ATI L3 2021-2022

Support de TD Unity

TD3 (Cours 4) – Fraise Boy

Présentation

Dans ce TD, nous allons créer un jeu reprenant les principes essentiels de Super Meat Boy.

- Mise en place d'un niveau
- Mouvements simples d'un personnage
- Instanciation (création d'objets)
- Détection de collisions et triggers
- Chronomètre et meilleur score

Démarche

Mise en scène

Un niveau et un personnage.

Objectifs	Tâche	Résultat
Créer le MeatBoy [FACILE]	Créez un cube rouge, et mettez-lui des petits yeux noirs.	Vous obtiendrez le portrait craché de Meat Boy.
Créer le niveau [FACILE]	Créez et placez quelques cubes par-ci par-là pour avoir un niveau avec des hauteurs différentes et des trous.	Vous aurez un level design digne des titres AAA.

Meat Boy Simple

L'idée est de réaliser un personnage qu'on peut contrôler avec les flèches directionnelles et de le faire sauter avec Espace. Il peut ainsi entrer en collision avec les murs, le sol, le plafond, ou tout type d'obstacle. Il peut aussi tomber. Tout ça grâce au CharacterController.

Objectifs	Tâche	Résultat
Créer le script [FACILE]	Créez un script C# « MeatBoy »	
Contraindre les Components nécessaires au script [FACILE]	Ajouter un RequireComponent de CharacterController **(support p.3)**	Lorsque vous appliquerez le script sur votre Meat Boy, il vous ajoutera un Component CharacterController dans la même action. Il est très conseillé de retirer le BoxCollider du MeatBoy à ce stade.
Déclaration des variables [FACILE]	Nous aurons besoin de trois float publics : speed, jumpSpeed, et gravity. Nous aurons également besoin d'un Vector3 privé nommé mouvement. Finalement, nous allons avoir besoin d'une variable privée d'un nouveau	Vous aurez 3 variables publiques et 2 privées. N'oubliez pas d'indiquer dans l'Inspector des valeurs aux publiques (par défaut, je vous recommande 5, 8, et

	type : un CharacterController, que nous nommerons controller.	10).
Assigner automatiquement le Character Controller [INTERMEDIAIRE]	Dans le Awake **(support p.4)**, assignez controller avec l'expression suivante : « `GetComponent<CharacterController>();` » **(support p.2-3)**	GetComponent permet de récupérer un autre Component sur l'objet. Comme CharacterController en est un, on peut y accéder. C'est lui qui gère le mouvement physique, et non les opérations de Transform.
Déplacement simple [INTERMEDIAIRE]	Dans le Update, assignez à mouvement.x la valeur du GetAxis Horizontal, multiplié par speed. Réalisez un Move avec le CharacterController **(support p.1-2)**, avec comme paramètre le vecteur mouvement multiplié par le deltaTime.	Lancez le jeu. Vous pouvez déplacer le personnage à gauche et à droite avec les flèches. S'il y a un mur, il ne le traversera pas.
Gravité [INTERMEDIAIRE]	Dans le Update, réduisez mouvement.y de gravity multiplié par le deltaTime. Réalisez une condition isGrounded **(support p.2)**, qui si vérifiée assigne zéro à mouvement.y.	Meat Boy tombe lentement lorsqu'il est en l'air, mais ne peut pas encore sauter.
Saut [FACILE]	Condition if dans Update pour un « `Input.GetButtonDown("Jump");` ». Si vérifiée, donnez à mouvement.y la valeur de jumpSpeed.	Le bonhomme saute, c'est magique ! L'effet en cloche est dû aux deux forces opposées jumpSpeed et gravity.
Double Saut [INTERMEDIAIRE]	On a besoin d'un compteur de sauts. Il nous faut deux variables : public int jumpsMax (égale à 2 par exemple), et un private int jumpsCount. Dans la condition du saut, ajoutez une autre condition pour savoir si jumpsCount est inférieur à jumpsMax. Si oui, incrémentez jumpsCount. Dans la condition isGrounded qu'on a créé plus tôt, ajoutez que jumpsCount doit être réinitialisé à zéro.	Vous pourrez faire autant de sauts consécutifs que de jumpsMax. JumpsCount compte les sauts et reveint à zéro si on touche le sol.

Goutte

L'objectif est de faire des petites gouttes de jus propulsées par le Meat Boy lorsqu'il court. Elles doivent ainsi partir dans le sens opposé au joueur, avoir un « mouvement de goutte », et rester affichées à l'écran. La goutte est un prefab pour pouvoir l'instancier en masse.

Objectifs	Tâche	Résultat
Créer le prefab Goutte [FACILE]	Créez un Cube rouge, plus petit que celui du MeatBoy. Ajoutez-lui un Rigidbody pour qu'il se comporte comme une goutte de jus de viande. Ou de fraise. Ou de tomate.	Vous aurez fait un petit Cube qui tombe vers le bas.

	S'assurer que UseGravity est coché. Contraindre la position en Z et dans tous les axes de rotation. Déplacez l'objet dans la fenêtre Project afin d'en faire un prefab.	
Créer le script [FACILE]	Créez un script C# « Goutte ».	
Projection de la goutte [FACILE]	Créez une variable publique Vector3 velocity (pour la direction) et Float force (pour la force). Dans le Start, faites un GetComponent<Rigidbody>(). AddForce avec pour valeur velocity * force. Dans l'Inspector, assignez (5,5,0) à velocity et 20 à force.	Le cube fait une espèce de saut très caractéristique des liquides fruités.
Au contact, retirez les components innutiles [FACILE]	Ajoutez un OnColliderEnter, dans lequel vous ferez : • un Destroy du GetComponent<Rigidbody>() • un Destroy du this	Il est possible de supprimer le script en cours avec un Destroy(this). C'est pratique pour éviter qu'Unity ne recalcule un tas de choses. Normalement, la goutte ne devrais plus rebondir ou avoir le mouvements superflus.
Création des gouttes quand MeatBoy court [INTERMEDIAIRE]	Revenons dans le script MeatBoy. Ajoutez une variable GameObject pour le prefab de la goutte. Appellons-le gouttePrefab. S'il devait y avoir une instance à chaque frame, il y aurait beaucoup trop de gouttes ! Nous allons donc créer un compteur **(support p.4)**. Nous aurons besoin d'une variable publique float « delayGoutte » et private float « cptGoutte ». Dans le Update, faire une condition qui vérifie si on ne se déplace pas (mouvement.x != 0f). Si vérifiée, décrémenter cptGoutte par le deltaTime. Posez ensuite encore une condition pour savoir si cptGoutte est inférieur à zéro. Si oui, alors faites une Instance de la goutte. Réinitialisez aussi le compteur à la valeur de delayGoutte.	Les gouttes devraient s'instancier avec le bon rythme, mais rester bloquées sur le personnage.
Calque de collision [FACILE]	Créez des Layers pour le Player et la Goutte. Dans la matrice de collision **(support Cours3)**, faire en sorte que la goutte et le player ne puissent pas rentrer en	Les gouttes sont instanciées, traversent le player, mais s'arrêtent bien au sol.

	collision.	
Les gouttes sont propulsées dans la direction opposée du Meat **[DIFFICILE]**	A la place de l'instance de la Goutte, récupérez son Component de script **(support p.3)**, et modifiez l'axe x de la vélocité de la goutte selon la valeur de l'axe x de mouvement.	public GameObject gouttePrefab; public Vector3 goutteOffset;

Scie

Les scies font partie des obstacles standards de Meat Boy. Elles tournent sur elles mêmes. Lorsque Meat Boy les touche, il explose dans un nuage de jus de fraise délicieux et gluant. C'est poétique.

Objectifs	Tâche	Résultat
Créer le script **[FACILE]**	Créez un script « Scie »	
Préparer la Scie **[FACILE]**	Créez un Quad, et assignez-lui un Material en CutOut utilisant la texture que je vous ai fourni. Mettez-lui un collider adapté.	Glissez l'image de sie sur le quad, il va créer un material sie. Changez le Rendering mode de ce material à Fade pour avoir transparence.
Faire tourner la scie sur elle-même **[FACILE]**	Variables float publique « speed ». Dans le Update, faites un Rotate avec Vector3.forward * speed * Time.deltaTime.	La Scie tourne, c'est magnifique.
Instance de goutte au contact **[INTERMEDIAIRE]**	Créer une variable GameObject gouttePrefab. Faites un OnTriggerEnter qui vérifie si le Collider a bien le tag du « Player ». Si oui, Instance de goutte.	Changez le tag de votre character à Player dans inspector.
Changer l'apparence de la scie **[FACILE]**	Créez une variable publique de type Texture « tache ». Après l'instance des gouttes, faites un : `GetComponent<Renderer>().material.mainTexture = tache ;`	Quand le joueur se fera piéger par la scie, celle-ci prend une belle couleur de sirop de fraises. Miam !
Le joueur meurt au contact et réapparait au début du niveau **[DIFFICILE]**	Dans le script MeatBoy. Créez une variable privée defaultPosition (un vecteur 3). Dans le Start, lui assigner la valeur transform.position. Ainsi, le script sauvegarde la position de départ du joueur. Créez une fonction publique **(support p.4)** nommée « Die ». Dedans, assigner à transform.position le defaultPosition. Dans le script Scie, après le changement d'apparence de la scie, faites un : `other.GetComponent<NomDuScriptMeatBoy>().Die() ;`	Le joueur réapparait à sa position d'origine au contact de la scie. C'est possible le CharacterController va empêcher le transform.psosition de Player remettre sur position default. Il faut désactiver le characterController pendant le transform.psosition et le mettra activé juste après. CharacterController cc = controller.GetComponent<CharacterController>();

```
        cc.enabled = false;
        transform.position =
defaultPosition;
        cc.enabled = true;
```

Girl

La Girl est une cerise qui attend que sa fraise vienne la cueillir (comme c'est mignon mignon mignon tout plein de bisous).

Objectifs	Tâche	Résultat
Créer la girl [FACILE]	Réalisez un bel objet pour la Girl. Moi j'ai fait une cerise, mais j'ai hésité avec une banane. C'est adorable.	Ce sera superbe.
Elle saute sur place [DIFFICILE]	Nous allons reprendre une partie du code du MeatBoy. Créez une script « Girl ». Faire un CharacterController pour la Girl. Lui appliquer une gravité. Si elle est isGrounded, la faire sauter.	La Girl saute sur place.

Chronomètre (facultatif)

Un des éléments fun d'un MeatBoy est son chronomètre. Le temps est compté et le but est ainsi de faire le meilleur temps possible.

Objectifs	Tâche	Résultat
Créer le script [FACILE]	Créez un script « Chronometre » (sans accent) et placez-le sur un empty nommé « ChronoManager ».	
Préparer le GUI [FACILE]	Créez un UI Text et placez-le où vous voulez **(support p.5)**. Paramétrez bien le Canvas pour qu'il s'adapte à la taille de l'écran.	Votre Chronomètre sera bien placé.
Incrémentation du temps [FACILE]	Créez une variable float privée « chrono » égale à zéro. Dans le Update, incrémentez cette valeur selon le deltaTime. C'est tout. Time.deltatime met une seconde pour arriver à 1. de fait, nous avons créé un chronomètre.	Votre variable augmente mais rien ne s'affiche sur le UI Text.

Traduction du temps en minutes, secondes, et dixièmes [INTERMEDIAIRE]	Déclarez trois floats privés : « minutes », « secondes », et « fraction ». Ajoutez aussi une variable publique Text « chronoUI » **(support p.6)**
	Après l'incrémentation du chrono :
	`minutes = (int)(chrono/60f);`
	`secondes = (int)(chrono%60f);`
	`fraction = (int)((chrono*100f)%100f);`
	
	`chronoUI.text = "Temps : " + minutes + ":" + secondes + ":" + fraction;`
		Les minutes correspondent à un entier, d'ou la conversion avec le (int). 60 secondes dans une minute, donc division par 60.
		% signifie « modulo » et désigne le reste d'une division.

Chargement du meilleur temps [DIFFICILE]	Déclarez une float bestScore, avec une valeur par défaut élevée (genre 600), et un string bestScoreString, et un string `ScoreSaving pour PlayerPrefs enregistre` Dans le Awake, créez un PlayerPrefs « bestScore » **(support p.6)**, puis assignez-lui la valeur de votre float bestScore. Toujours dans le Awake, déclarez trois variables float : minutes, secondes, et fraction. Réalisez la même opération que ci dessus pour le temps, mais avec le bestScore. Assignez à bestScoreString `"Best : " + minutes + ":" + secondes + ":" + fraction;` Dans le Update, à la ligne qui commence par « chronoUI.text = ... », ajoutez un « `bestScoreString + "\n" +` » avant « `"Temps : "` ».	Vous aurez le chronomètre qui augmente, et le meilleur temps d'affiché juste au dessus, qui indique la meilleure performance. Le « \n » (attention, il s'agit d'un backslash, AltGr+8) indique à Unity qu'il faut faire un retour à la ligne.

Sauvegarde du meilleur temps [DIFFICILE]	Créez un Private string ScoreSave pour enregistrer le bestScore.
	Créez une fonction publique End. Dedans, faites une condition qui demande si chrono est inférieur à bestScore.
	Si oui, Assignez au PlayerPrefs de bestScore la valeur de chrono.
	Après la condition, rechargez le niveau.
	
	Créez un script Fin.
	Créez un Cube et cochez isTrigger dans le BoxCollider. Décochez son Renderer.
	Prennez votre script et mettez-le dans votre cube.
	Déclarez une variable publique Chronometre (oui, le même nom que votre script), du nom de « chronoScript ».
	Glissez-déposez votre Empty avec le script Chronometre dans cette variable (dans l'Inspector).
	Faites un OnTriggerEnter qui vérifie le contact avec le joueur. Dedans, faites un chronoScript.End() ;
		Lorsque le joueur traversera le Cube, le score sera enregistré. Swag ?
		
		Vous pouvez placer le Cube autour de la Cerise Girl.
		
		Enregistrer le valeur de chrono au PlayerPrefs :
		
		`PlayerPrefs.SetFloat(ScoreSave, chrono);`
		
		récupérer le valeur de PlayPrefs :
		
		`bestScore = PlayerPrefs.GetFloat(ScoreSave);`

Améliorations Autonomes

La suite ne comporte pas de solution. C'est à vous d'utiliser ce que vous savez.

MeatBoy

Objectifs	Indices
Lorsque MeatBoy saute contre un mur, il peut sauter de nouveau (wall jump) **[INTERMEDIAIRE]**	Réinitialiser le compteur. Détection de collision d'un mur (pas du sol). Il y a autant d'algorithmes de walljump que de développeurs.
Les gouttes apparaissent au pied et non au centre **[FACILE]**	Offset de l'instance.

Scie

Objectifs	Indices
Belle explosion de gouttes **[INTERMEDIAIRE]**	Random x,y.