Exercice2 : Gameplay Framework

Sélectionner branche 'exercice2' comme point de départ.

Comparer avec branche 'master' pour la version complétée.

Mise en place initiale du GameRole: APotatoGameRole

- Définir APotatoGameMode
  - Créer nouvelle classe `APotatoGameMode`
  - Hériter de `AGameMode`
  - Laisser vide pour l'instant, suite à venir
- Créer un blueprint BP_PotatoGameMode
  - Créer nouveau blueprint class `BP_PotatoGameMode`
  - Hériter de `APotatoGameMode`
- Assigner le APotatoGameRole au niveau
  - Ouvrir l'onglet Window -> World Settings
  - Assigner `BP_PotatoGameMode` dans GameMode Override du niveau

Mise en place du PlayerState: APotatoPlayerState

- Définir EGameRoleType
  - Créer nouveau enum `EGameRoleType`

```
UENUM()
enum class EGameRoleType
{
    Planter,
    Gatherer,
    Eater
};
```

- Définir FPotatoGameRole
  - Créer nouvelle structure `FPotatoGameRole`
  - Définir 2 champs
    - Valeur numérique de rôle

```
UPROPERTY(EditAnywhere)
EGameRoleType RoleType;
```

    - Type de personnage pouvant être possédé par ce rôle

```
UPROPERTY(EditAnywhere)
TSubclassOf<APotatoBaseCharacter> CharacterType;
```

- Définir APotatoPlayerState
  - Créer nouvelle classe `APotatoPlayerState`
  - Hériter de `APlayerState`
  - Définir 1 champ
    - Role courrant

```
UPROPERTY(Transient)
FPotatoGameRole CurrentRole
```

- Créer BP_APotatoPlayerState
  - Créer nouveau blueprint class `BP_APotatoPlayerState`
  - Utiliser `APlayerState` comme classe de base
- Assigner le APotatoPlayerState au niveau
  - Ouvrir l'onglet Window -> World Settings
  - Assigner `APotatoPlayerState` dans Player State Class du niveau

Mise en place du PlayerController: APotatoPlayerController

- Définir APotatoPlayerController
  - Créer nouvelle classe `APotatoPlayerController`
  - Hériter de APlayerController
  - Définir 3 méthodes
    - Changer de rôle

```
void ChangeRole()
{
    // Récuperer le APotatoGameMode
    // Déléguer exécution sur APotatoGameMode::ChangeRole(this)
}
```

    - Quitter la partie

```
void QuitGame()
{
    // Récuperer le APotatoGameMode
    // Déléguer exécution sur APotatoGameMode::QuitGame(this)
}
```

- Connecter aux inputs

```
virtual void SetupInputComponent() override
{
    // Binder "Switch" sur APotatoPlayerController::Server_ChangeRole
    // Binder "Quit" sur APotatoPlayerController::QuitGame
}
```

- Créer BP_PotatoPlayerController
  - Créer nouveau blueprint class `BP_PotatoPlayerController`
  - Utiliser `APotatoPlayerController` comme classe de base
- Assigner le APotatoPlayerController au niveau
  - Ouvrir l'onglet Window -> World Settings
  - Assigner `BP_PotatoPlayerController` dans Player Controller Class du niveau

Mise en place du GameState: APotatoGameState

- Définir APotatoGameState
  - Créer nouvelle classe `APotatoGameState`
  - Hériter de `AGameState`
  - Définir 1 champ
    - Indicateur de termination de la partie

```
UPROPERTY(Transient)
bool GameEnded = false
```

- Créer BP_PotatoGameState
  - Créer nouveau blueprint class `BP_PotatoGameState`
  - Utiliser `APotatoGameState` comme classe de base
- Assigner le APotatoGameState au niveau
  - Ouvrir l'onglet Window -> World Settings
  - Assigner `BP_PotatoGameState` dans Game State Class du niveau

Compléter le GameMode: APotatoGameMode

- Compléter APotatoGameMode
  - Définir 2 champs
    - Ensemble des rôles disponibles

```

```

```
UPROPERTY(EditAnywhere)
TArray<FPotatoGameMode> Roles;
```

- Définir 4 méthodes
  - Vérification de la fin de la partie

```
void CheckGameEnded()
{
    // Collectionner UPotatoEatingComponent des APotatoBaseCharacters
    // Si aucun component est UPotatoEatingComponent::IsHungry(),
        // Récupérer APotatoGameState
        // Invoquer APotatoGameState::SetGameEnded(true)
}
```

  - Function de tick

```
virtual void Tick(float dt) override
{
    // Invoquer APotatoGameMode::CheckGameEnded()
}
```

  - Trouver un personnage à posséder en fonction d'un type

```
void test()
{
    // Itérer sur tous les APotatoBaseCharacter du world
        // Vérifier si APotatoBaseCharacter est du bon type...
            // avec AActor::IsA(type)
        // Vérifier character n'est pas possédé par un joueur...
            //  avec !APawn::GetController()
        // Si vrai au deux, retourner le character
}
```

  - Changer de rôle

```
void ChangeRole(APotatoPlayerController* requester)
{
    // Récupérer APlayerState associé au requester
    // Itérer sur roles autres que APlayerState::GetCurrentRole()
```

```
        // Trouver personnage (FindSuitableCharacter(type))
        // Si character trouvé
            // Posséder character...
                // avec APotatoPlayerController::Possess(character)
            // Assigner rôle...
                // avec APotatoPlayerState::SetCurrentRole(role)
            // Arrêter l'itération
}
```

- Assigner rôle au démarrage

```
virtual RestartPlayer(AController* playerController)
{
    // Caster playerController en APotatoPlayerController...
        // avec Cast<APotatoPlayerController>(playerController))
    // Si IsValid(potatoPlayerController),
        // Invoquer ChangeRole(potatoPlayerController)
}
```

- Vérifier le bon fonctionnement du changement de rôle et condition de victoire.