POLYTECH MARSEILLE
Aix*Marseille Université

# RAPPORT DE STAGE :

# Développer Unity :
*Réalisation de jeux 3D sur Unity*

*Stage de 5e année*


**Nom** : Ghislain LEVREAU

**Formation** : 5e année Ingénieur Polytech Marseille

**Année** : 2023-2024

**Entreprise** : Jungle VR

**Adresse** : 29 rue Etienne Dolet 94140 ALFORTVILLE FRANCE


**Stage du** : 7 février au 28 juillet 2024

*Partie Ski Rush*

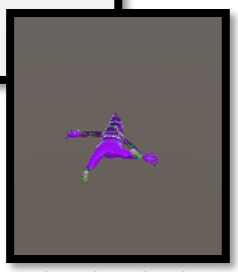
# Ski Rush

## I. Contexte et objectifs

### *a) Objectifs initiaux*

La deuxième mission de mon stage chez Jungle VR était de développer une série de jeux de type casual, voire hyper-casual, caractérisés par leur simplicité tant au niveau de la mécanique que de l'interaction utilisateur. Ces jeux, conçus pour être accessibles et engageants, visaient à toucher un large public grâce à une prise en main immédiate et une courbe d'apprentissage minimale. Ce choix de développement répond à une demande croissante sur le marché des jeux mobiles, où les utilisateurs recherchent des divertissements rapides, faciles à comprendre, et gratifiants dès les premières minutes de jeu.

*Maquette*



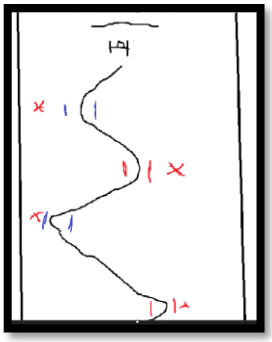
*Animation de chute*

Dans le cadre de ce projet, j'ai commencé par créer des maquettes pour explorer différentes idées de gameplay. Une des premières idées concrétisées impliquait un concept où le joueur pouvait se déplacer sur trois colonnes, avec des animations simples mais dynamiques de glissade, de course, de saut, et de chute. Ce concept servait de base pour tester et affiner les mécaniques de jeu, tout en respectant les contraintes de simplicité et de rapidité de développement qui définissent les jeux hyper-casuals. L'approche était de concevoir des jeux avec un gameplay immédiatement gratifiant et des graphismes minimalistes, adaptés à des sessions de jeu courtes mais stimulantes, alignant ainsi le projet avec les tendances actuelles du marché mobile.

Par la suite, l'idée de développer un jeu de ski a émergé, offrant une thématique particulièrement appropriée en période hivernale. Ce choix s'est révélé pertinent non seulement pour son attrait saisonnier, mais aussi pour les possibilités qu'il offrait en termes de gameplay dynamique et visuellement attrayant. Certaines des mécaniques initiales ont été conservées et adaptées, tandis que d'autres ont été modifiées pour mieux correspondre aux spécificités du ski alpin.

## b) Recherches initiales

Avant de nous lancer dans le développement, nous avons entrepris une phase de recherche pour explorer les concepts existants et sélectionner les éléments clés qui pourraient rendre notre jeu unique et attrayant. Notre première approche a été de concevoir un jeu en 2D avec des déplacements simples, où un tap permettait de changer de côté ou un slide modifiait la direction du drift.



*Croquis jeu 2D*


*Double Drift*

Nous avons été inspirés par un jeu appelé "Double Drift" qui possédait une mécanique intéressante et adaptée à ce que nous voulions réaliser. De plus, après avoir analysé divers jeux, nous avons décidé de basculer vers une vue en 3D, offrant une perspective arrière pour une expérience plus immersive.

Le concept initial de "Ski Rush" était un jeu de ski infini où le joueur devait passer entre des portes ou des drapeaux. Au fil du développement, le jeu a évolué pour inclure des éléments supplémentaires comme un système de skins/cosmétiques, une boucle de gameplay infinie avec une difficulté croissante, et divers obstacles augmentant le challenge et la durée de vie du jeu.

## c) Présentation du jeu

Ski Rush est un jeu de ski en 3D axé sur le slalom et la progression en distance. Le joueur incarne un skieur qui dévale des pistes enneigées, avec pour objectif principal de franchir des portes tout en évitant des obstacles variés. Chaque descente est une opportunité de battre son propre record de distance parcourue.

Le gameplay de Ski Rush est conçu pour être simple mais addictif. Le joueur utilise des gestes simples pour diriger le skieur : un glissement à gauche ou à droite permet de changer de direction. À mesure que le joueur progresse, la difficulté augmente avec l'apparition de nouveaux obstacles et des portes placées de manière plus complexe.

Pour maintenir l'intérêt des joueurs, Ski Rush propose également un système de personnalisation avec des skins et des cosmétiques déblocables avec des points du jeu, gagnable en passant les portes. Ces éléments esthétiques permettent aux joueurs de personnaliser leur expérience de jeu et d'afficher leur progression.


*Images de jeu*

En conclusion, Ski Rush vise à offrir une expérience de jeu rapide et satisfaisante, idéale pour des sessions de jeu courtes mais engageantes. Grâce à une recherche approfondie et une conception réfléchie, nous avons pu développer un jeu qui se distingue par sa simplicité, son esthétique attrayante et son gameplay captivant.

# II. Personnages

## a) Ressources graphiques

La création des modèles 3D pour les personnages de *Ski Rush* a été un élément clé du projet, nécessitant un équilibre entre qualité visuelle et optimisation des ressources. Étant donné la complexité et le temps nécessaire pour créer des personnages détaillés, j'ai opté pour l'utilisation de modèles téléchargés. Ces personnages ont été choisis en fonction de critères stricts, tels que la qualité graphique, la compatibilité avec Unity, et l'optimisation pour les performances mobiles. Cette approche m'a permis de gagner un temps précieux, tout en assurant que les personnages soient cohérents avec le style visuel du jeu et suffisamment low poly pour éviter tout lag sur les téléphones.


*Liste des sites web utilisés pour télécharger des modèles*

En parallèle, pour les autres modèles plus spécifiques au projet, comme les portes, les éléments de la piste, des textes 3D, j'ai utilisé Blender pour créer des assets personnalisés. Cette décision était motivée par le besoin de donner une identité unique au jeu, en concevant des éléments qui correspondent parfaitement à la vision artistique de *Ski Rush*.


*Modélisation du texte d'arrière-plan*

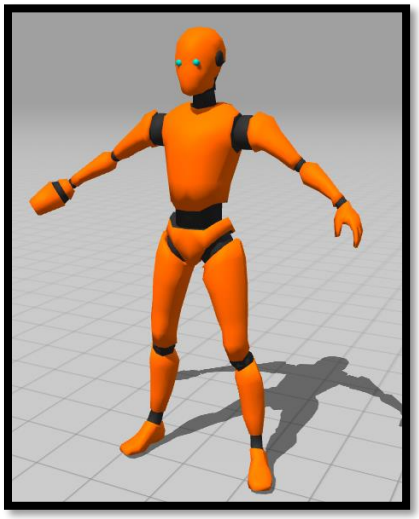


*Porte ski alpin*

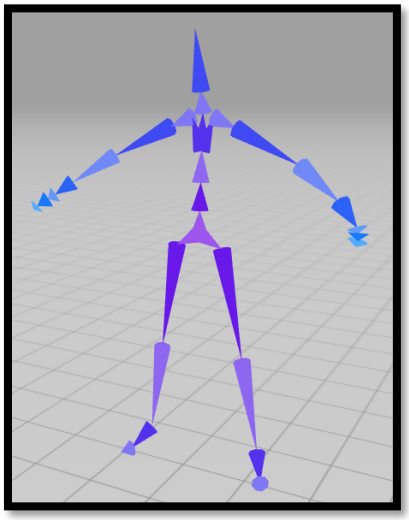
## b) Animation des personnages

Pour l'animation et la gestion des personnages dans *Ski Rush*, j'ai opté pour l'utilisation de Mixamo, une plateforme en ligne spécialisée dans la création et l'application d'animations 3D. *Ski Rush* se distingue par un style simple mais qui conserve des éléments de réalisme, rendant les animations de personnages cruciales pour une expérience de jeu immersive.

Avec le système de skins que nous verrons plus tard, il était essentiel d'avoir des animations génériques qui fonctionnent pour chaque skin. Mixamo s'est révélé particulièrement utile dans ce contexte, car il permet de créer un squelette générique pour chaque personnage à partir d'un modèle importé. Ce processus simplifie considérablement l'intégration des skins dans le jeu, accélérant ainsi la création et l'application des animations. En quelques étapes, Mixamo génère un squelette adapté au modèle, ce qui est très pratique et efficace.

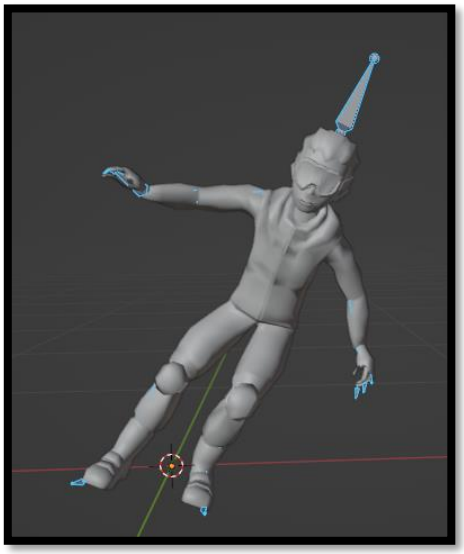


*Modèle importé sur Mixamo*

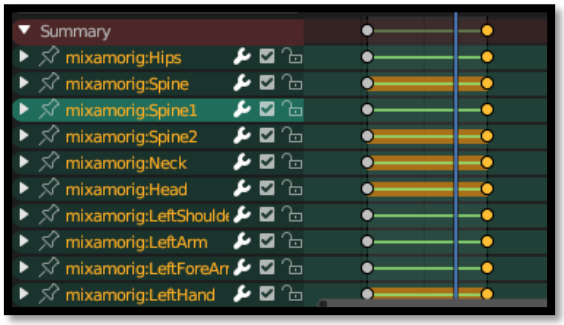


*Squelette généré par Mixamo*

Pour les animations de base, comme l'animation idle (position de repos) du personnage, j'ai utilisé une animation directement issue de la bibliothèque de Mixamo. Cependant, pour des animations plus spécifiques, comme le drift en ski, qui était trop particulier pour être trouvé sur Mixamo ou d'autres plateformes, j'ai choisi de créer l'animation moi-même à l'aide de Blender. Cette approche m'a permis de personnaliser le mouvement pour qu'il corresponde parfaitement aux exigences du gameplay de Ski Rush, tout en conservant la fluidité et la précision nécessaires pour une animation réaliste.



*Animation sur Blender*




*Animator Blender*

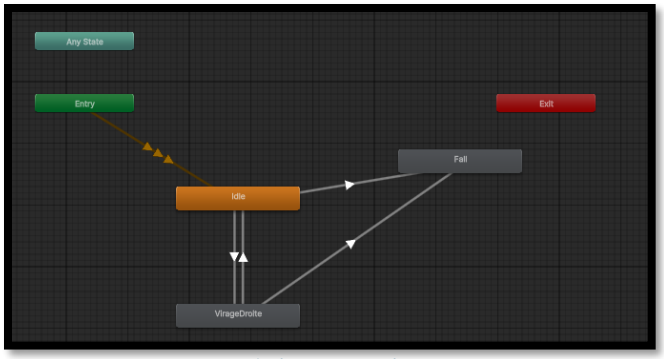
### c) *Gestion des entrées tactiles*

Le contrôle du personnage est géré dans le script *PlayerController* et principalement par la méthode HandleTouchInput(). Cette méthode capture les interactions tactiles sur l'écran, comme les mouvements de glissement, et les traduit en mouvements latéraux du personnage sur la piste de ski.

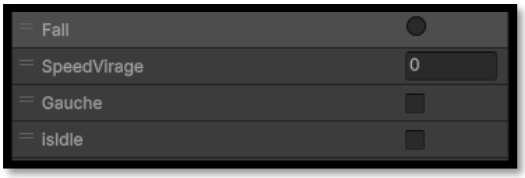
- **Détection et Gestion des Touches :** Lorsqu'une touche est détectée, le script capture la position initiale de la touche sur l'écran. Au fur et à mesure que le joueur déplace son doigt, le script calcule la différence entre la position actuelle et la position initiale, ce qui permet de déterminer la direction et l'intensité du mouvement souhaité par le joueur.
- **Application du Mouvement :** Ce mouvement est ensuite appliqué au personnage en ajustant sa position cible sur l'axe X. L'interpolation (Vector3.Lerp) est utilisée pour déplacer le personnage de manière fluide vers sa nouvelle position, simulant ainsi un glissement sur la neige.

Les animations du personnage sont gérées par un Animator Unity, qui réagit aux mouvements et aux actions du joueur.

- **Mise à Jour de l'Animation :** La méthode UpdateAnimation() détermine l'animation à jouer en fonction de l'état du personnage. Si le personnage est en mouvement (détecté par l'analyse des entrées tactiles), l'animation de virage est déclenchée. La direction du virage (gauche ou droite) est déterminée par la valeur de sliderAnimationValue, qui reflète l'intensité et la direction du mouvement latéral.
- **Gestion des États d'Animation :** Lorsque le personnage est immobile, l'animation "idle" est activée pour simuler une position de repos. Dans le cas où le personnage se déplace, le Animator adapte l'animation de virage en fonction de la direction et de la vitesse du mouvement. Cela permet une transition fluide entre les différentes animations, ce qui rend les mouvements du personnage plus naturels et réalistes.
- **Animations Spécifiques aux Événements :** Des animations spécifiques, telles que l'animation de chute, sont déclenchées lors de certains événements comme la collision avec un obstacle. Ces animations sont activées via des triggers dans l'Animator, en réponse aux interactions du joueur avec l'environnement du jeu.




*Animator Unity*




*Paramètres d'animation*

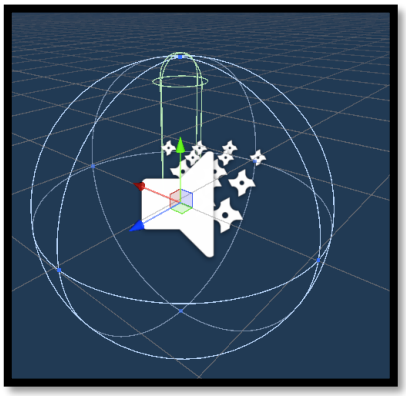
En combinant la gestion des entrées tactiles et le déclenchement des animations, le code de *Ski Rush* permet de créer une expérience de jeu où les actions du joueur sont directement et instantanément reflétées dans les mouvements du personnage, offrant ainsi une sensation de contrôle précis et une immersion accrue.

### d) *Structure*

Pour accélérer et simplifier la création de différents personnages, un prefab vide a été conçu avec tous les composants nécessaires. Cette approche permet de créer rapidement des variantes de personnages en dupliquant ce prefab de base et en y glissant simplement le modèle 3D ainsi que l'avatar correspondant. Voici les composants inclus dans ce prefab :

- **Rigidbody :** Composant essentiel pour gérer la physique du personnage, notamment ses interactions avec le terrain.
- **Capsule Collider (en trigger) :** Utilisé pour détecter les collisions avec les éléments interactifs du jeu tels que les collectibles, les obstacles, et les portes. Ce collider est configuré en mode "trigger" pour enregistrer les collisions sans affecter la physique du personnage.
- **Animator :** Responsable de la gestion des animations du personnage. En changeant simplement l'avatar dans l'Animator, les mêmes animations peuvent être appliquées à différents modèles.
- **Script de contrôle du personnage :** Le script principal qui gère les mouvements, les entrées tactiles, et les interactions du personnage avec l'environnement du jeu.
- **Audio Source :** Composants pour la gestion des effets sonores associés aux actions du personnage, comme le bruit des skis sur la neige ou les sons déclenchés lors des collisions.
- **Particle Systems :** Deux systèmes de particules sont inclus pour simuler l'effet de projection de neige par les skis, ajoutant une touche visuelle réaliste à chaque déplacement du personnage.


*Prefab parent* *Variantes*



*Vue du Prefab parent*

# III. Cosmétiques et personnalisation

## *a) Système de monnaie*

Le système de monnaie dans *Ski Rush* permet aux joueurs de gagner et de dépenser des points pour débloquer des personnalisations de skis, de bâtons et de personnages. Le script *CurrencyManager* gère la monnaie du joueur, permettant d'ajouter des points lorsqu'ils sont gagnés et de les déduire lors d'achats. Le solde de la monnaie est affiché à l'écran, et le joueur peut voir en temps réel les points disponibles.

Pour assurer que la progression du joueur est sauvegardée entre les sessions, la monnaie est stockée à l'aide de *PlayerPrefs*. Chaque fois que la monnaie change, elle est automatiquement sauvegardée, et lors du redémarrage du jeu, la monnaie est rechargée pour que le joueur retrouve son solde précédent. Cela garantit une continuité dans l'expérience de jeu, permettant aux joueurs de conserver leur progression.

## *b) Gains et feedback de récompense*

Le joueur peut gagner la monnaie virtuelle, représentée par des flocons de neige (snowflakes), en traversant des portes sur la piste. Chaque porte franchie rapporte un flocon, qui s'ajoute au compteur de monnaie du joueur.

Le feedback visuel et sonore lié au gain de monnaie a été soigneusement travaillé pour offrir une sensation de satisfaction au joueur. Lorsqu'une porte est franchie, un cercle bleu apparaît et grandit autour de la porte, signalant la réussite de l'action. Simultanément, un flocon de neige apparaît au-dessus du joueur et monte dans les airs, créant une animation qui souligne le gain de monnaie. Le compteur de monnaie est mis à jour en douceur à l'aide de la bibliothèque *Dotween*. Une légère vibration est également déclenchée pour ajouter une dimension tactile à l'expérience, tandis qu'un son spécifique est joué pour compléter le retour d'information.

Ce feedback multi-sensoriel contribue à rendre le passage dans les portes et l'accumulation de flocons particulièrement gratifiants pour le joueur.

***Passage d'une porte***

### c) Système de cosmétiques

Le système de cosmétiques dans *Ski Rush* permet aux joueurs de personnaliser leur expérience en choisissant différents personnages, skis, et bâtons. Cette fonctionnalité ajoute une dimension de personnalisation qui enrichit le gameplay et offre une motivation supplémentaire pour accumuler des snowflakes, la monnaie du jeu.

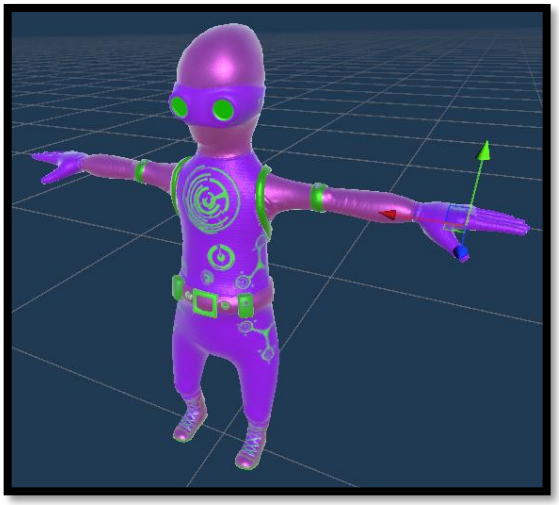
Les joueurs peuvent choisir leur personnage, leurs skis, et leurs bâtons directement dans le menu de personnalisation. Lorsqu'un joueur change de personnage, le *MenuManager* supprime le prefab existant du personnage actuel et réinstancie le nouveau personnage sélectionné. De plus, un autre script, *EquipmentManager*, gère l'attachement des équipements comme les skis et les bâtons au personnage.

Les noms des os dans les squelettes des personnages sont cohérents et uniformisés grâce à Mixamo, ce qui facilite la recherche des points d'attache corrects pour chaque élément, comme les pieds pour les skis et les mains pour les bâtons. Pour localiser ces points d'attache dans la hiérarchie du squelette, un petit script utilitaire a été créé. Ce script, *FindDeepChild*, parcourt récursivement tous les enfants du squelette pour trouver l'os correspondant au nom spécifié. Cette méthode garantit que chaque élément est attaché au bon endroit, peu importe la complexité de la hiérarchie du squelette.

```
public void EquipCharacter(GameObject character, GameObject skiPrefab, GameObject polePrefab)
{
    skiLeftAttachPoint = character.transform.FindDeepChild("mixamorig:LeftToeBase");
    skiRightAttachPoint = character.transform.FindDeepChild("mixamorig:RightToeBase");
    poleLeftAttachPoint = character.transform.FindDeepChild("LeftPolePosition");
    poleRightAttachPoint = character.transform.FindDeepChild("RightPolePosition");
    lightAttachPoint = character.transform.FindDeepChild("mixamorig:Head");

    // Appliquer l'équipement
    ChangeSkis(skiPrefab);
    ChangePoles(polePrefab); ;
}
```

*Code d'attachement des équipements*

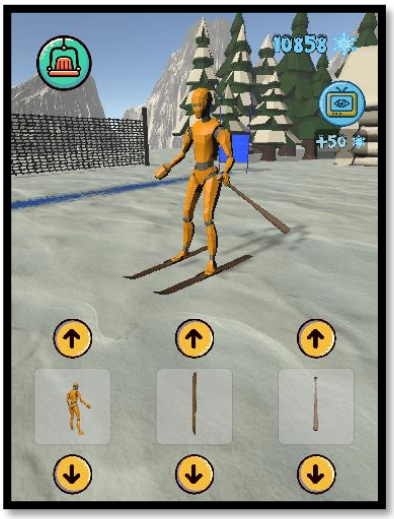


*Point de fixation du bâton gauche*

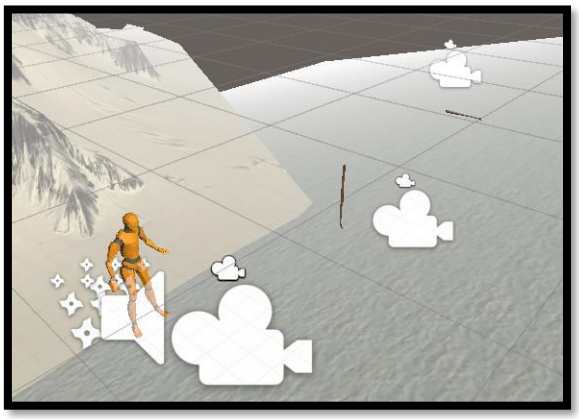

*Hiérarchie du squelette des personnages*

Les joueurs peuvent dépenser leurs snowflakes pour débloquer de nouveaux cosmétiques. Le système vérifie d'abord si le joueur dispose de suffisamment de monnaie pour effectuer l'achat. Si l'achat est validé, l'élément est débloqué, équipé, et réinstancié dans le jeu. Comme pour la gestion de la monnaie, les possessions et sélections de cosmétiques du joueur sont sauvegardées à l'aide de *PlayerPrefs*, garantissant que les choix du joueur sont conservés lors de ses futures sessions de jeu.


1. ***Achat d'un cosmétique de personnage ainsi que de bâtons.***
2. ***Dépense de snowflakes (monnaie)***
3. ***Changement des éléments en conséquent.***
4. ***Sauvegarde des nouveaux éléments.***




Pour créer l'illusion des cases avec les objets 3D à débloquer dans le menu, chaque élément est instancié dans un coin de la scène avec une rotation pour un effet visuel attrayant. Des caméras dédiées observent chaque élément, et leurs rendus sont affichés en temps réel dans le menu via des *RenderTextures*. Ce système permet aux joueurs de visualiser les cosmétiques avant de les acheter, ajoutant une dimension supplémentaire à l'expérience d'achat et de personnalisation.



***3 instances pour générer les Render Textures***

### *d) Publicités*

Dans *Ski Rush*, plusieurs types de publicités ont été intégrés pour monétiser le jeu tout en offrant des récompenses aux joueurs. Le système de publicités est géré par le script AdsInitializer, qui relie les identifiants iOS et Android du jeu à la plateforme Unity Ads, permettant ainsi de diffuser les publicités adaptées à chaque appareil.

Dans le menu des skins, un bouton "+50 ❄" permet aux joueurs de gagner 50 snowflakes en regardant une publicité. Cette publicité, appelée *Rewarded Ad*, dure 30 secondes et ne peut pas

être passée. Une fois la publicité terminée, les 50 snowflakes sont ajoutés à la monnaie du joueur, lui permettant ainsi de débloquer plus rapidement des cosmétiques. Ce type de publicité est volontairement choisi par le joueur, ce qui le rend moins intrusif et plus acceptable tout en offrant une récompense tangible.

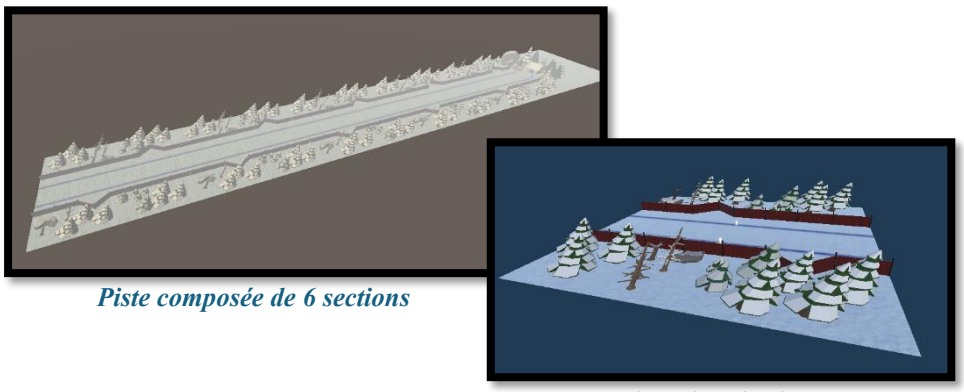
Un autre type de publicité utilisé dans *Ski Rush* est l'*Interstitial Ad*. Cette publicité recouvre l'écran entier et est généralement utilisée comme transition entre deux phases du jeu, par exemple à la fin de chaque partie. Ces publicités sont plus intrusives que les *Rewarded Ads*, mais elles sont placées à des moments stratégiques pour minimiser l'impact sur l'expérience de jeu tout en générant des revenus.

Il existe aussi les *Banner Ads*, qui sont des bannières publicitaires affichées en permanence sur l'écran. Cependant, ces publicités ne sont pas utilisées dans *Ski Rush*, car elles sont souvent considérées comme gênantes et mal perçues par les joueurs, impactant négativement l'expérience utilisateur.

# IV. Mécaniques du jeu

## *a) Structure de la piste*

Dans *Ski Rush*, la piste de ski est structurée comme une série de tuiles ou sections, chacune représentée par un prefab préconstruit. Ces sections contiennent les éléments de base tels que la texture de neige, les barrières, et l'environnement autour de la piste.



***Piste composée de 6 sections***

***1 section de piste***

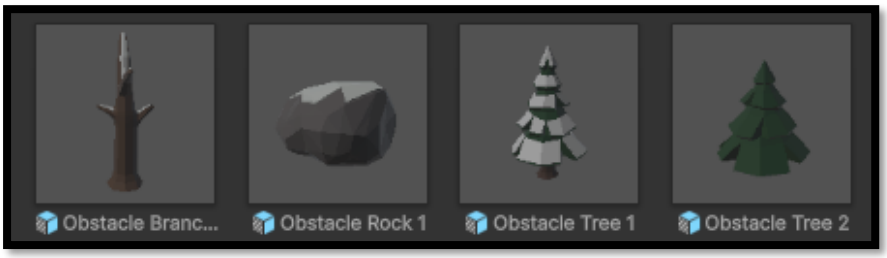
Le joueur, bien qu'apparemment en mouvement sur la piste, est en réalité fixe dans la scène. Ce sont les sections de la piste qui se déplacent vers le joueur, simulant ainsi le mouvement. Cette technique permet de maintenir la position du joueur constante tout en créant l'illusion d'une descente rapide et continue.

Pour optimiser les performances et éviter les coûts élevés associés à la suppression et à la réinstanciation des sections, un système de repositionnement est utilisé. Lorsqu'une section de la piste passe derrière la caméra du joueur, elle n'est pas détruite, mais simplement téléportée à

l'avant, devant le joueur. Cette technique permet de réutiliser les sections de manière continue, créant l'illusion d'une piste infinie sans nécessiter de nouvelles ressources. En déplaçant les sections plutôt que de les recréer, le jeu maintient une performance fluide, tout en assurant une transition transparente dans l'expérience de jeu.

### b) *Obstacles aléatoires*

Chaque section de piste est divisée en une grille de plusieurs colonnes et rangées (3x6). Les obstacles, tels que les rochers ou les arbres, sont placés dans les cellules de cette grille. Les positions des obstacles sont générées de manière aléatoire pour chaque section grâce au script *SpawnerManager*, ce qui garantit que chaque descente est unique. Les obstacles sont instanciés en tant qu'enfants des sections de piste, ce qui leur permet de se déplacer avec les sections au fur et à mesure que celles-ci défilent vers le joueur.




*3 instances pour générer les Render Textures*

Pour assurer que les niveaux restent jouables et équilibrés, plusieurs règles d'apparition ont été implémentées :

- **Évitement des Murs d'Obstacles :** Le système s'assure qu'aucune rangée entière n'est bloquée par des obstacles, ce qui laisserait le joueur sans passage. Cela est géré en limitant le nombre d'obstacles par rangée et en imposant des cellules vides obligatoires.
- **Espacement entre les Obstacles :** Les obstacles ne peuvent pas être placés trop près les uns des autres. Cette règle est mise en œuvre en vérifiant les cellules adjacentes avant de placer un nouvel obstacle, garantissant ainsi un espace suffisant pour que le joueur puisse manœuvrer.
- **Placement Alterné des Portes :** Les portes, qui sont essentielles pour gagner des snowflakes, ne peuvent apparaître que sur les rangées où l'indice est pair (x % 2 == 0). Par exemple, dans une grille de 3 rangées par 6 colonnes, les portes ne peuvent se placer que sur les deux rangées avec un indice pair. Cela permet d'alterner les rangées contenant des portes et celles contenant des obstacles, assurant ainsi un rythme de jeu équilibré et prévisible.

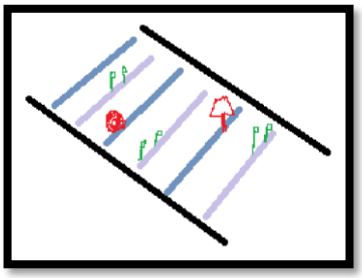


*Schéma d'apparition*

- **Intégration des Portes et Obstacles** : Les portes ne peuvent pas être bloquées par des obstacles, et elles sont placées de manière à offrir des trajectoires claires au joueur en alternant gauche et droite. De plus, cette alternance de placement entre les rangées de

portes et les rangées d'obstacles assure que le joueur doit constamment ajuster sa trajectoire pour progresser.

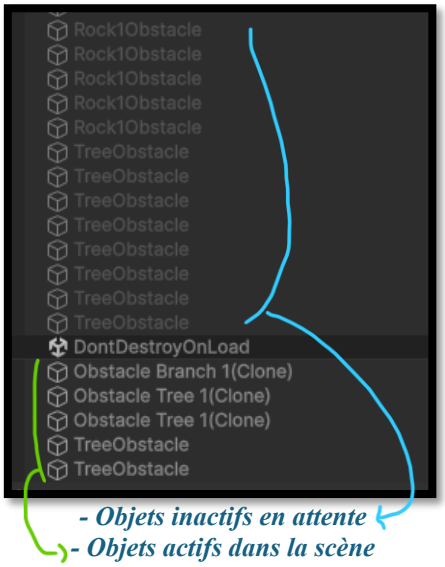
Lorsque la section de piste dépasse la position du joueur et doit être téléportée à l'avant, elle est d'abord nettoyée de tous ses obstacles existants. Ce processus consiste à renvoyer les obstacles au pool d'objets pour qu'ils puissent être réutilisés ultérieurement. Une fois nettoyée, la section est réinitialisée avec une nouvelle disposition d'obstacles, générée aléatoirement selon les règles décrites ci-dessus. De plus, au fil de la partie, la difficulté augmente progressivement, ce qui se traduit par une augmentation du nombre d'obstacles présents sur la piste et une accélération de la vitesse des sections. Cela garantit que chaque passage de la piste offre un défi nouveau et imprévisible tout en maintenant des performances optimales, tout en adaptant la difficulté à la progression du joueur.

### c) *Optimisation*

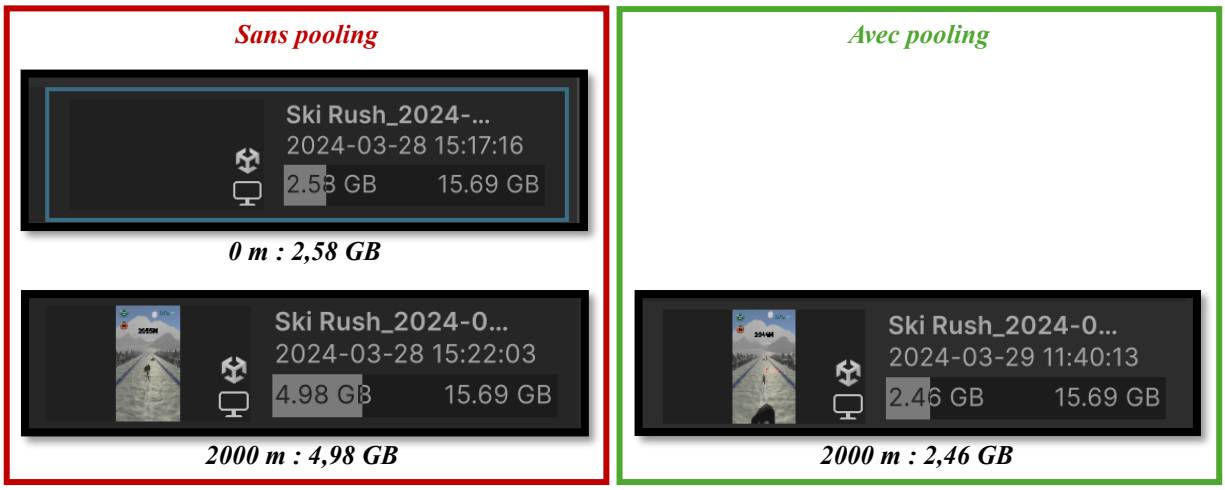
L'optimisation est un aspect crucial dans le développement de jeux mobiles pour assurer une expérience fluide et réactive. Dans *Ski Rush*, une des principales stratégies d'optimisation utilisées est le **pooling d'objets**. Cette technique améliore les performances en réduisant la charge sur le processeur et la mémoire, en particulier lors de la gestion des obstacles et des éléments récurrents du jeu.

Le pooling d'objets consiste à réutiliser des instances préfabriquées d'objets plutôt que d'en créer de nouvelles à chaque fois qu'un élément doit apparaître. Cela évite les coûts élevés associés à l'instanciation et à la destruction fréquentes d'objets, ce qui peut provoquer des baisses de performance, notamment des saccades (stuttering) dues à la gestion de la mémoire.

1. **Création du Pool** : Au début du jeu, le *ObjectPoolManager* crée un pool d'objets pour chaque type de prefab, comme les obstacles et les portes. Chaque pool contient un certain nombre d'instances préfabriquées, prêtes à être utilisées.
2. **Réutilisation des Objets** : Lorsqu'un obstacle ou une porte doit être placé sur la piste, le système ne crée pas un nouvel objet. Au lieu de cela, il récupère une instance inactive du pool, la positionne, la réactive, et l'attache à la section de piste correspondante. Cela permet de réduire significativement le nombre de nouvelles allocations de mémoire.
3. **Retour au Pool** : Une fois qu'une section de piste est déplacée derrière le joueur et téléportée à l'avant, tous les obstacles et portes qu'elle contient sont désactivés et renvoyés au pool. Cela les rend disponibles pour être réutilisés dans une autre section sans avoir besoin de recréer les objets.



Pour garantir que le jeu fonctionne de manière fluide sur une variété de téléphones, des tests de performance ont été réalisés en utilisant les outils intégrés d'Unity, tels que le Memory Profiler. Cela a été particulièrement utile pour surveiller l'utilisation de la mémoire, en identifiant les fuites de mémoire et les allocations excessives qui pourraient affecter les performances. En combinant ces analyses avec le système de pooling d'objets, nous avons pu réduire la charge mémoire et maintenir un taux de rafraîchissement élevé des images par seconde (FPS). Cela a permis de garantir une expérience de jeu fluide, même sur des appareils disposant de ressources matérielles limitées.



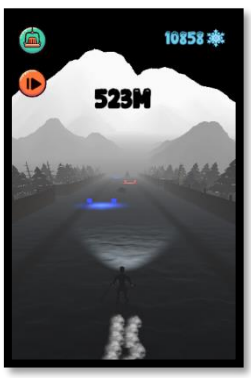

*Mémoire utilisée*

Comme on peut le constater sur les captures d'écran ci-dessus, sans l'utilisation du pooling, après avoir parcouru 2000 mètres dans le jeu, la mémoire utilisée double en raison des résidus de suppressions accumulés. Cela provoque une augmentation progressive de l'utilisation de la mémoire, ce qui peut entraîner des ralentissements et une baisse de performance. En revanche, avec le système de pooling, la mémoire utilisée reste stable, semblable à son niveau initial. Cette stabilité est due à la réutilisation des objets existants plutôt qu'à leur suppression et recréation, ce qui réduit significativement la charge mémoire et améliore la performance globale du jeu.

### d) Effets météorologiques

Dans *Ski Rush*, les effets météorologiques jouent un rôle crucial pour ajouter de la variété et du réalisme à l'expérience de jeu.

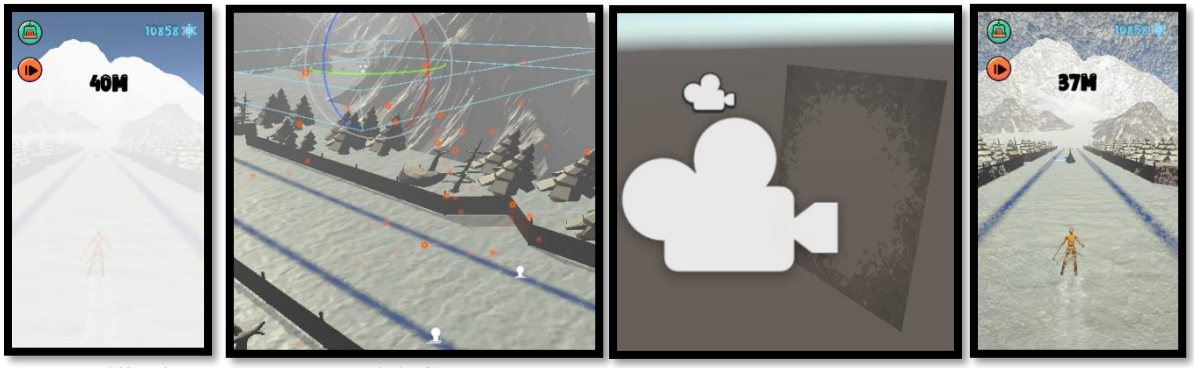
Un cycle jour/nuit est simulé en utilisant une *Directional Light* qui représente le soleil. Cette lumière tourne autour de la scène, créant ainsi des transitions naturelles entre le jour et la nuit. Le script qui gère ce cycle ajuste la rotation du soleil en fonction du temps écoulé, et lorsqu'une certaine rotation est atteinte, la lumière du personnage (une lampe frontale) s'allume automatiquement pour permettre au joueur de continuer à skier dans des conditions de faible luminosité.




*Nuit*

Pour ajouter de la diversité au gameplay, plusieurs effets météorologiques ont été implémentés, chacun apportant son propre défi visuel et influençant la jouabilité :

1. **Brouillard** : Le brouillard est créé en utilisant le système de *Fog* d'Unity. Lorsqu'il est activé, le *Fog* augmente progressivement, réduisant la visibilité sur la piste. Une fois l'effet terminé, le *Fog* est progressivement dissipé pour revenir à des conditions normales de visibilité.
2. **Neige** : L'effet de neige est réalisé avec un *Particle System* qui émet des particules blanches simulant la chute de neige. Cet effet peut être activé pour une durée limitée, plongeant la piste dans une ambiance hivernale immersive.
3. **Blizzard** : Plus intense que l'effet de neige, le blizzard est également créé à l'aide d'un *Particle System*, mais avec des particules plus denses et une vitesse accrue, simulant une tempête de neige. Cela réduit encore plus la visibilité et augmente la difficulté de la descente.
4. **Givre** : L'effet de givre est obtenu en plaçant un rectangle avec une texture de glace devant la caméra. Lorsqu'il est activé, la transparence de cette texture est progressivement réduite, simulant un givre qui se forme sur l'écran, ce qui obscurcit partiellement la vue du joueur.




*Brouillard* *Particle System* *Rectangle de glace* *Frost*

Les effets météorologiques sont déclenchés de manière aléatoire pendant le jeu, ajoutant un élément d'imprévisibilité. Le script *WeatherManager* contrôle l'apparition de ces effets, en les activant et en les désactivant à des intervalles aléatoires, en fonction du niveau de difficulté du jeu (distance parcourue). Cela permet de maintenir l'intérêt du joueur et de créer des défis visuels supplémentaires au fur et à mesure qu'il progresse dans le jeu.

# V. Conclusion de Ski Rush

Le développement de *Ski Rush* a été une expérience enrichissante qui m'a permis d'explorer divers aspects du développement de jeux mobiles, en particulier dans le genre casual. Ce projet m'a offert l'opportunité d'appliquer des concepts fondamentaux comme l'optimisation via le pooling d'objets et la gestion des performances, tout en concevant des mécaniques de jeu dynamiques, telles que la génération procédurale des obstacles et l'intégration d'effets météorologiques immersifs.

Un aspect particulièrement important et motivant de ce projet a été la mise en place d'un système de cosmétiques. Ce système permet aux joueurs de personnaliser leur expérience en sélectionnant différents personnages, skis, et bâtons, ce qui ajoute une couche supplémentaire d'engagement. La gestion des cosmétiques a nécessité une approche réfléchie pour garantir que chaque élément puisse être facilement intégré et changé en cours de partie, tout en assurant une performance optimale. L'utilisation de *PlayerPrefs* pour sauvegarder les sélections de cosmétiques et de *RenderTextures* pour afficher les objets en 3D dans le menu a permis de créer une interface utilisateur intuitive et visuellement attrayante.

Ce système de cosmétiques a non seulement enrichi l'expérience de jeu en offrant des options de personnalisation, mais il a également introduit une dynamique économique avec l'acquisition de "snowflakes" pour débloquer de nouveaux éléments. Cette fonctionnalité a ajouté une dimension de progression qui encourage les joueurs à continuer à jouer et à améliorer leurs compétences pour obtenir de nouvelles récompenses.

Tout au long du développement, j'ai cherché à équilibrer simplicité et engagement, créant un jeu accessible pour un large public tout en offrant des défis croissants pour maintenir l'intérêt des joueurs. L'intégration de cycles jour/nuit et d'événements météorologiques aléatoires, combinée au système de cosmétiques, a permis de rendre chaque session de jeu unique et personnalisée.

En conclusion, *Ski Rush* représente une étape cruciale dans mon parcours de développeur. Ce projet m'a permis d'acquérir une expérience précieuse en créant un jeu de A à Z, couvrant tous les aspects du développement. J'ai appris à gérer un projet dans sa globalité, de la conception des assets 3D à la mise en place du gameplay, en passant par l'optimisation des performances, l'intégration des musiques et des effets sonores, la création des animations, la gestion des interfaces utilisateur, et la mise en œuvre d'un système économique avec des cosmétiques.

# Annexes

## I. Liens de téléchargement

Télecharger Ski Rush