Hugo PROULX

Projet d’approfondissement 1

420-6X3-VI

JOURNAL D’EXPÉRIMENTATION

Travail présenté à

M. Mathieu Fréchette

Département d’informatique

Cégep de Victoriaville

12 février 2021

Table des matières

[But de l’expérimentation 3](#_Toc63065108)

[Comparaison de la technologie 3](#_Toc63065109)

[Tutoriel 4](#_Toc63065110)

[Problèmes rencontrés 9](#_Toc63065111)

[Conclusion 10](#_Toc63065112)

# ****But de l’expérimentation****

L’expérimentation avait pour but d’apprendre comment Unity fonctionne et créer la base d’un jeu vidéo trois dimensionnelles. Donc, si nous devions séparer les aspects de base d’un jeu vidéo, les points suivants seraient retenus; les mouvements du joueur, la collision avec l’environnement qui l’entourent, ainsi que la présence des « NPC », c’est-à-dire d’un personnage non joueur, ainsi que les mouvements de ceux-ci. En bref, le but de mon expérimentation n’est pas de créer un monde, mais plutôt programmer tous ce qu’il y est contenu, plus précisément le joueur, les NPC, le menu principal et la collision avec l’environnement. De plus, malgré la courte durée du cours, j’ai tenté d’expérimenter le plus possible avec les personnages non joueurs pour rendre leurs mouvements le plus fluide possible.

# Comparaison de la technologie

Plusieurs technologies étaient offertes à ma disposition pour la réalisation de ce projet, tel que Unreal Engine, Godot et Unity. Plusieurs différences sont présentes entre chacun de ces engines de développement. Par exemple, le langage de programmation entre chacun varie, Unreal Engine utilise le C++, Unity utilise le C#, alors que Godot utilise Python. Autres que simplement le langage de programmation, il y avait également plusieurs différences au niveau de la programmation et des composantes déjà présentes dans chacune des engines, si nous prenons Godot, nous devons programmer chaque touche, alors que Unity a déjà des touches pré programmées, telles que Unreal Engine.

Par conséquent, les raisons pour lesquelles quelle j’ai choisi Unity à la place des deux autres engines, sont les suivantes. Premièrement, la charge qui est appliqué sur mon ordinateur, à la suite de quelques testes avec Unreal Engine celui-ci à placer une charge trop grande sur mon ordinateur qui l’empêchait de fonctionner adéquatement, allant même jusqu’à un écran noir. Deuxièmement, le désire d’approfondir une technologie jamais vue, tel que le cours le demande, il n’est pas concevable d’approfondir une technologie déjà dans un cours précédent. Par conséquent, Godot devait une option non viable pour moi, puisque nous l’avons déjà utilisé dans le cours de simulation logicielle. En conclusion, la raison pour laquelle, j’ai retenu Unity pour la technologie de mon projet sont à cause de la demande plus légère sur mon ordinateur, ainsi que cette technologie m’étais totalement inconnu.

# ****Tutoriel (Comment refaire ce que j’ai fait.)****

Premièrement, si vous désiriez reproduire l’expérimentation, vous allez tous d’abord vous procurez tous les items suivants.

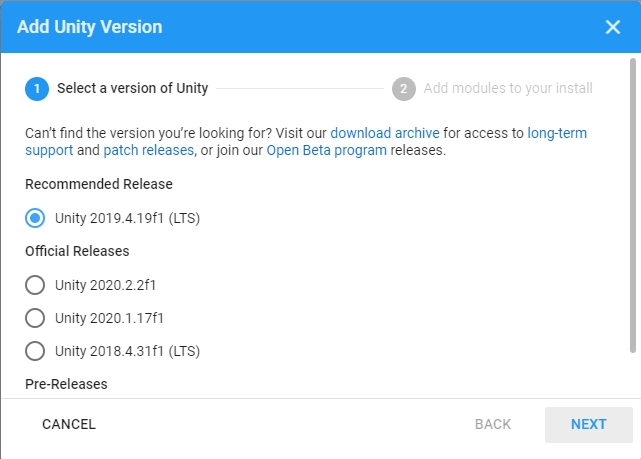
1. [Unity](https://unity.com/)
   1. Unity étant la platform de développement pour jeu vidéo.
2. [Visual Studio](https://visualstudio.microsoft.com/)
   1. Visual studio étant une platform qui permet de programmer avec le langage C#

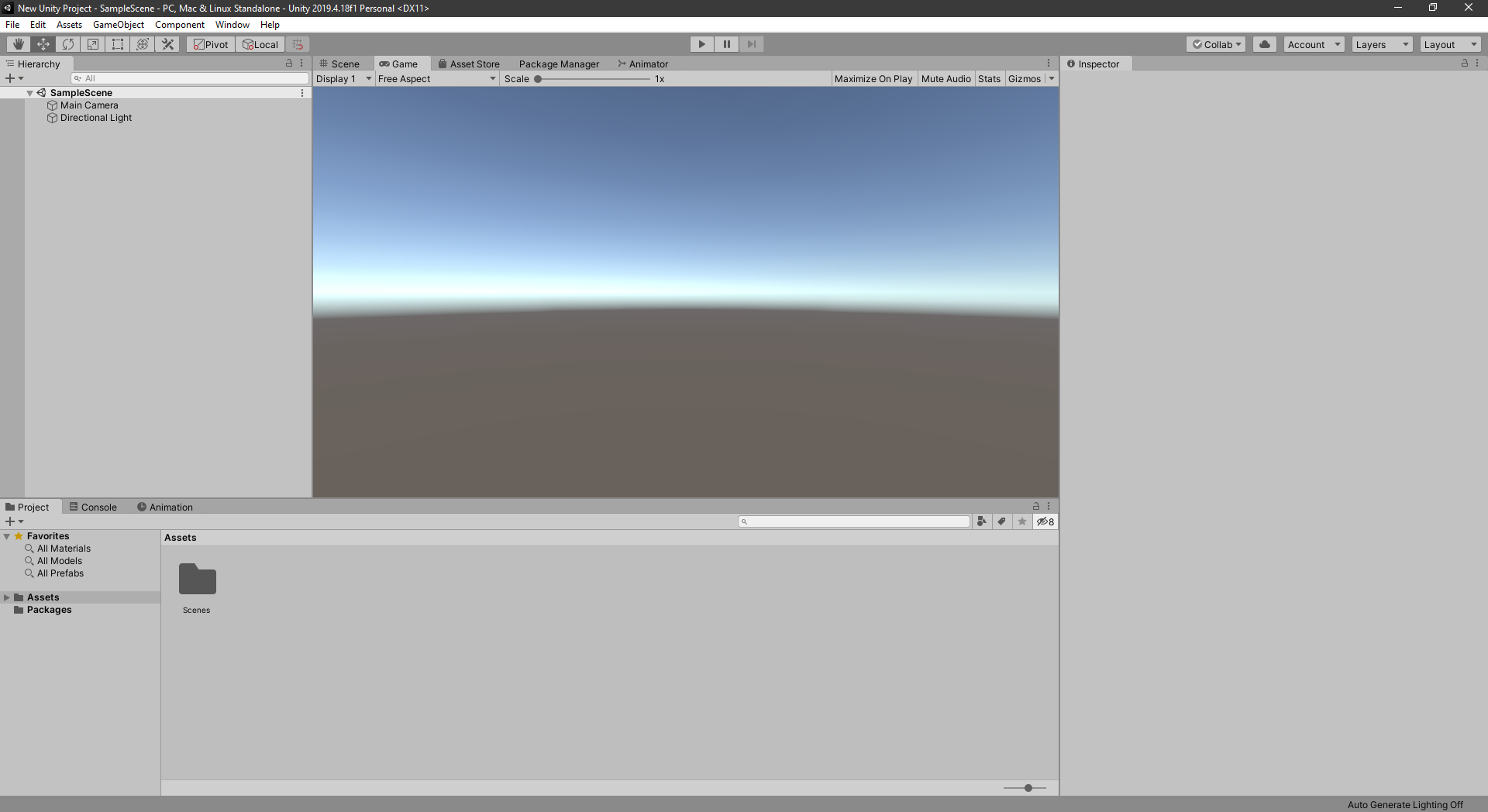
Par la suite, plusieurs options vous serons présentez tout d’abord, il va être très important de vous connectez avec votre compte. Par la suite, vous allez devoir vous rendre sur l’onglet suivant :



Lorsque vous allez être rendu, vous allez devoir appuyer sur le bouton : 

Vous allez devoir par la suite appuyer sur le bouton : « Next », c’est-à-dire « suivant »



Ensuite, vous allez devoir créer votre projet. Lors que cela est fait, vous allez faire face à une fenêtre tel que la suivante : 

Par la suite, si vous désirez créer votre propre monde ou niveau vous pouvez le créer de différente façon, la première étant à l’aide de forme et des matériaux et la deuxième est d’aller chercher un monde ou niveau déjà créer sur le « asset store » de Unity. Dans mon cas, j’ai utilisé l’option 2, donc si vous désirez utiliser le même monde que moi, voici le lien de celui-ci : [Monde](https://assetstore.unity.com/packages/3d/environments/fantasy/medieval-town-exteriors-27026). Par la suite, je vous recommande de créer votre personnage que vous allez utiliser dans votre jeu. Personnellement, j’ai décider de prendre un modèle de personnage déjà créer et gratuit. Donc, si vous désirez faire de même, vous pouvez trouver une multitude de ces modèles sur la page web suivante : [Mixamo](https://www.mixamo.com/). Lorsque vous allez avoir ajouter votre personnage dans la scène, vous allez devoir également ajouter une caméra sur votre personnage. Par la suite, vous allez devoir commencer à le programmer, mais pour faire ainsi, vous allez devoir au préalable avoir ajouter quelques composantes sur votre personnage. Pour faire ainsi, vous allez devoir appuyer sur le bouton suivant : 

Les composants que vous allez devoir ajouter sont les suivantes :

1. Le contrôleur de personnage (« Character Controller »)
   1. Cette composante permet d’avoir accès aux entrés lorsque le jeu est en marche.
2. Une capsule collider
   1. La capsule va devoir être ajuster à la grandeur du personnage
   2. Cette composante permet de faire des collisions avec l’environnement qui l’entoure.
3. Un animator
   1. Cette composante permet de gérer les animations du personnage
4. Ainsi qu’un script avec un nom quelconque, pour le bien du tutoriel, vous pouvez le nom comme suit : « mouvementJoueur »
   1. Ce script va permet de programmer le mouvement du personnage.

Après la création de toute ces composantes, vous allez vous retrouvez avec un objet qui comporte toutes les composantes nécessaires. Par la suite, vous allez devoir ouvrir le script : « mouvementJoueur ». Lorsque vous allez avoir celui-ci avec VisualStudio, vous allez devoir supprimer toute les codes déjà présent et passé le suivant :

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class joueurMouvement : MonoBehaviour

{

//Déclaration des variables pour tout ce qui influence le mouvement du personnage.

public float vitesse = 12f; //La vitesse à laquelle le personnage va pourvoir ce déplacé.

public float graviter = -20f; //La graviter est la force qui est appliquée au joueur lorsque celui-ci ne touche pas le sol.

public float sautPuissance = 3f; //La puissance de saut est la force avec laquelle le joueur peut sauter.

Vector3 velocity;

//Ici, nous vérifions la distance entre le sol et le joueur, pour ainsi nous assurer que la force qui est appliquée au joueur, lorsque celui-ci touche le sol, soit réinitialisée.

public Transform verificateurDeSol;

public float verificateurDistance = 0.4f;

public LayerMask solMask; //Le layer mask est ce qui permet de différencier les différentes partie du jeu.

bool touchSol;

//Cette variable est la variable qui exécute les mouvements pour faire bouger le joueur.

public CharacterController controller;

// Update est la méthode qui est appelée à chaque itération du jeu, plus communément appeler "frame".

void Update()

{

//Vérificateur de toucher au sol

touchSol = Physics.CheckSphere(verificateurDeSol.position, verificateurDistance, solMask);

//Réinisialisateur de vélocité en y, lorsque le joueur touche le sol.

if (touchSol && velocity.y < 0)

{

velocity.y = -1f;

}

//Gestion des entrées ('W', 'A', 'S' et 'D')

//Ces entrées sont des touches pré déterminées dans les paramètres de Unity.

float h = Input.GetAxis("Horizontal");

float v = Input.GetAxis("Vertical");

//Cette variable permet d'exécuter le mouvement pour sur les personnages.

Vector3 mouvement = transform.right \* h + transform.forward \* v;

//Ici, à l'aide du "CharacterController", nous faisons bouger le joueur. (Selon les entrées précédentes.)

controller.Move(mouvement \* vitesse \* Time.deltaTime);

//L'application de ce mouvement sur le joueur.

controller.Move(velocity \* Time.deltaTime);

}

}

//Vérificateur d'entrée permettant au joueur de sauter dans les aires, si les conditions sont remplies.

if (Input.GetButtonDown("Jump") && touchSol)

{

velocity.y = Mathf.Sqrt(sautPuissance \* -2f \* graviter);

}

//Cette méthode permet d'appliquer une force au joueur de manière plus réaliste.

velocity.y += graviter \* Time.deltaTime;

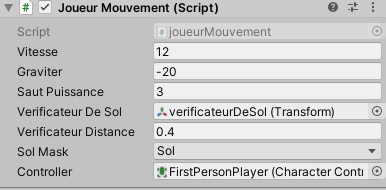
//L'application de ce mouvement sur le joueur.

controller.Move(velocity \* Time.deltaTime);

}

}

Lorsque vous allez avoir copier cette section du programme, il ne vous reste qu’a remplir les champs suivants avec vos items, pour faire ainsi, il ne vous suffit que de les « dragues » jusqu’à leur emplacement respectif, dans mon cas, voici le résultat :



Par la suite, vous serez en mesure de faire jouer et de voir le résultat, c’est-à-dire votre personnage qui est en mesure de se déplacer et également de sauter dans les aires.

En conclusion, pour pouvoir reproduire, plusieurs autres étapes sont nécessaires, et je vous recommande d’accéder au journal de veille, ainsi vous allez être en mesure de voir toutes autres fonctionnalités que vous pouvez créer pour votre propre jeu! Mais n’oublier pas, vous devriez créer un jeu qui vous est propre et non simplement copier ce que les autres font. De plus, vous allez certainement rencontrer des problèmes lors du développement de votre jeu, tel que tout le monde, voici quelques problèmes que j’ai rencontrés lors du développement de mon jeu.

# Problèmes rencontrés

Lors de la réalisation de mon expérimentation plusieurs problèmes ont fait surface et que j’ai par conséquent dû les surmonter.

Premièrement, le premier problème auxquelles j’ai dû faire face était un problème au niveau du hardware. En effet, lorsque j’ai commencé la réalisation de mon projet d’approfondissement la température de mon ordinateur à augmenter a une vitesse fulgurante. La raison était relativement simple, mon ordinateur portable n’a pas de carte graphique, de plus celle-ci n’a pas un processeur super efficace. Donc, la solution à ce problème fût relativement simple a trouvé, mais celle-ci me demandais d’autant plus de travail hors des heures de cours. En effet, la solution qui m’était le plus simple était de réaliser le projet sur mon ordinateur à la maison qui est beaucoup plus puissante.

Deuxièmement, lorsque j’ai débuté l’animation de mes personnages, un problème est survenu. Ce problème étant le fait que mes personnages lors de la création de l’animation, ceux-ci ne répondaient pas à aucun des tentatives d’animation que les personnages devaient exécuter, mais plutôt celui-ci lorsque le jeu débutait, il commençait à voler dans le ciel. Donc, pour remédier à la situation, après plusieurs recherches sur internet, j’ai finalement appris que la structure du personnage que je tentais d’utiliser n’étais pas adéquate à l’animation. Finalement, j’ai réussi à trouver une page web avec une multitude de modèle 3D qui son gratuit et également avec plein d’animation qui son compatible avec ces personnages.

Troisièmement, j’ai programmé le déplacement aléatoire de personnage non joueur, mais ceux-ci avaient un problème majeur. Ces personnages percutaient des murs et restaient pris sur ces murs, peu importe combien de temps passait. Donc, pour remédier à la situation, j’ai dû ajuster le mouvement des personnages à l’aide d’une détection de leurs alentours. C’est-à-dire que les personnages sont dorénavant en mesure de voir s’il y a un mur ou tout autre objet qui peuvent leur bloquer le passage. Ainsi, les NPC lorsqu’ils détectaient ces obstacles modifiaient leurs trajectoires pour ne pas rester bloqué. En bref, l’ajout d’un détecteur de collision permet corriger le problème des personnages non joueurs qui restent bloqués sur les murs.

En bref, malgré tous les problèmes que j’ai rencontrés tout au long du développement de mon projet, j’ai réussi à trouver des solutions et finalement créer le produit que j’avais en tête lors du début de l’expérimentation.

# Conclusion

En conclusion, l’expérimentation que j’ai faite au cours des 3 dernières semaines mon appris vraiment beaucoup sur le développement des jeux vidéo en trois dimensions et également sur les prérequis au niveau des systèmes informatiques. Malgré mon but très précis dans cette expérimentation, plusieurs problèmes sont survenus lors de la réalisation de celui-ci, tels que la surcharge de mon ordinateur, l’animation des personnages et le déplacement de ceux-ci. Donc, pour aider toute autre personne qui désire reproduire mon expérimentation avec moins d’embuches, la majorité des solutions ont été indiquées dans le tutoriel. Finalement, ce que je retiens de cette expérimentation est que faire un jeu dans son entièreté seule n’est pas facile.