Conception de nouvelles interfaces pour la locomotion dans des environnements virtuels en interaction avec les humains virtuels

Rémi Pineau

Rapport de stage

Conception de nouvelles interfaces pour la locomotion dans des environnements virtuels en interaction avec les humains virtuels

Tuteurs: *Maud MARCHAL & Julien PETTRÉ*

INRIA-IRISA - Equipe MIMETIC/VR4I

PINEAU Rémi

Master 2 Recherche Informatique Rennes 1

Résumé

Les interfaces de locomotion en environnement virtuel améliorent la sensation de déplacement de l'utilisateur en stimulant plusieurs de ses sens. Ces interfaces sont gérées par des fonctions de transfert qui transforment le mouvement réel en un déplacement virtuel le plus réaliste possible.

Dans ce rapport, nous décrivons certaines interfaces et nous nous focalisons sur l'évaluation de l'une d'entre elles, le Joyman, développé au sein de l'Inria-Irisa. Nous développerons des modèles pour la locomotion physique.

1 Table des matières

1 Introduction

1.1 Présentation générale :

Dans le cadre de mon stage de fin d'études de Master 2 Recherche Informatique Images et Interactions de l'Université de Rennes 1(ISTIC), j'ai effectué un stage à l'IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires) au sein des équipes VR4i et Mimetic.

L'IRISA est associé à INRIA à travers un nombre important d'équipes communes au sein du centre Inria Rennes-Bretagne Atlantique.

L'IRISA, créé en 1975, est une unité mixte de recherche (UMR 6074) dont les établissements partenaires sont le CNRS, l'Université de Rennes 1 (établissement principal), l'INSA de Rennes et l'ENS Cachan (antenne de Bretagne).

1.2 Contexte

La locomotion dans un environnement virtuel est un besoin essentiel en réalité virtuelle, les utilisateurs ayant besoin de se mouvoir pour interagir avec les objets virtuels.

Nous allons porter notre étude sur une interface dédiée à la marche dans les mondes virtuels : le Joyman.

Nous présentons les différentes étapes de contrôle de la locomotion du Joyman

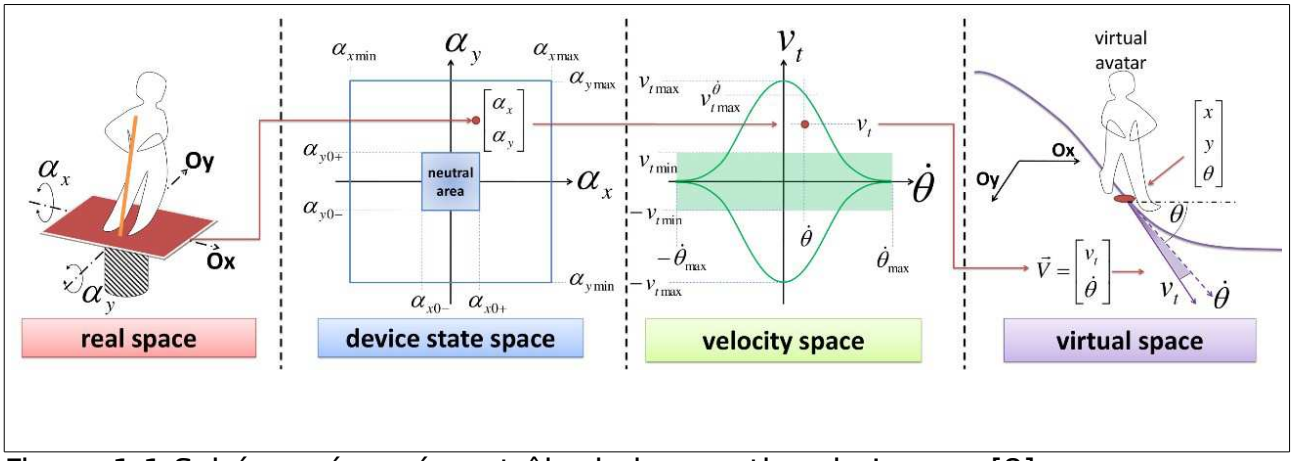


Figure 1.1 Schéma résumé contrôle de locomotion du Joyman[8]

1.3 Sujet du stage

Les objectifs du stage sont l'élaboration de modèles réalistes pour la locomotion de l'utilisateur dans un environnement virtuel.

Nous allons étudier la locomotion physique à travers le Joyman.

2 Etat de l'art

2.1 Interface de locomotion dans les environnements virtuels

Les neurosciences ont beaucoup étudié la marche humaine. Elle repose sur trois sens : la vision, la proprioception, l'équilibrioception(sens de l'équilibre). Ces sens mobilisent trois systèmes : oculaire, somato-sensoriel et vestibulaire. Le corps humain les combine pour piloter sa marche. La plupart des interfaces jouent sur la proprioception.

Nous sélectionnons des critères définis dans le traité de la réalité virtuelle[1] pour nous permettre de classer les différentes interfaces.

Nous avons choisi comme critères : **l'encombrement** qui se représente en terme de volume et de poids, **l'immersion** qui représente l'aptitude de l'interface à immerger l'utilisateur au sein de l'environnement virtuel, la **précision** qui est la mesure des distances et des angles, la **prise en main** qui représente la facilité d'utilisation et enfin **la liberté de mouvement** qui est l'un des critères les plus importants. Si l'interface ne permet les déplacements que dans une direction et au sein d'un environnement limité, alors l'immersion et le retour des sensations sont faibles.

On compte de nombreuses interfaces de locomotion, allant du simple Joystick au tapis roulant multi-directionnel. Dans cette sous-partie nous dressons un descriptif sommaire de plusieurs d'entre elles.

- **Joystick ou périphérique manuel**

Figure 2.1 photo joystick [2]

Descriptif matériel/fonctionnement :

L'interface la plus facile à employer reste le joystick ou tout autre périphérique d'entrée(clavier,souris....). La précision, la liberté de mouvement et la prise en main sont maximales, mais l'immersion est minimale. L'encombrement est infime.

- **Gait master :**

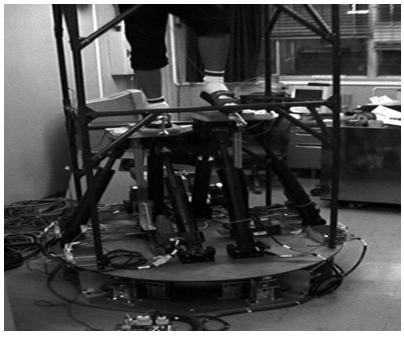


Figure 2.2 Prototype du Gaitmaster omnidirectionnel[3]

Descriptif matériel/fonctionnement :

C'est une interface de déplacement sur place.

Il s'agit de deux plates-formes mono-directionnelles montées sur une plate-forme tournante. Le système enregistre le mouvement des pieds et détermine la direction et la vitesse du déplacement. Le Gait Master[3] est peu encombrant, assez précis et permet une grande liberté de mouvement. Cependant, la prise en main et la complexité de mise en place en font une interface assez peu adaptative.

- **CyberCarpet :**

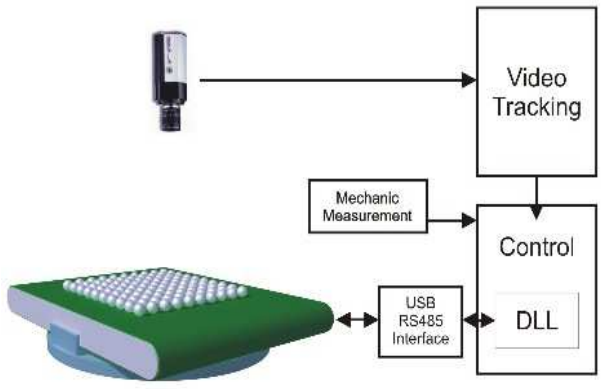


Figure 2..3 schéma représentant le fonctionnement du Cybercarpe[4]

Descriptif matériel/fonctionnement :

Le Cybercarpet [4] est un tapis roulant monodirectionnel monté sur une plate-forme tournante. L'utilisateur entreprend le mouvement et le tapis se met en marche, tournant suivant son orientation de l'utilisateur.

Néanmoins, cette interface a comme principaux défauts une imprécision relative, la liberté de mouvement restreinte par la taille limitée du tapis et le fait de devoir toujours se placer au centre pour que les réponses soient bonnes. [5]

- **CyberSphere** :

Figure 2.4 représentation cybersphere [6]

Descriptif matériel/fonctionnement :

La CyberSphere [6] est l'interface de locomotion omnidirectionnelle la plus intuitive.

L'utilisateur en avançant fait tourner la sphère dans laquelle il est confiné, lui permettant ainsi de rester sur place.

Cependant, même si la sensation de déplacement est importante, l'immersion et l'encombrement font que cette interface est surtout utile à titre de recherche.

- **ChairIO** :

Figure 2.5 ChairIO [7]

Descriptif matériel/fonctionnement :

Le ChairIO [7] se compose d'un siège pivotant monté sur une plate-forme tournante 4. C'est l'une des rares interfaces de locomotion à utiliser les fonctions équilibrioceptives. Cependant,l'immersion et la précision sont limitées (du fait de la position de l'utilisateur), en dépit d'un encombrement réduit.

– **Joyman** :

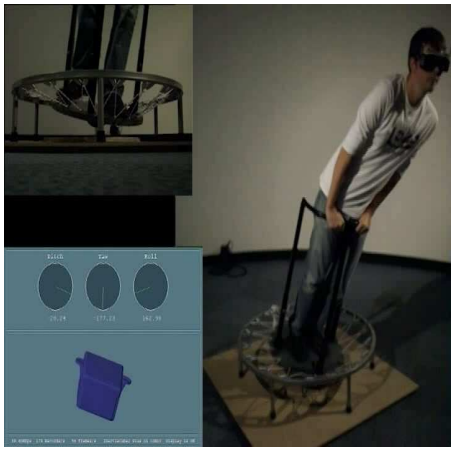


Figure 2.6 Joyman [8]

Descriptif matériel/fonctionnement :

Le Joyman [8] est développé dans les équipes VR4I/Mimetic. L'utilisateur se tient debout sur une plate-forme articulée et se penche pour indiquer la direction souhaitée de déplacement dans le monde virtuel. Son fonctionnement est semblable à celui d'un Segway sauf que l'utilisateur reste statique.

Figure 2.7 photo segway [8]

Le Joyman est basé sur les fonctions Equilibrioception de l'homme. Le corps humain devient joystick. Il suffit de se pencher pour avancer.

Nous avons ainsi décrit un aperçu des différentes interfaces de locomotion pour la réalité virtuelle.

Nous dressons un tableau avec les différentes interfaces et les critères de sélection(encombrement,immersion,précision,complexité,mouvement) que l'on a choisis précédemment.

Critère / Interface	encombrement	immersion	précision	complexité	mouvement
joystick	minimale	minimale	maximale	minimale	maximal
gaitmaster[2]	moyen	bonne	bonne	forte	bon
cybercarpet[3]	faible	faible	moyenne	moyenne	moyenne
cybersphere[4]	important	faible	bonne	importante	grand
chairIO[6]	faible	très moyenne	moyenne	faible	grand
Joyman[7]	faible	bonne	À préciser	faible	grand

2.2 Lois de commande du Joyman

Nous allons dresser un aperçu des différentes lois de commandes qui régissent le Joyman.

Pour qu'un mouvement puisse d'être réaliste il faut qu'il soit analogue à celui d'un humain, la principale difficulté vient du fait que la locomotion n'est pas holonome(toutes les positions de l'espace ne sont pas atteignables directement).[10] [11]

Dans le cas du Joyman, l'utilisateur se tient debout sur une plate-forme articulée et se penche pour indiquer la direction souhaitée de déplacement dans le monde virtuel. L'inclinaison induite est l'entrée de la loi pilotant la locomotion virtuelle.

La loi de commande est inspirée par la nature non-holonomic de la marche humaine[12]

De la position et l'orientation du Joyman, nous pouvons calculer le vecteur vitesse virtuelle en utilisant notre modèle de locomotion et la loi de commande[7].

On peut déterminer également la nature du mouvement.

En analysant, la locomotion à partir du centre de gravité d'un humain au cours d'un déplacement, on constate une trajectoire oscillatoire.

On s'appuie sur une étude [13] qui présente une méthode de détection automatique permettant de détecter les virages à partir d'une trace capturée de locomotion humaine, en se projetant dans l'espace de représentation de vitesse-courbure.

Une démarche envisageable est celle de l'utilisation de l'espace des vitesses .

Les auteurs [12] ont étudié les limites du déplacement humain et ont pu intégrer des valeurs minimales et maximales aux différents paramètres utilisés .

Ces lois de commandes diffèrent suivant la nature de l'interface et prennent part à la finesse, la précision et le réalisme du déplacement au sein de l'environnement virtuel. L'optimisation de ces lois est un critère déterminant de la qualité de l'interface.

Après avoir dresser les lois de commandes déjà existantes, nous allons mettre en œuvre la modélisation physiques de la locomotion avec le Joyman.

3 Modélisation physique de la locomotion

3.1 Introduction

Nous allons essayer d'élaborer un modèle permettant d'utiliser les propriétés physiques du monde réel dans un univers virtuel à l'aide du Joyman, principalement la gravité. L'ensemble est réalisé à l'aide de unity 3D, outil permettant la création de contenus interactifs tels que des visualisations architecturales ou des animations 3D en temps réel.

Ce logiciel a pour principale caractéristique d'utiliser un éditeur de script *monodevelop* qui peut exploiter les langages c#, *javascript, Boo* et de contenir un environnement de développement intégré.

Plusieurs scènes ont été conçues pour exploiter la locomotion avec les modèles de skieurs que nous avons créés.

Nous présenterons ces scènes par plusieurs captures d'écrans qui ont été prises en particulier une scène contenant une piste créée pour tester les modèle de skieur, et une scène de démonstration permettant la locomotion dans un environnement virtuel avec un décor que nous avons créé.

On décrit à l'aide de captures d'écran prises la scène « piste ».

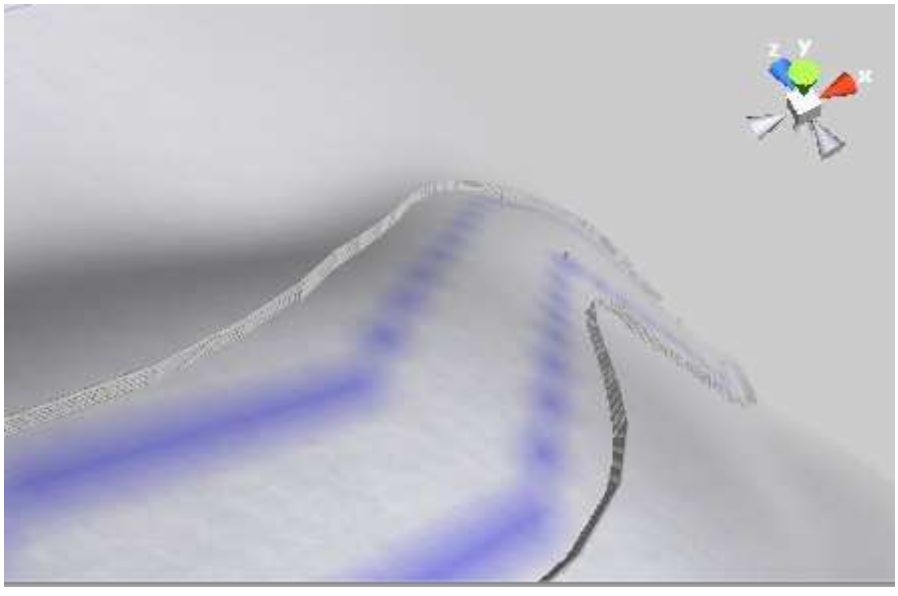


Figure 3.1 Capture d'écran 1 : vue d'une partie de la « piste »

Cette vue permet d'avoir un aperçu du relief de la piste. On peut remarquer que la pente est plutôt raide et les virages assez serrés. Cette piste est donc tout indiquée pour le test.

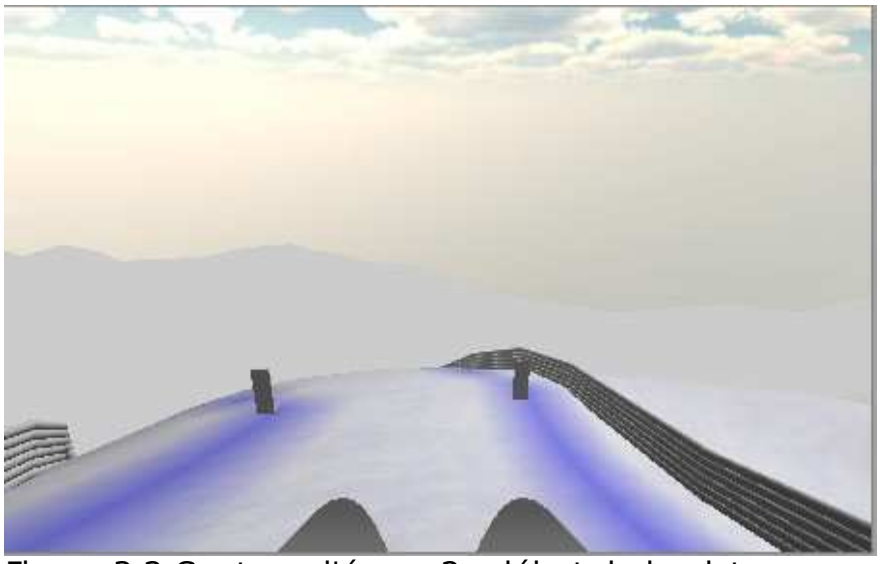


Figure 3.2 Capture d'écran 2 : début de la piste

Le skieur prend ainsi de la vitesse rapidement, il est intéressant de pourvoir tester les modèles de skieur sur cette pente pour s'assurer de leur stabilité en présence de virages et avec une vitesse importante.

On décrit maintenant la scène « démonstration ».

Figure 3.3 Capture d'écran 1 scène démonstration

Figure 3.4 Capture d'écran 2 scène démonstration

La scène « démonstration » comporte un environnement virtuel avec du décor. Elle permet de parcourir de larges distances dans un monde virtuel.

Sous Unity 3D, il est possible d'importer des modèles 3D avec différents logiciels : *maya,cinema 4d,3ds max,Cheetah 3d, Modo, Lightwave, Blender.* Pour la construction des scènes et modèles de skieur, on utilise le logiciel *Blender*, et l'on exportera de préférence avec le format fbx qui est le mieux géré par Unity même s'il gère de nombreux autres formats. Les meshs représentent les modèles importés dans la scène.

Unity 3D fonctionne avec 3 langages : c#, javascript et boo. Ces langages intègrent par défaut la conception objet. Seuls le javascript et le c# ont été utilisés. Nous expliquerons quelques extraits du controller pour les modèles de skieur.

L'information permettant et utilisant la captation de l'axe vertical et horizontal se notera respectivement dy et dx.

On utilisera toujours ce procédé pour récupérer les déplacements dx et dy de l'utilisateur dans la suite de ce rapport.

Les différents de types d'algorithmes utilisant cette méthode sont ainsi exploitables à partir d'un Joystick ou d'un Joyman.

```
function FixedUpdate () {

// Get the horizontal and vertical axis.
    // By default they are mapped to the arrow keys.
    // The value is in the range -1 to 1
    var dy : float = Input.GetAxis ("Vertical");
    var dx : float = Input.GetAxis ("Horizontal");

}
```

C'est la méthode FixedUpdate qui sera employée par rapport à la méthode update pour le déplacement pour assurer une meilleur stabilité des comportements physiques.

Pour visualiser le skieur nous utilisons deux types de vues :

- vue à la troisième personne (la caméra est directement placée à l'intérieur de l'objet, les postions sont exprimées dans le repère local x,y,z).
- caméra qui suit le skieur ou les skis.

Nous avons réalisé un script *camera manager* permettant de passer directement d'une vue à une autre simplement à l'aide d'un raccourci clavier ou joystick. Nous allons pouvoir utiliser ce script pour chacun des deux modèles de skieur présentés dans la suite.

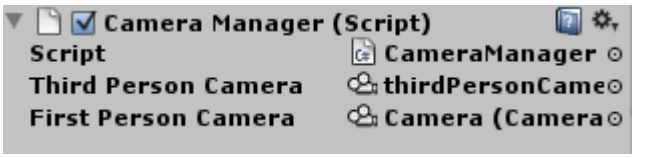


Figure 3.5 vue générale script camera manager

Sur la Figure 3.5 on a fait un lien avec les différentes caméras. La « firstPerson camera » se situe dans l'objet, la « thirdpersoncamera » est positionnée à l'extérieur de l'objet. Avec les raccourcis clavier on peut ainsi passer d'une caméra à une autre. Nous allons donc utiliser les modèles de caméras sur les modèles de skis que nous avons réalisés.

La « thirdpersocamera » contient un script « SmoothFellow » qui permet de suivre le skieur.

Figure 3.6

Nous allons maintenant passer à la mise en place de modèles de skieurs dont on a cherché à maximiser le réalisme.

3.2 Modèle physique avec frottement

Nous allons commencer par décrire la physique d'un skieur qui descend une pente.

$$m \cdot \vec{a_x} = \vec{P_x}$$

$$m \cdot a_x = m \cdot g \cdot \sin(\alpha)$$

$$a_x = g \cdot \sin(\alpha)$$

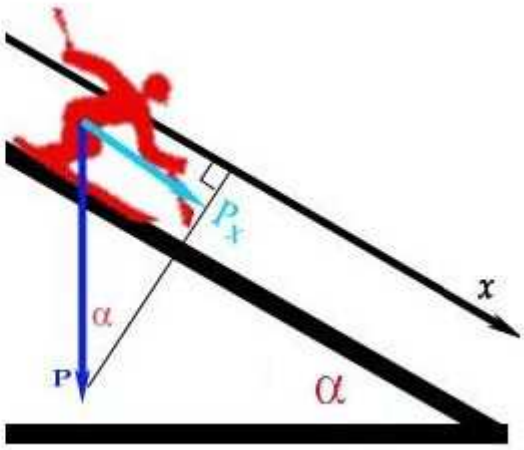


Figure 3.7 schéma extrait article [14]

L'accélération du skieur dépend de l'inclinaison de la piste, le skieur est ralenti par les frottements des skis sur la neige.

La loi de Coulomb exprime sous une forme simplifiée l'intensité des forces de frottements qui s'exercent entre deux solides. Selon que ces solides glissent ou non l'un contre l'autre, on parle de glissement (frottement dynamique) ou d'adhérence (frottement statique).

Dans notre cas, nous étudions le frottement entre le skieur et la pente.

Nous allons essayer de construire un modèle réaliste qui respecte la loi de Coulomb, afin que les calculs de frottement soient directement pris en compte par le moteur de Unity 3D. Nous allons créer un matériel qui sera ajouté à chacun des colliders qui sont en jeu dans la locomotion du skieur.

A partir d'une table, on cherche les coefficients statique et dynamique du contact des skis avec la neige.

Johnson, Clifford V. Friction. Microsoft Encarta Online Encyclopedia 2007.	Materials in Contact Static Friction Kinetic Friction Waxed Ski on Snow 0.1 0.05	0.1 (static) 0.05 (kinetic)

Figure 3.8 Extrait table [15]

On commence par créer un matériel « snow », dans lequel on affecte les coefficients de friction vus précédemment. A partir de ces coefficients on conçoit le matériel :

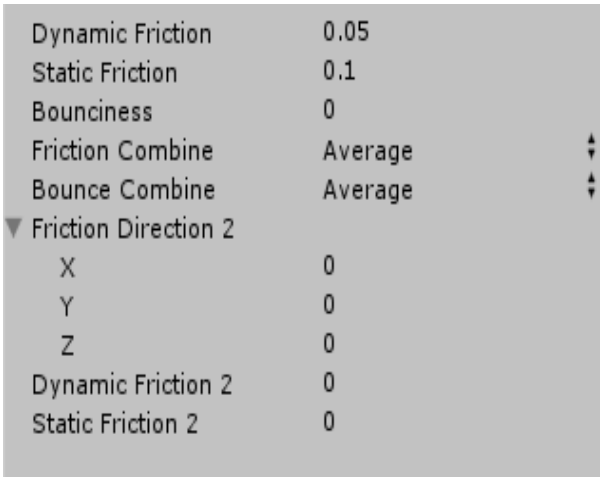


Figure 3.9 Vue du matériel créé

Nous allons utiliser ce matériel sur les différents colliders pour que le moteur physique puisse faire des calculs exacts selon les coefficients affectés. Nous allons donner un aperçu de la classe « mère » collider, dont hérite tous les colliders et nécessaire au calcul des forces de frottements.

enabled	Enabled Colliders will collide with other colliders, disabled Colliders won't.
attachedRigidbody	The rigidbody the collider is attached to.
isTrigger	Is the collider a trigger?
material	The material used by the collider.
sharedMaterial	The shared physic material of this collider.
bounds	The world space bounding volume of the collider.

Figure 3.10 [16]

Nous avons conçu un terrain pour chaque scène. Nous associons un terrain collider pour chacun d'entre eux. Ce dernier gère le contact avec le skieur. Nous donnons un aperçu du terrain collider, cette classe hérite de la classe collider.

Variables

terrainData	The terrain that stores the heightmap

Inherited members

Inherited Variables

enabled	Enabled Colliders will collide with other colliders, disabled Colliders won't.
attachedRigidbody	The rigidbody the collider is attached to.
isTrigger	Is the collider a trigger?

Figure 3.11 [17] classe terrain collider

Nous avons affecté le matériel « snow » au terrain collider.

Figure 3.12 vision générale terrain collider

Une fois que nous avons appliqué ce matériel au terrain, nous devons également l'appliquer au modèle de skieur que l'on a choisi.

Pour construire le premier modèle de skieur, nous avons employé des box colliders pour gérer la collision du skieur. Nous dressons un aperçu partiel de cette classe qui hérite aussi de la classe collider.

Variables

center	The center of the box, measured in the object's local space.
size	The size of the box, measured in the object's local space.

Inherited members

Inherited Variables

enabled	Enabled Colliders will collide with other colliders, disabled Colliders won't.
attachedRigidbody	The rigidbody the collider is attached to.
isTrigger	Is the collider a trigger?
material	The material used by the collider.

Figure 3.13 [18]

Voici un descriptif de l'aspect général d'un box collider :

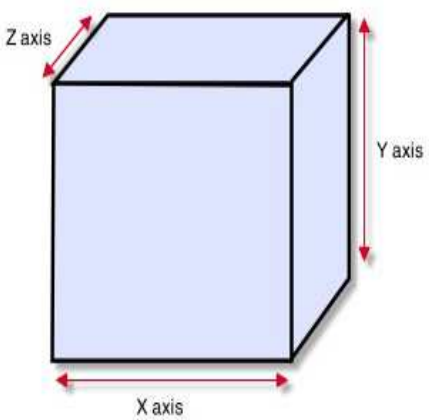


Figure 3.14 [23]

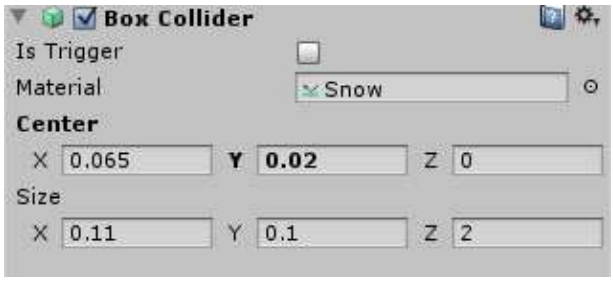


Figure 3.15

Sur chaque ski, droit et gauche on affecte un box collider.

Nous avons donc deux box colliders, chacun sur un ski(droit et gauche),qui sont représentés en vert auxquels on affecte le matériel « snow ». Figure 3.16

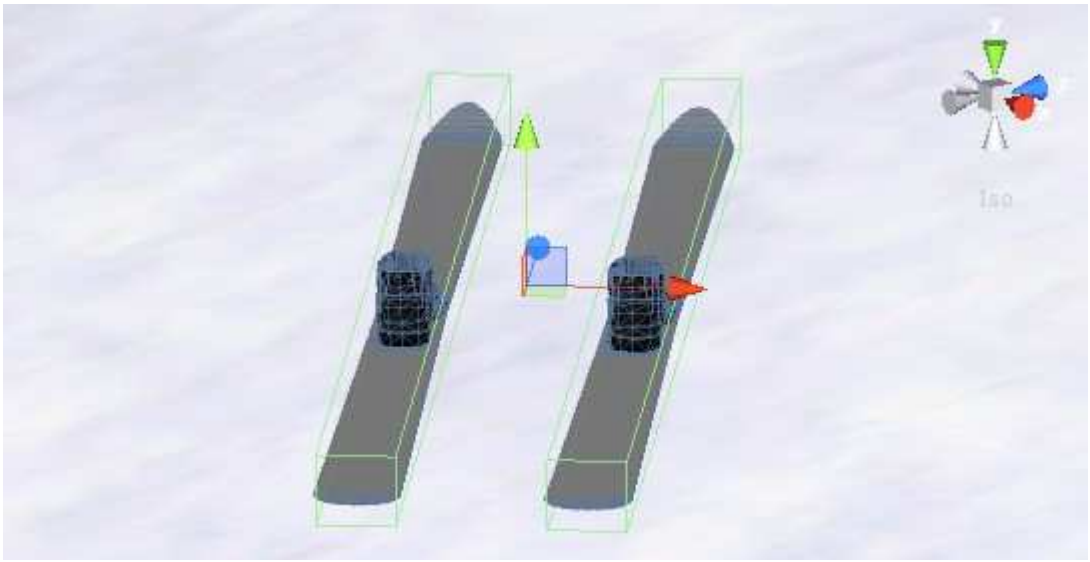


Figure 3.16 modèle de ski avec collider en vert

Le gamobject « skis » a donc la structure hiérarchique suivante

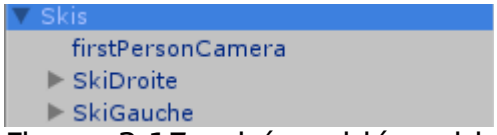


Figure 3.17 schéma hiérarchie skis

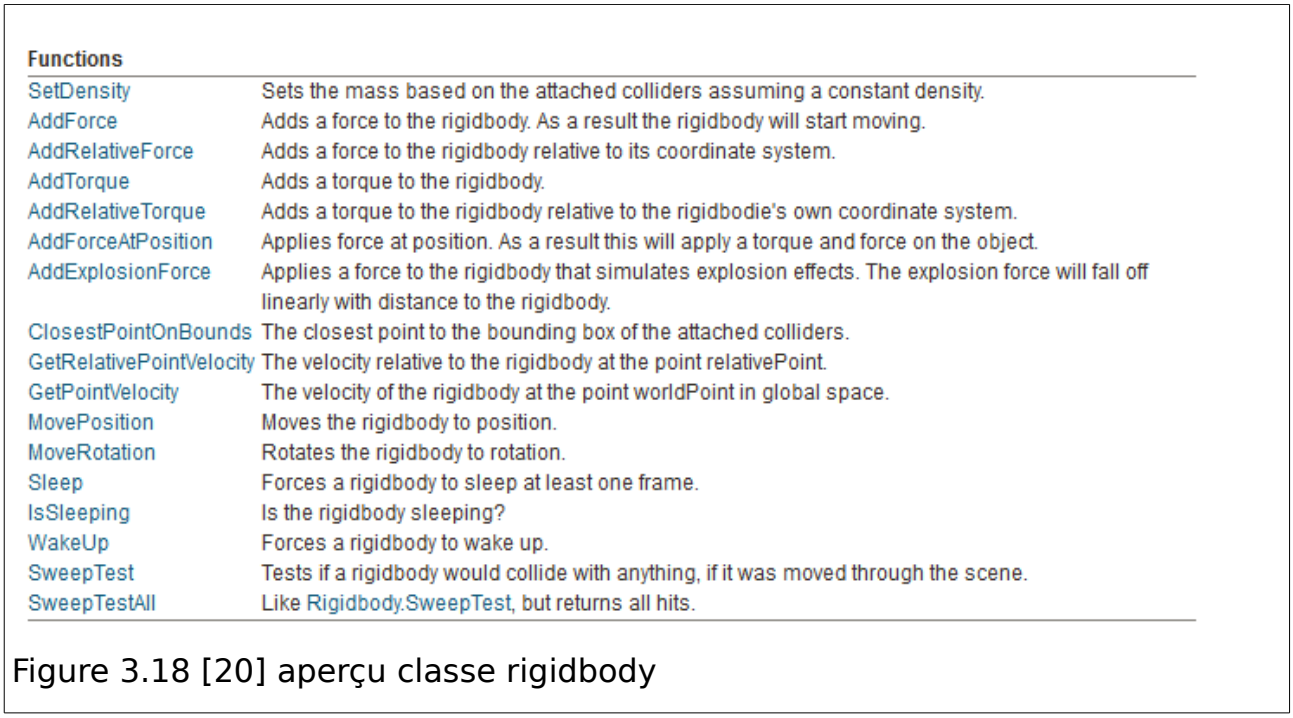
La « firstPersonCamera » sur l'objet permet d'avoir une vue à la première personne. On équipe le gameobject « skis » d'un corps physique(rigidbody), pour qu'il obéisse aux lois de la Physique, en particulier à la gravité.

Les forces de frottements sont calculées par le moteur physique entre le collider du terrain et celui des deux skis. Ainsi de manière automatique, lorsque la pente diminue les deux skis ralentissent et peuvent s'arrêter au bout d' un certain temps si la pente devient nulle.

Le script du controller centralisera tous les traitements, le freinage du skieur, l'accélération, le déplacement et la rotation des skis.

Le déplacement s'effectue sur le corps physique de l'objet entier auquel on ajoute des forces, ce qui permettra le déplacement du skieur dans l'environnement virtuel. A l'aide de la documentation, nous donnons un aperçu de quelques fonctions qui s'appliquent au corps physique(rigidbody).

Voici un aperçu de la classe rigidbody.



Functions	
SetDensity	Sets the mass based on the attached colliders assuming a constant density.
AddForce	Adds a force to the rigidbody. As a result the rigidbody will start moving.
AddRelativeForce	Adds a force to the rigidbody relative to its coordinate system.
AddTorque	Adds a torque to the rigidbody.
AddRelativeTorque	Adds a torque to the rigidbody relative to the rigidbodie's own coordinate system.
AddForceAtPosition	Applies force at position. As a result this will apply a torque and force on the object.
AddExplosionForce	Applies a force to the rigidbody that simulates explosion effects. The explosion force will fall off linearly with distance to the rigidbody.
ClosestPointOnBounds	The closest point to the bounding box of the attached colliders.
GetRelativePointVelocity	The velocity relative to the rigidbody at the point relativePoint.
GetPointVelocity	The velocity of the rigidbody at the point worldPoint in global space.
MovePosition	Moves the rigidbody to position.
MoveRotation	Rotates the rigidbody to rotation.
Sleep	Forces a rigidbody to sleep at least one frame.
IsSleeping	Is the rigidbody sleeping?
WakeUp	Forces a rigidbody to wake up.
SweepTest	Tests if a rigidbody would collide with anything, if it was moved through the scene.
SweepTestAll	Like Rigidbody.SweepTest, but returns all hits.

Figure 3.18 [20] aperçu classe rigidbody

Nous allons décrire le script de contrôle à travers l'aperçu de méthodes qui le composent. Certaine méthodes qui le composent : OnCollisionEnter(), OnCollisionExit() permettent de gérer la collision du skieur avec le sol. Ces méthodes sont utiles pour déterminer si les skis sont sur le sol ou non et de déclencher la méthode move().

La script comporte également OnApplicationQuit() qui permet de quitter l'application, void OnGUI() pour afficher des informations. On peut citer une méthode de gestion des bugs dans la méthode update().

Nous détaillons les parties du controller qui permettent la locomotion des skis

```
void Move() {

        float dy = Input.GetAxis("Vertical");
        float dx = Input.GetAxis("Horizontal");
```

pour l'accélération

```
        if(dy > 0){
```

on limite l'accélération lorsqu'on a atteint une certaine vitesse

```
            if(transform.rigidbody.velocity.magnitude*3.6f < 30){
                transform.rigidbody.AddForce(transform.forward * dy*30);
            }
```

```
                else {
                        //print ("Avant");
                        transform.rigidbody.AddForce(transform.forward * dy);
                }
        }

        // If speed not equals 0
        if (transform.rigidbody.velocity.magnitude != 0){
```

pour le freinage

```
                if(dy < 0){
                        //print ("Arrière");
                        transform.rigidbody.AddForce(transform.forward *
Input.GetAxis("Vertical")*30);
                }
```

lorsqu'on tourne à droite

```
                if(dx > 0){

                        transform.rigidbody.AddForce(transform.right *
dx*transform.rigidbody.velocity.magnitude*5);
                                Quaternion deltaRotation =
Quaternion.Euler(eulerAngleVelocity * 0.5f);
                        rigidbody.MoveRotation(rigidbody.rotation * deltaRotation);
                            transform.rigidbody.AddForce(-transform.rigidbody.velocity*0.2f);

                }
```

lorsqu'on tourne à gauche

```
                if(dx < 0){

                                transform.rigidbody.AddForce(-transform.right * dx*-
transform.rigidbody.velocity.magnitude*5);
                        Quaternion deltaRotation = Quaternion.Euler(eulerAngleVelocity *
0.5f);
                        rigidbody.MoveRotation(rigidbody.rotation * deltaRotation);
                                transform.rigidbody.AddForce(-
transform.rigidbody.velocity*0.2f);
                           }


                }
        }
        }
```

Nous avons ajouter une instruction pour le ralentissement du skieur dans les virages, on ajoute une force négative proportionnelle au vecteur vitesse.

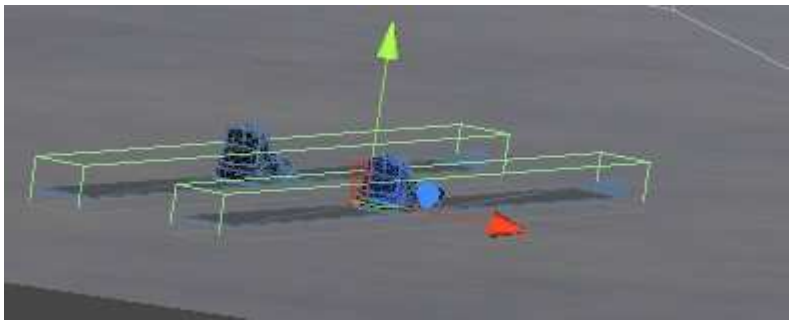


Figure 3.19 skis avec collider en vert

Comportement des deux skis lors d'une rotation, nous détaillons la méthode skiRotation() qui permet la rotation des skis, en fonction du vecteur vitesse qui à été modifié lors du déplacement.

```
void skiRotation() {

		float angleYbis = Mathf.Atan2(transform.rigidbody.velocity.z,
transform.rigidbody.velocity.x)*Mathf.Rad2Deg;

		if(transform.rigidbody.velocity.magnitude*3.6f > 1f){
			transform.localEulerAngles = new Vector3(transform.eulerAngles.x,
-angleYbis+90, transform.eulerAngles.z);
		}
	}
```

L'option Max Time permet de limiter la durée de la démonstration.

Figure 3.20 character controler

Le modèle est testé sur les différentes scènes que nous avons créées. Sur la scène « piste », le modèle de skieur reste stable avec une vitesse supérieure à 130 km/h. Il est cependant nécessaire d'utiliser le freinage pour arriver en bas de la piste.

Pour la scène « démonstration », il est assez facile de faire le parcours de début jusqu'à la fin, le modèle est donc stable sur les grandes distances.

Nous avons pu créer un modèle qui permet un calcul des frottements selon la loi de Coulomb, nous allons maintenant chercher à maximiser le réalisme au niveau du matériel.

3.3 Modèle avec roue

Le but cette fois est de chercher à maximiser le réalisme du mouvement avec les composants matériels du Joyman. Les roues sont les composants de capteur inertiel. Cette fois on ne cherchera pas exactement un réalisme physique, c'est à dire à l'exactitude des calculs des forces de frottements selon la loi de Coulomb qui s'exercent sur le skieur.

De plus, l'utilisation de roues permet de glisser sur la neige. Ainsi le fait de pouvoir déraper peut considérablement augmenter le réalisme du comportement du skieur sur la neige. L'utilisateur aura des sensations analogues à la réalité.

Au cours de mes recherches [19], j 'ai trouvé une idée d'architecture de skis avec des roues qui m'a semblé intéressante.

Nous utiliserons une architecture avec 4 roues, elle paraît en effet pouvoir assurer la stabilité du skieur. Pour ce modèle nous utiliserons des wheel colliders, des spheres colliders peuvent-être aussi utilisées. Ce type collider est semblable à une roue. Nous dressons un aperçu des champs de classe wheelcollider, mais il existe également des méthodes associées que l'on ne détaillera pas.

center	The center of the wheel, measured in the object's local space.
radius	The radius of the wheel, measured in local space.
suspensionDistance	Maximum extension distance of wheel suspension, measured in local space.
suspensionSpring	The parameters of wheel's suspension. The suspension attempts to reach a target position
mass	The mass of the wheel. Must be larger than zero.
forwardFriction	Properties of tire friction in the direction the wheel is pointing in.
sidewaysFriction	Properties of tire friction in the sideways direction.
motorTorque	Motor torque on the wheel axle. Positive or negative depending on direction.
brakeTorque	Brake torque. Must be positive.
steerAngle	Steering angle in degrees, always around the local y-axis.
isGrounded	Indicates whether the wheel currently collides with something (Read Only).
rpm	Current wheel axle rotation speed, in rotations per minute (Read Only).

Figure 3.21 [21] aperçu classe wheel collider

Le collider peut être caractérisé comme une roue avec un centre et un radius (rayon).

Il nous faut développer un controller qui permette un traitement approprié.

De manière analogue au premier modèle, le script du controller centralise tous les traitements.

Contrairement au modèle précédant, la locomotion se fait directement à partir des colliders. On va décrire un modèle de skieur assez simpliste avec 4 wheels colliders. Ce composant est le plus souvent utilisé pour les modèles de voiture et mais peut-être utilisé pour d'autres véhicules terrestres.

On équipe le gameobject « skis » de quatre wheels collider qui sont représentés en vert sur le schéma, les roues seront invisibles pour l'utilisateur, lorsque lors du déroulement de la scène.

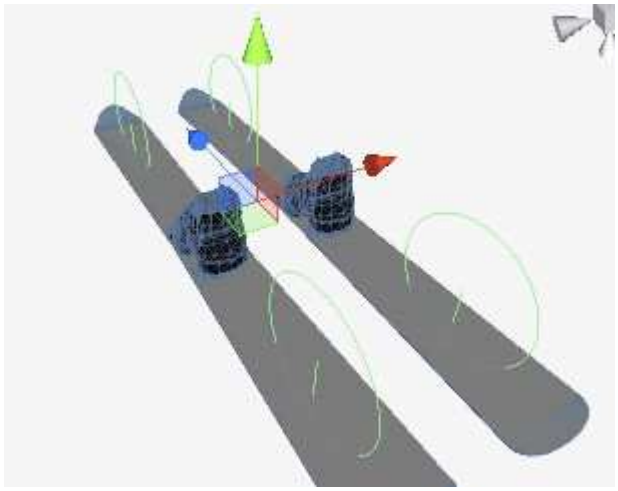


Figure 3.22 schéma skieur architecture globale

Le positionnement de chaque collider est dans le repère local (x,y),(-x,y),(-x,-y), (x,-y). Nous dressons un aperçu de la hiérarchie du gameobject créé. Les colliders sont en vert et invisible par l'utilisateur lors de l'exécution de la scène.

Figure 3.23 hiérarchie skieur avec roues

Nous avons construit le modèle de gestion du controller de manière analogue à celui du premier skieur. Nous avons équipé le gameobject « skiroue » d'un rigidbody de la même manière que pour le premier modèle afin qu'il soit soumis aux lois physiques en particulier à la gravité. On configure un centre de masse rigidbody.centerOfMass = Vector3(0, -0.05, 0), dans le repère local. Dans le premier modèle on a laissé le centre de masse se calculer automatiquement par le système à partir de collider rattaché au corps physique de l'objet. Le skieur est plus stable avec un centre de gravité bas qui est légèrement négatif, ainsi le skieur est moins susceptible de se renverser.

Le contrôle d'un wheel collider est dirigé essentiellement par 3 paramètres :

- *motorTorque* qui fait la force motrice des roues
- *brakeTorque* qui fait la force de freinage des roues
- *steerAngle* (angle de conduite), correspond à l'angle de la rotation de la roue

Les forces de frottements qui s'appliquent sur les roues sont calculées à part par le moteur physique. On ne peut pas utiliser le terrain collider dans ce cas comme dans le premier modèle.

Le contrôleur affectera les réglages de frottements identiquement aux quatre roues.

Deux types de frottements sont configurables et s'appliquent à chacune des roues: sideways(latéral) et forward(avant). Il possède la même structure de type WheelFrictionCurve. Ce sont les réglages de ces paramètres qui permettent le dérapage des roues.

WheelFrictionCurve

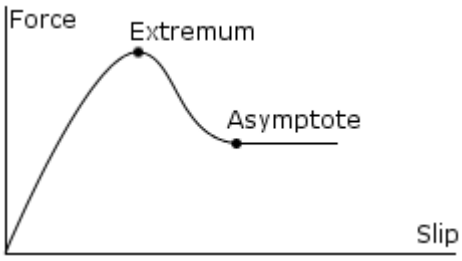


figure 3.24 [22]

Extremum	Slip/ValueCurve's extremum point
Asymptote	Slip/ValueCurve's asymptote point
StiffnessMultiplier	for the Extremum Value
	Asymptote Value (default is 1).

Nous pouvons donc paramétrer pour les deux types de frottements sur chacun les paramètres suivants :

- extremum slip/valuecurve
- asymptote sllip/valueCurve's
- StiffnessMultiplier

Pour les autres paramètres de la roue que l'on peut configurer sur le wheel collider on utilisera également : *Suspension Distance* (distance maximum de suspension de la roue), *Target Position* (valeur distance de suspension au repos 0 extention, 1 compression), *Spring*(plus cette valeur est grande plus la Target Position est atteinte rapidement), *Damper*(plus plus cette valeur est grande plus Target Position est atteinte lentement).

Dans le cas d'un modèle de skieur les valeurs Suspension Distance et Target Position doivent être très faibles.

L'architecture globale peut-être vue comme analogue à celle d'une voiture mais il est important que le contrôle se fasse sur le collider de chacune des roues, sans centralisation du déplacement à travers l'ajout de force sur le corps physique en entier.

En effet, lors de nombreux tests que j'ai effectués, l'utilisation de modèles de voiture ne permet pas le réalisme et apporte un comportement trop mécanique. De plus si on veut se rapprocher du réalisme il faut que le mouvement soit impulsé directement dans le collider.

Le script du controller a accès aux quatre colliders des roues.

Pour plus de simplicité et de cohésion, l'ensemble des paramètres est appliqué de façon identique aux quatre roues par le controller.

Nous allons expliquer le fonctionnement du controller qui permet de la locomotion de l'objet, qui est réalisé en c#.

La fonction start() lance les fonctions storage() et setupfrottement, et initialise le centre de masse de l'objet. La méthode storage() met les liens avec les colliders dans un tableau pour pouvoir appliquer des traitements sur les quatre colliders.

On va détailler les quelques parties de la fonction fixedupdate qui permettent le déplacement.

On applique les forces motrices au quatre roues :

```
for(var w:WheelCollider in wheels){
w. motorTorque=motor_max * motor;
}
```

On applique les forces de freinage :

```
for(var w:WheelCollider in wheels){
w.brakeTorque = brake_max * brake;
}
```

Les variables motor et brake sont calculées à partir de dy et sont comprises respectivement entre [-1,0] et [0,1].

Les quatre instructions suivantes permettent de déclencher la rotation des roues à partir d'un déplacement dx :

```
frontWheel1.steerAngle = steer_max * dx;
frontWheel2.steerAngle = steer_max * dx;
rearWheel1.steerAngle = -steer_max * dx;
rearWheel2.steerAngle = -steer_max * dx;
```

On va montrer de manière graphique le fonctionnement algorithmique de la rotation d'une roue.

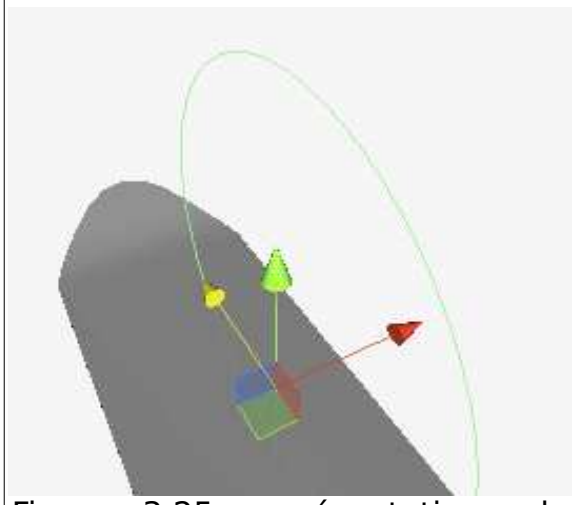


Figure 3.25 représentation wheel collider

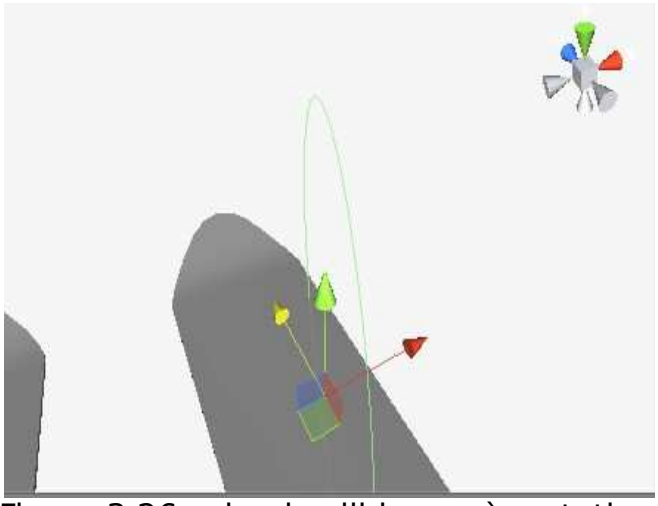


Figure 3.26 wheel collider après rotation

Exemple de rotation d'une roue à partir du dx capturé : dans le repère local composé de trois flèches directionnelles, la rotation s'effectue selon l'axe y (flèche verte).

Il faut de préférence que l'angle de conduite(steermax) soit inférieur à 45 degrés pour assurer une bonne cohérence de la rotation sur la roue.

Mouvement des quatre roues dans un virage

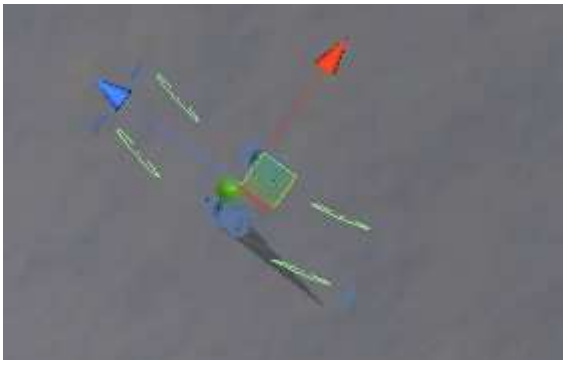


Figure 3.27 rotation ski avec 4 roues

Cette capture d'écran nous montre le comportement des quatre roues tournant ensemble, qui permettent la rotation du skieur.

On peut remarquer que la liste d'instructions pour le déplacement est plus courte que lors du premier modèle.

La vision du script dans l'interface graphique nous permet d'avoir une vision globale, et autorise l'utilisateur à régler de manière graphique les forces de frottements, angle de navigation, force de freinage et d'accélération maximum, qui s'appliqueront directement aux quatre roues.

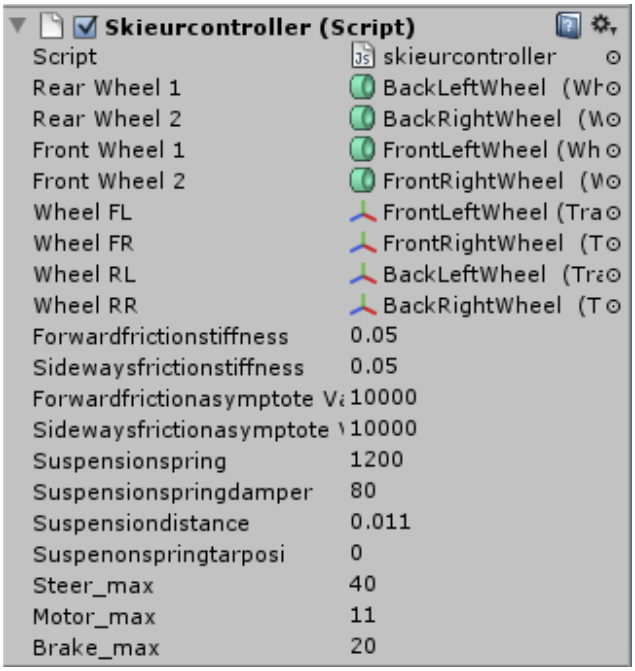


Figure 3.28 vision globale controller

Le controller à accès aux quatre colliders

On peut régler directement les frottements sur les quatre roues.

L'angle maximum de conduite pour chaque roue
L'accélération maximum
Le freinage maximum

Nous testons le modèle de skieur sur la scène « piste ». On constate qu'il est très difficile d'arriver jusqu'au bout de la piste avec les paramètres *forward friction* et *sideways friction* inférieurs à 8000. En dessous de cette valeur les dérapages sont très importants et le skieur se retourne très rapidement.

Sur la scène « démonstration » le comportement du skis demeure instable.

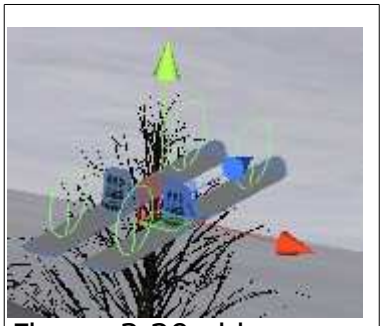


Figure 3.29 skis

Il faut tester et réduire l'échelle de l'objet selon les dimensions (x,y,z) jusqu'à atteindre une bonne stabilité. Une fois le modèle stabilisé on peut faire la démonstration d'un bout à l'autre en parcourant des distances importantes.

On peut aussi rajouter une vitesse de confort à l'aide d'un script.

Maintenant nous allons décrire quelques procédés qui permettent d'augmenter l'immersion de l'utilisateur.

3.4 Facteurs pouvant améliorer l'immersion de l'utilisateur

Figure 3.30 capture d'écran scène « piste »

Plusieurs facteurs peuvent améliorer l'immersion de l'utilisateur. :

- lors du contact du skis avec neige, il peut y avoir émission de particules(neige)
- faire tomber de la neige.
- laisser une traînée sur la neige
- rajouter un ciel sur la scène,

- mettre en place un brouillard(fog) sur le décor, qui permet d'afficher le relief au fur et à mesure.

Nous allons maintenant détailler le fonctionnent du Joyman et intégrer les modèles de skieurs avec les environnements virtuels que nous avons créés.

4 Intégration avec le Joyman

4.1 Description du Joyman

Nous allons faire une description des techniques du Joyman. Comme décrit précedemment, le Joyman est une interface équilibrioceptive développée par l'équipe VR4i/Mimetic. La capture des mouvements de l'utilisateur se fait via une centrale inertielle.

On récupère ainsi les deux angles de rotation qui peuvent aussi être vus comme des quantités dx et dy utilisées par les lois de commandes . La tension de la corde au niveau de la plate-forme permet d'ajuster l'assiette et la stabilité du Joyman en position initiale. Initialement, les mains ne servent que d'appui, de pouvoir se pencher sans avoir peur de tomber. Mais, ces dernières n'intervenant pas directement dans la locomotion, nous pouvons développer des techniques d'interaction supplémentaires (on peut imaginer un clavier ou autre interface d'entrée sur la rambarde du Joyman). Le Joyman peut être considéré comme un joystick à taille humaine, l'utilisateur contrôlant l'inclinaison du périphérique non pas avec sa main mais à l'aide de tout son corps.

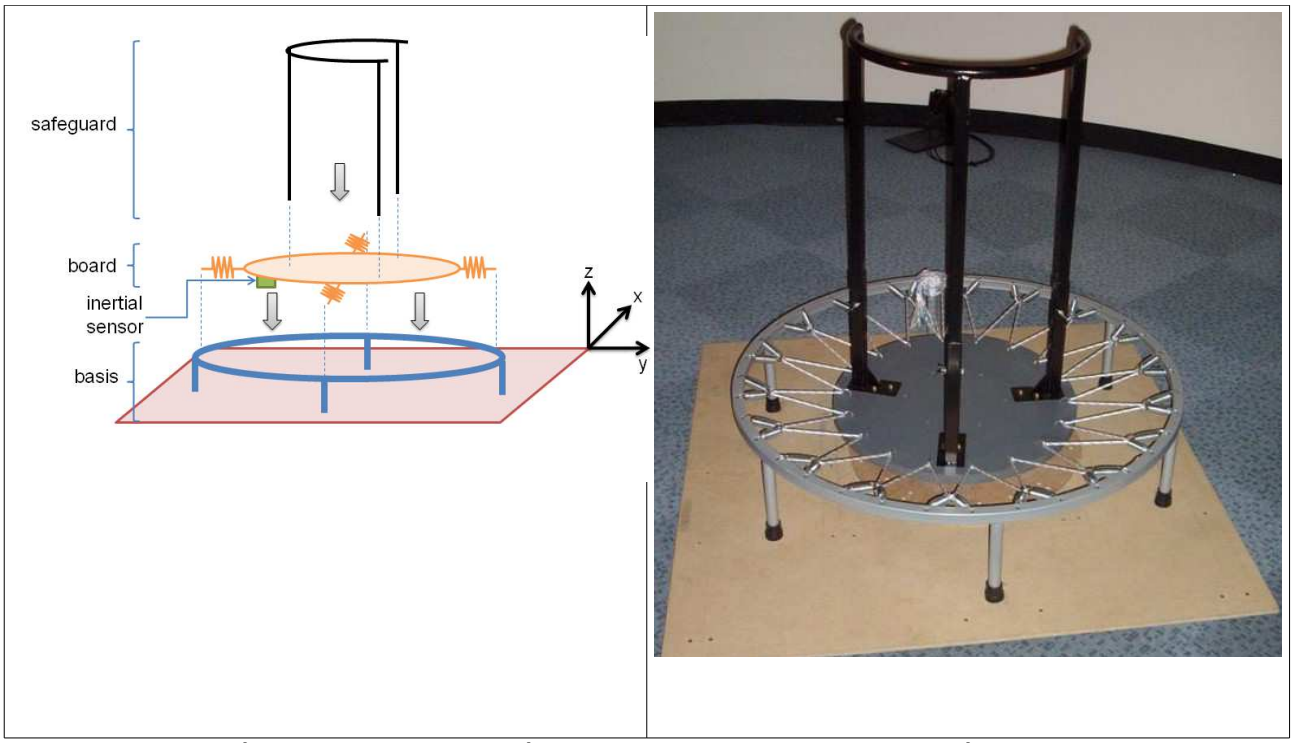


Figure 4.1 Schéma avec les différents composants et vue réelle du Joyman [8]

Nous allons tester les modèles de skieurs dans les environnements virtuels qui ont été créés avec le Joyman.

4.2 Intégration du modèle pour son utilisation avec le Joyman

Le Joyman est donc notre principal périphérique d'entrée, utilisé au sein d'un environnement virtuel crée grâce au moteur de unity 3D.

Lorsque l'utilisateur se penche, la centrale inertielle mesure l'inclinaison du plateau central(dx et dy), les paramètres sont récupérés par le système, déclenchant ainsi le mouvement du skieur dans la scène. Le déroulement est ensuite projetée à l'écran ou dans un casque.

Figure 4.3 utilisatur utilisant un Joyman[7]

Le Joyman est employé avec la plate-forme de réalité virtuelle Immersia, permettant ainsi une forte immersion de l'utilisateur dans l'environnement de réalité virtuelle.

5 Conclusion

Cette partie regroupe la liste de mes travaux, ainsi que les problèmes ou éléments pouvant être résolus, voir améliorés.

5.1 Résumé du travail réalisé

☐ Réalisation d'un système de caméra pour les skieurs:

- système de changement de vue

☐ Réalisation de deux modèles de skieurs :

- modèle respectant les lois physiques de frottements du monde réel
- modèle permettant permettant le glissement sur la neige

☐ Réalisation de plusieurs scènes permettant l'exploitation de la locomotion :

– une scène « piste » de contenant une piste pour tester les modèles

– une scène « démonstration » permettant la locomotion dans un environnement virtuel assez étendu

☐ Création de systèmes de particules :

- système dans lequel les particules peuvent tomber par le haut (comme de la neige)

- système qui émet des particules lorsque les skis touchent la neige

5.2 Limites et perspectives

☐ Limites :

Le deuxième modèle proposé (avec les roues) reste à améliorer au niveau de la physique(respect de la loi de Coulomb).

☐ Perspectives :

On peut le développement d'un simulateur d'apprentissage pour le ski en utilisant le deuxième modèle. L'utilisateur aura des sensations analogues à la réalité grâce la possibilité d'effectuer des dérapages.

Autre possibilité envisageable est la mise en place d'exercices de rééducation des membres inférieurs

Bilan :

Lors de ce stage j'ai rencontré quelques difficultés, j'ai eu besoin d'un tempas d'adaptation. En effet je ne connaissais pas la programmation graphique.

Ce stage m'a ainsi permis de mettre en pratique les connaissances théoriques et pratiques acquises lors de ma formation, et d'en acquérir de nouvelles sur l'univers de la réalité virtuelle.

Ce projet a également été l'occasion de me plonger dans le milieu de la recherche. Ce stage à l'IRISA m'a permis de rencontrer des chercheurs et de partager leurs expériences.

Des améliorations restent à apporter aux modèles proposés durant le mois de stage qui me reste.

Ce stage est une expérience enrichissante, il me permet de clore ma deuxième année de Master 2 Informatique et me donne un aperçu du travail dans le milieu de la recherche.

6 Références

[1] P. Fuchs, B.Arnaldi, A. Chauffaut, S. Donikian, and T. Duval. *Traité de la Réalité Virtuelle*, volume 1,chapter Les interfaces spécifiques de la localisation corporelle, pages 234–241. Presses de l'Ecole des Mines de Paris, second edition edition, 2006

[2]http://radialmind.blogspot.fr/2010/05/joystick-pong-how-to-write-custom-usb.html

[3] H.Iwata, H.Yano, and F.Nakaizumi. Gait master: A versatile locomotion interface for uneven virtual terrain. Virtual Reality Conference, IEEE, 0:131, 2001.

[4] De Luca, A.; Mattone, R.; Giordano, P.R. The motion control problem for the cybercarpet. In ICRA, pages 3532–3537. IEEE, 2006.

[5] Schwaiger, M.C., Thummel, T., Ulbrich, H. A 2d-motion platform: The cybercarpet.*World Haptics Conference*, 0:415–420, 2007.

[6] K.J. Fernandes, V. Raja, and J. Eyre. Cybersphere: the fully immersive spherical projection system. Commun. ACM, 46:141–146, September 2003.

[7] Beckhaus, S., Blom, KJ, Haringer, M . ChairIO – The Chair-Based Interface.

[8]M.Marchal, J.Pettré, S.Pineau and A.Lécuyer. Joyman: a Human-Scale Joystick for Navigating in Virtual Worlds IEEE Symposium on 3D User Interfaces (3DUI), 2011

[9] http://public.iutenligne.net/etudes_realisations/Nardi/Segway/general/index.html

[10] G. Arechavaleta, J.-P. Laumond, H. Hicheur, A. Berthoz. An Optimality Principle Governing Human Walking. [Arechavaleta et al. 2008] IEEE Transactions on Robotics 24(1): 5-14, 2008

[11] G. Arechavaleta, J-P. Laumond, H. Hicheur, and A. Berthoz. The nonholonomic nature of human locomotion: a modeling study

[12] Y.Zhang, J.Pettre, Q.Peng, and S.Donikian. Data based steering of virtual human using a velocity-space approach. In Motion In Games, Utrecht Pays-Bas, 11 2009.

[13] Olivier, A.-H.; Kulpa, R.; Pettré, J. & Crétual, A. Overmars, M. & Egges, A. (Eds.) A velocity-curvature space approach for walking motions analysis Proceedings of the 2nd International Workshop on Motion in Games, MIG '09: Proceedings of the 2nd International Workshop on Motion in Games, Springer-Verlag, 2009, 104-115

[14] http://sciences.blog.lemonde.fr/2010/02/26/la-physique-du-ski/

[15] http://hypertextbook.com/facts/2007/TabraizRasul.shtmlen

[16]http://unity3d.com/support/documentation/ScriptReference/Collider.html

[17]http://unity3d.com/support/documentation/ScriptReference/TerrainCollider

[18]http://unity3d.com/support/documentation/Components/class-BoxCollider.html

[19]http://answers.unity3d.com/questions/41710/best-physics-choice-for-a-ski-game-.html

[20]http://unity3d.com/support/documentation/Components/class-Rigidbody.html

[21] http://unity3d.com/support/documentation/Components/class-WheelCollider.html

[22]http://unity3d.com/support/documentation/ScriptReference/WheelCollider.html

[23] http://unity3d.com/support/documentation/ScriptReference/BoxCollider.html