I.I

Home sweet home.

Dossier de conception

Ouri Levin
Marius Ballot
Luc Miramont
Michael De Laborde

Sommaire

Concept

Idéation

Sujet

Le sujet à aborder était la « **mobilité connectée** ». Nous avions carte blanche pour imaginer un projet faisant référence à cette notion, sans partir dans le cliché des transport connectés.
Nous avons donc essayé de jouer avec les mots et nous sommes partis sur l'idée de la **mobilité virtuelle, et la notion d'immobilité connectée** en est aussi ressortie. Nous avons **réinterprété le thème** pour ouvrir des perspectives plus larges.

L'idée de parler d'Internet est donc venue naturellement, mais nous ne voulions pas le faire de manière banale. c'est donc pour l'option d'une histoire narrative et interactive que nous avons opté.

Idée

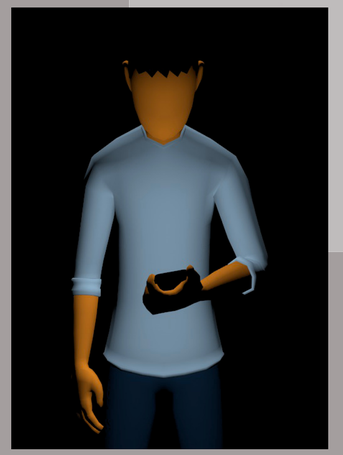
C'est en écoutant une musique du groupe **The 1975** (ci-contre) que l'histoire nous est apparue comme une évidence, puisqu'elle correspondait parfaitement au sujet que nous voulions traiter.

C'est donc vers la relation entre un homme seul et Internet que se tourne notre expérience. Dans un ton à la fois triste et merveilleux.

Nous souhaitions partager la même émotion que celle que nous avions ressentie lors de notre écoute du morceau.

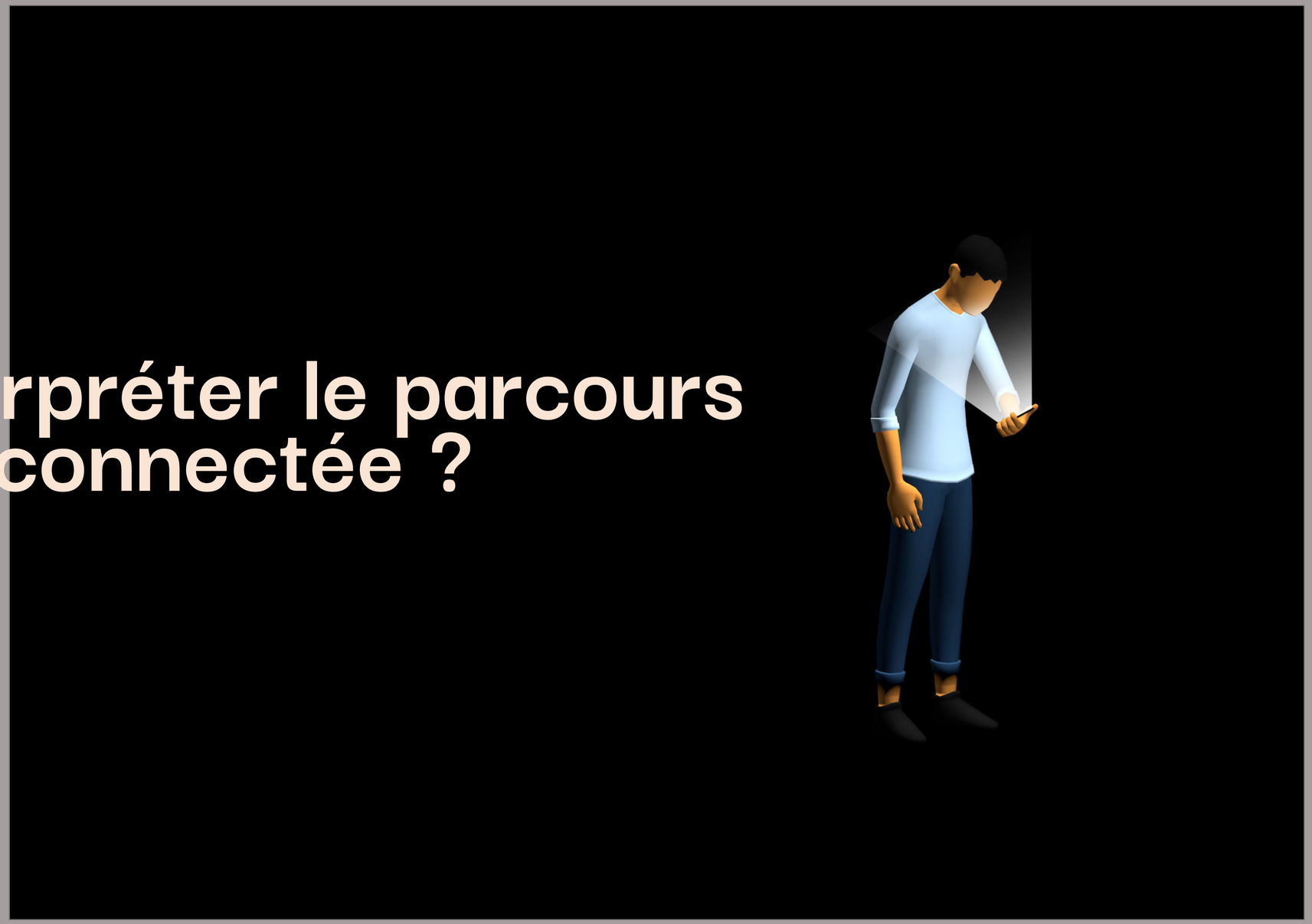
The 1975 Song

« The Man Who Married A Robot / Love Theme »



Problématique

Comment interpréter le parcours d’une vie sur-connectée ?



Cible et contexte

Cette expérience vise **un public très large.**
Elle peut, en effet, être vécue par toute personne étant intriguée par celle-ci. Elle est aussi destinée aux passionnés du Web

Cependant plusieurs contextes peuvent être imaginés pour son utilisation. Il serait possible d'envisager un **atelier de découverte lors d'évènements tech ou dans un salon de jeux vidéo.**

Une simple utilisation après recommandation est aussi imaginable, **simplement via le site et le téléchargement.**

Une expérience qui demande du temps.

Un moment pour échanger Et partager les points de vue.

Une réflexion à mener seul face à l'histoire.

Objectifs

Notre objectif est de **provoquer de l'émotion chez l'utilisateur.**
Une sorte de compassion pour le personnage.

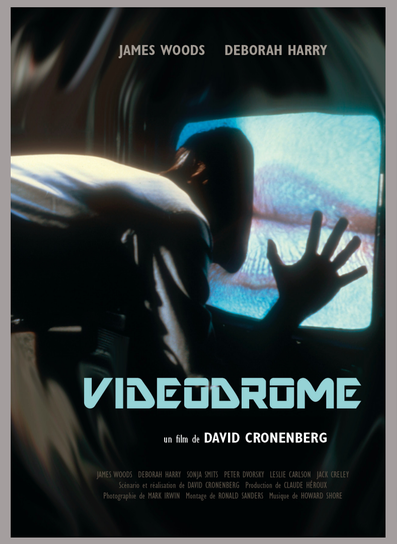
Cette expérience a pour but de bousculer l'utilisateur, de le **pousser réfléchir une fois l'expérience achevée.**

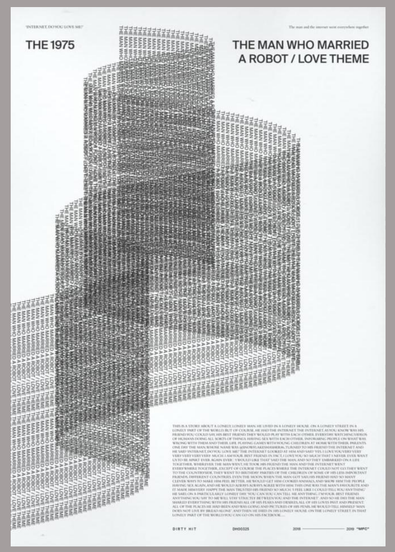
Intentions

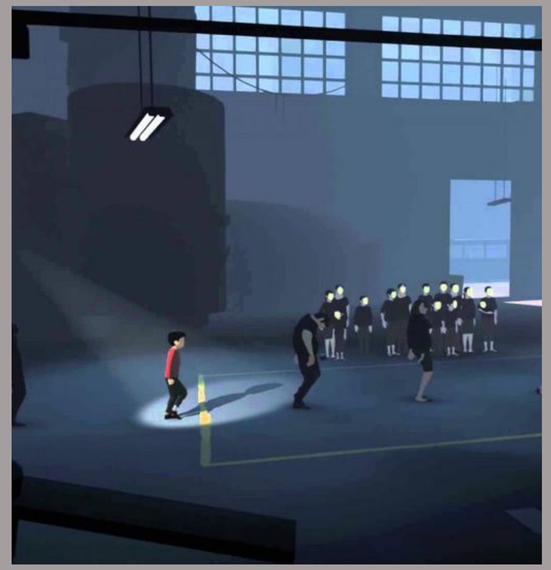
Inspiration

Nous avons listé toutes nos inspirations, en gardant les plus importantes. Chacune d'entre elles ayant servi à la construction et surtout à la cohérence globale du projet. Que ce soit d'un point de vue visuel ou de liens en terme d'histoire.

1/ HER - Spike Jones
2/ Blade Runner 2049 - Denis Villeneuve
Moon - Duncan Jones
Taxi Driver - Martin Scorsese
Tony Takiani - Jun Ichikawa
3/ Videodrome - David Cronenberg
Ex Machina - Alex Garland
4/ Le Daim - Mr. Oizo
Requiem for a Dream
5/ The 1975 - The man who married a robot
6/INSIDE - Playdead
The Long Dark - Hinterland Studio
Another World - Éric Chahi
Return to Obra Dinn - Lucas Pope
7/Firewatch - Campo Santo
Untitled Goose Game - House House







Scénario

Nous racontons l'histoire d'un homme seul, très seul, ayant comme unique ami Internet.

L'homme va tisser des liens avec Internet jusqu'à en tomber amoureux. Nous allons donc à travers toute l'experience, assister à la montée fulgurante de cet amour inconditionnel.

C'est donc à travers 6 étapes que nous allons découvrir cette relation. En effet la totalité de l'histoire sera séparée en deux points de vue :

- Un point de vue isométrique décrivant ses conditions de vie, qui deviennent progressivement plus funestes;

- Un point de vue subjectif "miroir", ancré dans son imaginaire;

Début de l'expérience

Lieux du monde réel (Visible sur desktop)		Lieux du monde virtuel (Visible sur mobile)
Appartement le matin	1	Jardin (avec chien)
Rue en début de journée	2	Parc, Forêt (avec oiseaux)
Appartement à midi	3	Environnement abstrait
Métro	4	Désert (avec dromadaire)
Rooftop au coucher du soleil	5	Montagnes enneigées (avec moutons)
Appartement le soir	6	Lune (avec la Terre)

Mort du personnage

Narration

Toute la narration de l'histoire est évidemment inspirée de la musique
« The Man Who Married a Robot / Love Theme » de The 1975.

Nous avons naturellement utilisé **une voix synthétique** pour la totalité des interventions narratives.

Direction Artistique

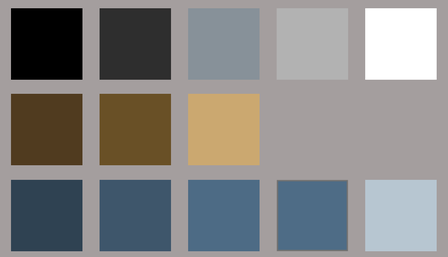
Identité graphique

Typographie

Darker Grotesque Bold
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

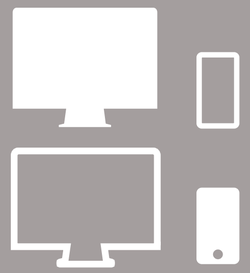
Darker Grotesque Medium
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

Typographie



Nous avons choisi des couleurs assez proches pour laisser la lumière agir sur les environnements 3D.

Interface graphique



Notre interface est très simple pour favoriser l'immersion dans les décors.

Logo

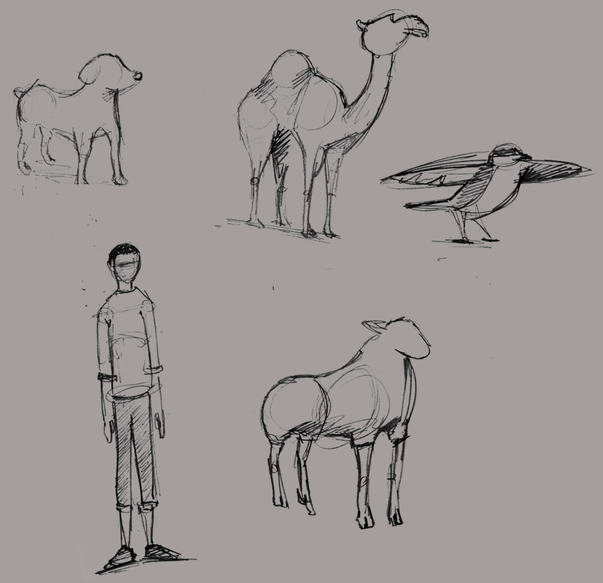
Home sweet home.

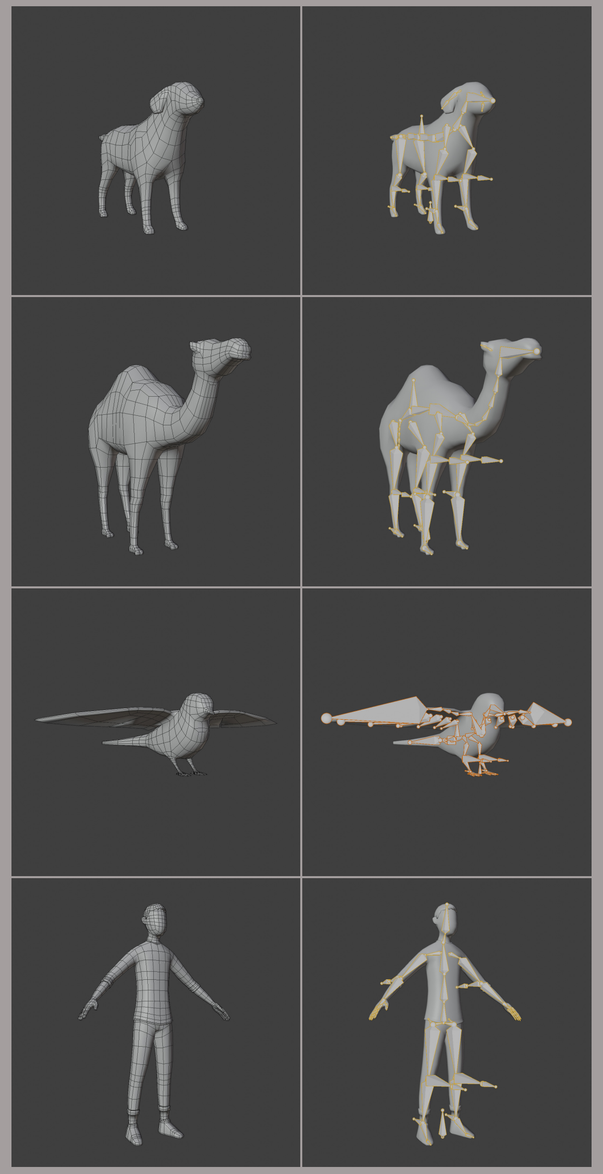
I.I

Nous avons deux logos, le premier est simplement fait de typographie et le deuxième est une représentation simplifiée de l'histoire racontée dans notre experience : un individu partagé entre deux mondes.

Character design

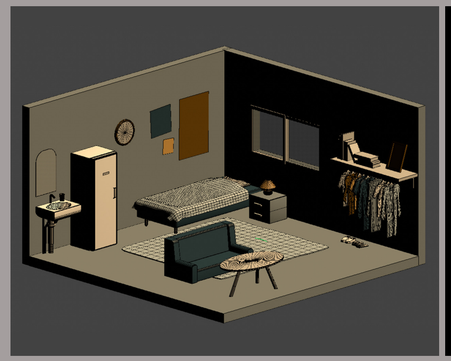
Les 7 modélisations sont en low-poly. La palette de couleur est appliquée sur chaque personnage afin d'avoir une cohérence globale.

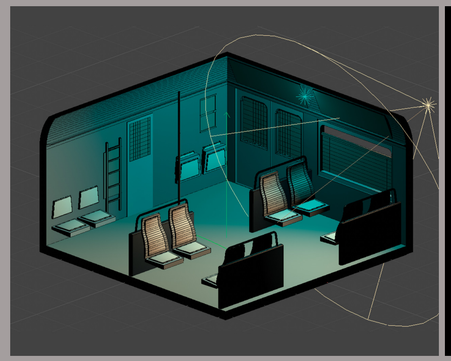
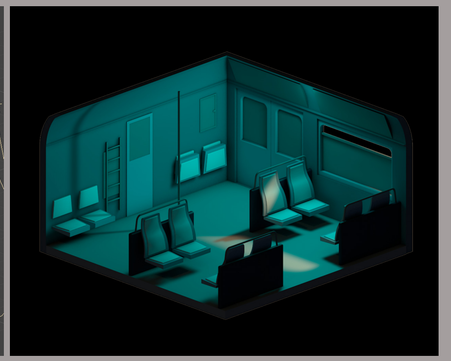
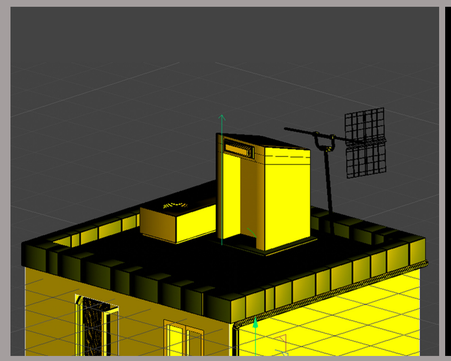
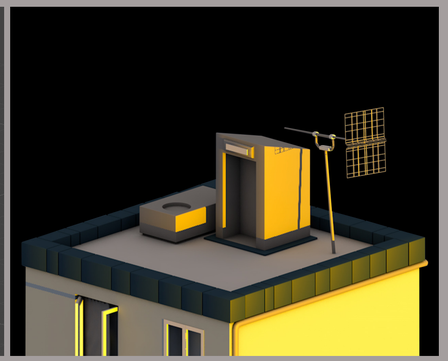
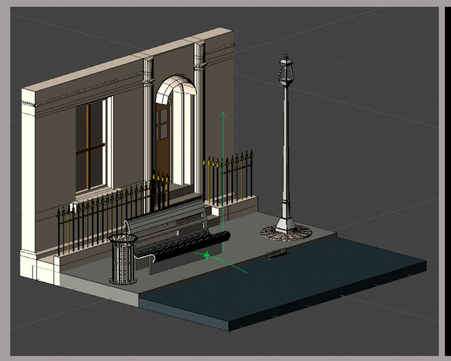
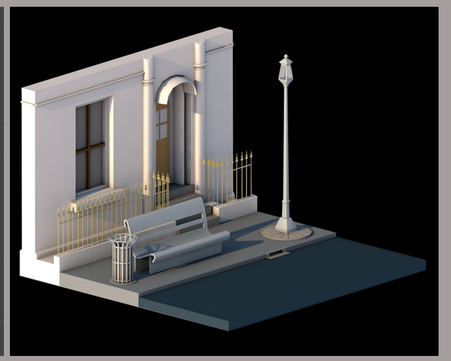


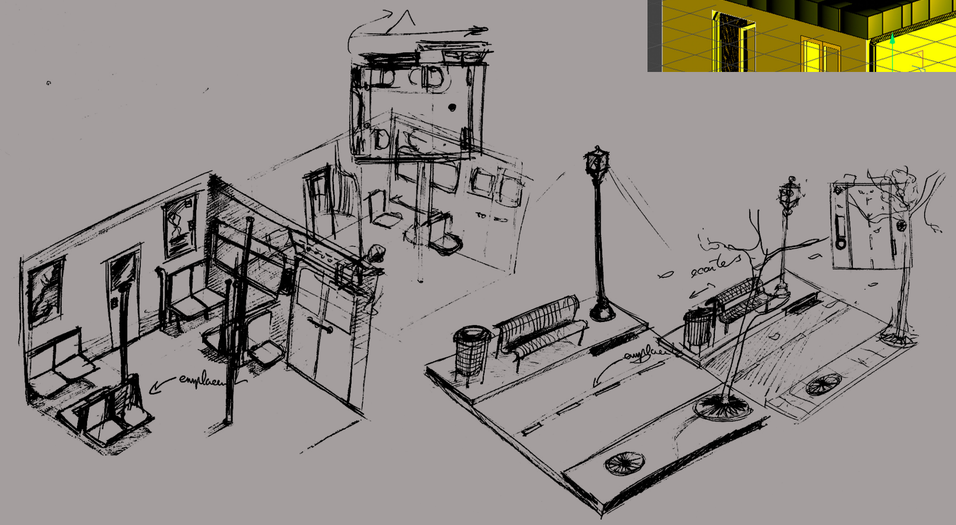


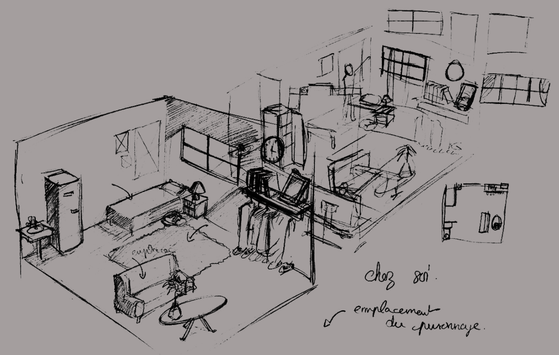
Décors monde réel

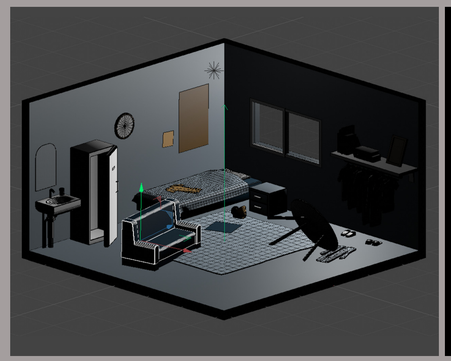
Les lieux de vie où le personnage va évoluer, avec son téléphone à la main. La vision isométrique permet de garder une vision restreinte et d'accentuer le sentiment d'isolement du personnage. Une fois la palette de couleur appliquée, c'est la lumière qui va donner vie aux couleurs.









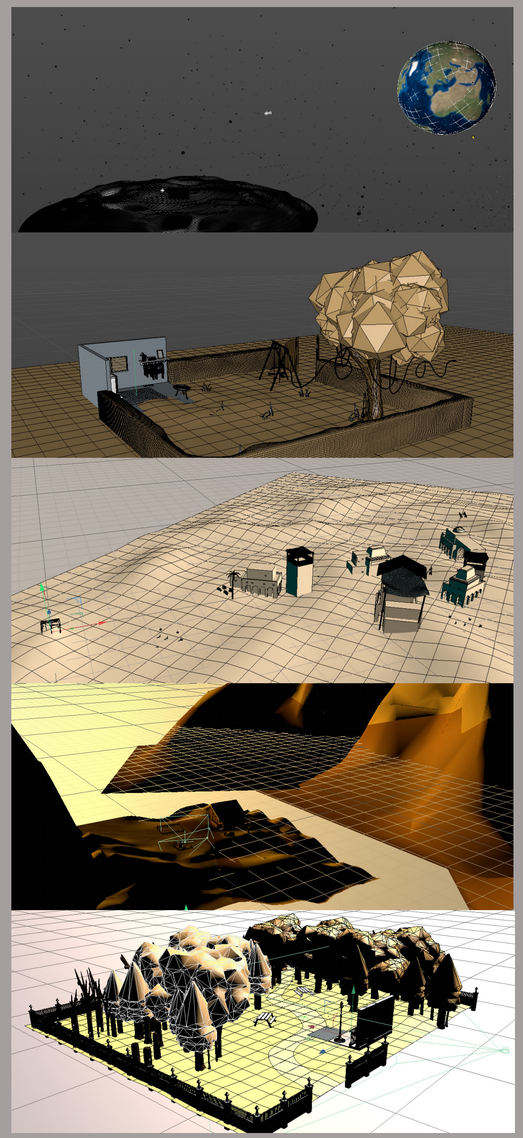

Décors monde virtuel

Les décors virtuels reprennent toujours la 3D du monde réel pour y intégrer une sorte d'extension plus belle. Nous avons décidé de préserver la scène afin que l'utilisateur puisse garder ses repères dans l'espace.

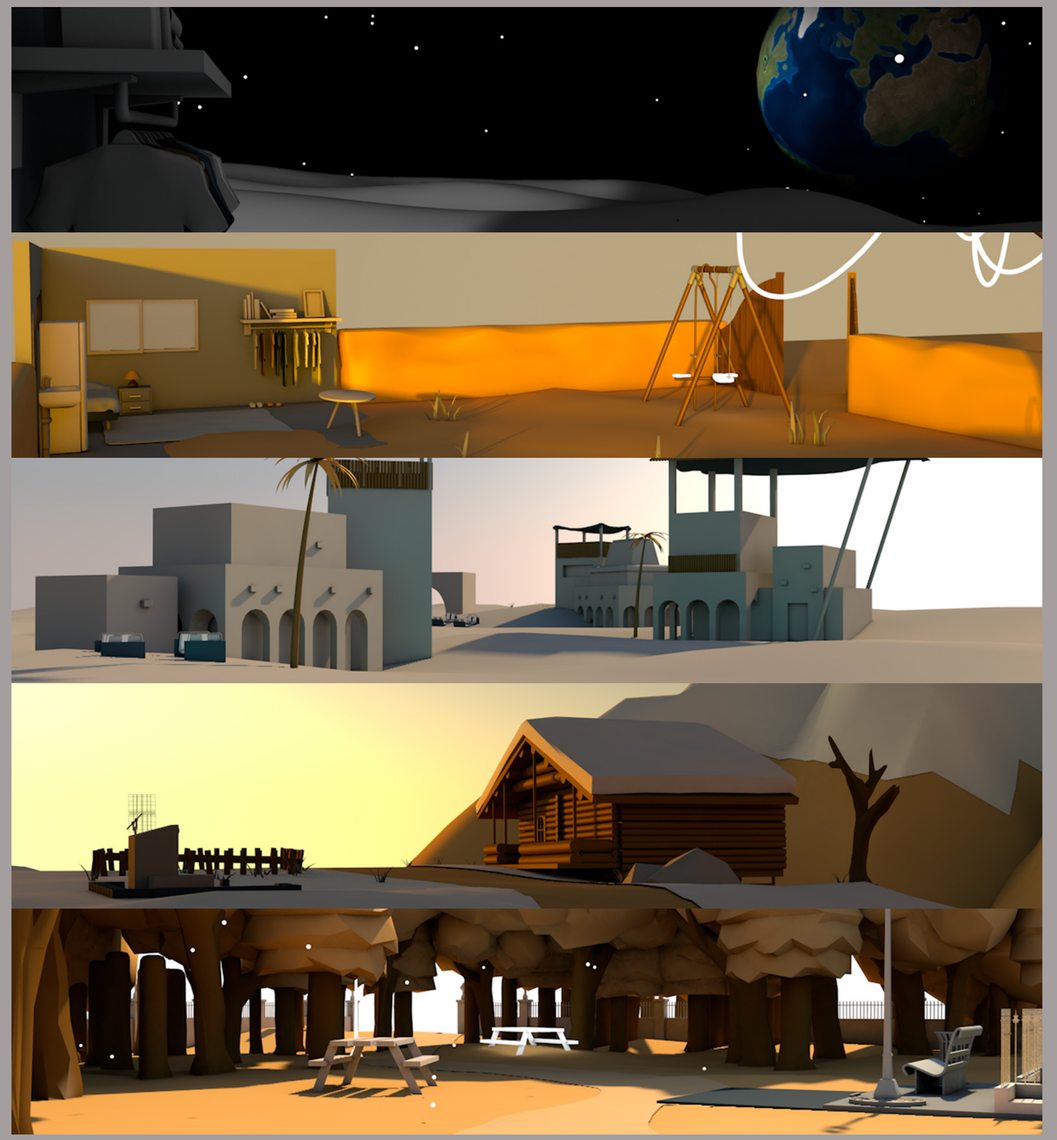
Les modélisations 3D sont par conséquent plus vastes, mais utilisent toujours la même palette.

Afin de marquer une difference entre les deux univers (Réel & Virtuel), un filtre blanc sera ajouté sur l'UI de la vision Virtuelle.
Dans le but de faire comprendre que le monde en question n'est pas reel mais n'est qu'une illusion.

Modélisation

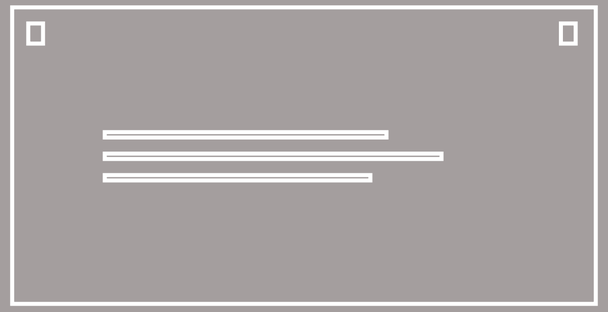


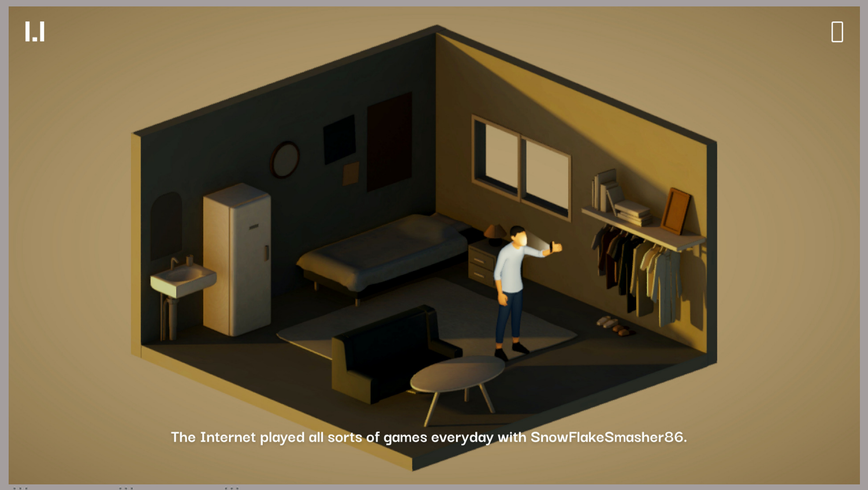
Rendu



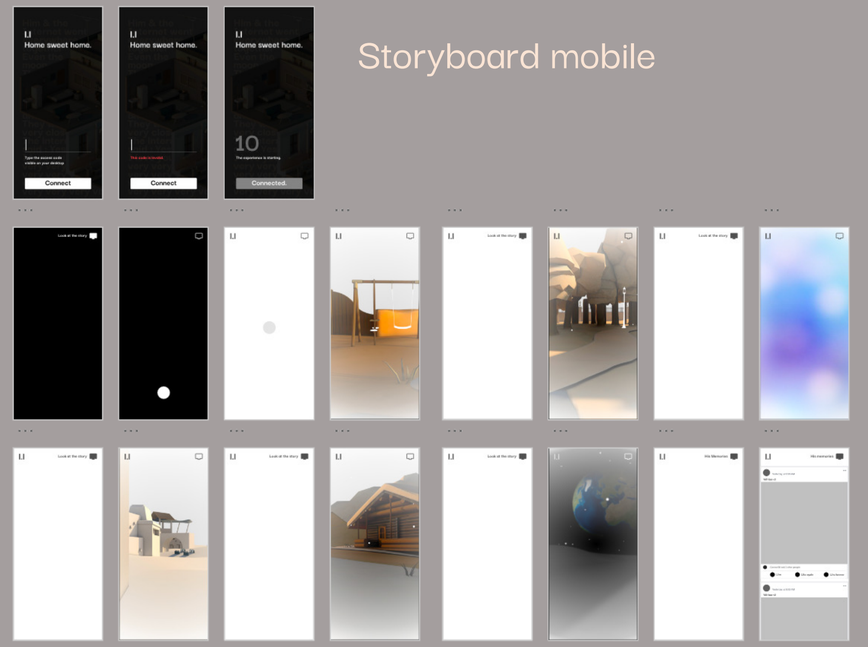
Storyboard

Les storyboards visibles ci-contre sont des guides pour les designers et développeurs, Il définit l'UI et ses evolutions, autant en terme de pictogramme que de texte ou narration.





Storyboard desktop



Storyboard mobile

Sound design

La musique du projet est une musique ambiante. Le logiciel utilisé pour sa création est FIstudio.
Elle accompagne le personnage à travers toute l'histoire. À chaque nouvelle scène, de nouvelles mélodies viennent s'ajouter à la composition orchestrale, jusqu'au « grand final ».

Nous avons d'abord séparé la musique en plusieurs sections, avec des marqueurs pour bien les distinguer. Chaque section devait pouvoir se répéter, sans que celle-ci soit entendue.

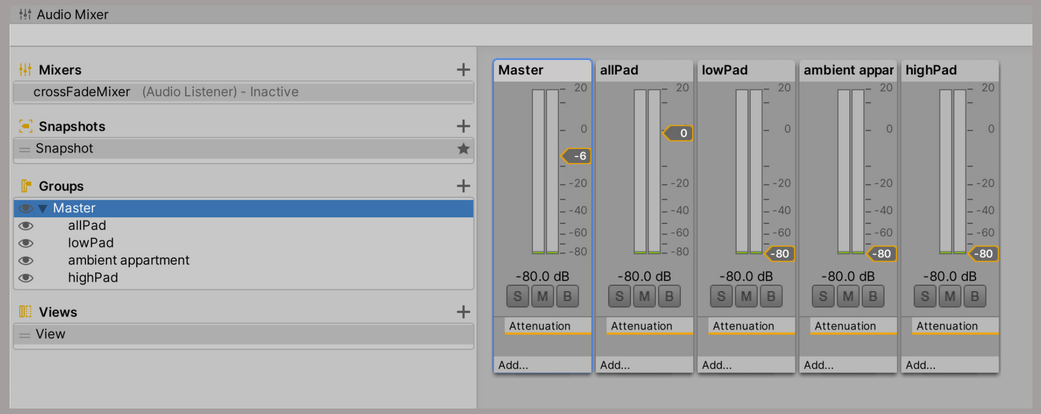
C'est pourquoi nous avons choisi que la plupart des sons, y compris la musique, seraient émis depuis l'ordinateur, qui a le plus souvent, une qualité audio supérieure.

Afin de s'assurer que l'utilisateur puisse balayer les deux écrans de manière fluide et naturelle, nous avons créer un contraste sonore suffisamment fort pour attirer son regard vers l'écran opposé, sans pour autant le sortir de l'expérience.

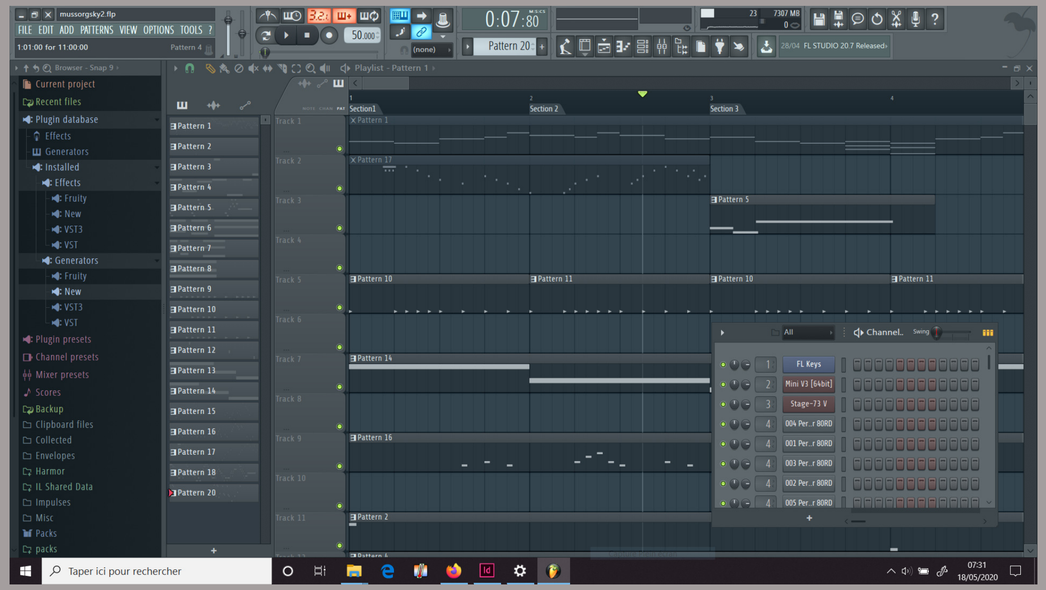
Nous avons décidé d'utiliser des bruitages concrets, qui correspondent aux éléments auxquels ils sont associés. Par exemple, pour attirer le regard sur l'action d'un oiseau, la notification sera le chant de celui-ci. Pour accentuer cela, le volume de ces bruitages est considérablement plus fort que le reste.

Après plusieurs tests, nous avons aussi décidé de laisser à l'utilisateur suffisamment de temps pour passer d'un écran à l'autre sans rater le déroulement de l'action. L'utilisateur a besoin d'un délai d' 1 à 3 secondes. Lorsqu'un évènement est déclenché, l'animation est retardée, tandis que le son se joue tout de suite.

Audio Unity

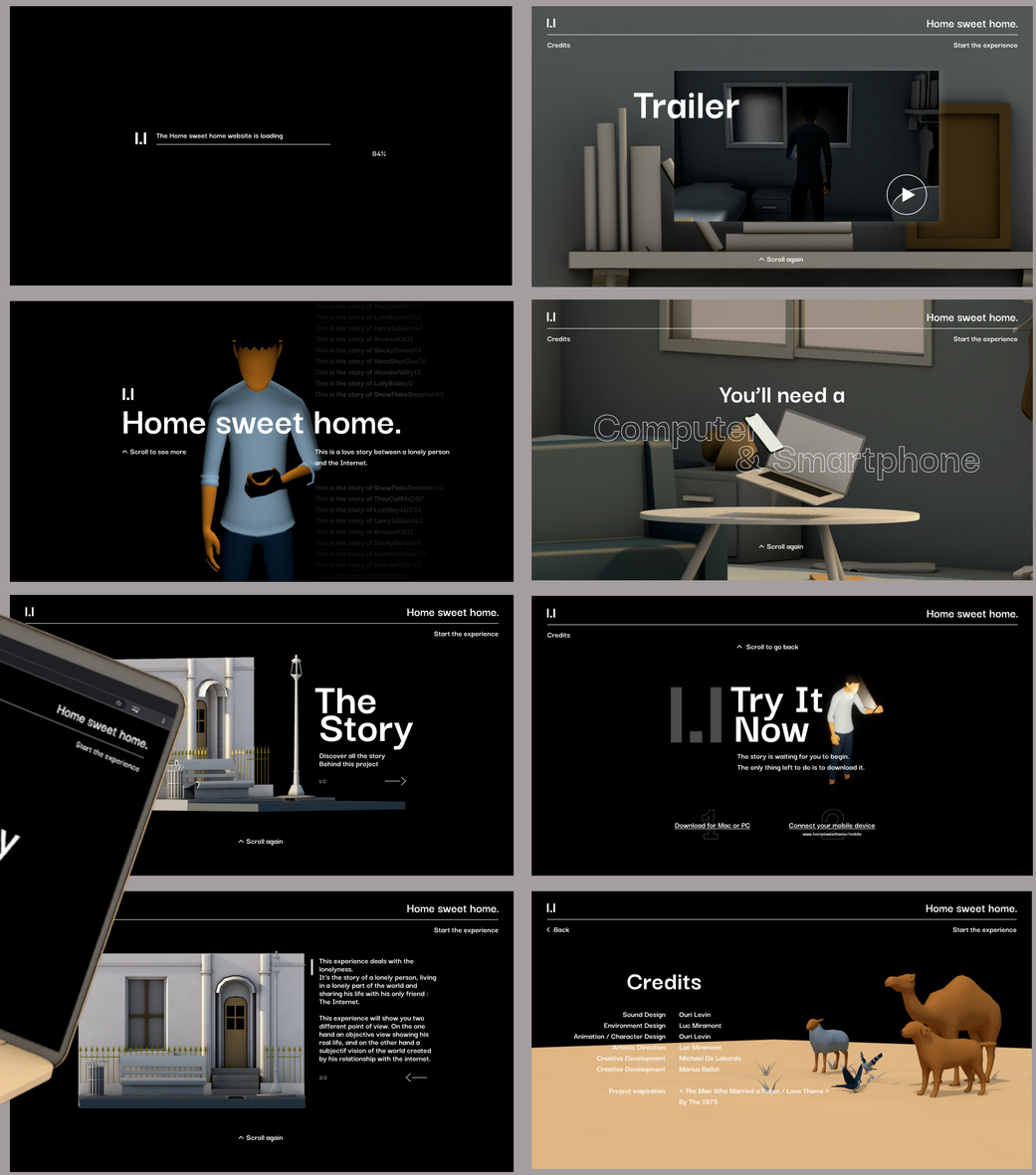


FL STUDIO



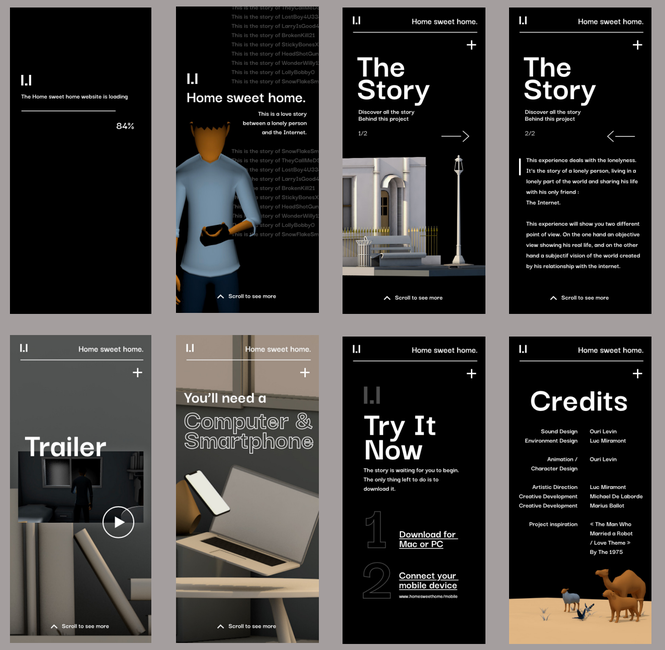
Site web

Le site web a pour but de faire comprendre à l'utilisateur l'univers de l'experience intéractive tout en donnant envie de l'essayer.
Ci-contre sont présentées les maquettes.



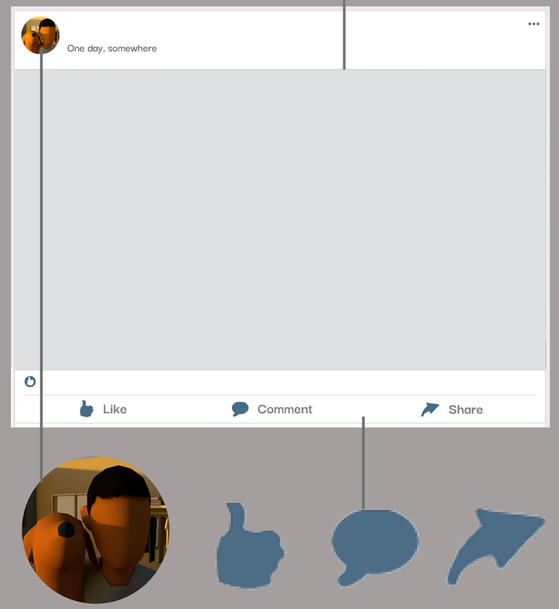
Le site internet comprend nombre d'éléments 3D interactifs extraits de l'experience afin d'appuyer toutes les informations données.

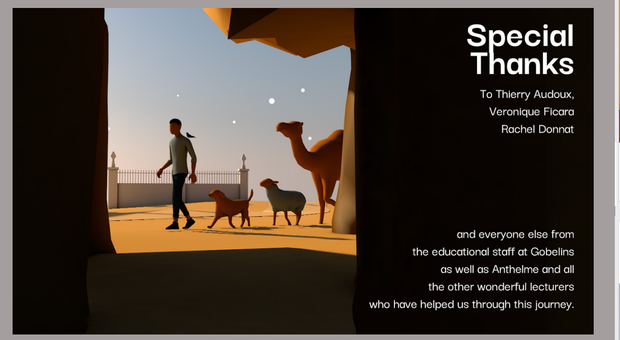
Nous avons choisi de créer un site assez court pour ne pas trop en dévoiler sur l'expérience mais surtout pour favoriser comprehension claire du thème menant vers un téléchargement rapide.

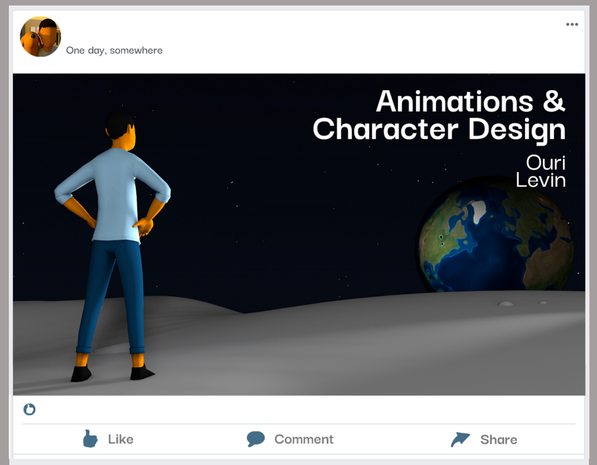


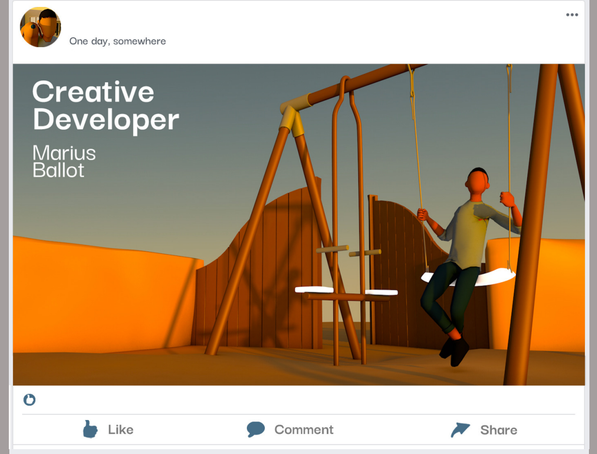
Credits

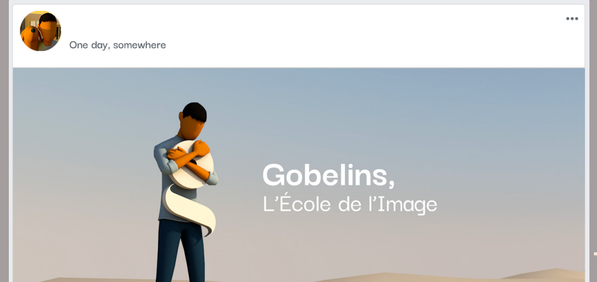
Le générique de l'experience est fait exclusivement de souvenirs du personnage principal qu'il aurait laissés sur ses réseaux sociaux. Ceci expliquant le style du générique fait exclusivement de publications.











Développement

Introduction

Les spécificités de notre projet ont nécessité la concrétisation d'un stack technique innovant.

En effet, nous souhaitons montrer simultanément et interactivement les deux points de vue de notre personnage.

Nous avons choisi de représenter la première dans une application **Unity** tournant sur un ordinateur et la deuxième dans un site web conçu en **VueJS** et **ThreeJS**, destiné à être visité avec un smartphone. Nous rentrerons plus en détail sur ces choix par la suite.
Ayant besoin de relier deux devices en temps réel pour synchroniser le déroulement de l'expérience, interagir avec les capteurs du smartphone et établir un système de codes d'accès, nous avons employé extensivement la technologie **Websocket**.

Unity

Quoi ?

Unity est une plateforme de développement en temps réel. Initialement conçue comme un moteur de jeu facile à prendre en main, Unity est maintenant utilisé pour réaliser des projets en 3D, 2D, Réalité Augmentée et Réalité Virtuelle dans des champs aussi divers que la visualisation d'architecture ou l'animation.

Pourquoi ?

Plusieurs facteurs sont venus influencer notre décision :

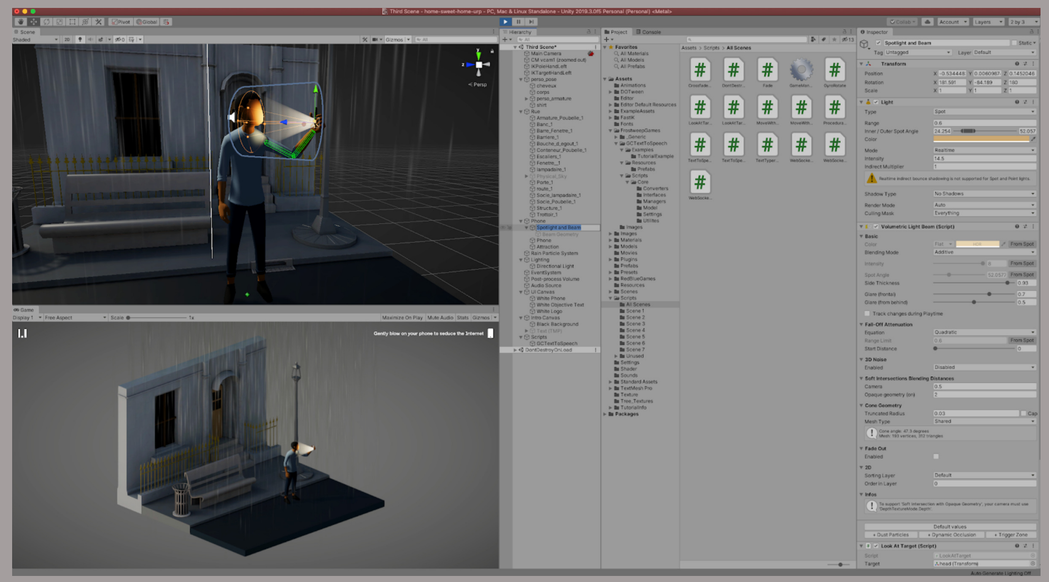
1/ Notre envie d'en apprendre plus sur cette plateforme prépondérante dans les métiers de l'interactif.

2/ La grande disponibilité d'informations, tutoriels, et documentation sur le Web.

3/ Des bonnes performances et une prise en main plus aisée comparées à du WebGL, ce qui nous a permis de viser un bon rendu en peu de temps.

Comment ?

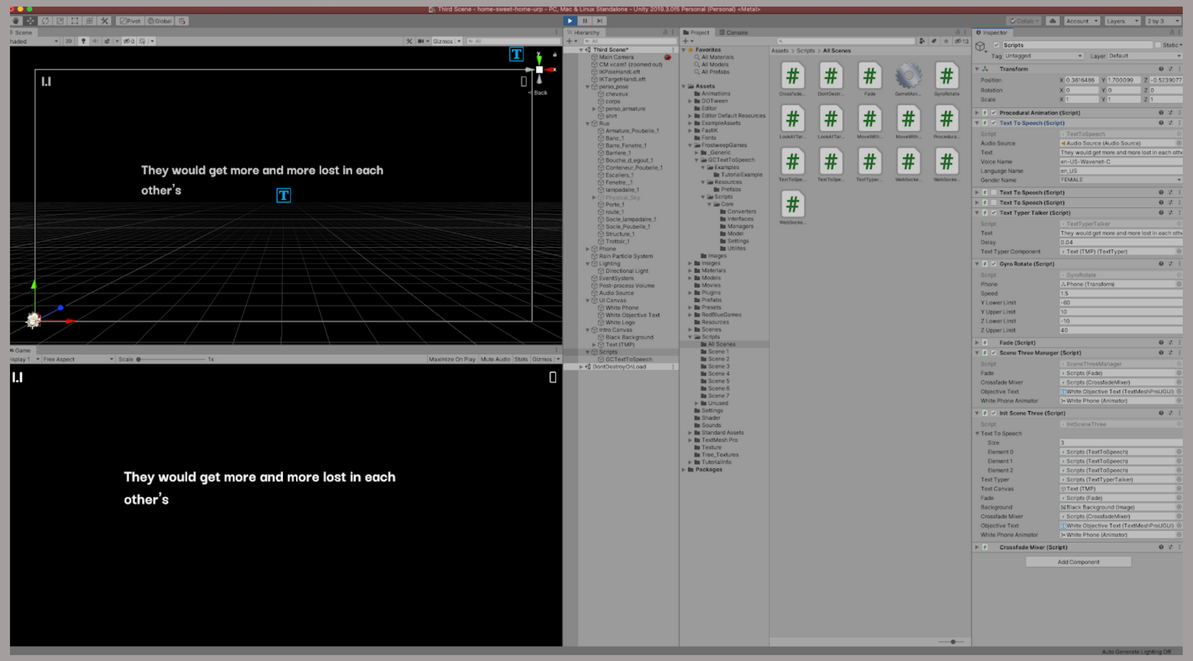
Le projet tourne sous Unity 2019.3.0f5 et nous avons choisi l'Universal Render Pipeline pour des raisons de performance, car ayant initialement essayé la pipeline de rendu de haute définition (HDRP), celle-ci n'a pas apporté beaucoup d'avantages à notre projet, et a un grand coût de performance, instabilité et complexité.



L'animation du personnage tenant son portable sur la vision isométrique a été très complexe à mettre en place. Nous avons finalement opté pour l'utilisation d'Inverse Kinematics à travers le plugin "Fast IK" avec une implémentation personnalisée pour gérer les valeurs d'angles provenant du gyroscope de la partie smartphone.
Notre pipeline n'offrant pas de lumières volumétriques, nous avons du opter pour le plugin Volumetric Light Beam pour simultanément éclairer le visage du personnage à travers l'écran de son smartphone et montrer la lumière provenant de celui-ci.

Unity

Pour la réalisation de la voix narrative,
nous nous sommes inspirés de la chanson
”The Man Who Married A Robot / Love Theme“ du groupe The 1975.
Après avoir essayé plusieurs solutions de Text-to-Speech (la chanson en question utilisant Siri pour la narration), Nous avons employé la plateforme Google Cloud Text-to-Speech, à travers un plugin Unity réalisé par FrostWeepGames.



Certains effets visuels, comme cette disparition / apparition, ont été faits avec l’aide de Shader Graph.

Websocket

Quoi ?

WebSocket est une technologie évoluée qui permet d'ouvrir un canal de communication bidirectionnelle entre un navigateur (côté client) et un serveur.
Avec cette API nous pouvons envoyer des messages à un serveur et recevoir ses réponses de manière événementielle sans avoir à aller consulter le serveur pour obtenir une réponse.

Pourquoi ?

Les deux grands avantages de Websocket pour notre projet sont les mêmes qui justifient son utilisation très répandue auprès d'autres projets interactifs sur le Web:

1/ Très bonnes performances, non seulement en raison de la petite taille des messages échangés entre le serveur et le client, mais aussi de l'utilisation d'un système d'événements qui nullifie la nécessité de poll le serveur.

2/ Prise en main facile, dû à une API simple, une documentation claire et des solutions faciles à trouver.

Comment ?

La pièce centrale de notre communication Websocket est notre serveur **Node.js**, hébergé sous l'adresse https://home-sweet-home--ws.herokuapp.com/.

Notre implémentation Unity envoie des messages codés en binaire, nous avons donc dû décoder ces derniers puis utiliser une Regex pour retrouver les informations envoyées,
L'application **VueJS** envoie des messages au format JSON au serveur.

La gestion des access code est exclusivement faite côté serveur, pour empêcher des manipulations néfastes côté client.
Pour ce faire, nous avons imaginé une solution complètement personnalisée, sans librairies ou bases de données.

Web dev et tool building

Programmer sans interface :

Programmer pour le web, dans l'histoire, se composait juste de 3 grandes parties. HTML pour le contenu et la segmentation d'une page, autrement dit le squelette; CSS pour le style de notre contenu, la peau, couleur des yeux pour continuer sur cette métaphore et enfin le JavaScript, le système nerveux.
Programmer pour le web était rapide, simple de compréhension et ne nécessitait pas de planification, d'architecture, d'optimisation ou autre. C'est pourquoi développer pour le web se fait sans interface.

Le problème étant que le web a énormément évolué. JavaScript est devenu plus complet, des pré-processeur sont là pour remplacer le CSS, un tas de framework sont disponibles et bien sûr l'arrivée fracassante de WebGL, un dérivé de OpenGL utilisé notamment pour des moteurs de jeux vidéos, software de modélisation et simulation 3D. Cela a repoussé les limites de ce qui était possible sur le web, mais les processus de programmation, eux, restent les mêmes. Un ordinateur, un éditeur de texte, un clavier et point.

Pas de prévisualisation, pas de gestion de models, pas de end-points, pas d'asset center, pas de garbage collector. Juste du texte, compilé directement à la vue finale.

Développer des projets/expériences créatives est donc re-développer à chaque fois de zéro un moteur graphique modulable et adapter à une tâche bien spécifique.
Pour éviter de perdre du temps à tout reconstruire à chaque fois, nous créons des petits morceaux de codes : des objects, functions, classes qui sont ensuite réutilisés comme des Légos dans d'autres projets.
C'est le principe du tool building.

L'importance du tool building

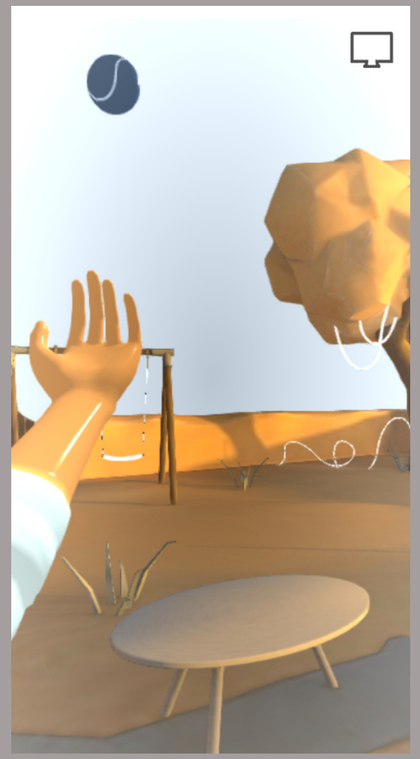
Comme stipulé plus tôt, le tool building nous permet de travailler plus efficacement, tout en restant adapté au scope de base. Cela consiste à créer de petites briques d'ingénierie, pour ensuite les brancher à notre projet.
Le tool building n'est pas seulement un moyen de ré-utiliser du code, mais également de faire évoluer le tool en question à chaque nouvelle instanciation.
Un tool n'est pas seulement un morceau de code mais une architecture avec des variables dynamique modifiable au moment du clonage.

La prochaine partie se concentre sur quelques-uns de ces tools pour la partie **Three.js** de Home Sweet Home.

Quelques tools

Physics Engine

Sweet Home possède de multiples interactions utilisateurs. Certaines de celles-ci, couplées avec le mouvement de caméra contrôlé par l'utilisateur, doivent être intégralement génératives.



C'est le cas de la scène numéro 1 où l'utilisateur doit lancer une balle dans un jardin. Pour ce faire, nous pouvions tout simplement baker l'animation mais cela aurait enlevé beaucoup de caractère à l'expérience. C'est pourquoi nous avons opté pour l'utilisation de **cannon.js**, un moteur physique pour JavaScript.

Le problème de cannon.js est sa verbosité lors de son utilisation. Par exemple, créer une simple sphere avec des propriétés physiques par défaut revient à écrire ce morceau de code : (en plus, évidemment, du setup de la librairie)

```
this.world = new CANNON.World()
this.timeStep = 1 / 60
this.world.gravity.set(0, -9.18, 0);
this.world.broadphase = new CANNON.NaiveBroadphase();
this.world.solver.iterations = 10

let sphereParams = mesh.geometry.parameters
let groundMaterial = new CANNON.Material();
groundMaterial.friction = .4;
bodyObj.body = new CANNON.Body({
    mass: bodyObj.mass,
    shape: new CANNON.Sphere(sphereParams.radius)
});
bodyObj.body.angularDamping = 0.8
bodyObj.body.material = groundMaterial
bodyObj.body.position.copy(mesh.position)
bodyObj.body.quaternion.copy(mesh.quaternion)

update() {
    this.world.step(this.timeStep);
    bodyObj.mesh.position.copy(bodyObj.body.position)
    bodyObj.mesh.quaternion.copy(bodyObj.body.quaternion);
}
```

Ce qui n'est pas très viable sur le long terme avec plusieurs dizaines d'objets a implémenter.

C'est pourquoi nous avons crée ce tool : https://github.com/MariusBallot/home-sweet-home-WebGL/blob/master/src/classes/PhysicsEngine.js

Quelques tools

Ce qui transforme le précédent code comme ceci :

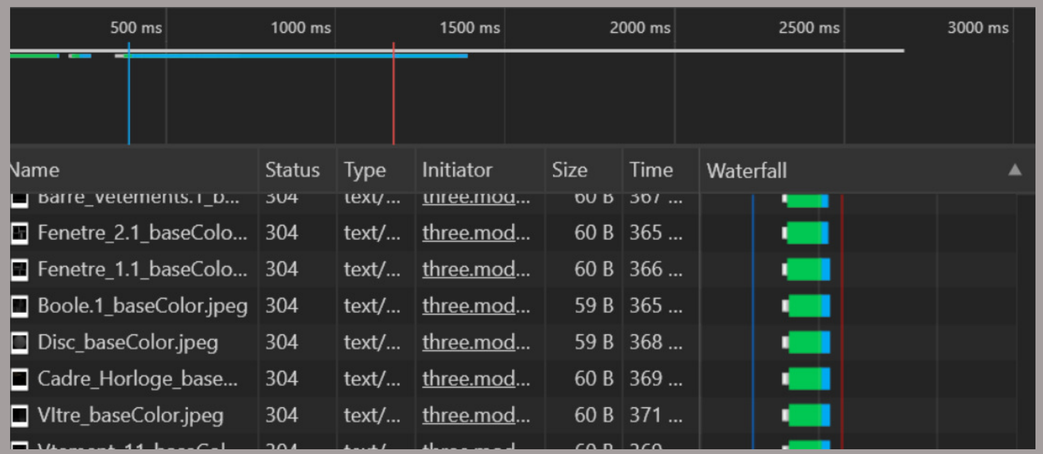
```
1 import PhysicsEngine from './PhysicsEngine'
2
3 PhysicsEngine.start()
4
5 let sphereBod = PhysicsEngine.addBody({
6     name: "Scene0Ball",
7     type: "sphere",
8     mesh: this.sphereCol,
9     active: false
10 })
```

Ce tool d'environ 140 lignes nous a permis de réaliser des scènes physiques fonctionnelles en un temps record. Évidemment ce tool n'est pas sans bug et, comme dis plus tôt, il va énormément évoluer avec le temps.

Comme tous les autres tools que nous verrons, celui-ci est disponible en open source et aidera plus d'un a l'utilisation de cannon.js.

Model manager

La 3D étant un très gros topic de notre projet, il nous fallait un moyen clair et efficace de pouvoir générer nos models 3D.



Dans l'ordre, le tool devait remplir ces points :

1/ Attendre le loading de tous les assets avant démarrage de l'app.

2/ Permettre de savoir quel objet est chargé et à quel moment.

3/ Avoir un gestionnaire de callback pour pouvoir utiliser cet event à plusieurs endroit dans l'application

4/ Disposer tous les models et informations de ces derniers dans un objets accessible a n'importe quel endroit dans l'appli.

Quelques tools

Le LoadingController permet de répondre à tous ces points. Il utilise à sa base le LoadingManager de Three.js et nous y avons implémenté tout ce systeme de gestion.

Il suffit d'y plugger un modelLoader, puis utiliser le controller selon nos besoins.

```
import LoadingController from "@/classes/HomeClasses/LoadingController";

let loader = new GLTFLoader(LoadingController.manager)

LoadingController.addOnLoad("homeOnLoad", () => {
    //Do a little some some
});
```

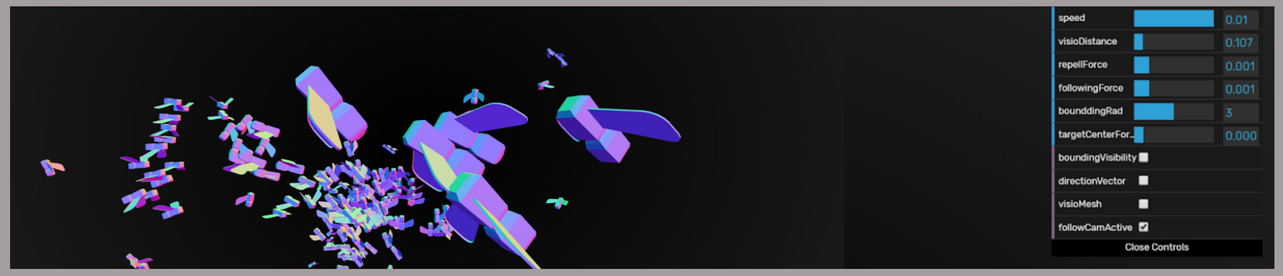
Tous les models pluggés sont ensuite input dans un seul et même object accessible partout.

```
import ModelLoader from "../ModelLoader"

//...
start() {
    this.room = ModelLoader.models[2].scene
    this.scene.add(this.room)
}
```

Boids manager

La scène numéro 2 de home sweet home demande a l'utilisateur de souffler sur le device pour faire voler des oiseaux.
La gestion de position et rotation de ces derniers a été déjà le sujet plusieurs études et est aujourd'hui designé par le nom de " Simulation de boids ".
Voir document explicatif : https://www.red3d.com/cwr/boids/



En nous basant sur ces différents documents, nous avons construit ce tool: https://github.com/MariusBallot/05-2020-threeBoids

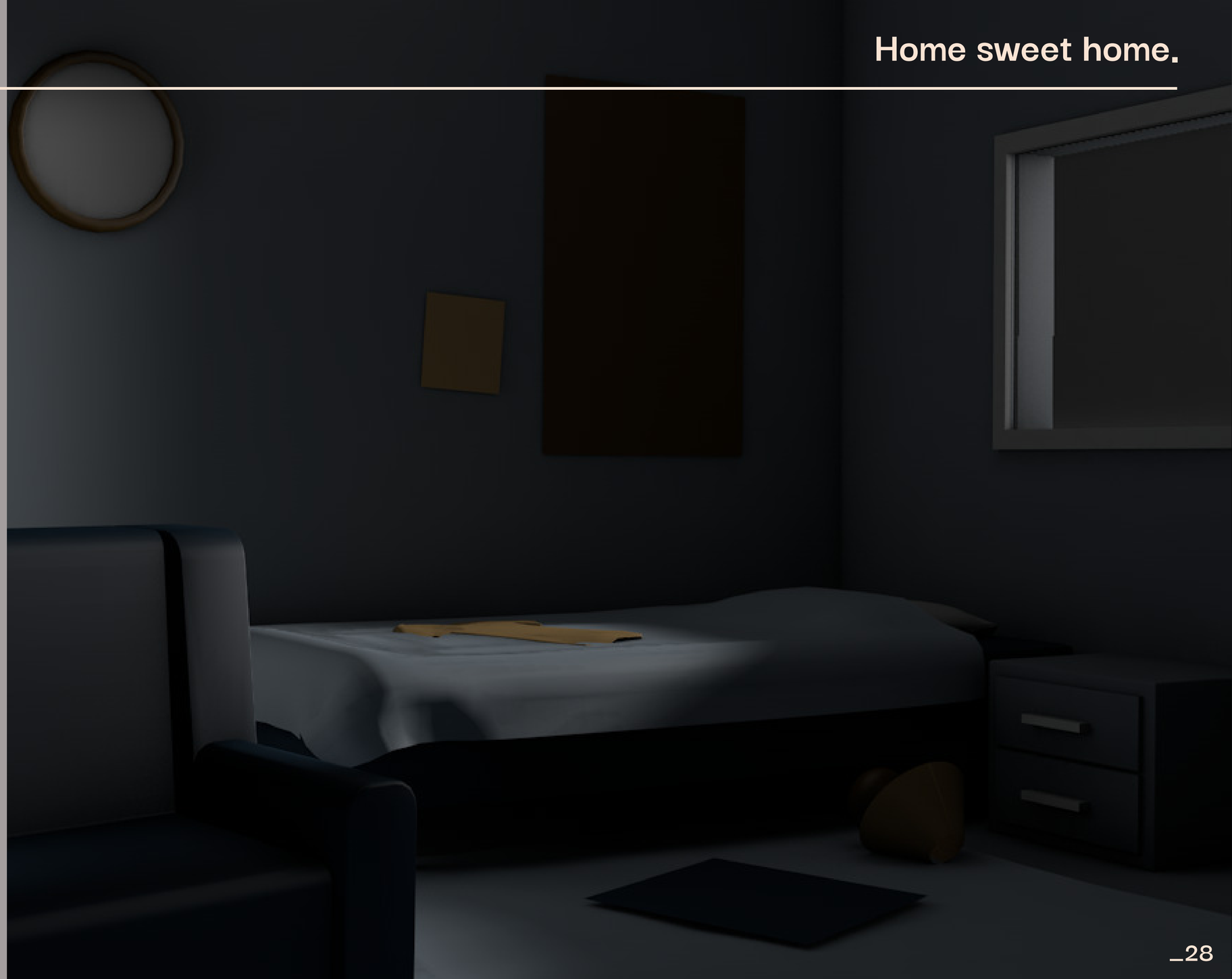
Il permet d'instancier un certain nombre d'entités et suivent les 3 règles principales des boids :

1/ Les autres boids les plus proches dans un certain rayon, appelé boids voisins, doivent s'éloigner les uns des autres

2/ Les boids voisins doivent suivre le vecteur directionnel des autres boids

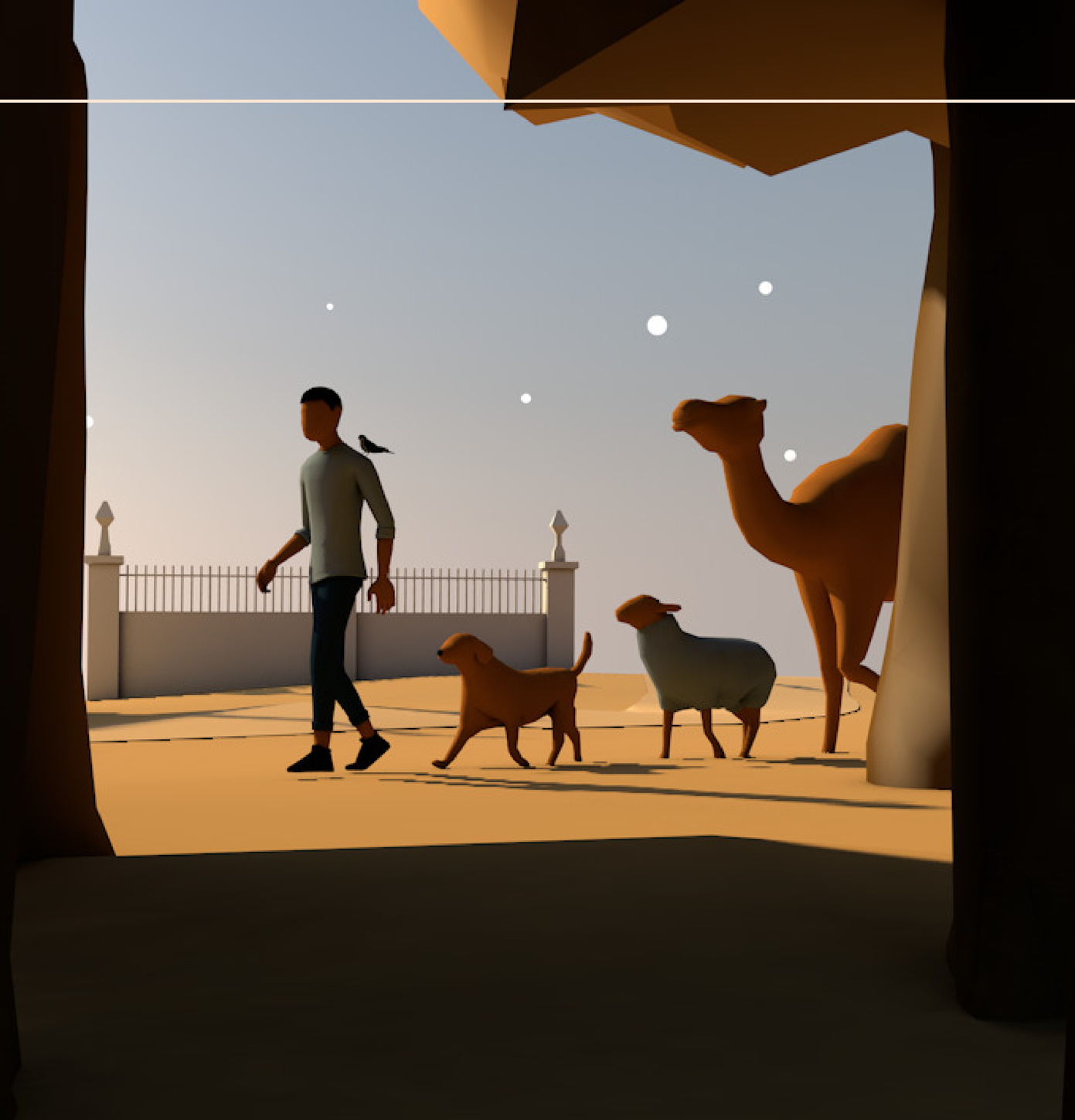
3/Un boids a une force directionnelle vers le center volumique des boids voisins.

Final



L’Equipe

Sound Design	Ouri Levin
Environements	Luc Miramont
Animations / Character Design	Ouri Levin
Direction Artistique	Luc Miramont
Creative Developer	Michael De Laborde
Creative Developer	Marius Ballot



Merci

Pour votre temps.