

# Technical Design Document

Projet LeapMotion Session 2015-2016

**M2 M2I ITESCIA**

14 octobre 2015

Christophe Froberville – Nicolas Guerrault – Julien Nguyen – Geoffrey Diop – Antony Bontour

# Technical Design Document

---

Projet LeapMotion Session 2015-2016

## ***I - Outils***

- Génération des données
- Code
- Audio
- LeapMotion
  - Système de coordonnées
  - La main
  - Gestes
  - Mouvements

## ***II - Techniques de rendu***

- Vue
- Éléments de gameplay
- Type d'élément
  - Cubes
    - Cubes rotation simple
    - Cubes rotation père/fils
    - Cubes pivot d'angle rotation père/fils
  - Mur
    - Mur bloquant le retour
  - Piège
- Élément interactif
  - Levier
  - Anamorphose
  - Objets récupérables
- Texture
- Niveau
- Comportement du personnage
- Risques techniques

## ***III - Détail sur le processus de création de données***

- Niveau
- Objets animés
- Processus

## ***IV - Données graphiques***

- Personnage
  - Modélisation et intégration
  - Animation
  - Contraintes
- Interface
- Liste des éléments du décor

## I - Outils

DeadWeight est un jeu 3D qui se joue sur PC avec la LeapMotion.

Configurations mini du PC recommandée :

- Windows® 7/8 or Mac® OS X 10.7
- AMD Phenom™ II or Intel® Core™ i3/i5/i7 processor
- Chipset graphique minimum : Radeon R7-250
- 2 GB RAM
- USB 2.0 port
- Internet (Software Leap Motion)

Concernant la résolution nous nous sommes basé sur les données de joueurs de septembre fournies par steam pour choisir les 3 résolutions les plus utilisées :

1360 x 1024	0.01%	0.00%
1366 x 768	26.84%	-0.24%
1368 x 768	0.03%	0.00%
1400 x 1050	0.07%	0.00%
1414 x 794	0.02%	0.00%
1421 x 800	0.01%	0.00%
1440 x 810	0.01%	0.00%
1440 x 900	5.04%	+0.01%
1440 x 960	0.10%	-0.01%
1474 x 829	0.01%	+0.01%
1536 x 864	2.62%	+0.20%
1536 x 960	0.02%	0.00%
1600 x 900	7.23%	-0.11%
1600 x 1024	0.05%	0.00%
1600 x 1200	0.24%	0.00%
1680 x 945	0.02%	0.00%
1680 x 1050	4.59%	-0.04%
1707 x 960	0.02%	-0.01%
1707 x 1067	0.01%	0.00%
1768 x 992	0.10%	0.00%
1776 x 1000	0.06%	-0.01%
1804 x 1014	0.01%	0.00%
1824 x 1026	0.03%	0.00%
1842 x 1026	0.01%	0.00%
1842 x 1036	0.02%	-0.01%
1862 x 1048	0.01%	0.00%
1920 x 1080	34.51%	+0.47%
1920 x 1200	1.52%	0.00%
2048 x 1152	0.10%	0.00%

-1920\*1080 sera la résolution standard

-1366\*768 et 1600\*900 seront d'autres résolutions supportées par notre jeu.

## Génération des données

Toutes les données 3D seront générées avec un éditeur 3D, nous avons choisi 3DsMax.

Nous utilisons la version 3DsMax 2011.

Les objets et données seront exportés depuis 3DsMax en utilisant le format .fbx ; C'est un logiciel libre et gratuit.

Dans 3DsMax seront créés les décors, le personnage et les animations.

## Code

L'ensemble du développement se fera par Unity en C#.

Nous utilisons la version Unity 5.2.1f1 (64-bit)

### Convention d'écriture:

Quoi?	Comment?	Exemples:
Variables Locales	*Minuscule *Si plusieurs "mots", une majuscule pour première lettre de chaque mot sauf le premier	int local; int maVariableLocal;
Variables Membre public	*Minuscule * Chaque mot séparé par un "_"	public int variable; public int variable_public;
Variables Membre protected	*Minuscule * Chaque mot séparé par un "_"	public int variable; public int variable_protected;
Variable Membre privée	*Majuscule première lettre *Chaque mots séparé par un "_" et se termine par _	private int Variable_; Private int Variable_

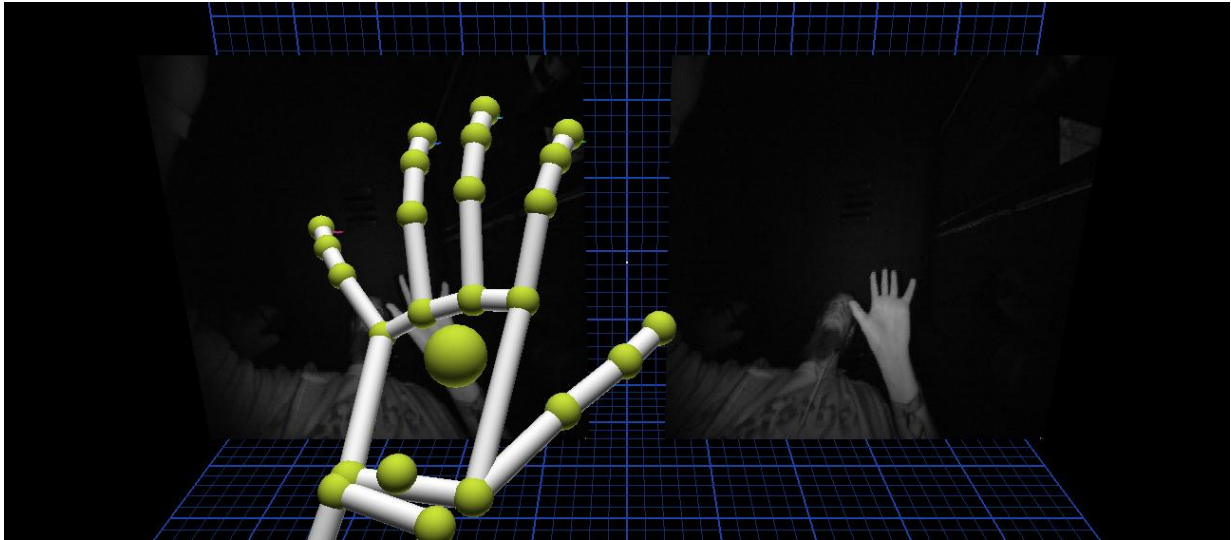
Avec un éditeur de texte comme MonoDevelopUnity ou Visual Studio 2015.

## Audio

Les musiques et les sons de notre jeu sont créés via le logiciel Reason puis exporté en format .wav pour être utilisé sur Unity 3D.

## LeapMotion

Le contrôleur LeapMotion utilise des capteurs optiques et la lumière infrarouge qui reconnaît et suit les mains, les doigts et les outils (type stylo) avec un champ de vision d'environ 150 degrés.



La version du SDK utilisé pour le développement sera la version 2.3.X fournie par LeapMotion à l'adresse suivante: [https://developer.leapmotion.com/downloads#release\\_notes](https://developer.leapmotion.com/downloads#release_notes)

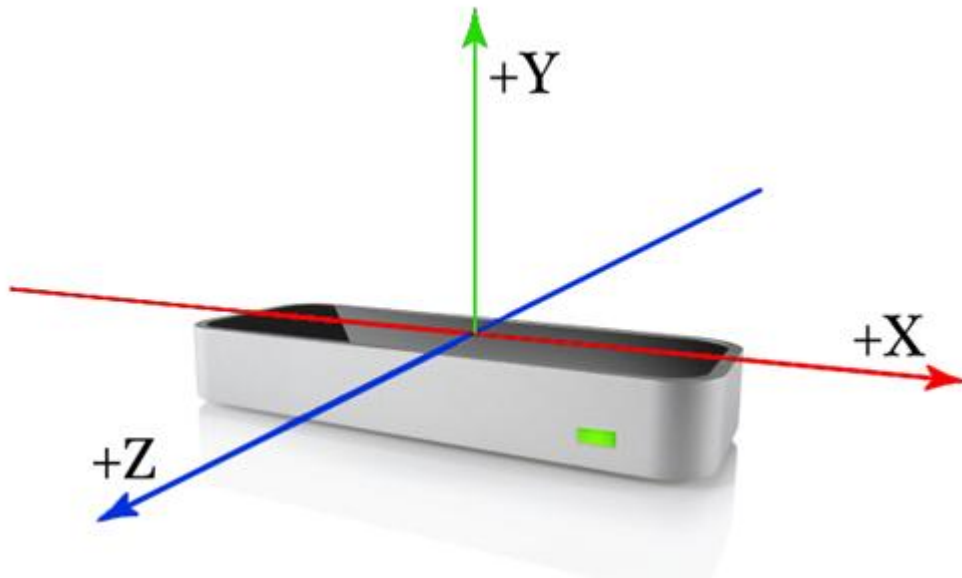
Cette version est la dernière en date et la plus stable, elle permet entre autre à l'appareil d'être utilisé avec des casques de réalité virtuelle.

Le SDK fournit également un ensemble de fonctions et de scripts écrit en C# et utilisable dans les scènes Unity. Une documentation complète est fournie par le constructeur à l'adresse suivante: <https://developer.leapmotion.com/documentation/csharp/index.html>

Le principale objet exploité dans le SDK est *HAND*, en effet c'est lui qui fournit la plupart des informations relative au déplacement et actions de la main dans l'espace. L'objet fournit un historique de toutes ces informations frames par frames, il est ainsi possible de savoir ce que l'utilisateur a fait et pourquoi pas de prévoir ces futurs mouvement.

### Système de coordonnées

Le système LeapMotion utilise un système de coordonnées cartésiennes droitier. L'origine est centrée en haut du contrôleur LeapMotion. X et z se situent dans les axes du plan horizontal, avec l'axe x parallèle à l'arête longue de l'appareil et l'axe y est l'axe situé verticalement perpendiculairement à l'axe x.



L'API Leap Motion mesure de grandeurs physiques avec les unités suivantes:

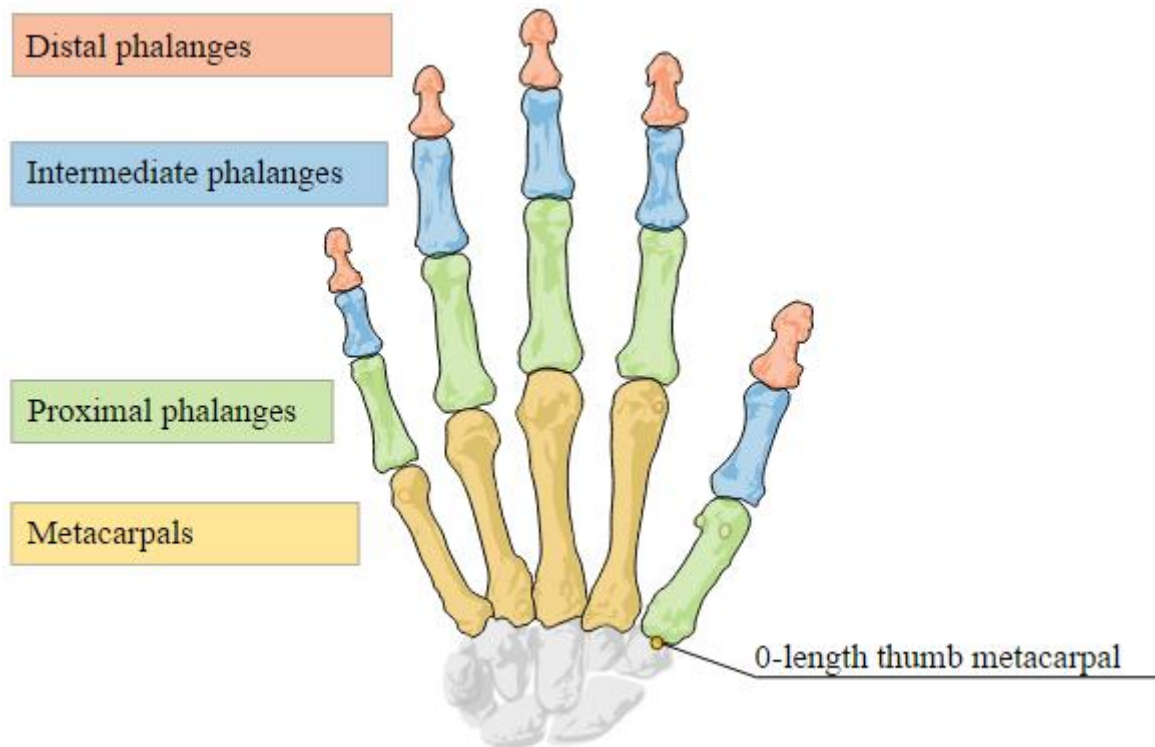
Distance	Millimètre
Temps	Microsecondes
Vitesse	Millimètres / secondes
Angle	Radian

### **La main**

Le contrôleur LeapMotion fournit des informations sur chaque doigt de la main. Si tout ou partie d'un doigt est pas visible, les caractéristiques de doigts sont estimés sur la base des observations récentes et le modèle anatomique de la main. Les mains sont identifiés par le nom du type, à savoir *le* pouce, l'*index*, milieu, *anneau*, et *rosâtre*.

Ce qui nous permet en tant que développeurs d'avoir Finger [tipPosition \(\)](#) et [direction \(\)](#) vecteurs fournissent la position d'un bout de doigt et de la direction générale dans laquelle un doigt est pointé.

Un doigt objet fournit un os objet décrivant la position et l'orientation de chaque doigt osseux anatomique. Tous les doigts contiennent quatre os commandés à partir de la base au sommet.

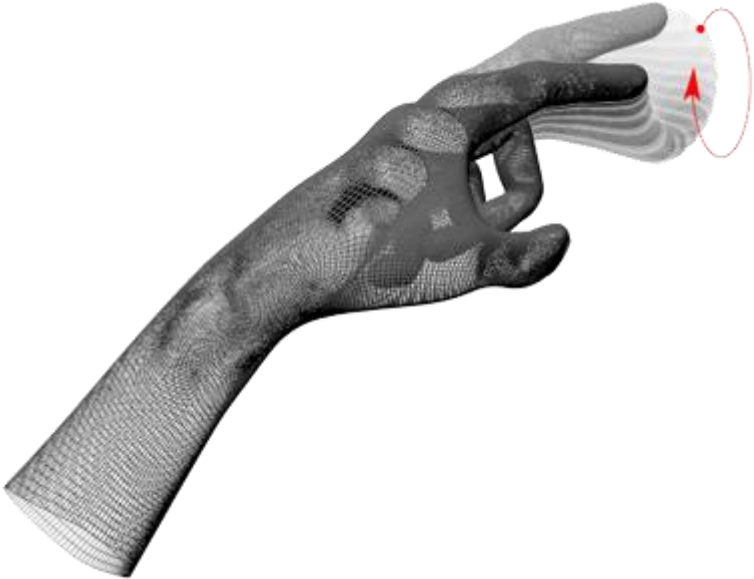



Les os sont identifiés comme suit:

- Métacarpal - l'os intérieur de la connexion le doigt vers le poignet (sauf le pouce)
- Proximal Phalange - l'os à la base du doigt, reliée à la paume
- Intermédiaire Phalange - l'os milieu du doigt, entre la pointe et la base
- Distal Phalange - l'os terminal à l'extrémité du doigt

## Gestes

Le logiciel LeapMotion sait également reconnaître des gestes appartenant à quatre catégories:

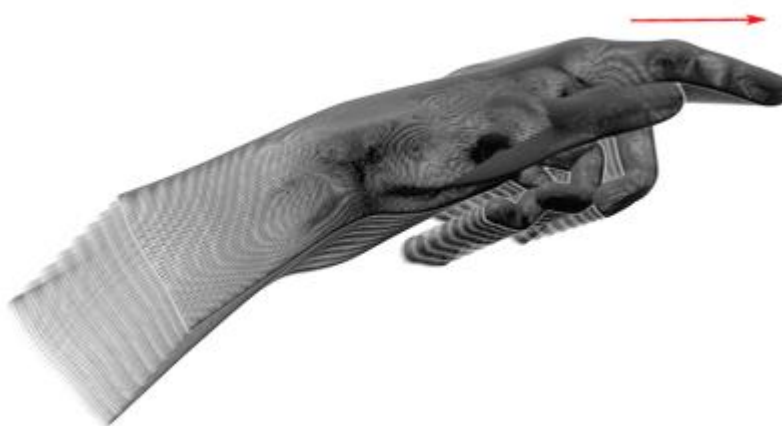
Catégorie	Geste
Cercle	
Balayage	



Pression (Type touche)



Pression sur l'écran



## Mouvements

Les mouvements sont le résultat de la différence de position et/ou de rotation faite entre deux frames.

Type de mouvement	Frame	Main
<b>Échelle (Scale)</b>	Reflète le mouvement des objets de la scène plus ou moins loin de l'autre. Par exemple, une main se rapproche de l'autre.	Mise à l'échelle de la main reflète le changement de l'écartement des doigts.
<b>Rotation</b>	Reflète un mouvement différentiel des objets dans la scène. Par exemple, une main se déplace vers le haut et l'autre vers le bas	Changement dans l'orientation d'une seule main.
<b>Translation</b>	Reflète la variation moyenne de la position de tous les objets de la scène. Par exemple, les deux mains se déplacent vers la gauche, vers le haut ou vers l'avant.	Changement de position de la main.

Plus de deux mains peuvent apparaître dans la liste de la main pour un cadre si les mains ou d'autres objets comme à la main de plus d'une personne sont en vue. Cependant, il est recommandé de conserver au maximum deux mains dans le champ du Contrôleur LeapMotion pour une qualité optimale de suivi de mouvement.

## II - Techniques de rendu

### Vue

La caméra de notre jeu sera rotative, elle pourra tourner selon l'axe Y autour d'un point de pivot centré sur le milieu de la scène.

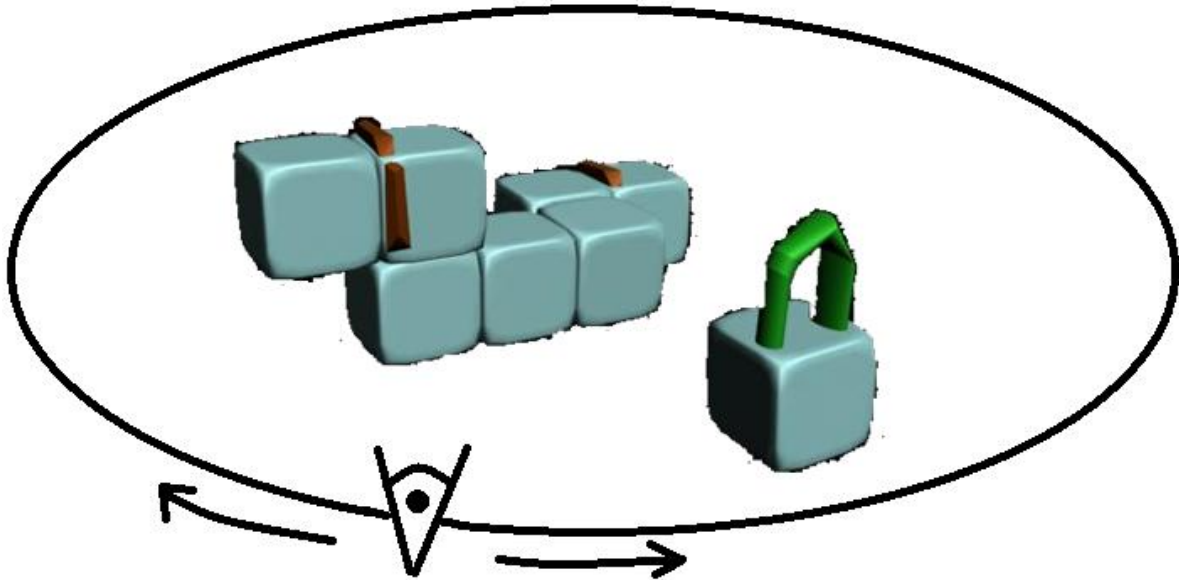


Schéma des positions possible de la caméra

La caméra est légèrement surélevée afin de visualiser le chemin que notre personnage empruntera. Il suffira de placer ses deux mains dans un coin en haut à droite/gauche de l'écran afin de faire une rotation autour de cet axe.

### Eléments de gameplay

L'environnement de notre jeu est totalement en 3D, nous avons opté pour une caméra qui effectue une rotation autour d'un point de pivot fixe (le personnage) en horizontal.

Les décors des levels seront portés sur un système de cube, pouvant effectuer des rotations. Les décors visuels autour du levels seront constitués d'anamorphoses ainsi que divers objets et images cohérentes aux scénarios adoptés pour chaque niveau.

Les décors en dehors du level seront appliqués sur la skybox avec une méthode de post process, le tone mapping afin d'améliorer la qualité graphique.

Le fond de notre écran de jeu sera une sphère sur laquelle on appliquera une skybox.

## Type d'élément

### **Cube:**

Les cubes sont les éléments principaux des niveaux, chacun d'entre eux sera créer à partir d'une série de cube, les bonus et les différents éléments d'interaction seront greffés sur ces derniers.

Étapes de création d'un cube:

- Modélisation d'un cube simple sous 3DS MAX
- Modification du mesh du cube afin d'arrondir les angles
- Création de la texture du cube sous Substance
- Export/import du fichier de texture .sbs vers 3DS MAX
- Application de la texture au mesh du cube
- Export/import du fichier.fbx de 3DS MAX à Unity

### **Cube rotation simple:**

Pour ce type de cube, le centre de gravité sera placé au centre de ce dernier et les rotations autour des axes Y et Z seront bloqués afin de laisser le cube tourner autour du seul axe X.

### **Cube rotation père/fils:**

Semblable aux cubes à rotation simple à la différence que la rotation du cube principal entraîne celle d'autres cubes à la suite.

### **Cube pivot d'angle rotation père/fils:**

Le concept est identique aux cubes à rotation père/fils à la seule différence que la rotation n'est plus la même: elle s'effectue par rapport à un centre de gravité placé dans un angle du cube, ce qui engendrera une rotation de manière à ce que le cube ai des sortes de gonds et s'articule autour de ces derniers.

### **Murs**

Des obstacles simples qui arrêtent le personnage dans sa marche et permet d'anticiper la suite du niveau.

Création d'un mur:

- Modélisation du mur dans 3DS max
- Création de la texture du cube sous Substance
- Export/import du fichier de texture .sbs vers 3DS MAX
- Application de la texture au mesh du cube
- Export/import du fichier.fbx de 3DS MAX à Unity
- Ajout du mur sur une face d'un cube
- Ajout d'un colider sur le mur afin que le personnage s'arrête en conséquence

## **Murs bloquant retour**

Modélisation du mur dans 3DS max

- Création de la texture du cube sous Substance
- Export/import du fichier de texture .sbs vers 3DS MAX
- Application de la texture au mesh du cube
- Export/import du fichier.fbx de 3DS MAX à Unity
- Ajout d'un trigger sur une case adjacente a celle contenant le mur
- Ajout d'un script permettant l'affichage du mur avec collider lors de l'activation du trigger

## **Piège:**

Les piège font partie des éléments d'interaction principaux, il en existe de plusieurs type: à trou/téléportation/inversement, ils s'enclencheront lorsque le personnage franchira le trigger de ces derniers. L'ensemble des éléments de type pièges seront créés de la même manière, globalement seul le code couleur changera

Etapes de création d'un piège:

- Modélisation d'une plateforme plate et ovale sur 3DS MAX
- Création de particules autour du piège
- Création de la texture du piège
- Export/import du fichier de texture .sbs vers 3DS MAX
- Application de la texture
- Export/import du fichier.fbx de 3DS MAX à Unity
- Application et liaison d'un piège sur un cube
- Creation et positionner le trigger sur le cube

## **Éléments d'interaction:**

### **Leviers:**

Les leviers seront des éléments d'interaction placée sur un cube de la Mapp, ils permettront lorsque le personnage passe dessus d'ouvrir une porte jusque-là bloquée.

Étapes de création d'un levier:

- Modélisation du levier sur 3DS MAX
- Création de la texture du levier sur substance
- Export/import du fichier de texture .sbs vers 3DS MAX
- Application de la texture
- Export/import du fichier.fbx de 3DS MAX à Unity
- Application et liaison du levier sur un cube
- Créer et positionner le trigger sur le cube
- Modélisation de la porte 3DS MAX
- Création de sa texture sur substance

- Export/import du fichier de texture .sbs vers 3DS MAX
- Application de la texture
- Export/import du fichier.fbx de 3DS MAX à Unity
- Application et liaison de la porte sur un cube
- Création du script qui permet de déplacer la porte lorsque le trigger du levier est activé.

### **Anamorphose**

Pour construire l'anamorphose de chaque niveau, nous avons toujours procédé de la même manière:

- Création de séries de cube sur 3DS max
- Association de l'image représentant l'anamorphose au cube en utilisant la technique du mapping
- Lien de un ou plusieurs cube de l'anamorphose sur les cubes pivots de notre niveau
- Exportation du fichier en .fbx pour être exploitable sur Unity 3D

### **Objet récupérables**

Ces objets permettront d'ajouter les 3 composantes RGB au niveau initialement sans couleur.

Création des objets récupérables :

- Modélisation des objets récupérables sur 3DS MAX
- Création de la texture des objets sur substance
- Export/import du fichier de texture .sbs vers 3DS MAX
- Application de la texture
- Export/import du fichier.fbx de 3DS MAX à Unity
- Placer les objets sur le niveau
- Ajouter les triggers respectifs
- Ajouter les scripts sur les triggers permettant de rajouter les palettes de couleurs personnalisées faites avec le tone mapping

### **Texture**

L'ensemble de nos textures seront créée avec Substance Designer

Charte de tailles des textures :

- 256\*256 pour les images de textures des cubes
- 512\*512 pour la texture du personnage

Post process utilisés :

- Sepia Tone

- Noise
- Tone Mapping

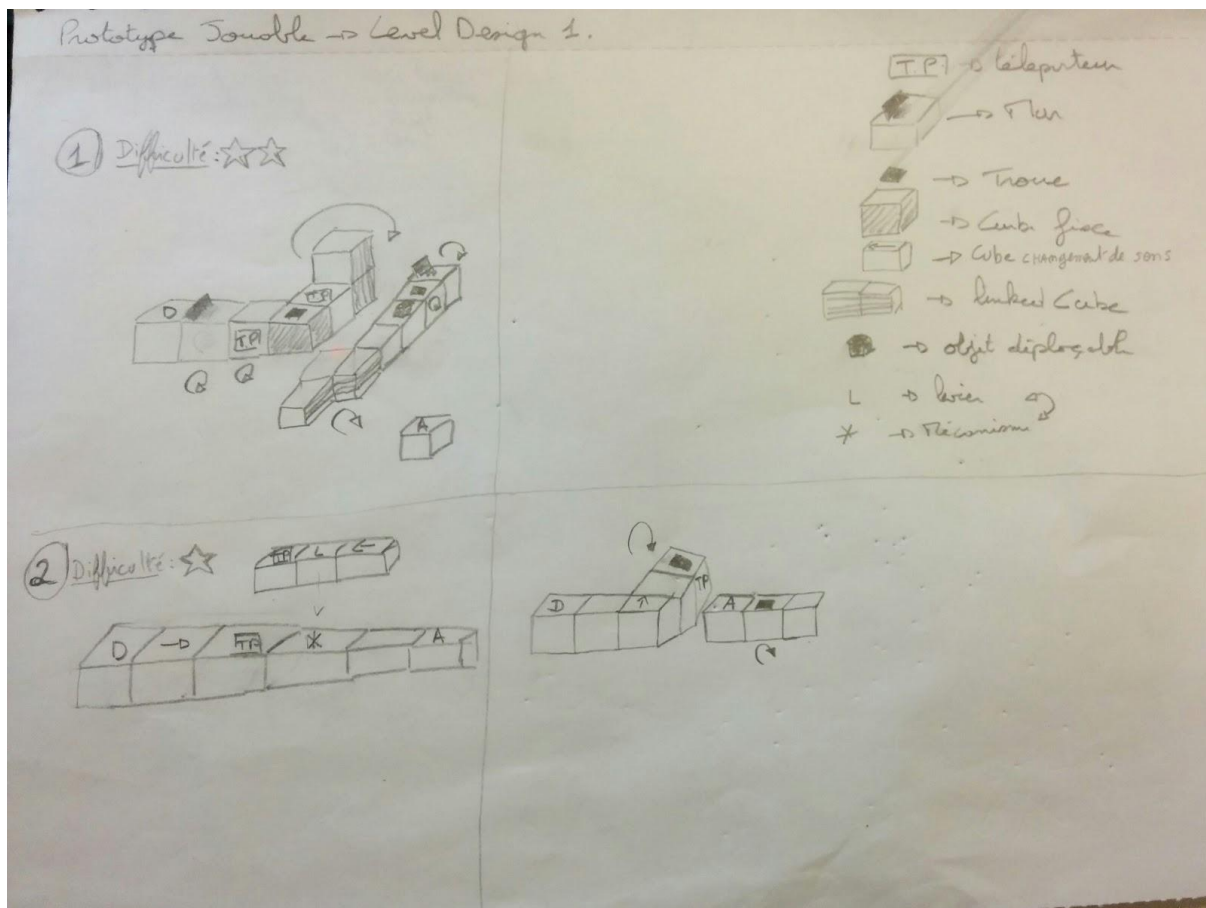
Utilisation du tone mapping :

Aux débuts de tous nos niveaux, ces derniers seront avec un filtre sepia, plutôt fade et le joueur pourra faire en sorte que le personnage prennes les 3 objets récupérables. Le tone mapping permettra grâce aux palettes de couleur personnalisées d'ajouter cet effet.

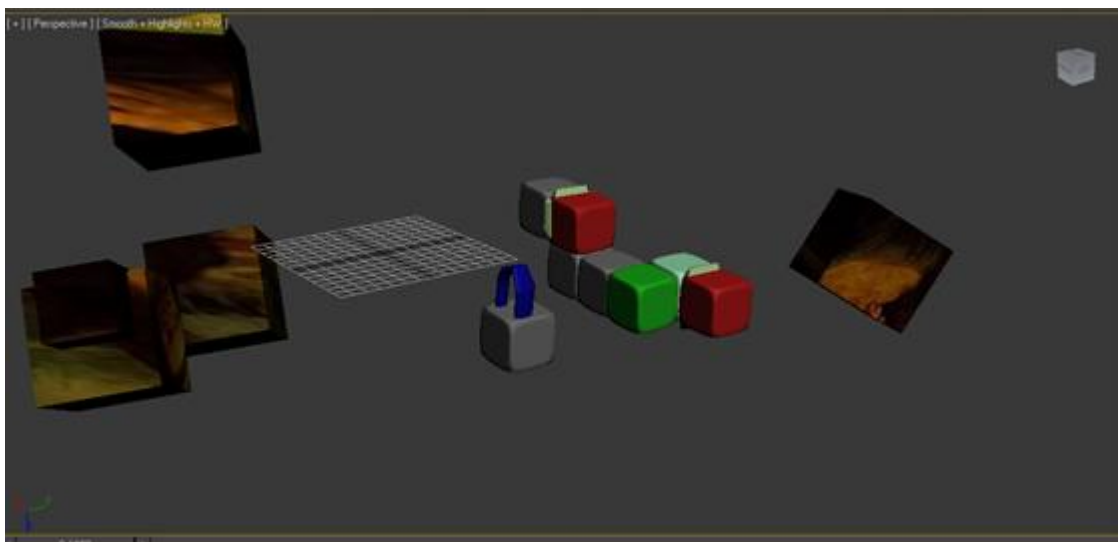
## Niveau

Tout d'abord pour concevoir chaque niveau nous commençons par les dessiner sur papier en y incorporant tous nos éléments de Gameplay.

La taille maximale cube d'un niveau est de 60 cubes.



Une fois le niveau validé par tous les membres du groupes, la personne en charge modélise le niveau sur 3DS max. Nous créons ensuite un fichier .fbx exportable sur Unity3D.



## Comportement du personnage

Notre personnage se déplacera seul sur les faces de nos cubes, c'est à dire que nous n'avons aucun contrôle sur lui. N'ayant pas de choix à faire quand à la route qu'il devra prendre, notre personnage sera donc dépourvu d'intelligence artificielle mais changera de comportement selon des triggers et dans certains cas :



- Quand il rencontre un mur, il devra s'arrêter
- Quand il arrive sur une bordure d'un cube isolé, il devra s'arrêter
- Quand il arrive à la fin d'un niveau, il devra s'arrêter
- Quand il arrive sur un trou, il le traversera et engendrera un retry du niveau

## Risques techniques

Les différents risques techniques sont :

- Risque de ne pas retransmettre un mouvement intuitif avec la leap motion
- Risque sur le rendu d'un point de vue modélisation, animation

### III - Détail sur le processus de création de données

#### Niveau

Level	Nombre de cube		Elements interactifs
1	11	Chris	Rotation Axe Y et X, mur
2	12	Geo	Rotation, ChangSens
3	11	Chris	Rotation, Téléporteur, ChangSens
4	16	Geo	Rotation, Trou, mur AR
5	20	Chris	rotation, trou, mur, téléporteur, levier
6	25	Geo	Rotation Double, ChangSens, Teleport, trou, mur
7	25	Chris	rotation, trou, mur, téléporteur, levier
8	30	Geo	rotation, ChangSens, Teleport, trou, mur, mur AR
9	20	Chris	rotation, S A. G, trou, mur
10	40	Geo Chris Antho	rotation, S A. G, mur, murAR, trou, téléporteur, levier, ChangSens

	ordre d'arrivé des E.I
	rotation simple+mur, rotation père/fils X, Y, Z
1	
2	mur
	changement de sens(ChangSens)
3	
4	téléporteur
5	trou
6	levier
7	mur anti-retour(mur AR)
	surface anti-gravité(S A.G)
8	

#### Objets animés

L'animation de base dans notre jeu vidéo est la rotation de cube, avec la possibilité de faire une rotation  $+90^\circ$  et  $-90^\circ$  dans une direction via un mouvement avec la LeapMotion afin de permettre au personnage d'avancer. Un Cube avec rotation père/fils: Une rotation d'un cube peut entraîner la rotation d'un autre cube afin de créer un chemin.

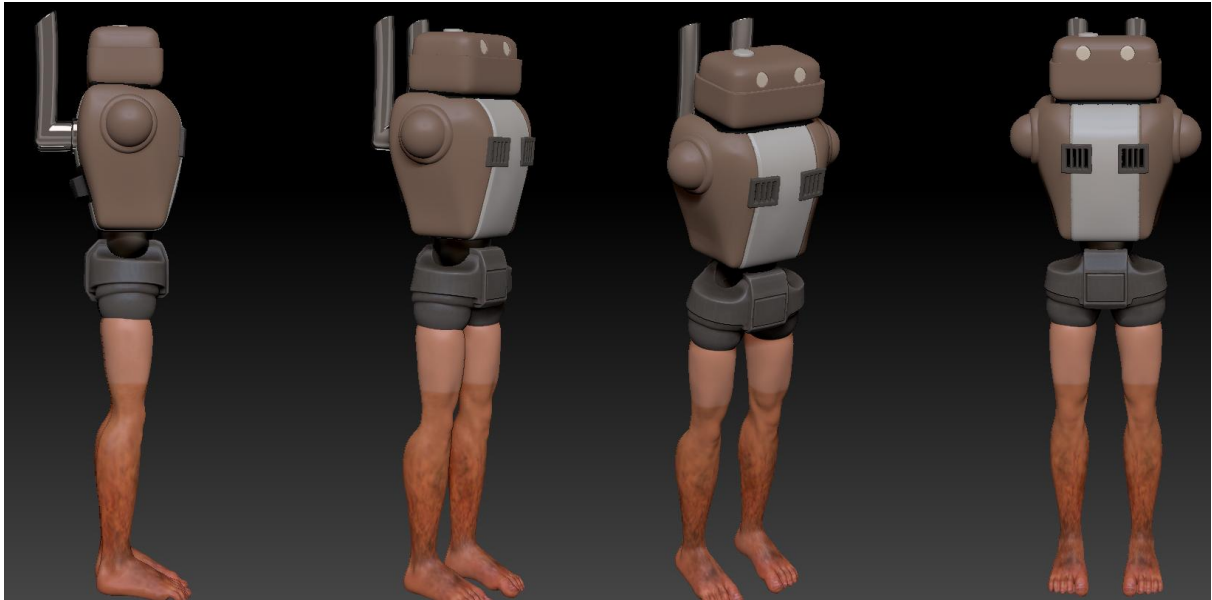
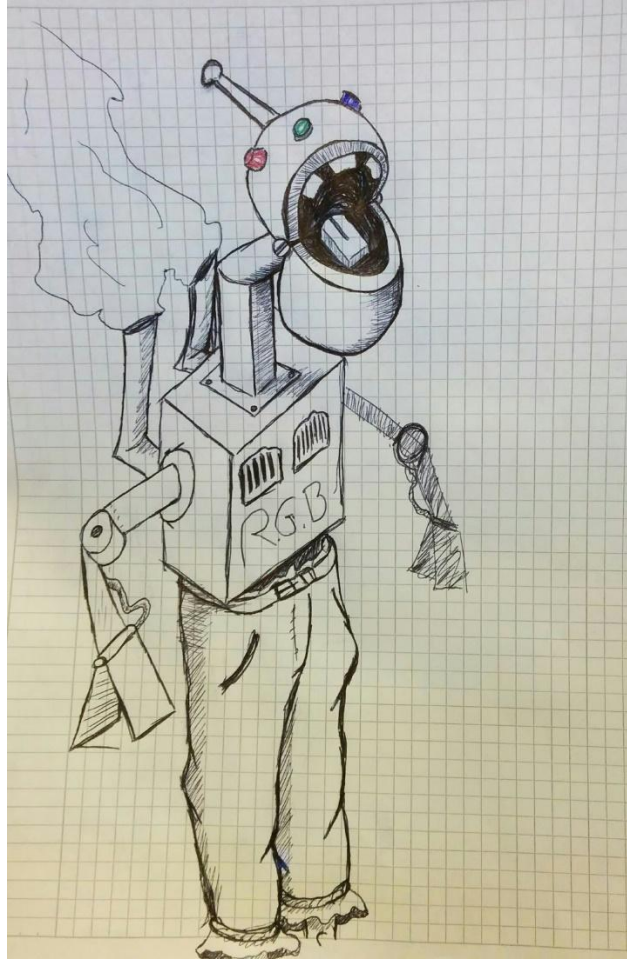
Le tout développé via Unity en liaison avec la LeapMotion

Nos animations seront créées grâce au logiciel 3dsMax, de plus nous avons accès à un outil d'animation avec la Kinect, ce qui nous permet d'avoir nos propres animations personnelles, que l'on retouche via notre logiciel de modélisation.

#### Processus

Capture d'animation avec logiciel "Ipi Motion Capture" ou Pro Body v2 et deux Kinects.







Notre personnage possède des pots d'échappements sur le dos qui émettent de la fumée, ces derniers seront faits avec grâce à l'outil émetteurs de particules sur 3DS MAX. Il possède également trois LED sur sa tête qui s'allumeront en temps voulu.

Pour les textures appliquées au personnage nous avons utilisé Substance Designer. (À détailler)

Pour appliquer la texture créée sur notre personnage nous avons exporté le fichier depuis Substance au format .sbs pour ensuite l'utiliser sur 3DS Max.

Une fois le personnage modélisé et les textures appliquées nous l'avons exporté en fichier .fbx pour l'importer sur Unity 3D.

### ***Animations***

Pour l'animation nous avons utilisé une Kinect pour capturer les mouvements couplée avec Brekel Pro Body v2 qui est un logiciel permettant ensuite de les interpréter pour pouvoir en faire des animations applicables à notre personnage. Tout ceci est effectué sur l'ordinateur LDLC PC10 Forcer fourni par l'ITESCIA qui est équipé de très bon composants pour pouvoir effectuer la motion capture sans ralentissement.

Nous avons ensuite fait des retouches pour affiner les animations sur le logiciel 3DS Max.

Le fichier final à importer sur Unity 3D sera au format .fbx et contiendra le personnage constitué de ces polygones, bones et textures ainsi que ces animations.

Pour une fluide animation du personnage nous avons choisi de partir sur 60 images par secondes.

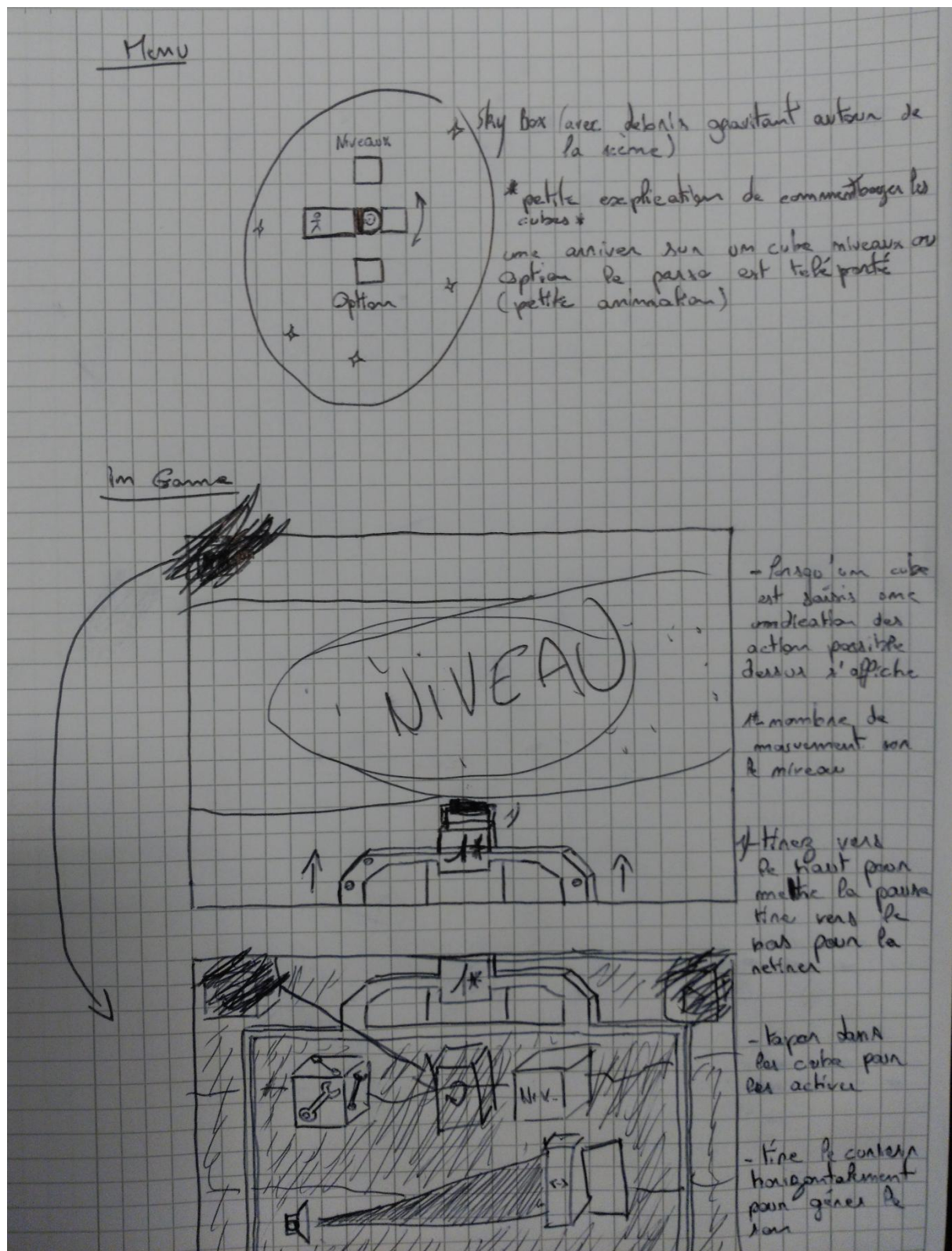
### **Contraintes**

Les contraintes globales pour le personnage sont:

- 1500 à 4000 polygones pour la modélisation
- Apparence visuelle ressemblant un maximum à notre croquis en prenant en compte nos compétences en modélisation 3D



## Interface



## Liste d'éléments de décor

- Skybox
- Cube
- Anamorphose