Année 2017/2018

Rapport de projet - EI4 SAGI

Plateforme pour la rééducation de la sclérose en plaques

Projet réalisé par:
Axel Raimbault
Denis Sanchez

Projet encadré par:
Paul Richard

IstiA

École associée Polytech

1°)Sommaire

2°)Table des figures

3°)Remerciements

Nous tenons à remercier de nombreuses personnes pour ce projet.

-Mlle. Mathilde DE CHIARA pour avoir proposé ce sujet.

-M. Paul RICHARD pour avoir encadré ce projet et mis à notre disposition le matériel que nous avons utilisé.

-M. Jules MACAIRE pour nous avoir accueilli dans sa salle et nous avoir prodigué de précieux conseils.

-M. Corentin TURCAUD et M. Antonin THILLOUX pour nous avoir partagé une partie d'un de leurs anciens codes dont nous avons pu nous inspirer.

-Pauline Rouls pour avoir produit le dessin de Unity-chan utilisé plus loin dans le rapport. (figure 11)

4°)Engagement de non-plagiat

Nous, soussigné Axel Raimbault et Denis Sanchez, déclarons être pleinement conscients que le plagiat de documents ou d'une partie d'un document publiés sur toutes formes de support, y compris l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée. En conséquence, nous nous engageons à citer toutes les sources que nous avons utilisées pour écrire ce rapport.

5°)Conduite de projet

5.1°)Origines du projet et objectifs

5.1.1°)Origines du projet

Ce projet consiste à réaliser une plate-forme de réalité virtuelle pour aider à la rééducation de personnes atteintes par la sclérose en plaques. Le projet initial a été demandé par une étudiante des Beaux-Arts d'Angers dans le cadre de sa thèse. Nous devions créer une plateforme en trois dimensions suivant des contraintes graphiques qu'elle nous donnait. Nous avons toutefois rapidement arrêté cette collaboration et avons travaillé sur le projet de la même manière afin d'avoir une plate-forme la plus aboutie possible.

5.1.2°)Objectifs

L'objectif principal est de créer une plate-forme de réalité virtuelle permettant à une personne atteinte de la sclérose en plaques de faire des exercices de manière ludique. Les exercices doivent être variés et d'une durée raisonnable pour ne pas épuiser les muscles de l'utilisateur. La plate-forme doit être agréable à regarder tout en suivant la charte graphique qui nous a été fournie. Enfin, elle doit être facilement utilisable par un nouvel utilisateur qui n'aurait jamais utilisé une plate-forme de ce genre.

5.1.3°)Contraintes

La première contrainte est une contrainte temporelle. Nous avons besoin de terminer ce projet avant le 13 avril, date de rendu du rapport. Une autre contrainte est la contrainte matérielle et de local. Nous n'avions en effet accès jusqu'en mars qu'aux salles informatiques en libre-service pour réaliser ce projet. Nous étions généralement installés dans la salle 319, seule salle où le logiciel Unity était installé. Cependant, il y avait régulièrement des cours dans cette salle, ce qui nous obligeait à changer de salle et à installer Unity sur place.

5.1.4°)Evaluation

L'évaluation de la plate-forme se fait par le jury composé de M. Paul Richard et M. Jean-Baptiste Fasquel selon les critères évoqués précédemment. Il faut que notre plate-forme soit avant tout fonctionnelle mais également intuitive et esthétique.

5.2°)Méthode de travail

5.2.1°)Matériel

Pour ce projet, nous avons principalement utilisé Unity, le logiciel qui nous permet de créer la plate-forme. La forme globale des éléments de la scène principale a été créée sur le logiciel de création 3D Blender. Le code a lui été réalisé sur Visual Studio.

Figure 1: Logos des différents logiciels utilisés.

Pour Le matériel, nous avons utilisé une Kinect 2.0. Nous avons estimé que cet outil nous permet de récupérer facilement la position du corps entier et calculer rapidement les mouvements relatifs sur Visual Studio.

Figure 2: Une Kinect.

5.2.2°)Planning

Nous avons décidé de commencer par la création de la partie visuelle avant de nous occuper de la partie codage.

Voici notre diagramme de Gantt. Une version zoomée et lisible est en annexe.

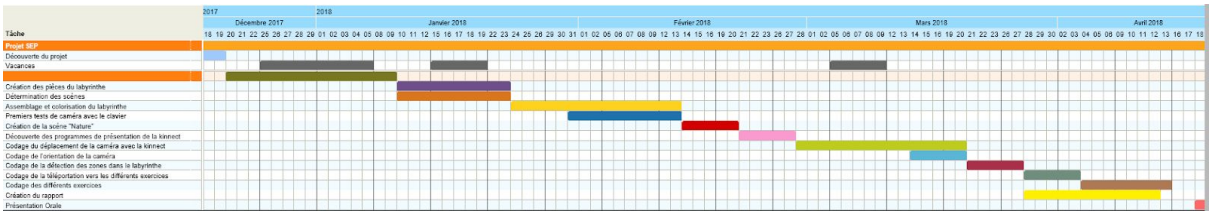


Figure 3: Diagramme de Gantt (voir Annexe 1).

Voici le diagramme montrant la durée relative des tâches.

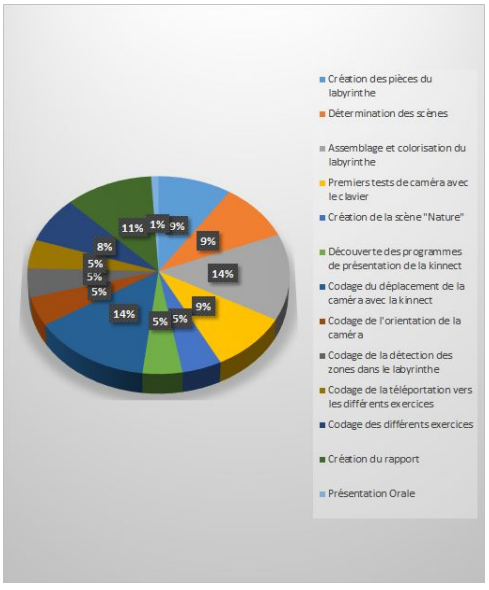


Figure 4: Diagramme circulaire montrant la part des différentes activités dans le projet. (voir Annexe 2)

5.3°)La sclérose en plaques

La sclérose en plaques est une maladie neurologique qui affecte environ deux millions et demi de personnes dans le monde et qui se déclenche surtout chez les jeunes adultes entre 20 et 40 ans. Cette maladie affecte le système nerveux et empêche la personne de faire des mouvements comme elle le souhaite. Les symptômes sont multiples mais globalement, il est difficile pour les personnes concernées de bouger leurs muscles. Pour certaines personnes, le problème est de bouger les paupières, pour d'autres, c'est la parole qui est la plus affectée mais la plupart du temps, ce sont les bras, les jambes et le dos dont les mouvements sont rendus difficiles.

Contrairement à la sclérose latérale amyotrophique (aussi appelée maladie de Charcot) qui touche la moelle épinière, cette maladie n'est généralement pas mortelle. Elle fait tout de même diminuer l'espérance de vie du malade d'entre 5 et 10 ans.

Il va de soi que les personnes atteintes de sclérose en plaques ont des difficultés à faire du sport ou tout autre effort physique. Elles n'ont toutefois pas d'autre choix que de faire de l'exercice fréquemment pour rester en forme. C'est pourquoi il est utile de changer la manière de faire ces exercices. Une plate-forme de réalité virtuelle est l'outil idéal puisqu'elle est très ludique et permet de faire des exercices concrets.

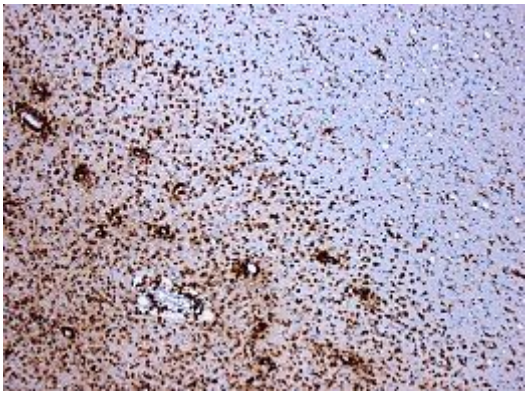


Figure 5: destruction de la gaine de myéline qui entoure et protège les fibres nerveuses, caractéristique de la sclérose en plaques.

5.4°)Présentation détaillée des méthodes de travail

5.4.1°) Travail de documentation et premières ébauches

Avant de commencer officiellement ce projet, nous avons assisté à une petite réunion de lancement en compagnie de Mlle De Chiara et M. Richard. L'objectif de cette réunion était de définir la charte graphique pour le menu. Il s'agit de mélanger les univers de Pac-Man et Tron. La couleur dominante doit être le bleu foncé tandis que les couleurs secondaires doivent être celles présentes dans l'univers de Tron (rouge et orange en priorité).

La première étape du projet a été de déterminer les différents exercices que nous coderons ainsi que les environnements adéquats.

Après une concertation, nous nous sommes décidés sur la création d'une scène carrée qui serve de "hub" et de quatre scènes accessibles via les quatre coins du hub.
Scènes:

- Scène de "menu" sur une carte inspirée de Pac-Man et Tron avec un exercice de marche pour le déplacement et l'orientation des épaules pour l'orientation générale.
- Exercices haut du corps avec plusieurs possibilités:
    - vertical face au sujet (enfonçage d'un clou)
    - vertical aux côtés du sujet
    - "pompe"
- Exercice bas du corps avec plusieurs possibilités:
    - battements de jambe
    - montée de genou
    - "squats"
- Exercices d'endurance
    - marche dans un environnement naturel
    - exercice canoë
- Etirements
    - mollets : fente

Nous ne savions pas s'il était préférable d'utiliser le même environnement pour les quatre exercices ou en faire un pour chacun. Nous avons donc décidé de commencer par créer le hub et de réfléchir plus tard aux exercices.

Pour l'environnement du menu, la charte graphique à suivre était un mélange entre Pac-Man et Tron. L'objectif est de recréer le labyrinthe de Pac-Man en trois dimensions. Pour cela, la carte du labyrinthe a été calquée sur le logiciel de création 3D Blender. Une fois ces pièces créées, nous avons pu les importer sur un nouveau projet Unity. Il a fallu remettre les pièces du labyrinthe à leur place pour le recréer le plus fidèlement possible. Des premiers tests de caméra qui bouge avec les flèches du clavier ont été réalisés.

5.4.2°) Création du labyrinthe

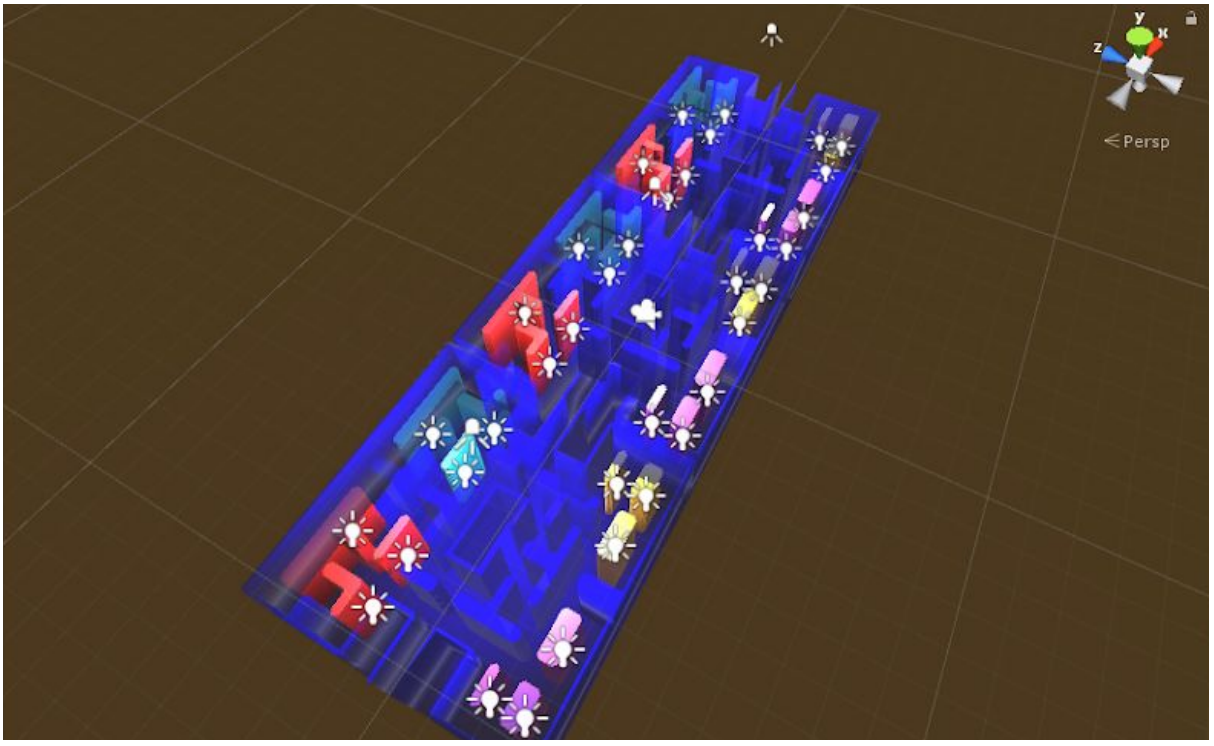


Figure 6: Vue d'ensemble de notre labyrinthe: Mélange de Tron et Pac-Man.

La colorisation du labyrinthe a également été une grande étape. Nous souhaitions avoir des murs bleus pour rappeler la couleur principale de l'univers Tron. La partie difficile était de rendre les murs à moitié transparents. En effet, le résultat était soit trop clair ce qui ne permettait pas une bonne immersion dans l'univers Tron, soit trop foncé et nous ne pouvions pas distinguer la transparence des murs. Qui plus est, lorsque nous nous approchions trop d'un mur, nous voyions à travers, ce qui empêchait toute immersion. Nous avons toutefois réussi à obtenir le résultat escompté et enlevé tous les bugs mineurs que nous avons pu rencontrer. L'objectif était atteint puisque nous avions une carte de Pac-Man avec les couleurs et le style "néon" de Tron.

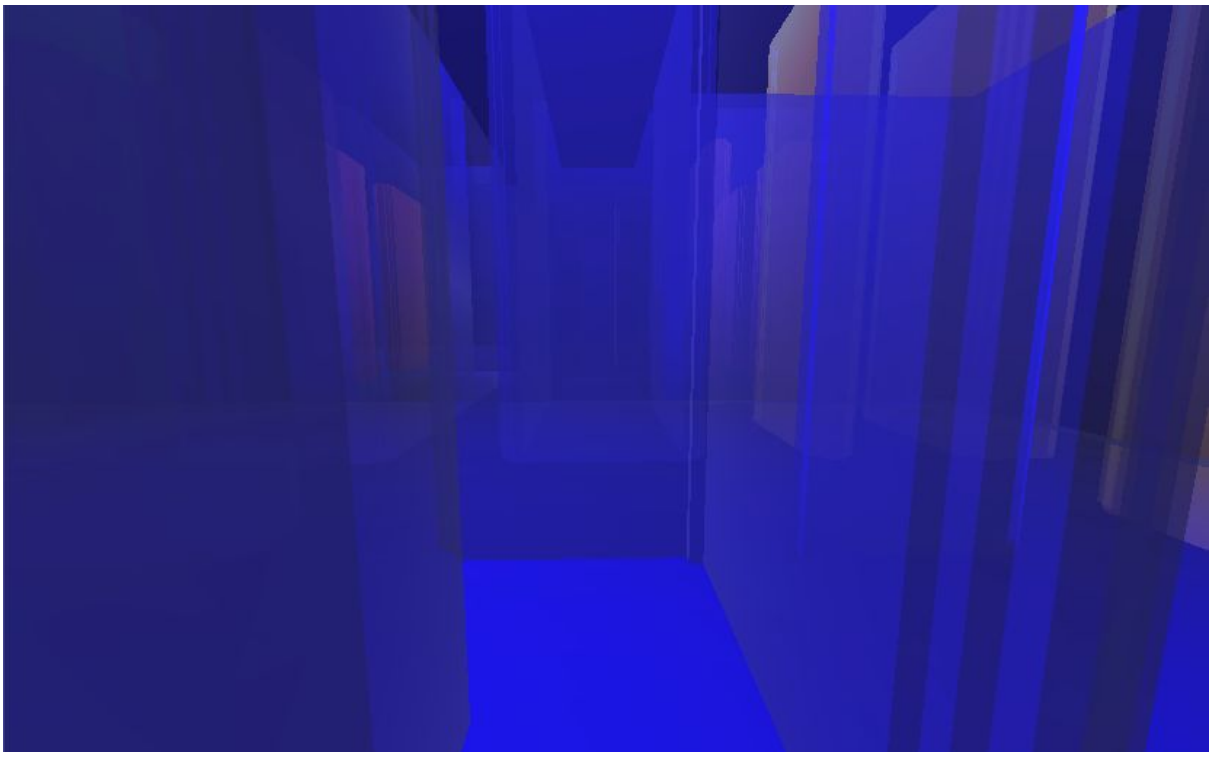


Figure 7: Vue depuis de labyrinthe illustrant le style “néon”.

Une fois la forme globale du labyrinthe prête, nous l’avons enregistré et avons aligné trois carrés afin que le joueur puisse se déplacer dans un environnement d’une taille de trois fois le carré plutôt qu’une seule.

Nous avons pris ensuite tous les murs et lui avons affecté un “box collider” afin qu’ils soient tangibles et que la caméra ne puisse plus les traverser. Le projet commence alors vraiment à prendre forme.

5.4.3°) Elaboration des quatre zones de téléportation du labyrinthe

Pour la partie suivante, Il fallait déterminer dans notre carte carrée quatre zones qui permettent de se rendre chacune dans un exercice différent. Nous avions imaginé plusieurs solutions pour désigner ces zones:

- un son qui se jouerait dans la zone.
- un texte qui s'afficherait dans la zone pour expliquer l’exercice.
- une couleur différente sur le mur
- une lumière colorée dans la zone

Nous avons envisagé toutes ces solutions et avons testé les deux dernières. Cependant, aucune des deux ne nous convenait parfaitement car nous perdions le style “néon”, nécessaire pour correspondre à l’univers de Tron. Nous avons alors

décidé de combiner les deux. Nous avons mis les murs de quatre autres couleurs: rouge, orange et bleu clair qui sont les trois couleurs secondaires de la charte graphique de Tron. ainsi que le rose qui était la couleur qui complétait le mieux les autres. Nous y avons ajouté des lumières de type spotlight dans les murs des quatre zones, de la même couleur que le mur ainsi qu'un spotlight blanc qui éclaire toute la carte et qui augmente encore l'effet néon.

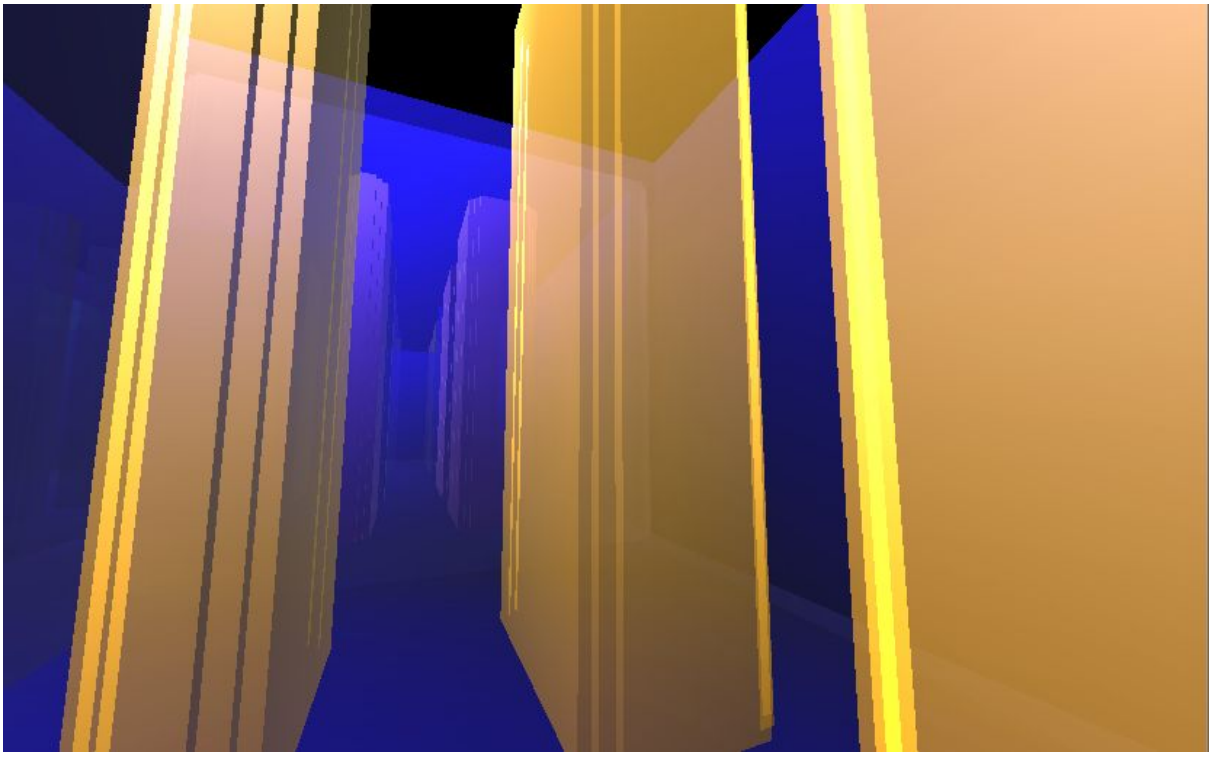


Figure 8: Ici, le personnage se situe devant la zone jaune.

5.4.3°) Création de l'environnement des exercices

Une fois le hub finalisé, il fallait commencer à réfléchir aux autres plates-formes. Nous avons décidé d'être prudents et de ne commencer que par la création d'une seule plate-forme secondaire pour le moment: la plus grande. Il s'agit d'une forêt que nous avons créée à l'aide des assets d'arbres présents sur le dépôt à notre disposition. Ces arbres ont été placés sur un terrain que nous avons légèrement terraformé afin qu'il ait du relief mais qu'il ne ressemble pas à une montagne. La zone créée est suffisamment grande pour faire les exercices d'endurance et il sera toujours possible de faire les autres exercices statiquement dans cet environnement si nous manquons de temps pour en créer d'autres.

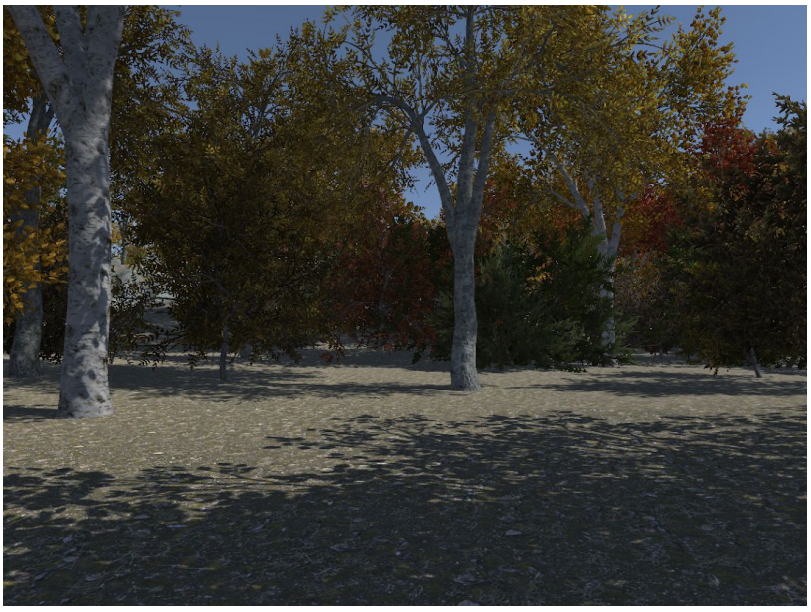


Figure 9: La scène secondaire: une forêt.

5.4.4°) Découverte de la Kinect

Nous avons nos scènes principales et les tests que nous avons faits pour le déplacement de la caméra ont été concluants. C'est donc le moment de passer à l'étape suivante: changer le mode de déplacement. Pour cela nous utilisons une kinect 2.0 qui est disponible à l'ISTIA car elle est utilisée par les étudiants d'EI5 IHMRV. Nous ne savons toutefois pas comment elle fonctionne et devons passer du temps sur le code de démonstration.

La kinect détecte le corps de la personne située devant elle et l'analyse. Elle rend au programme une série de points qui correspondent à la position dans l'espace des différentes parties du corps de la personne. On peut voir sur l'image ci-dessous qu'il y a de nombreux points (tête, mains, épaules, torse, genoux, pieds…) .

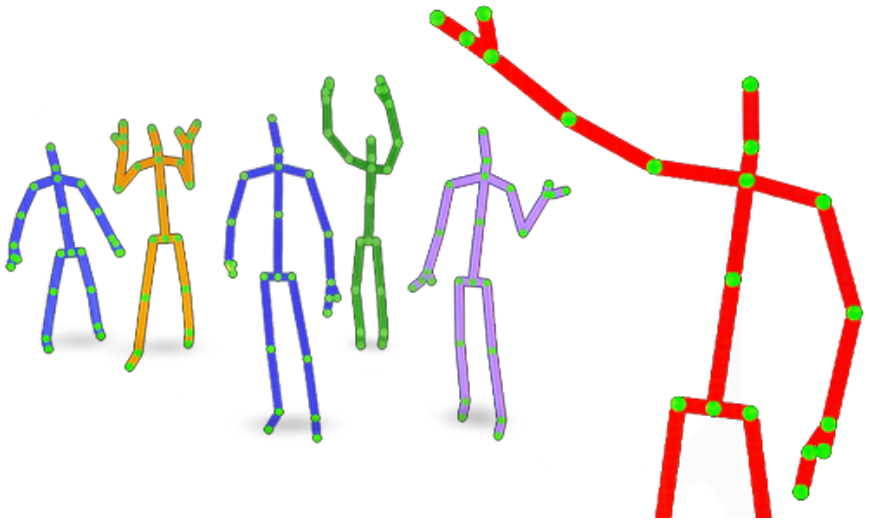


Figure 10: Représentation des points renvoyés par la Kinect.

Ces points peuvent être utilisés dans notre futur programme, par exemple, en détectant une variation de la hauteur du point du pied droit par rapport au point du pied gauche, il est possible de dire que la personne est en train de lever la jambe. C'est ce principe que nous avons utilisé pour simuler la marche du joueur et faire avancer la caméra à la place du clavier.

5.4.5°) Codage du déplacement

Nous avons donc établi une première classe permettant de détecter un écart sur la coordonnée verticale de la position des pieds. Cette classe nous donnant l'écart entre les deux pieds, nous l'avons lié la vitesse du joueur. De cette manière plus le pied est levé haut plus le joueur se déplace rapidement. Ce principe de fonctionnement est semblable au code emprunté au projet de Corentin Turcaud et Antonin Thilloux qui lui modifiait l'orientation du joueur en fonction de la position des épaules, script que nous utilisons pour atteindre ce même but mais que nous avons tout de même modifié afin d'arrêter la rotation si le joueur regarde en face de lui afin de ne pas avoir à onduler des épaules afin de regarder droit devant soi.

Le player que nous utilisions n'étant pas celui de Unity il nous a de même été nécessaire d'implémenter la gestion des collisions. Cette partie n'a pas été des plus compliquées car les scripts Unity étant aisément compréhensibles, nous avons pu adapter notre player controller afin de l'empêcher de devenir un passe-murailles.

5.4.6°) Codage des changements de scènes et des exercices

La question du déplacement dans le hub étant résolue, nous nous sommes attelés aux changements de scène. L'état actuel est qu'une fois arrivé dans l'une des quatre zones définies par les murs de couleurs différentes, le joueur doit placer sa main gauche sur son cou et sa main droite sur son bassin pour une durée de 1s afin de changer de scène. Ce geste a été choisi car il ne risquait pas d'être repéré comme étant réalisé au cours d'un exercice, les utilisateurs ayant la possibilité de sortir d'un exercice au moment où ils le souhaitent. Le joueur peut ensuite retourner au hub en réitérant l'action.

Figure 11: Dessin de Unity Chan qui effectue le geste permettant de changer de scène.

Ce résultat a été obtenu par la création d'un script utilisant le SceneManager de Unity auquel on a affilié deux listes de strings et un int privés afin de récupérer, à l'aide de triggers autour dans les zones susmentionnées, la zone dans laquelle le joueur souhaite se déplacer.

Ainsi l'arrivée dans une zone envoie un message au scene manager lui indiquant le nom de la zone dans laquelle il arrive, le scene manager assigne ensuite à l'int la place de la scène au sein de la première liste regroupant les noms de toutes les scènes. Si le joueur souhaite accéder à cet exercice, le scene manager va récupérer le nom de la scène dans ladite liste mais fait aussi correspondre cette scène avec un exercice de la seconde liste, chaque exercice et scène correspondantes possédant la même place dans leurs listes respectives. Si le joueur souhaitait réaliser un autre exercice, le fait de sortir du trigger réinitialiserait la valeur de l'int à celle du hub.

Une fois la scène changée, le scene controller, à l'aide de la commande DontDestroyOnLoad lui permettant, ainsi qu'au player, de ne pas disparaître à chaque changement de scène, modifie le mode de déplacement imposé au player afin de coller aux besoins de l'exercice. Ces exercices étant définis par leur héritage d'une classe aux attributs protégés, leur démultiplication est plus simple que leur intégration. Néanmoins, les quatre exercices actuellement intégrés demandent au player de réaliser soit des montées de genoux, soit des battements latéraux de jambes, des fentes ou des levées de bras mains liées. Chacun de ces exercices possède un compteur s'affichant en haut à droite mais sans objectif de répétitions afin de permettre aux utilisateurs de jauger par eux-mêmes de la qualité de leur effort.

Toute interface utilisateur étant manquante pour le moment, nous avons eu l'idée, afin de montrer l'exercice à réaliser, de placer un avatar face à l'utilisateur afin de lui montrer un geste bien réussi et de l'encourager à perfectionner le sien. Cet avatar est celui de Unity-chan, un asset complet mis à disposition gratuitement sur l'asset store auquel nous avons ajouté des animations de notre cru réalisées à l'aide de final IK.

Le scene manager étant partie de la scène du hub nous avons rencontré, pour une première partie du projet, un problème du fait que la kinect arrêtait bonnement et simplement de fonctionner lorsque certaines conditions étaient rencontrées. La première étant la présence de deux kinect manager dans la scène, script affilié au même objet portant le scene manager, et le second étant une surcharge de la kinect quant aux conditions requises afin d'effectuer une action.

Ces problèmes ont été très bien accueillis au sens où ils nous ont aussi permis de nettoyer le code et d'optimiser nos conditions. Nous en avons aussi profité pour étudier les multiples possibilités de remplacement quant à la présence du scene manager dans la scène du hub allant de la création d'un prefab instancié par la suite par un autre code jusqu'à la solution grossière de produire une seconde scène de hub sans la composante gênante.

6°)Bilan et perspective

Denis Sanchez:

Pour le moment, ce projet ne propose que 4 exercices. L'ajout d'exercices supplémentaires, eux-mêmes assez simples à rajouter car pouvant être créés en récupérant nos scripts actuels et en modifiant les joints utilisés et les valeurs afin de correspondre à des actions différentes, pourrait mener à une subdivision des zones afin de choisir un exercice en particulier. Ceci ajouterait une réflexion supplémentaire qui pourrait être intéressante au vu des multiples solutions disponibles. Néanmoins, ce projet nous a tout de même permis de nous familiariser avec un hardware au fonctionnement très intéressant qu'est la kinect 2.0. Il a aussi permis à mon collègue de se familiariser un tant soit peu à l'environnement proposé par Unity et nous a aussi forcé à nous intéresser à la modélisation 3D, le labyrinthe ayant été réalisé par nos soins. Nous avons eu à faire face à des problèmes de programmation qui nous ont demandé de nous rappeler de nos cours de C# mais aussi de réfléchir par nous- mêmes à des solutions pratiques à ensuite retraduire en code.

Néanmoins, loin de moi l'idée que ce projet est terminé. A part une documentation sur le côté nous n'avons pas de moyens d'indiquer au joueur quelle zone correspond à quel exercice ni le geste à réaliser afin de changer de zone. Il serait donc intéressant de proposer de continuer ce projet l'année a suivre afin de permettre à des étudiants de travailler sur l'interface utilisateur et la créativité encadrée.

Axel Raimbault:

Ce projet a été pour moi l'occasion de découvrir le domaine de la réalité virtuelle en trois dimensions que je ne connaissais pas vraiment auparavant. J'ai dû pour cela découvrir de nombreuses fonctionnalités d'Unity que je n'avais pas utilisées en cours. J'ai également appris beaucoup au sujet de la sclérose en plaques et des problèmes qu'ont les malades au quotidien.

En ce qui concerne le projet, nous avons réussi à atteindre tous nos objectifs. Puisque nous avons créé nous-mêmes nos objectifs, nous avons pu les changer à loisir. Nous avions par exemple décidé de faire un exercice d'endurance mais puisque nous avons codé les exercices de manière à ce qu'il soit possible de les arrêter à tout moment, un tel exercice n'avait plus de sens.

Il est toutefois impossible d'affirmer que ce projet est terminé. En effet, il existe de nombreux axes d'amélioration possibles que nous aurions pu réaliser si nous avions disposé d'un peu plus de temps. Je pense notamment aux environnements des exercices. Nous avions anticipé un manque de temps pour en créer quatre et n'en avons créé qu'un seul: la forêt. Nous aurions pu télécharger des scènes toutes faites dans l'asset store d'Unity mais nous tenions à les réaliser par nos propres moyens. D'autres axes d'améliorations seraient d'ajouter des exercices et de créer une notice d'utilisation pour que les joueurs ne soient pas perdus quant à l'exercice qu'ils doivent réaliser.

J'espère que ce projet pourra être utilisé de manière concrète un jour par des personnes atteintes de la sclérose en plaques afin qu'il ait eu une réelle utilité.

Bilan Global:

Nous avons réussi à terminer toutes les fonctions que nous souhaitions créer et nous sommes vraiment satisfaits du travail que nous avons fourni durant toute la durée du projet.

Chaque exercice et la plate-forme du menu sont parfaitement fonctionnels et plusieurs fonctionnalités auraient pu être ajourées avec un peu plus de temps.

Ce projet nous aura permis de mettre en oeuvre un grand nombre de connaissances que nous avons pu acquérir depuis notre entrée à l'ISTIA, en particulier en programmation en C# sur Visual Studio et la création de scènes sur Unity.

Ce projet nous a également permis de mettre en pratique tous les éléments de gestion de projets que nous avions abordés en cours ainsi que de constater leur importance.

7°)Sitographie

https://developer.microsoft.com/fr-fr/windows/kinect
https://forum.ni-mate.com/t/documentation-kinect-for-windows-2-kinect-for-xbox-one/287

8°)Annexes

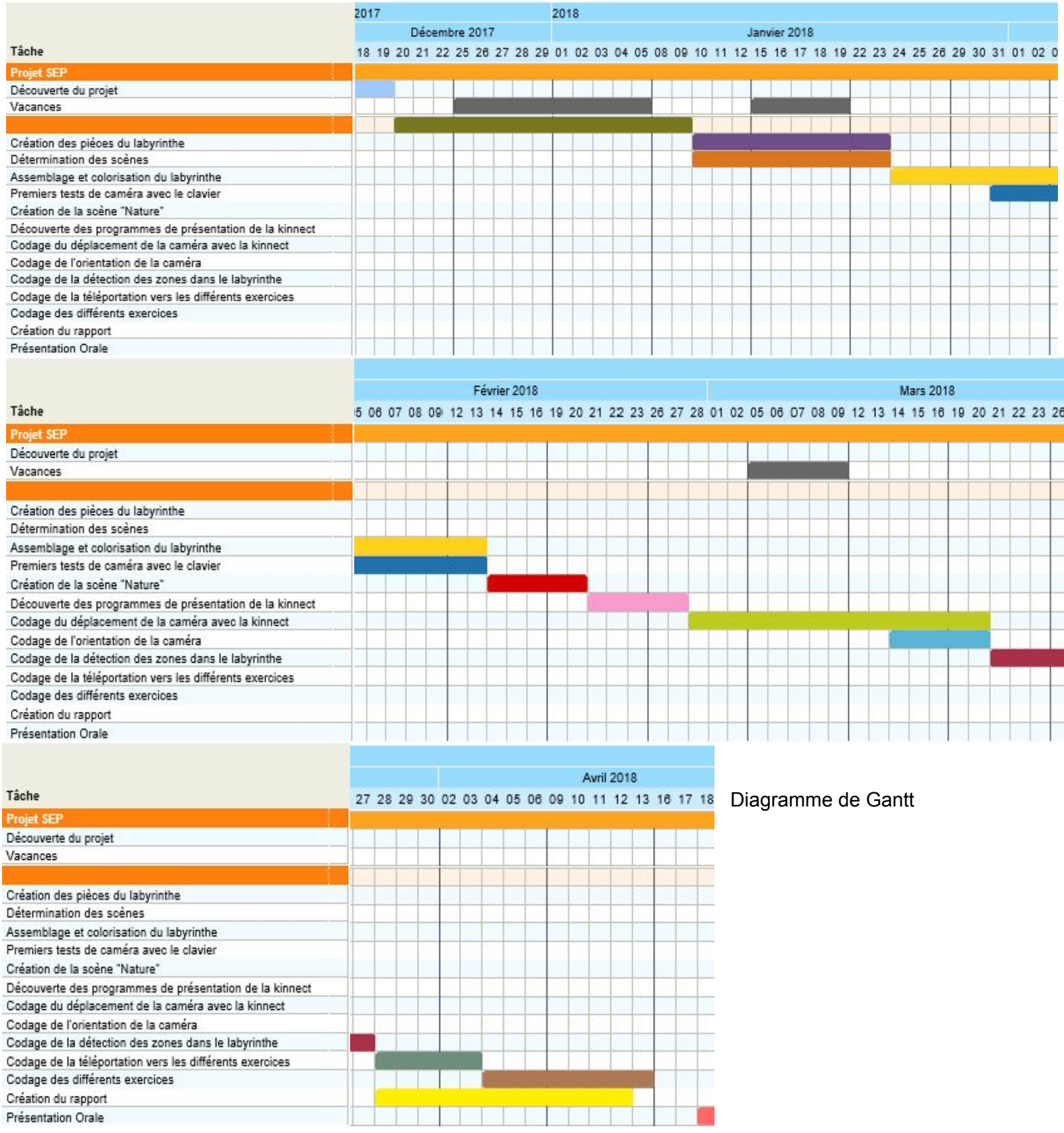
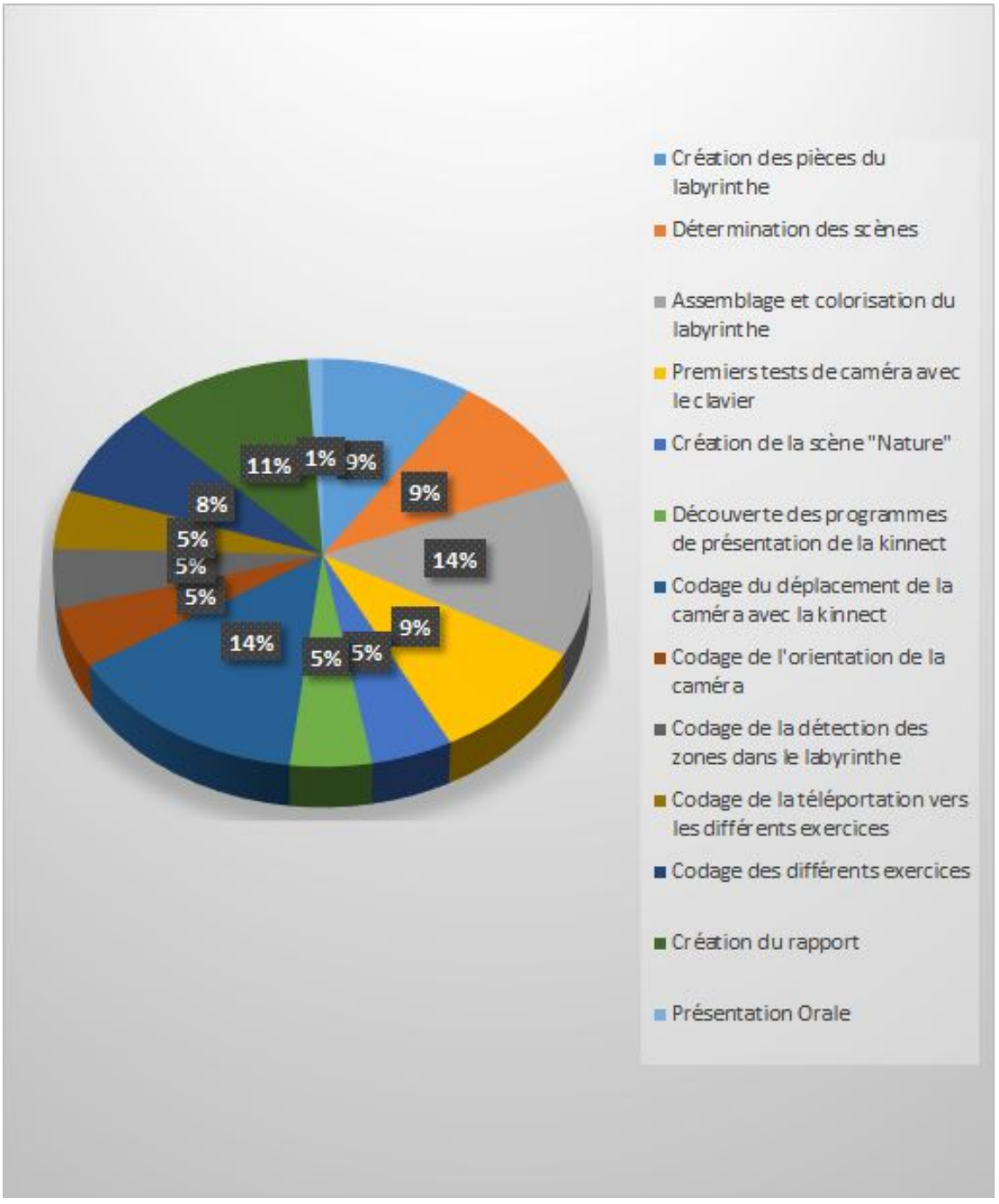


Diagramme de Gantt

Répartition en pourcentage de notre travail



Extraits de nos codes:

```
public class ZoneSwitch : MonoBehaviour
{

    public JointType _LeftHand = JointType.HandLeft;
    public JointType _RightHand = JointType.HandRight;
    public JointType _jointNeck = JointType.Neck;
    public JointType _RightHip = JointType.HipRight;
    public GameObject _bodySourceManager;
    private BodySourceManager _bodyManager;

    // Use this for initialization
    void Start()
    {
        _bodySourceManager = GameObject.Find("FPSController");
    }

    // Update is called once per frame
    void Update()
    {
        if (_bodySourceManager == null) return;
        _bodyManager = _bodySourceManager.GetComponent<BodySourceManager>();
        if (_bodyManager == null) return;
        Body[] data = _bodyManager.GetData();
        if (data == null)
        {
            return;
        }
        else
        {
            foreach (var body in data)
            {
                if (body == null) continue;
                if (body.IsTracked)
                {
                    var posMainDroite = body.Joints[_RightHand].Position;
                    var posMainGauche = body.Joints[_LeftHand].Position;
                    var posCou = body.Joints[_jointNeck].Position;
                    var posHip = body.Joints[_RightHip].Position;

                    float ecarthx = posMainGauche.X - posCou.X;
                    float ecartcx = posMainDroite.X - posHip.X;

                    //Debug.Log (ecart);
                    if (Math.Abs(ecarthx) < 0.1 && Math.Abs(ecartcx) < 0.1)
                    {
                        gameObject.GetComponent<SceneController>()._cameo = true;
                    }
                    else { gameObject.GetComponent<SceneController>()._cameo = false; }
                    break;
                }
            }
        }
    }
}
```

```
using System;
using UnityEngine;
using UnityEngine.UI;

namespace navigation.movement
{
    public class ExerciseReps : MonoBehaviour
    {

        private Text _compte;
        private velocity.KinectExercise _exercice;
        // Use this for initialization
        void Start()
        {
            _exercice = FindObjectOfType<velocity.KinectExercise>();
            _compte = GetComponentInChildren<Text>();
        }

        // Update is called once per frame
        void Update()
        {
            if(_exercice == null) { _exercice = FindObjectOfType<velocity.KinectExercise>(); }
            if (SceneController.GetCurrentScene() == "Hub")
            {
                _compte.text = "";
            }
            else
            {
                _compte.text = "" + _exercice.getcompteur()+_exercice.getflag();
            }
        }
    }
}
```

```
using UnityEngine;
using Windows.Kinect;
using System;

namespace navigation.movement.velocity
{
    public class KinectEcartPieds : AbstractVelocity
    {
        public JointType _PiedDroit = JointType.FootRight;
        public JointType _PiedGauche = JointType.FootLeft;
        public GameObject _bodySourceManager;
        private BodySourceManager _bodyManager;

        void Start()
        {
            _bodySourceManager = GameObject.Find ("FPSController");
        }


        void Update()
        {
            if (_bodySourceManager == null) return;
            _bodyManager = _bodySourceManager.GetComponent<BodySourceManager> ();
            if (_bodyManager == null) return;
            Body[] data = _bodyManager.GetData ();
            if (data == null)
            {
                Velocity = 0;
                return;
            }
            foreach (var body in data)
            {
                if (body == null) continue;
                if (body.IsTracked)
                {
                    var posPiedDroit = body.Joints [_PiedDroit].Position;
                    var posPiedGauche = body.Joints [_PiedGauche].Position;

                    float ecart = posPiedDroit.Y - posPiedGauche.Y;
                    //Debug.Log (ecart);
                    if (ecart > 0.1 || ecart < -0.1)
                    {
                        Velocity = System.Math.Abs(ecart) * 125;
                    }
                    else {Velocity = 0;}
                    InputRecieved ();
                    break;
                }
            }

        }
    }
}
```

```
public class SceneController : MonoBehaviour
{

    private static Scene _currentscene;
    private static string[] _scenes = { "Hub", "RedZone", "BlueZone", "PinkZone", "YellowZone" };
    private static string[] _deplacements = { "KinectEcartPieds", "KinectBrasPompe", "KinectEcartPiedsFente", "KinectEcartHancheGenou", "KinectEcartPiedsBattement" };
    private static Deplacement _deplacement;
    private int _scenezone;
    public bool _cameo = false;
    public bool _isRunning = false;
    private Transform _hubpos;

    public string getzone()
    {
        return _scenes[_scenezone];
    }

    public void setzone(string zn)
    {
        for (int i = 0; i < _scenes.Length; i++)
        {
            if (_scenes[i] == zn)
            {
                _scenezone = i;
                return;
            }
        }
    }

    private void Awake()
    {

        DontDestroyOnLoad(this.gameObject);

    }
```

```
    // Update is called once per frame
    void Update()
    {
        if(_cameo == true && _isRunning == false)
        {
            StartCoroutine("Switch");
            _isRunning = true;
        }
        else
        {
            if(_cameo == false && _isRunning == true)
            {
                StopCoroutine("Switch");
                _isRunning = false;
            }
        }
    }

    public static string GetCurrentScene()
    {
        _currentscene = SceneManager.GetActiveScene();
        //Debug.Log(_currentscene.name);
        _deplacement = FindObjectOfType<Deplacement>();
        return _currentscene.name;
    }

    public static void SwitchScene(int i)
    {
        SceneManager.LoadScene(_scenes[i]);
        GetCurrentScene();
        FindObjectOfType<Rigidbody>().isKinematic = true;
        FindObjectOfType<Rigidbody>().isKinematic = false;
        _deplacement.SetVelocity(_deplacements[i]);
        if (i == 0)
        {
            Destroy(FindObjectOfType<SceneController>().gameObject);

        }

        else
        {
            _deplacement.SetSteering("KeySteering");
            FindObjectOfType<SceneController>().transform.position = Vector3.zero;
            FindObjectOfType<SceneController>().transform.rotation = Quaternion.identity;
        }
    }

    IEnumerator Switch()
    {
        yield return new WaitForSeconds(1.0f);
        SwitchScene(_scenezone);
        setzone("Hub");
    }
}
```

```
    // Update is called once per frame
    void Update()
    {
        if(_cameo == true && _isRunning == false)
        {
            StartCoroutine("Switch");
            _isRunning = true;
        }
        else
        {
            if(_cameo == false && _isRunning == true)
            {
                StopCoroutine("Switch");
                _isRunning = false;
            }
        }
    }

    public static string GetCurrentScene()
    {
        _currentscene = SceneManager.GetActiveScene();
        //Debug.Log(_currentscene.name);
        _deplacement = FindObjectOfType<Deplacement>();
        return _currentscene.name;
    }

    public static void SwitchScene(int i)
    {
        SceneManager.LoadScene(_scenes[i]);
        GetCurrentScene();
        FindObjectOfType<Rigidbody>().isKinematic = true;
        FindObjectOfType<Rigidbody>().isKinematic = false;
        _deplacement.SetVelocity(_deplacements[i]);
        if (i == 0)
        {
            Destroy(FindObjectOfType<SceneController>().gameObject);

        }

        else
        {
            _deplacement.SetSteering("KeySteering");
            FindObjectOfType<SceneController>().transform.position = Vector3.zero;
            FindObjectOfType<SceneController>().transform.rotation = Quaternion.identity;
        }
    }

    IEnumerator Switch()
    {
        yield return new WaitForSeconds(1.0f);
        SwitchScene(_scenezone);
        setzone("Hub");
    }
}
```

Plateforme pour la rééducation de la sclérose en plaques

Projet réalisé par Axel Raimbault et Denis Sanchez
Projet encadré par Paul Richard

Résumé:
Ce projet a été réalisé afin d'aider à la rééducation des personnes souffrant de la sclérose en plaques, une maladie neurologique qui cause généralement des problèmes pour bouger les muscles. La mission était de créer une plate-forme de réalité virtuelle sur le logiciel Unity qui permette aux malades de faire des exercices qui soient ludiques et adaptés à leurs capacités. La méthode de capture des mouvements choisie est la Kinect de Microsoft qui rend possible des calculs pour le code du projet. En ce qui concerne l'environnement en trois dimensions, la charte graphique a été respectée: le menu est le labyrinthe de Pac-Man avec des couleurs et un jeu de lumières inspirés des films Tron. Certaines zones du labyrinthe permettent de se téléporter dans d'autres environnements pour effectuer les exercices.

Mots-Clés:
Rééducation, maladie neurologique, plate-forme, réalité virtuelle, exercices.

Summary:
This project was made in order to help the reeducation of persons suffering from multiple sclerosis, a neurological disease which often causes problems to move one's muscles. The objective was to create a virtual reality platform on the software Unity which allows the ill person to do playful and well adapted exercices. The method to capture the movements that was chosen is Microsoft's Kinect as it allows to make calculations in the project's code. In regards to The 3D environment, the graphical charter was respected: The menu is Pac-Man's maze with the colors and lights inspired from the Tron movies. Some of the maze's zones allow to teleport the camera in other environments to do the exercices.

Keywords:
Reeducation, neurological illness, platform, virtual reality, exercices.