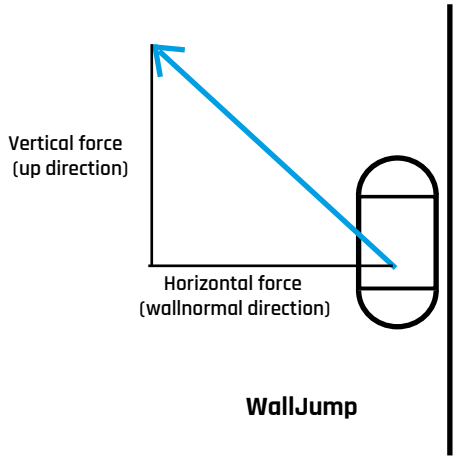
WallJump :

Si le joueur est en train de Wallrun et qu'il appuie sur la touche de saut, une force lui est appliquée en « up » et dans la direction de la normale du mur. Ainsi, le joueur saute en l'air et dans la direction opposée du mur:

```
Vector3 forceToApply = transform.up * wallJumpUpForce + wallNormal * wallJumpSideForce;
rb.velocity = new Vector3(rb.velocity.x, 0f, rb.velocity.z);
rb.AddForce(forceToApply, ForceMode.Impulse);
```

Quand le joueur walljump, un booléen « isExitingWall » est activé pendant quelques dixièmes de secondes, il agit comme un « cooldown » de wallrun pour éviter les bugs.



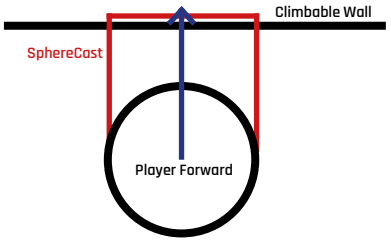


2. Code
2.7 Climbing

Conditions de climb :

Pour que le joueur puisse grimper un mur, il doit être dans une layer comprise dans le layermask utilisé pour la détection du climb, comme les layers «Pillar1» ou «Pillar 2» (qui correspondent aux deux types d'extrusions).

La détection de murs grimpables est faite avec un SphereCast partant du joueur vers son forward.

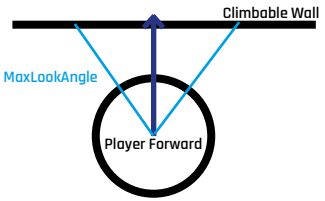


Si ce Spherecast touche un mur grimpable, l'angle du forward du joueur par rapport à l'angle de la normale du mur est comparé :

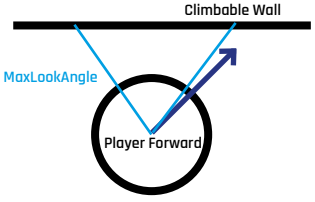
```
wallLookAngle = Vector3.Angle(orientation.forward, -frontWallHit.normal);
```

Ainsi, si le joueur avance (input Z), que l'angle de regard du joueur par rapport au mur est en-dessous de l'angle maximum, qu'il ne sort pas du mur (exactement comme pour le wallrun) et qu'il est en l'air, alors le climb est enclenché.

```
if (WallFront && Input.GetAxisRaw("Vertical") == 1 && wallLookAngle < maxLookAngle && !exitingWall && !cc.grounded)
```



Can Climb



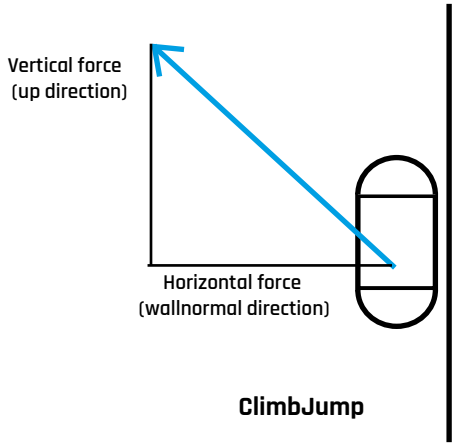
Cant Climb

Climbing :

Lorsque le joueur grimpe un mur, une force dans l'axe vector3.up lui est appliquée à chaque fixedupdate, soit continuellement. Sa vitesse, comme pour tout déplacement, est régulée par la fonction «SpeedControl». Lorsque le joueur commence à climb, un timer est démarré. Quand ce timer arrive à 0, le joueur lâche le mur et tombe.

Climb Jump :

Le climbJump fonctionne comme le walljump : il applique une force à la fois en l'air et hors du mur et un cooldown «exitingWall» est enclenché.



ClimbJump

2. Code
2.8 Extruding

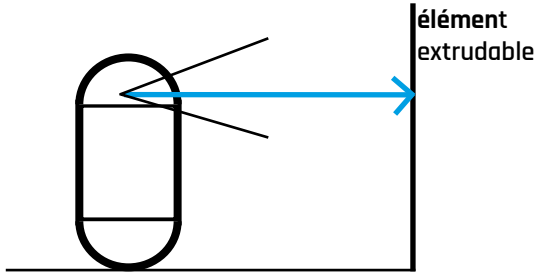
La mécanique principale de notre toy est l'extrusion. Un raycast de la caméra vers l'avant s'exécute en permanence, limité par une distance maximale et un cooldown après l'action, pour extruder une surface.

- Le joueur appuie sur Clic gauche (Mouse0) ou clic droit (Mouse1) pour le type d'extrusion qu'il désire.
- Il peut extruder (le cooldown est enclenché).
- Si le raycast touche une surface avec le tag «extrudable» :
    - Un spherecast est lancé depuis cette surface avec une longueur prédéfinie. Si ce spherecast touche un autre mur ou une extrusion, alors le jeu considère que cette extrusion n'aura pas la place de grandir, et n'a donc pas lieu.

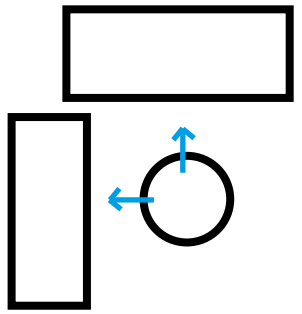
Cette dernière condition empêche le joueur de placer une extrusion qui n'aura ensuite pas la taille de grandir. Si elle n'existait pas, les extrusions s'arrêtant toujours à une distance définie du mur, un élément placé plus proche que cette distance existerait sans grandir. Ce n'est pas la réaction recherchée par le joueur, nous avons donc considéré qu'il serait plus simple d'empêcher tout simplement ce comportement. (Tous les raycasts sont effectués dans «fixedupdate» pour éviter les bugs).

Si toutes ces conditions sont réunies, alors un gameobject Pillar1 (pour Mouse0) ou Pillar2 (pour mouse1) est instancié à l'endroit où le raycast touche le mur. L'angle auquel ce game object est instancié est l'angle de la normale de la face sur laquelle l'extrusion se fait.

De plus, si l'extrusion est un panneau et que ce panneau est posé sur le sol (ceci est calculé sur une comparaison entre l'ange de la normale de la surface extrudée et le world space), alors le panneau est posé dans la largeur par rapport à la position du joueur.
Ceci permet 2 choses :
-L'angle des panneaux placés au mur a toujours le même angle : ils sont placés horizontalement contre le mur
-L'angle des panneaux posés au sol dépendent de l'angle du joueur, ce qui est plus intuitif



Extrusion Raycast(40m)



Ici, un panneau est posé sur le sol, il est donc orienté dans la largeur par rapport à l'angle du joueur

2. Code

2.9 Extrusions

(Le terme «piliers» désigne ici les deux types d'extrusion, le panneau étant également un pilier avec une forme différente.)

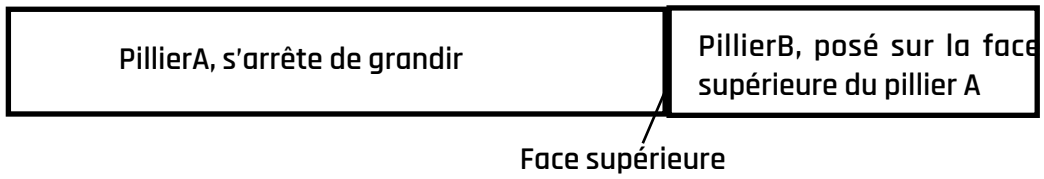
Quand un pilier est instancié, son échelle locale z est mise a 0.01f, puis ses collisions sont activées. De cette manière, le joueur ne se heurte pas à un mur invisible s'il la place avant une fixedupdate.

A chaque frame, son échelle locale est égale à « elle-même + le temps final de l'extrusion*deltaTime » (la taille est en pourcentage, ainsi 1 vaut 100%, et si l'extrusion dure 10 secondes il grandira donc de 10% par seconde).

Le point de pivot du pilier est à sa base. Quand son échelle z grandit, il «pousse» dans sa direction forward.

L' extrusion du pilier s'arrête si :
- Un élément se trouve a x mètres devant lui (cette donnée est acquise avec un spehercast)
- Une extrusion est réalisée sur la face avant du pilier
- L'échelle locale du pilier atteint 1 (100% de sa taille maximale)

Ainsi le pilier ne grandit pas à travers les murs et ne pousse pas le joueur à travers les murs, car il s'arrête avec juste assez d'espace entre lui et le mur pour que le joueur puisse y passer.
Nous avons créé plusieurs fonctions de débug. Par exemple, quand le joueur pose un pilier sous ses pieds, son rigidbody est réactualisé pour que l'extrusion ne grandisse pas à travers son corps.
Une autre fonction permet que lorsque le joueur place une extrusion, le jeu vérifie si cette extrusion est placée sur un pilier, puis vérifie si la face sur laquelle l'extrusion est posée est la face «avant» du pilier.
Si c'est le cas, le jeu accède au code du pilier et désactive son booléen «canGrow». Sans cela, le pilier continuerait de grandir à travers le nouveau pilier.



forward
pilier, échelle z actuelle (0.1)
échelle z finale (1)

```
//if it can grow
if (IsGrowing)
{
    //calculate the distance it will grow in one frame
    float growth = 1 / TimeToFinaleSize * Time.deltaTime;

    //grow (modifying local scale)
    gameObject.transform.localScale += new Vector3(0f, 0f, growth);
}
```

forward
sphereCast : si un élément est dans cette zone, le pilier cesse de grandir. Cette zone fait la taille du joueur (approximativement).
échelle z finale, si le pilier l'atteint, il s'arrête de grandir

```
RaycastHit hit;
bool GetPillar1 = Physics.Raycast(camHandle.transform.position, Camera.transform.forward, out hit, extrudeDistance, pillar1layer);

if (GetPillar1)
{
    pl0bj = hit.collider.gameObject;

    if (hit.normal == pl0bj.transform.forward && hit.collider.gameObject == pl0bj && GetPillar1)
    {
        pls = pl0bj.GetComponent<PillarScript>();
        pls.IsGrowing = false;
    }
}
```

2. Code
2.10 Parallel Boost

Parallel Boost

Le parallel boost est la conséquence du déplacement du joueur entre deux murs parallèles rapprochés. Quand le joueur passe entre ces deux murs, la fonction booléenne est active.

Il est utilisé comme une action qui donne du flow au personnage et pour le feeling dans le camerawork (changement de FOV).

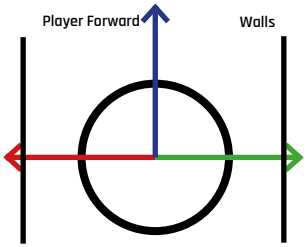
Il existe deux types de parallel boost, le parallel boost horizontal et vertical.

Parallel Boost Horizontal

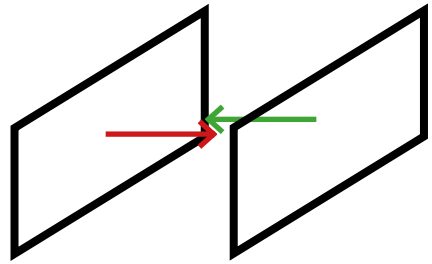
Le parallel Boost Horizontal envoie simplement deux raycast de portée limitée à gauche et à droite du joueur.
Si les raycasts touchent deux murs, alors les murs sont assez proches, et si l'angle entre la normalRight et (-)normalLeft est en dessous de 5°,
les murs sont considérés comme parallèles, et le booléen WallCheckHorizontal est en true.

Parallel Boost Vertical

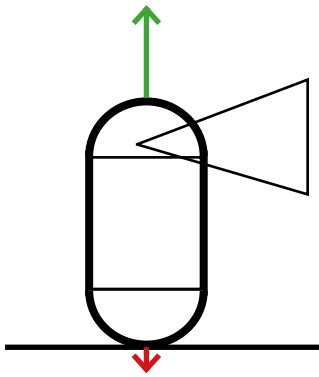
Il en va de même avec la fonction WallCheckVertical, cependant les raycasts sont calibrés pour être plus courts, et elle ne s'effectue que quand le joueur est en slide.
Le wallCheck vertical ne s'effectue donc que lorsque le joueur slide en dessous d'une surface parallèle au sol et où il n'aurait pas eu la place de passer debout.



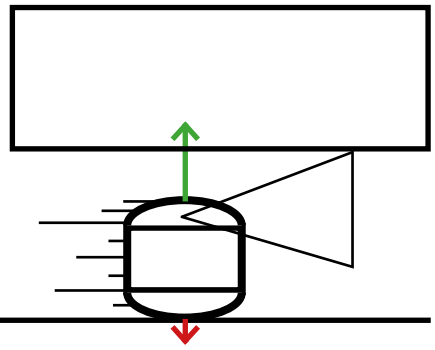
Horizontal Wallcheck
Les murs sont parallèles et dans les raycast



l'angle entre les deux vecteurs > 5 degrés, les deux murs sont parallèles.



Le joueur est debout, il ne peut pas vertical parallel-Boost



Le joueur slide en dessous de deux surfaces parallèles, il vertical parallelBoost

2. Code

2.11 Flow

Mécanique de flow :

La mécanique de « flow » marche ainsi : le joueur effectue des actions, ses actions rapportent du flow. En gardant son flow pendant un certain temps grâce à un « timer montant », le joueur atteint un nouveau niveau de flow, allant de 0 à 3. Un autre timer descend en permanence, et s'il atteint 0, le joueur perd un niveau de Flow. Lorsque le joueur perd un niveau de flow, le « timer montant » redescend jusqu'à atteindre la moitié de l'état précédent. Pour que le « timer descendant » n'atteigne pas 0, le joueur doit effectuer des actions de flow, qui réinitialisent le timer.

Plus l'état actuel est élevé, plus le timer descendant est rapide.
Quand le joueur est en l'air, ce timer avance légèrement plus lentement.

Flow:

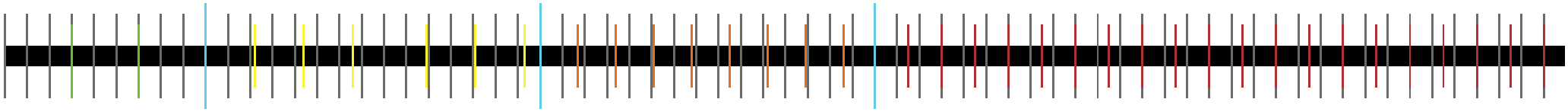
Les actions de flow sont gérées dans une fonction Bool appelée « Flow() » qui est en true si le joueur :
-wallrun ou walljump -climb ou climbJump -slide en pente
-pillarJump -parallelBoost

Fonction Booléenne Flow()

```
public bool Flow()
{
    // climbing, wallrunning, sliding on slope, destroying, parallel boost
    if (cc.climbing || cc.wallrunning || pboost.WallCheck() || pboost.WallCheckHorizontal())
    {
        return true;
    }
    //sliding on slope
    if (cc.sliding && cc.OnSlope())
    {
        return true;
    }
    //pillarJumping
    if (Input.GetButton("Jump") && cc.CanPillarJump() && cc.grounded || Input.GetButton("Jump") && cc.CanPillarJump() && cc.OnSlope())
    {
        return true;
    }
    //else no flow
    else { return false; }
}
```

Schéma des timers de flow

-De vert à rouge, les timers descendants, le joueur doit performer une action de flow entre chacune de ces barres pour ne pas passer a la moitié du niveau précédent,
- En bleu, les passages de niveaux, si le joueur ne performe pas, il reviendra a la moitié du niveau d avant.
- En gris les secondes



2. Code

2.12 CameraWork

CameraWork général :

En général, il y a 2 actions principales de camerawork utilisées ici : le tilt et le changement FOV.

Ces deux actions utilisent des Tweens que nous avons codé. La camera change de FOV selon le flow, mais également lors du slide, du wallrun et du parallel boost.

Changement de FOV :

Les changements de FOV principaux s'effectuent lors des changements d'état de flow. Plus l'état de flow est élevé, plus le FOV le sera également, offrant ainsi une sensation de vitesse sans changer directement la vitesse du personnage ni les metrics du jeu.
Le parallellBoost rend également un ajout léger de FOV. Enfin, un ajout minime de Fov est effectué lorsque le joueur wallrun.

Tilt :

Les Tilts de caméra correspondent à une rotation sur l'axe z local de cette dernière. Ils s'enclenchent lors :
- d'un wallrun : elle tilt du côté inverse du mur, à 20°
- d'un strifing : elle tilt d'avantage au sol qu'en l'air
- d'un slide : elle tilt vers la droite ; ce strifing est réduit en l'air
- d'un atterrissage : quand le joueur atterrit, la camera tilt très légèrement vers la gauche puis revient à la normale, créant une sensation légère d'impact ou de déséquilibre à l'atterrissage.

Valeur des changements de FOV, des tilt, et leurs vitesse up et down respectives

Fov Speed Multiplier	45
State 0 Fov	70
State 1 Fov	95
State 2 Fov	120
State 3 Fov	140

Parallel Boost	
FOV Boost	15
FOV Boost Multiplier	120
FOV Boost Multiplier Down	25

General	
Actual Fov	0
Wallrun	
Wallrun Cam Tilt	20
Tilt Speed Multiplier Down	6
Wr Tilt Speed Multiplier	6
Wr Tilt Speed Multiplier Down	6
Wallrun Cam Fov	10
Wallrun Cam Fov Speed	50
slide	
Slide Fov	10
Slide Fov Speed	50
Slide Tilt	-10
Slide Tilt Speed Multiplier	2
Slide Tilt Speed Multiplier Down	4
Land Max Time	0.25
Land Tilt Speed	0.75
Land Tilt Speed Down	0.75
Land Left Tilt	3
SideWalk	
Side Walk Speed	1.25
Side Walk Tilt	4
Side Walk Speed Down	1.25
Side Walk Air Speed	1
Side Walk Air Tilt	5
Side Walk Air Speed Down	1

2. Code

2.12-13 CameraWork & terrain dans l'éditeur

HeadBob :

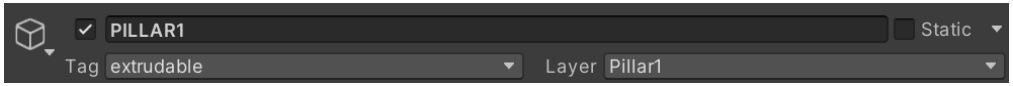
Lorsque le joueur marche au sol il y a aussi un léger HeadBob, pour que le joueur n'ait pas l'impression de glisser au sol.
Cet headBob est un mouvement extrêmement léger de caméra en utilisant les fonctions sinus et cosinus.

```
private Vector3 FootStepMotion()
{
    Vector3 pos = Vector3.zero;
    pos.y += Mathf.Sin(Time.time * frequency) * amplitude;
    pos.x += Mathf.Cos(Time.time * frequency / 2) * amplitude * 2;
    return pos;
}
```

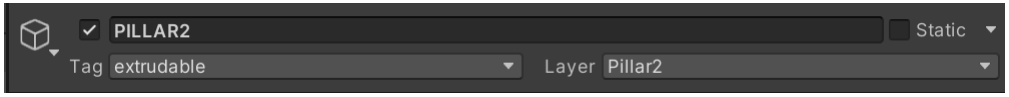
Terrain dans l'editeur :

Voici les éléments de terrain placés dans le LD (ainsi que les piliers). Leur layer et leur TAG permettent de différencier les interactions qu'ils proposent.
Tous les terrains ont un matériau physique dont la friction est réduite à 0.1f afin que le joueur ne colle pas aux murs.

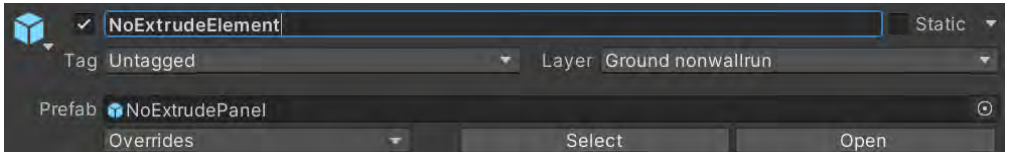
Pillar1 : extrudable, grimpable, wallrunable



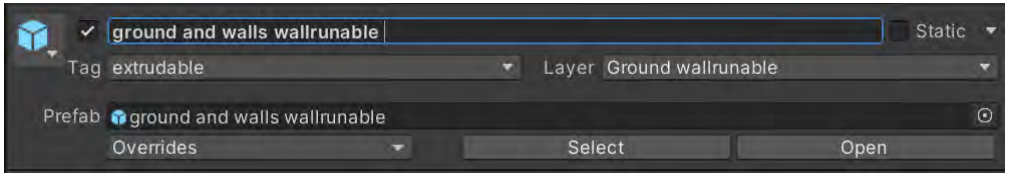
Pillar2 : extrudable, grimpable, wallrunable



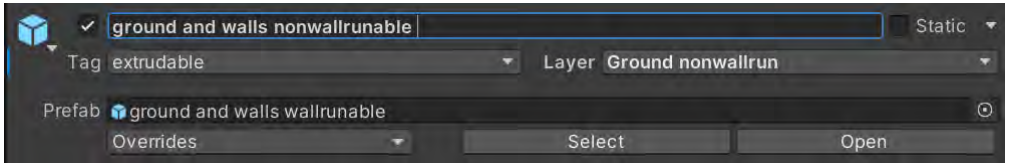
NoExtrudeElement : non extrudable, non grimpable,non wallrunable



ground&wallsWallrun : extrudable, non grimpable, wallrunable



ground&wallsNonWallrun : extrudable, non grimpable, non wallrunable

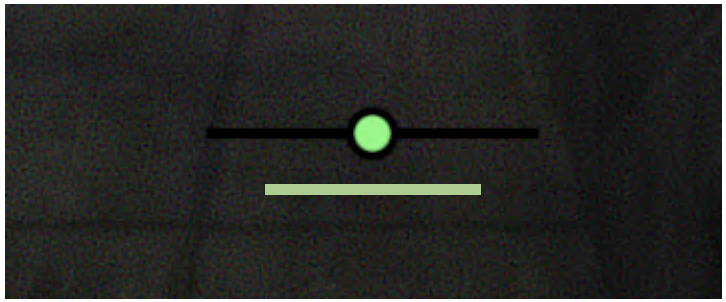


2. Code
2.14 HUD

HUD

Le HUD dispose de plusieurs éléments :
-Le pointeur : centré au milieu de la caméra, il fait office de «crosshair» pour que le joueur sache d'où il va extruder. Il change de sprite pour indiquer si une extrusion est possible ou non.
-Le timer : une barre verte dont l'échelle x et l'opacité dépendent de l'état de flow. Il réduit en taille x selon le timer de flow descendant pour visualiser l'information. Plus l'état de Flow est haut (0-3) plus la barre est opaque.
-La FlowBar : cette barre au centre du HUD réagit aux actions donnant du flow.

Fonctionnement de la FlowBar :



La FlowBar (la barre noire traversant le curseur) réagit en effectuant des rotations sur l'axe z et en changeant de scale ; elle est basée sur une série de Tweens, de la même manière que la caméra.
Voici ses réactions selon les actions de Flow :

- Wallrun : tilt pour s'aligner avec l'horizon, scale x +
- Slide : scale x ++
- Climb : tilt a 90° scale x+
- PillarJump, WallRunJump, ClimbJump : pulse (scale y ++ en 0.1s)
- ParallelBoost Vertical : se sépare en 2 verticalement (en utilisant un Lerp)
- ParallelBoost Horizontal : se sépare en 2 horizontalement (en utilisant un Lerp), tilt a 90°

Ces mouvement de la Flowbar peuvent être combinés. Les mouvements liés à des actions spontanées (PillarJump, WallrunJump, ClimbJump) déclenchent un timer de 0.1 secondes et le scale positif s'effectue dans ce timer.
Chaque mouvement possède sa propre vitesse et sa vitesse de « retour à la normale », ce qui permet des mouvement variés et smooth sans bug.

3. Direction artistique visuelle

3.1 Texture des extrusions

Randomisation

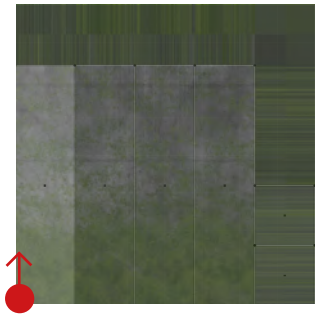
Lors de l'extrusion, les panneaux et piliers instanciés sont piochés aléatoirement dans une liste. Ainsi, les textures de chaque pilier ou panneau sont différentes. Pour chaque type d'extrusion on trouve 2 listes : «onGround» et «OnPillar». Si l'extrusion se fait sur une extrusion pré-existante, les textures seront différentes et plus végétalisées au niveau de la base que si l'extrusion se fait sur le sol ou un mur. Il y a donc 6 textures par type d'extrusion pour 12 textures et 24 matériaux.

Tiling des textures

Pour donner l'impression qu'il grandit du sol, le pillier change d'échelle, comme expliqué précédemment. Ce fonctionnement pose néanmoins problème au niveau des textures, car elles s'étirent à mesure que le pilier grandit.

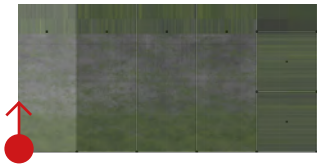
Pour pallier à ce problème, les textures changent de tiling en y en même temps que le pillier change d'échelle z, pour donner l'impression qu'elles poussent du sol et ne se « révèlent pas ». L'UV map a été modifiée pour que le haut des faces du pillier coïncide avec l'origine de la texture ; ainsi, le tiling part bien du haut du pillier l'impression recherchée de « pousse » est correctement rendue.

Les faces en haut et en bas des piliers ne sont pas soumises au problème d'étirement, et ne nécessitent pas d'appliquer le même traitement, c'est pourquoi ces faces dépendent d'un autre matériau, immobile.

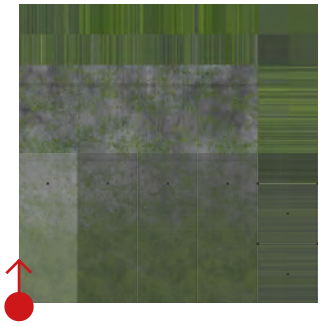


origine de l'UV et direction de la texture

Ici, le pilier a une taille de 1, tout comme la texture.



Le pilier est a 50% de son échelle z ; la texture se déforme automatiquement en même temps que le modèle ; les textures sont étirées.



Le Tiling en z de la texture est donc étiré proportionellement en y. Combiné a la base de l'UV map en haut du pilier, on a l'impression que celui ci sort du sol.

3. Direction artistique visuelle

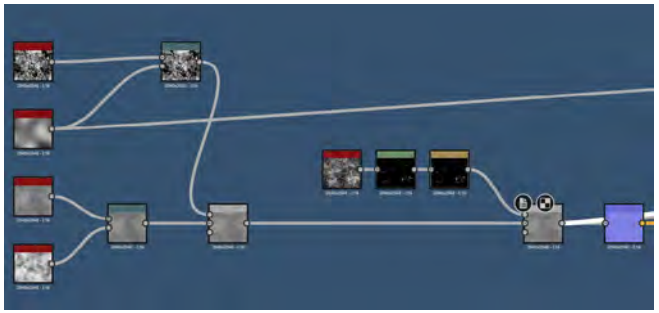
3.2 Textures nodales

Béton

Notre DA brutaliste fait que la plupart de la réalisation DA provient des textures, nous avons donc pris un soin tout particulier à ce que notre texture de béton soit la meilleure possible, car elle est présente en permanence dans l'environnement.
Cette texture a été réalisée sur Substance Designer.

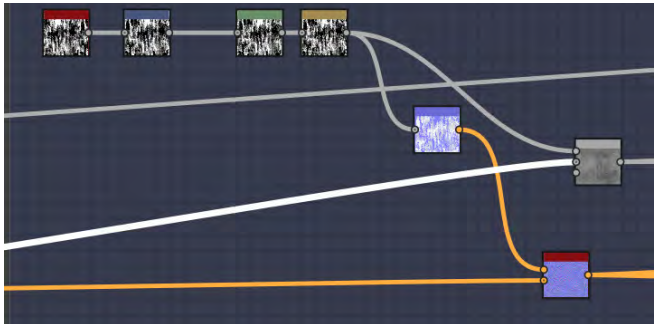
PREMIERE PASSE

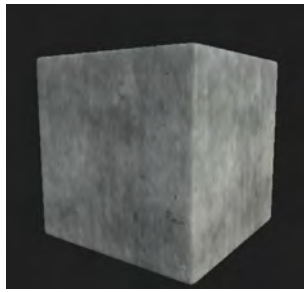
La base du béton est posée grâce à des grunge maps, des fractales, des « clouds » fusionnés selon différentes opérations. Il s'agit seulement de poser une base, puis des spots noir et blancs sont filtrés par niveaux et appliqués. De cette manière notre base présente déjà quelque petites tâches par endroits.
Cette base est également convertie en normal map.



SECONDE PASSE

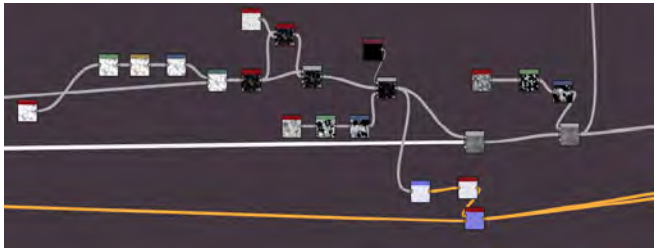
La seconde passe ajoute une grunge map en soustraction (assombrissant ainsi très légèrement certaines parties de la texture).
Cette grunge map agit également sur la normal map en ajoutant de la rugosité et du relief.

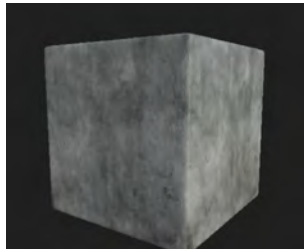




TROISIÈME PASSE

La troisième passe ajoute quelques craquelures à la couleur et la normal map. Ces craquelures sont faites en utilisant des cellules, des color ramp pour les toutes petites craquelures, puis elles sont appliquées seulement par endroits grâce à un masque. Ainsi, les craquelures sont localisées.
Puis d'autres taches sont appliquées à la couleur seulement en suivant un noise de moisture filtré par niveaux.



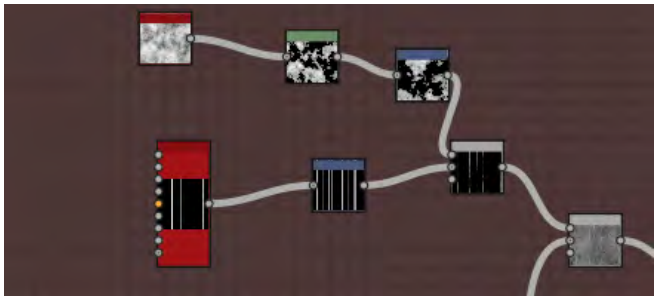


3. Direction artistique visuelle

3.2 Textures nodales

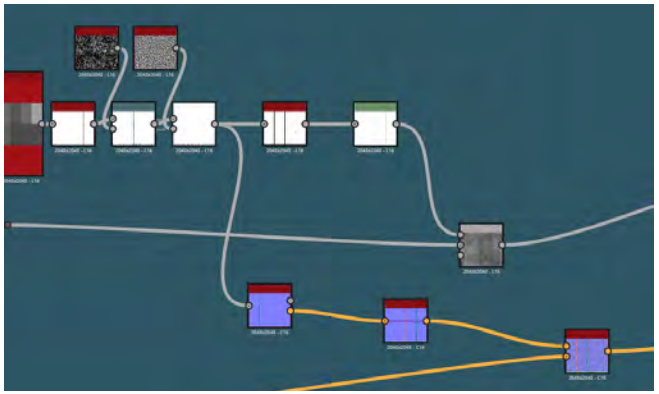
QUATRIÈME PASSE

La quatrième passe ajoute quelques lignes sur la couleur. Ces lignes sont générées grâce a un tile sampler puis appliquées à la texture avec un masque de clouds filtré par niveaux puis modifié par une transformation 2D



CINQUIÈME PASSE

La cinquième passe ajoute des lignes délimitant les blocs. Ces lignes sont créées grâce à un tile sampler + edgeDetect, puis légèrement déformées avec du noise. Elles sont ensuite « bevel » avec la normal map, et assombries.
On a donc des lignes de séparation assombries et creusées.
Le nombre de blocs est paramétrable, via le tile sampler, notre texture de base est en 4x2, mais d'autres textures du jeu sont en 1x2 par exemple.



Cette texture de béton est réutilisée pour «l'armure» du gant du personnage ainsi que la base des textures des piliers et panneaux.

Rendu In Game

3. Direction artistique visuelle

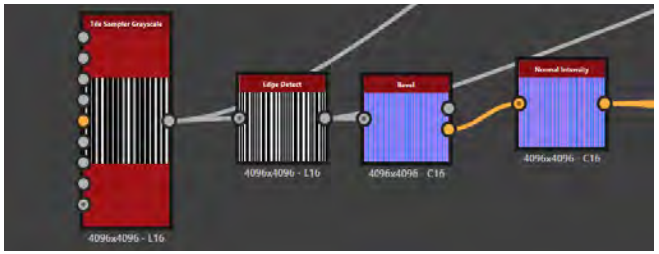
3.2 Textures nodales

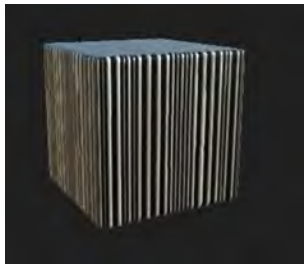
Non Extrudable

La texture non extrudable devait clairement être identifiable par le joueur et différente du béton tout en restant en niveaux de gris, pour s'intégrer naturellement dans notre choix de direction artistique. Nous avons donc réalisé sur Substance Designer une texture métallique géométrique.

PREMIERE PASSE

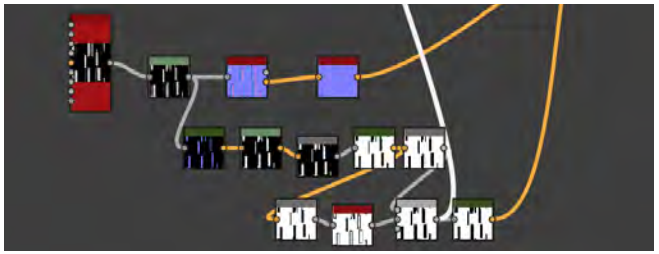
Cette texture est d'abord composée d'un tile sampler créant ces stries verticales. Ces stries sont beveled et reliées à la normal map.

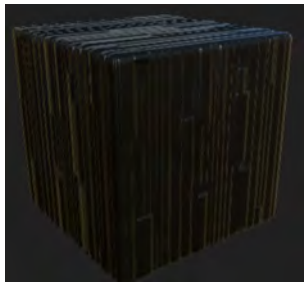




SECONDE PASSE

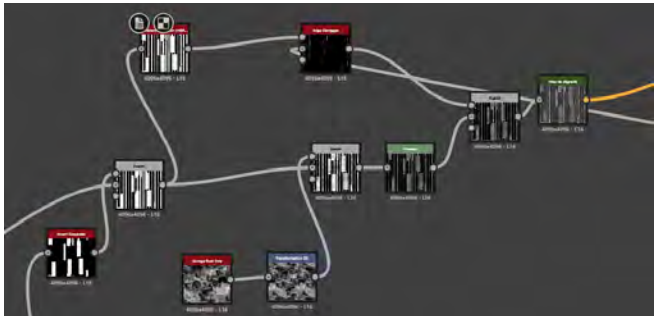
Un tile sampler place des rectangles aléatoirement sur la texture. Ils sont triés par niveaux de gris, puis la normal map est «aplatie» partout ou ces rectangles sont placés. Les rectangles sont ensuite beveled et placés dans ces slots.
Ainsi, on a des tries en relief, interrompues par des rectangles eux aussi en relief.

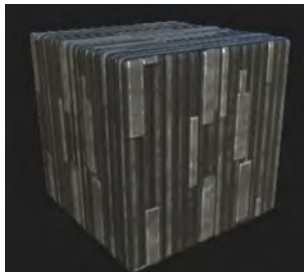




TROISIÈME PASSE

Pour la couleur, la partie intérieure des stries est assombrie, une grunge map et des edge damages sont ajoutés. Le tout est filtré à travers une map de dégradé pour égaliser les couleurs.





3. Direction artistique visuelle

3.3 Modélisation et Texturing des mains

Modélisation des mains et rigging

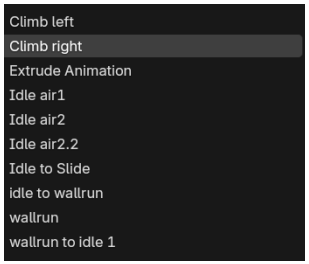
Les mains et leur Rig ont été réalisés sur Blender, tout comme leurs animations.

Le rig utilise différente techniques ; certaines parties sont déformées selon du Weight paint comme la main ou les tuyaux, d autres comme la machine ou l'arche au niveau du poignet sont rig en enveloppe weight.

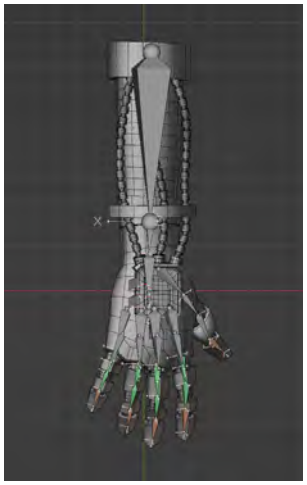
Le modèle 3D de la main comporte des discontinuités volontaires. La partie bétonnée du gant est moins dense que la main elle-même, afin qu'elle se déforme de manière plus artificielle.

Main seule

Modèle complet main + armure



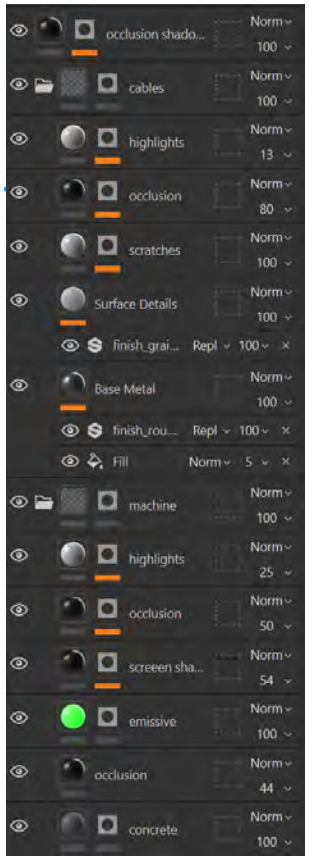
Liste des animations

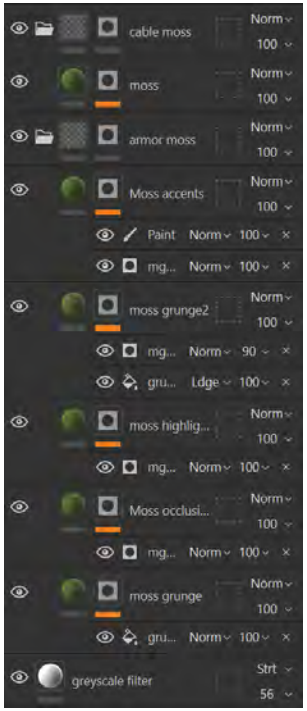


Rigging de la main

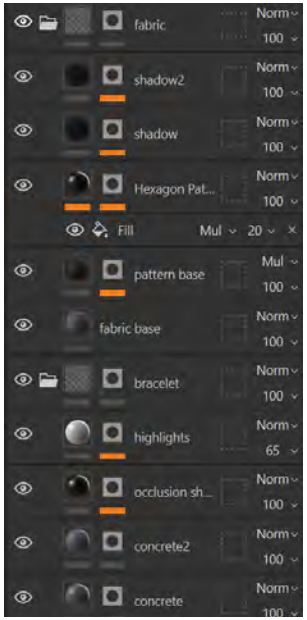
UV texturées

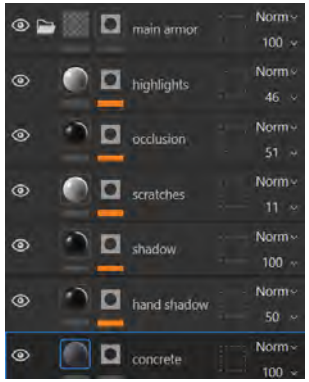
Main texturée





Layers de textures Substance Painter





Texturing

Les textures des mains ont été réalisées sur Substance Painter. On retrouve sur la partie « gant » de la main la texture de béton, et sur cette texture la mousse des piliers. La main respecte la règle de colorimétrie du projet: tout en grayscale, sauf une unique couleur, le vert.

3. Direction artistique visuelle

3.4 Animation des mains

Placement des mains

Nous ne pouvions pas placer les mains devant la Main camera, car celle-ci change de FOV pendant la partie et leur présence aurait brisé l'effet d accélération que le changement de FOV procure, car elles auraient également été déformées. Nous avons donc placé les mains à part, elles ont leur propre camera qui ne filme que leur layer.

Déplacement des mains

Les modèles 3D des mains animées sont placés en enfants de GameObjects vides, qui se déplacent et changent d'angle selon les actions.

Un GameObject ArmsManager gère ces positions. Ainsi, dès que le joueur effectue une action, les mains se déplacent en conséquent en Lerp à une position donnée et à une vitesse donnée.

Avant même d'être animées, les mains bougent donc à certaines positions à chaque actions.

Par exemple, lorsque le personnage court, les mains «bobbent» à la même fréquence que la caméra, avec plus d'amplitude, ou lors du wallrun, celles-ci vont se positionner sur le côté et tourner vers le mur, pour frôler ce dernier.
Chaque action (walk,extrude,air,slide,climb,wallrun) a sa propre position, rotation et vitesse de changement de main.

3. Direction artistique visuelle

3.4 Animation des mains

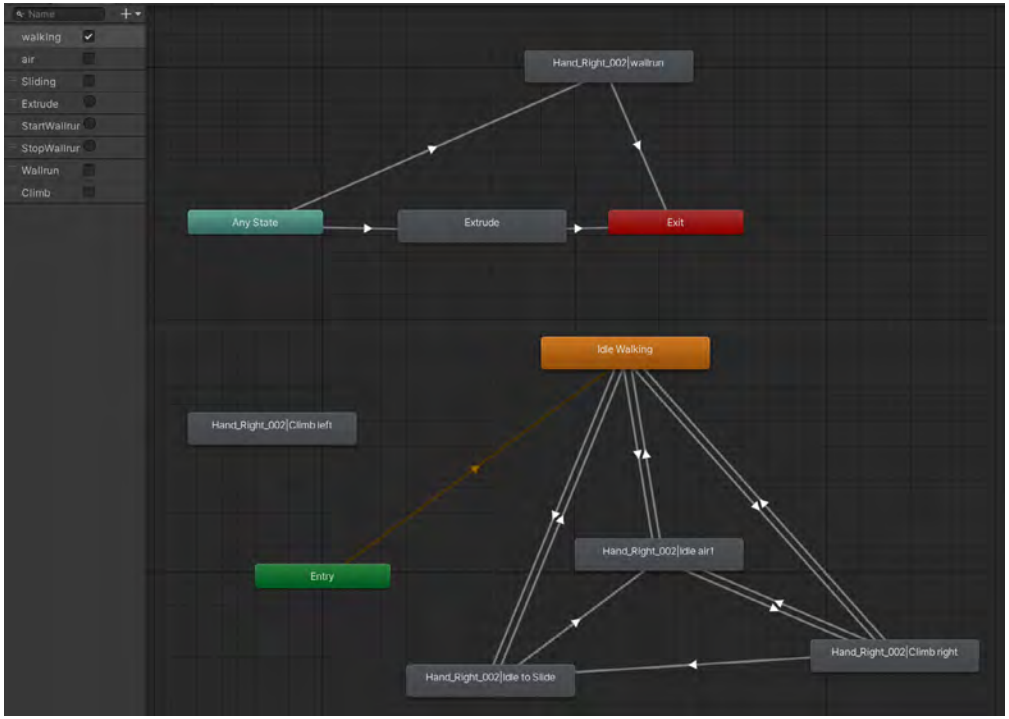
D'autres logiques sont ensuite appliquées. Par exemple, si le personnage Wallrun sur un mur à sa droite, la main droite longe le mur, et inversement s'il wallrun à sa gauche.
L'extrusion se fait de la main droite, mais lorsque le joueur wallrun sur sa droite, la main droite étant occupée, c'est la main gauche qui s'en charge.

La main gauche est rarement visible car nous avons choisi d'intégrer les mains à la manière d'un FPS, ou seule la main droite, qui tient habituellement l'arme, est visible. Ici, il ne s'agit pas d'une arme mais l'usage de la main est similaire.

Animation

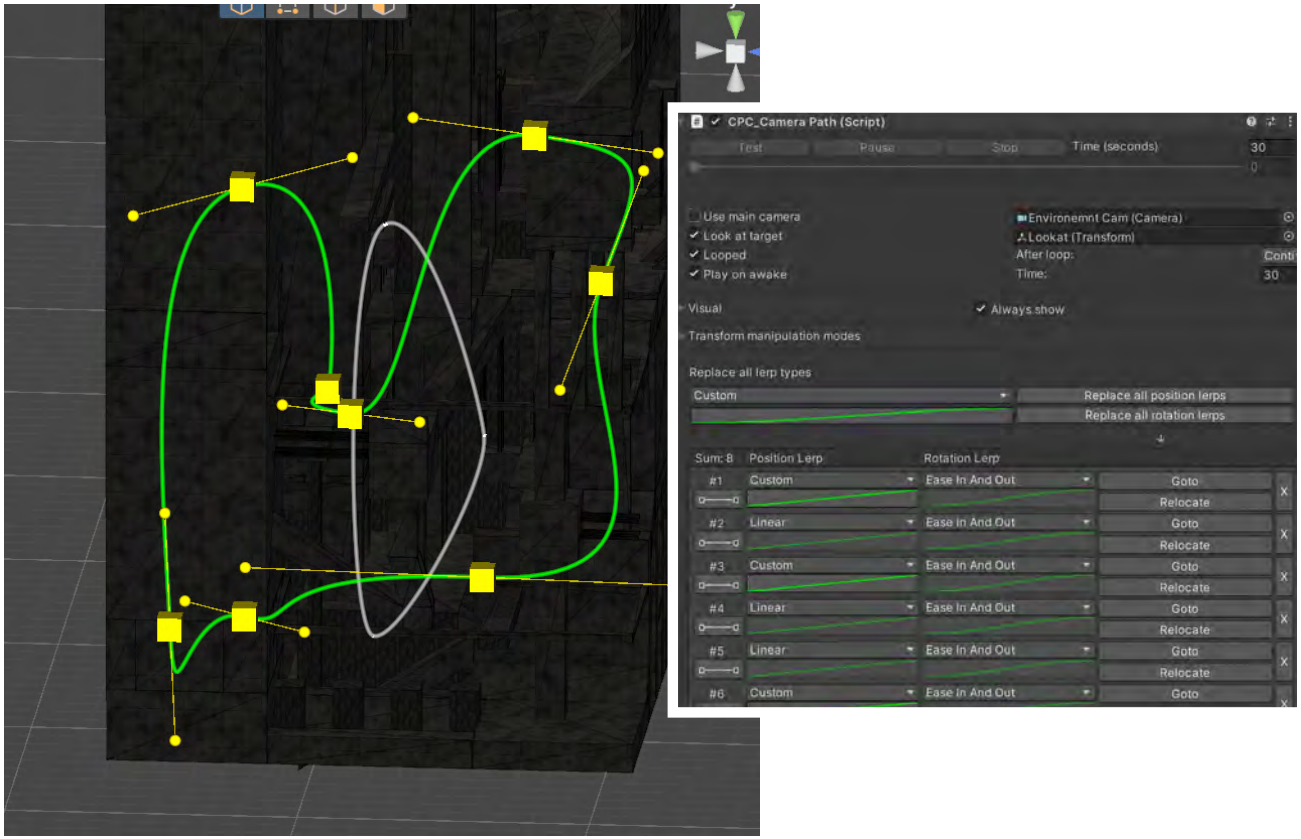
Les mains, en plus d'être déplacées par code, sont riggées et animées en utilisant l'animation controller de Unity.
L'animator dispose de plusieurs états correspondants aux états du joueur.

Les deux mains ont le même animator, mais l'animation d'extrusion en elle-même est toujours lancée sur la main droite seulement (sauf en cas de wallrun à droite).
Additionnellement, lors du climb, l'animation de la main gauche se lance en différé, pour donner l'impression que le personnage grimpe une main après l'autre.



4. Caméra du Main menu

Le fond du Main menu est une vue du niveau. Pour réaliser cela, nous avons utilisé un addon permettant de fixer la caméra sur un rail éditable ; ce rail est composé de courbes de bézier et différents paramètres sont modifiables comme la vitesse de la caméra à travers le rail, les courbes d'accélération, de rotation et ce que regarde la caméra.
La camera regarde donc un GameObject au centre du niveau, qui lui-même est sur un rail.



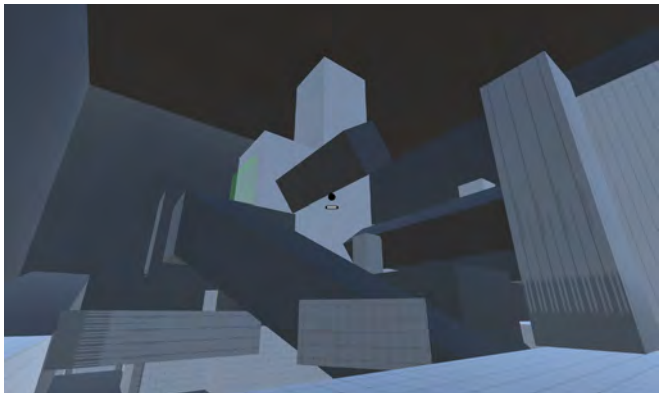
Evolutions du projet et améliorations potentielles

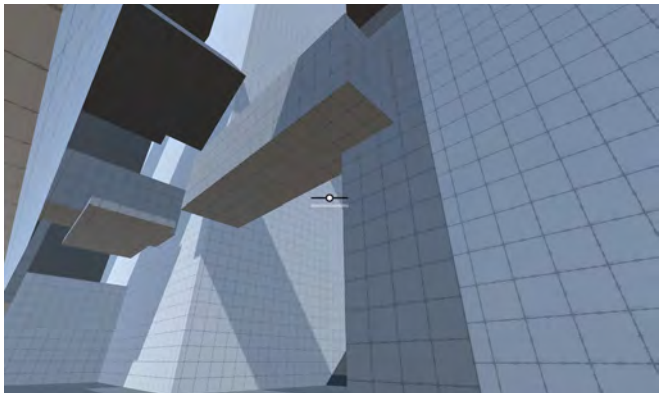
Builds du:

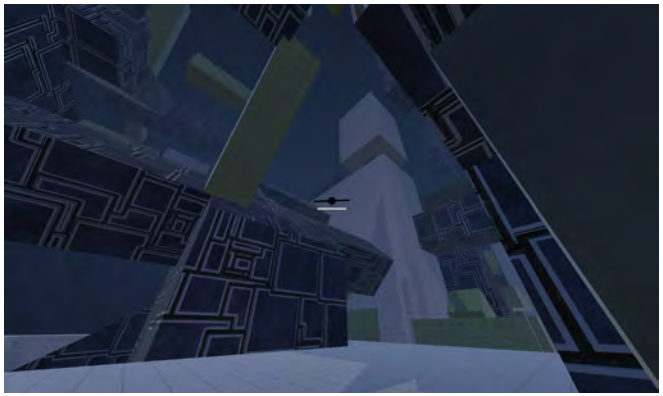
29.10.24 ..... 13.11.24

23.11.24 ..... 28.12.24

10.01.25 ..... 17.01.25







Évolutions et améliorations potentielles:

- Amélioration du système de flow pour aligner mieux le flow ressenti sur le flow in game.

- Amélioration du système de flow pour qu'il impacte plus le gameplay.

- Équilibrage de metrics plus avancé.

- Niveau additionnel de musique.

- Full Body pour le personnage.

Merci d'avoir lu !

Remerciment spéciaux a Nathan Miniere pour la musique, Nô , le corps ensignant de l'ICAN

Arthur (Haida) GUERRY , Aymeric LAVAL, Elliott SAUVENAY - 2GD - ICAN 2024-2025