

Exercice1 : Les bases

Sélectionner branche 'exercice1' comme point de départ.

Comparer avec branche 'master' pour la version complétée.

Material des personnages

- Pour tous les personnages de Content/Potato/Characters
 - Créer un MaterialInstance
 - Définir M_Male_Body comme parent
 - Observer l'existence d'un paramètre BodyColor dans M_Male_Body
 - Assigner une couleur au paramètre BodyColor
 - Assigner le MaterialInstance au personnage
 - Ouvrir le blueprint d'un personnage Characters/BP_{character}
 - Sélectionner le mesh
 - Assigner Material -> Element0 avec le MaterialInstance défini
- Valider que les personnages de scène ont bien un nouveau matériel

Mise en place de l'acteur Potato

- Créer FNutritionalInformations
 - Créer une nouvelle structure FNutritionalInformations
 - Anoter de `USTRUCT()`
 - Définir 4 champs
 - Calories

```
UPROPERTY(EditAnywhere)  
float Calories;
```

- Glucides

```
UPROPERTY(EditAnywhere)  
float Carbs;
```

- Matières grasses

```
UPROPERTY(EditAnywhere)  
float Fat;
```

- Proteines

```
UPROPERTY(EditAnywhere)  
float Proteins;
```

- Créer APotato

- Créer une nouvelle classe APotato
- Anoter de UCLASS()
- Définir comme descendante de AActor
- Définir 2 champs

- Poids

```
UPROPERTY(EditAnywhere)  
float Weight;
```

- Information nutritionnelles

```
UPROPERTY(EditAnywhere)  
FNutritionalInformations NutritionalInformations;
```

- Créer le StaticMesh

- Importer ImportAssets/Potato/potatoes.FBX
- Considérer utiliser un Import Uniform Scale de 3

- Créer la Texture

- Importer une texture de ImportAssets/Potato/potato_texture.jpg

- Créer le Material

- Créer nouveau Material
- Définir Texture Sample pointant sur la précédente texture et fournir dans la borne "Base Color"
- Définir Constant 0 à "Metalic" et "Specular"
- Définir Constant 1 à "Roughness"

- Créer le blueprint

- Choisir APotato comme classe parent
- Ajouter un StaticMeshComponent
 - Assigner la propriété StaticMesh avec le StaticMesh précédement défini
 - Assigner la propriété Material avec le Matériel précédement défini
- Assigner la propriété Weight et NutritionalInformations

- Valider le blueprint Potato dans la scène en glisser déposer l'acteur dans la scène

Mise en place de l'interaction "Plant"

- Définir classe UPotatoPlantingComponent
 - Créer une nouvelle classe UPotatoPlantingComponent
 - Définir comme descendante de `USceneComponent`
 - Anoter de `UCLASS(meta=(BlueprintSpawnableComponent))`
 - Définir 3 champs
 - Nom du socket de spawn

```
UPROPERTY(EditAnywhere)
FName SpawnSocketName = TEXT("socket_spawn");
```

- Vitesse de spawn

```
UPROPERTY(EditAnywhere)
float SpawnVelocity;
```

- Type de potato à spawn

```
UPROPERTY(EditAnywhere)
TSubclassOf<APotato> PotatoType;
```

- Définir 4 méthodes
 - Enregistrement sur `APotatoBaseCharacter::OnSetupPlayerInput`

```
virtual void InitializeComponent() override
{
    // Enregistrer UPotatoPlantingComponent::OnSetupPlayerInput sur
    l'évènement APotatoBaseCharacter::OnSetupPlayerInput du owner
}
```

- Désenregistrement du `APotatoBaseCharacter::OnSetupPlayerInput`

```
virtual void UninitializeComponent() override
{
    // Désenregistrer UPotatoPlantingComponent::OnSetupPlayerInput de
    l'évènement APotatoBaseCharacter::OnSetupPlayerInput du owner
}
```

- Connecter les inputs

```
void OnSetupPlayerInput(UInputComponent* inputComponent)
{
    // Binder l'input 'fire' sur la méthode
    UPotatoPlantingComponent::PlantPotato
}
```

- Planter une pomme de terre

```
void PlantPotato()
{
    // Récupérer le socket SpawnSocketName sur le modèle
    // Obtenir la world transform du socket
    // Déterminer une vitesse aléatoire dans un cône face au
    personnage d'une magnitude SpawnVelocity
    // Instancier un actor Potato de type PotatoType à la transform
    et vitesse calculée
}
```

- Ajouter UPotatoPlantingComponent au PotatoPlanterCharacter
 - Ajouter champ pour stocker le component

```
UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Interaction)
UPotatoPlantingComponent* PotatoPlantingComponent;
```

- Créer et enregistrer component dans constructeur UPotatoPlanterCharacter()

```
UPotatoPlantingComponent()
{
    PotatoPlantingComponent =
    CreateDefaultSubobject<UPotatoPlantingComponent>
    (TEXT("PotatoPlantComponent"));
    PotatoPlantingComponent->SetupAttachment(RootComponent);
}
```

- Assigner les valeurs au PotatoPlantingComponent
 - Ouvrir BP_PotatoPlanterCharacter
 - Sélectionner PotatoPlantingComponent dans la hiérarchie de composants
 - Assigner "Spawn Socket Name" à "socket_spawn"
 - Assigner "Potato Type" au BP_Potato créé précédemment
 - Assigner "Spawn Velocity" d'une valeur positive entre 5 et 30

- Testez si le Potato Planter est maintenant capable de spawner des potatoes

Mise en place de l'interaction 'PickUp'

- Définir la classe UPotatoPickUpComponent
 - Créer une nouvelle classe UPotatoPickUpComponent
 - Définir comme descendante de `USceneComponent`
 - Anoter `UCLASS(meta=(BlueprintSpawnableComponent))`
 - Définir 2 champs
 - Nom du socket de tenue

```
UPROPERTY(EditAnywhere)
FName heldSocketName = FName("socket_hand_r");
```

- Potato actuellement tenue

```
UPROPERTY(Transient)
APotato* HeldPotato;
```

- Définir 4 méthodes
 - Se connecter aux évènements du owner

```
virtual void InitializeComponent()
{
    // Enregistrer UPotatoPickUpComponent::OnSetupPlayerInput sur
    l'évènement APotatoBaseCharacter::OnSetupPlayerInput du owner
    // Enregistrer UPotatoPickUpComponent::OnOwnerOverlap sur
    l'évènement APotatoBaseCharacter::OnActorBeginOverlap du owner
    // Enregistrer UPotatoPickUpComponent::OnOwnerHit sur l'évènement
    APotatoBaseCharacter::OnActorHit du owner
}
```

- Se déconnecter des évènements du owner

```
virtual void InitializeComponent()
{
    // Dénregistrer UPotatoPickUpComponent::OnSetupPlayerInput sur
    l'évènement APotatoBaseCharacter::OnSetupPlayerInput du owner
    // Dénregistrer UPotatoPickUpComponent::OnOwnerOverlap sur
    l'évènement APotatoBaseCharacter::OnActorBeginOverlap du owner
    // Dénregistrer UPotatoPickUpComponent::OnOwnerHit sur
```

```
l'évènement APotatoBaseCharacter::OnActorHit du owner  
}
```

- Connecter les inputs

```
void OnSetupPlayerInput(UInputComponent* inputComponent)  
{  
    // Binder l'input 'release' sur la méthode  
    UPotatoPickUpComponent::DropPotato()  
}
```

- Prendre les pommes de terre avec lesquels un overlap se produit

```
void OnOwnerOverlap(AActor* owningActor, AActor* otherActor)  
{  
    // Si otherActor est un APotato, invoquer PickupPotato()  
}
```

- Prendre les pommes de terre avec lesquels une collision se produit

```
void OnOwnerHit(AActor* owningActor, AActor* otherActor, FVector  
normalImpulse, const FHitResult& hit)  
{  
    // Si otherActor est un APotato, invoquer PickupPotato()  
}
```

- Prendre une potato

```
void PickupPotato(APotato* potato)  
{  
    // Si IsHoldingPotato() est faux, alors invoquer  
    SetHeldPotato(potato)  
}
```

- Relâcher une potato

```
void DropPotato()  
{  
    // Si IsHoldingPotato() est vrai, alors invoquer
```

```
SetHeldPotato(null)
}
```

- Informe si une potato est tenue

```
bool IsHoldingPotato() const
{
    // Informer si HeldPotato est assigné
}
```

- Exécuter l'action de prendre / relâcher une potato

```
void SetHeldPotato(APotato* potato)
{
    // Définir previous = HeldPotato
    // Assigner HeldPotato = potato
    // Si HeldPotato est défini, désactiver physique + collision sur
    HeldPotato et attacher HeldPotato au socket heldSocketName
    // Si previous est défini, activer physique + collision sur
    previous et détacher previous du socket heldSocketName
}
```

- Ajouter UPotatoPickUpComponent aux PotatoPlanterCharacter, PotatoGathererCharacter et PotatoEaterCharacter
 - Ajouter champ pour stocker le component

```
UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Interaction)
UPotatoPickUpComponent* potatoPickUpComponent = nullptr;
```

- Créer et enregistrer component dans constructeur

```
UPotatoPlanterCharacter() // et UPotatoPlanterGatherer() et
UPotatoEaterCharacter()
{
    potatoPickUpComponent = CreateDefaultSubobject<UPotatoPickUpComponent>
(TEXT("PotatoPickUpComponent"));
    potatoPickUpComponent->SetupAttachment(RootComponent);
}
```

- Testez si tous les personnages sont maintenant capables de prendre et déposer des patates

Mise en place de l'interaction "Eat"

- Définir UPotatoEatingComponent
 - Créer classe UPotatoEatingComponent
 - Définir comme Descendante de USceneComponent
 - Annoter de UCLASS(meta=(BlueprintSpawnableComponent))
 - Définir 1 champ
 - Calories mangées

```
UPROPERTY(Transient)
float CaloriesEaten;
```

- Définir 4 méthodes
 - Enregistrement sur APotatoBaseCharacter::OnSetupPlayerInput

```
virtual void InitializeComponent() override
{
    // Enregistrer UPotatoEatingComponent::OnSetupPlayerInput sur
    l'évènement APotatoBaseCharacter::OnSetupPlayerInput du owner
}
```

- Désenregistrement du APotatoBaseCharacter::OnSetupPlayerInput

```
virtual void UninitializeComponent() override
{
    // Désenregistrer UPotatoEatingComponent::OnSetupPlayerInput de
    l'évènement APotatoBaseCharacter::OnSetupPlayerInput du owner
}
```

- Connecter les inputs

```
void OnSetupPlayerInput(UInputComponent* inputComponent)
{
    // Binder l'input 'fire' sur la méthode
    UPotatoEatingComponent::EatHeldPotato
}
```

- Manger la potato tenue


```
void EatHeldPotato()
{
    // Récupère le PotatoPickUpComponent du owner
    // Si UPotatoPickUpComponent::IsHoldingPotato()
    // Invoquer UPotatoPickUpComponent::Authority_DropPotato()
    pour relâcher et obtenir la potato
    //Invoquer EatPotato(potato)
}
```

- Manger la potato spécifiée

```
void EatPotato(APotato* potato)
{
    // Augmenter CaloriesEaten par
    APotato::NutritionalInformation::Calories
    // Invoquer AActor::Destroy() sur la potato pour la détruire
}
```

- Ajouter UPotatoEatingComponent au PotatoEaterCharacter
 - Ajouter champ pour stocker le component

```
UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Interaction)
UPotatoEatingComponent* potatoEatingComponent = nullptr;
```

- Créer et enregistrer component dans constructeur

```
UPotatoPlanterCharacter() // et UPotatoPlanterGatherer() et
UPotatoEaterCharacter()
{
    potatoEatingComponent = CreateDefaultSubobject<UPotatoEatingComponent>
(TEXT("PotatoEatingComponent"));
    potatoEatingComponent->SetupAttachment(RootComponent);
}
```

- Testez si le Potato Eater est maintenant capable de manger la potato tenue