Start new C++ basic project

Copy assets to content folder

Create a pawn C++ class called PacmanPawn

Add following code to PacmanPawn.h file

protected:

UPROPERTY(BlueprintReadOnly)

bool Frozen = true;

public:

void SetDirection(const FVector Direction);

bool IsFrozen() { return Frozen; }

UFUNCTION(BlueprintCallable)

void SetFrozen(bool Value) { Frozen = Value; }

private:

UFUNCTION()

void OnOverlapBegin(AActor\* PlyaerActor, AActor\* OtherActor);

Create a Blueprint folder

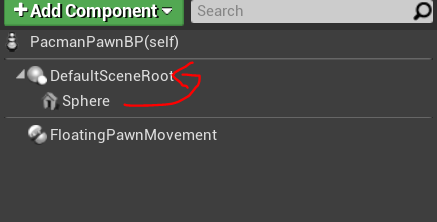
Create a BP version of the PacmanPawn

Add 2 components

* + Floating Pawn Movement
  + Static mesh Sphere

Give pacman material for sphere and make sphere the default root.

Scale sphere to 0.5

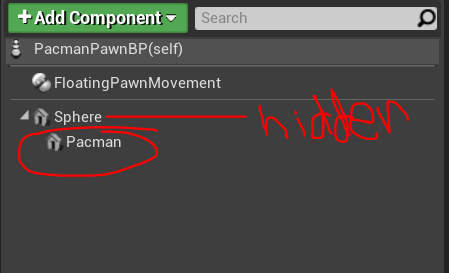


Add another static mesh and name it pacman

Select pacman static mesh from details panel

Change scale to 1.5

Set visibility hidden for the root sphere



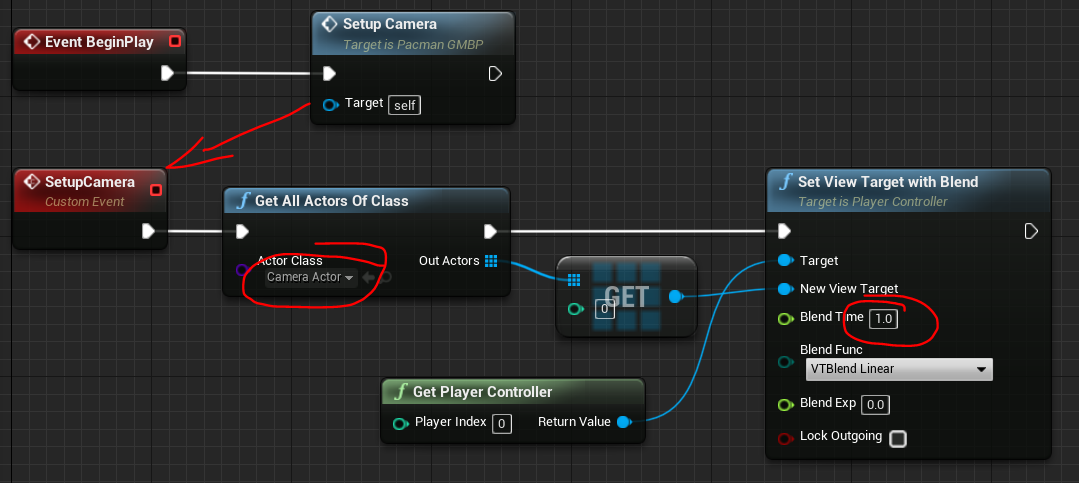
Place the pacman in level to see the scale is right.

Level can be found in Content\Pacman\Levels

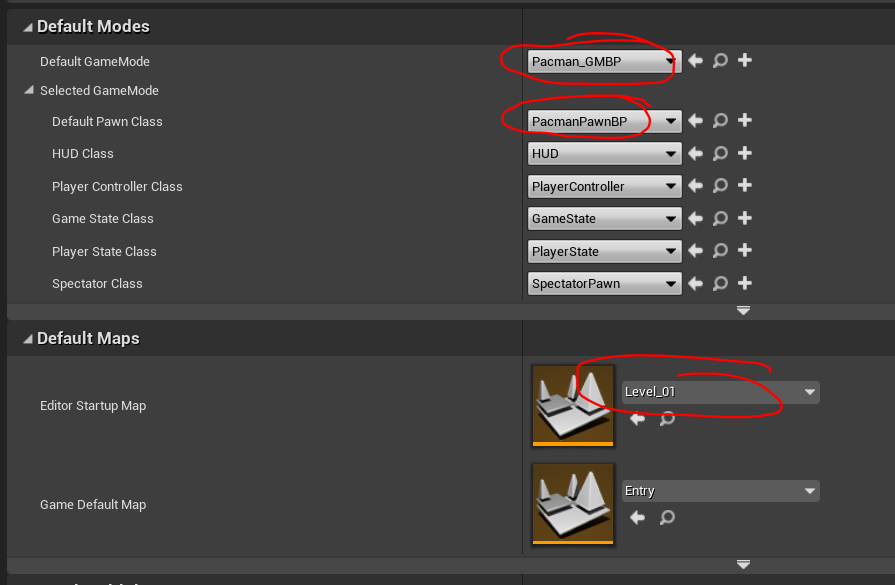
When we play we will not be able to see the whole game.

To fix do the following.

Add a custom event called “SetupCamera” and add the following BP



In the project settings make the following changes



When we start the game we should see the whole level.

Now add following code to begin play of Pawn

OnActorBeginOverlap.AddDynamic(this, &APacmanPawn::OnOverlapBegin);

Add following to Tick

if (!Frozen) {

AddMovementInput(GetActorForwardVector());

}

Add following to set Direction method

if (Direction == FVector::UpVector) {

SetActorRotation(FRotator(0.0f, 270.0f, 0.0f));

}

else if (Direction == FVector::DownVector) {

SetActorRotation(FRotator(0.0f, 90.0f, 0.0f));

}

else if (Direction == FVector::RightVector) {

SetActorRotation(FRotator(0.0f, 0.0f, 0.0f));

}

else if (Direction == FVector::LeftVector) {

SetActorRotation(FRotator(0.0f, 180.0f, 180.0f));

}

Now we need to create a controller.

So in the editor create a C++ Player Controller called PacmanController

Add following code to header file.

protected:

void SetupInputComponent() override;

APacmanPawn\* GetPacmanPawn() const;

public:

void MoveUp();

void MoveDown();

void MoveRight();

void MoveLeft();

forward decleration required

class APacmanPawn;

create definition for all methods in the CPP file

Change definition of GetPacmanPawn as following

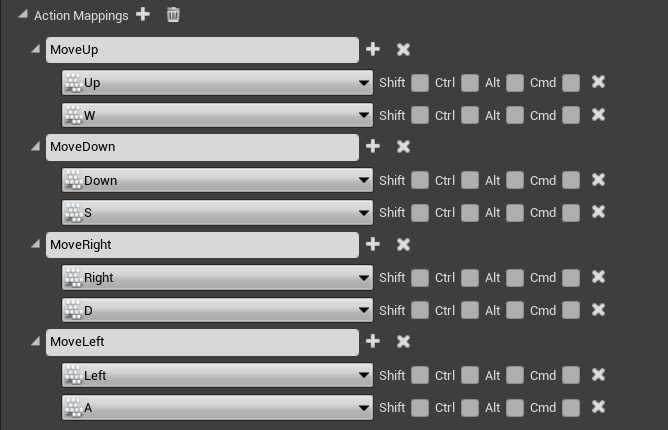
APacmanPawn\* APacmanController::GetPacmanPawn() const

{

return Cast<APacmanPawn>(GetPawn());

}

And input binding as following.



Add the following code to SetupInputComponent method

Super::SetupInputComponent(); //Dont mind the error line below SetupInputComponent

InputComponent->BindAction("MoveUp", IE\_Pressed, this, &APacmanController::MoveUp);

InputComponent->BindAction("MoveDown", IE\_Pressed, this, &APacmanController::MoveDown);

InputComponent->BindAction("MoveLeft", IE\_Pressed, this, &APacmanController::MoveLeft);

InputComponent->BindAction("MoveRight", IE\_Pressed, this, &APacmanController::MoveRight);

Change the move methods as following

void APacmanController::MoveUp()

{

if (GetPacmanPawn() != nullptr) {

GetPacmanPawn()->SetDirection(FVector::UpVector);

}

}

void APacmanController::MoveDown()

{

if (GetPacmanPawn() != nullptr) {

GetPacmanPawn()->SetDirection(FVector::DownVector);

}

}

void APacmanController::MoveRight()

{

if (GetPacmanPawn() != nullptr) {

GetPacmanPawn()->SetDirection(FVector::RightVector);

}

}

void APacmanController::MoveLeft()

{

if (GetPacmanPawn() != nullptr) {

GetPacmanPawn()->SetDirection(FVector::LeftVector);

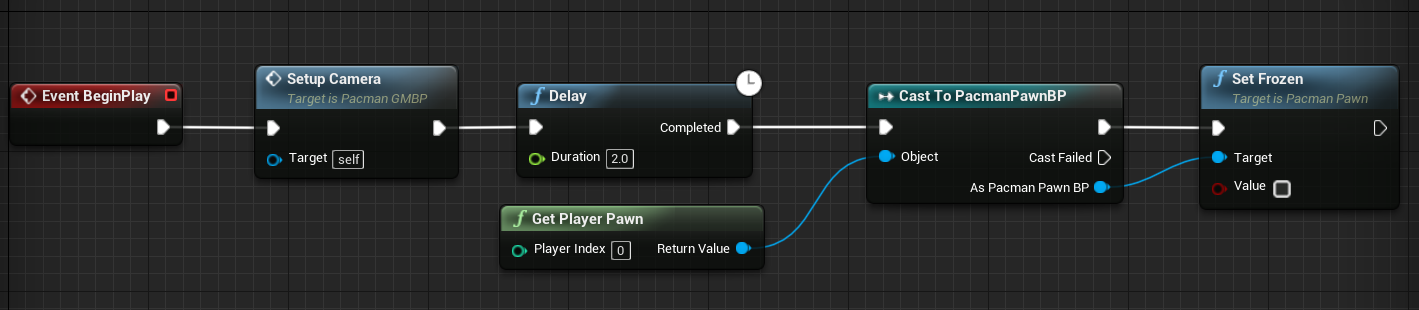
}

}

Change the player controller to PacmanController in the project settings

Pacman should be able to turn but not move.

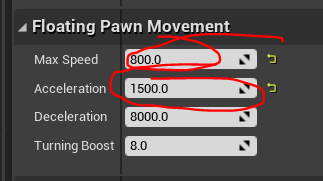
Change the begin play of GameModeBP as follows



Now pacman should be able to move.

If it penetrates through wall adjust the size of sphere and pacman in pacmanpawn.

Make following changes in FloatingPawnMovement.



Create a C++ actor class called Food.

In the Header just after includes create an enum as following

UENUM(BlueprintType)

enum class EFoodType : uint8 {

Regular,

PowerUp

};

DECLARE\_DYNAMIC\_MULTICAST\_DELEGATE\_OneParam (FFoodEatenEvent, EFoodType, FoodType); //Multicast delegate to lookup (kind of subscription)

Add the following code in the end

void Consume();

UPROPERTY(EditAnywhere)

EFoodType FoodType = EFoodType::Regular;

UPROPERTY(CPF\_BlueprintAssignable, BlueprintCallable)

FFoodEatenEvent FoodieEatenEvent;

private:

UPROPERTY(EditAnywhere)

USoundCue\* ConsumtionSound;

Need to forward declare USoundCue;

In the C++ file add the includes

#include "Kismet/GameplayStatics.h"

#include "Sound/SoundCue.h"

Create implementation of consume method as follows

void AFood::Consume()

{

UGameplayStatics::PlaySound2D(this, ConsumtionSound);

FoodEatenEvent.Broadcast(FoodType);

Destroy();

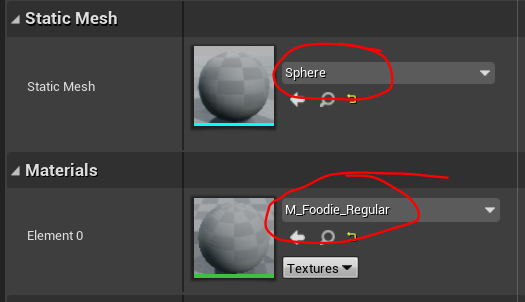
}

Create a BP version of Food

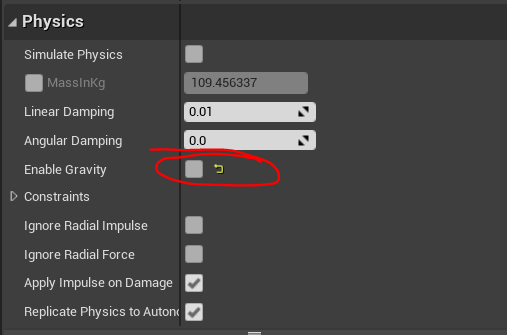
In the class defaults select sound Eat Regular



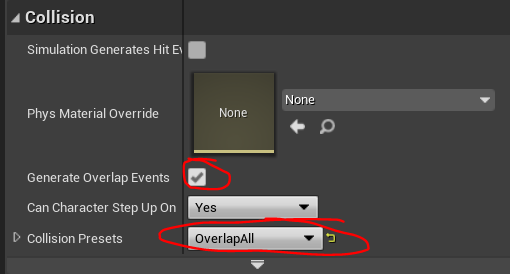
Add a sphere in components and select food regular material



Disable gravity



In collision select overlap all and make sure generate overlalp event is on

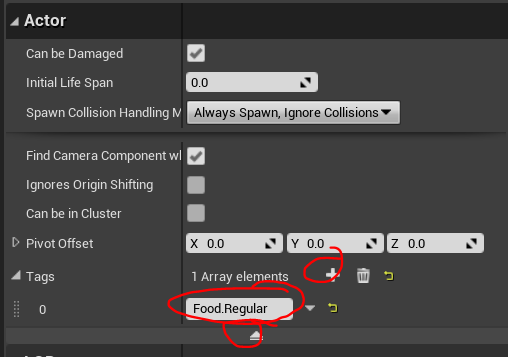


Place the food in level.

We need to create a tag for food

Open FoodBP and click ClassDefaults button

Expand actor section click on + sign and add a tag called “Food\_Regular”



Compile and save

Open PacmanPawn.cpp

Add following code to OnOverlapBegin

void APacmanPawn::OnOverlapBegin(AActor\* PlyaerActor, AActor\* OtherActor)

{

if (OtherActor->ActorHasTag("Food.Regular")) {

Cast<AFood>(OtherActor)->Consume();

}

}

We need to include Food.h

#include "Food.h"

Food should be able to be consumed

Distribute the food along the level.

**Create Enemy**

Create a C++ Pawn class and name it EnemyPawn.

In the enemy header file add the following code

UENUM(BlueprintType)

enum class EEnemyState : uint8 {

Default,

Idle

};

DECLARE\_DYNAMIC\_MULTICAST\_DELEGATE\_OneParam(FEnmemyStateChangedEvent, EEnemyState, NewState);

Delete event tick, begin play SetupPlayerInputComponent and contructor

Add the following code

public:

// Sets default values for this pawn's properties

UPROPERTY(EditAnywhere, BlueprintReadWrite)

EEnemyState State = EEnemyState::Default;

UFUNCTION(BlueprintCallable)

void Hunt();

UFUNCTION(BlueprintCallable)

void Idle();

FEnemyStateChangedEvent& OnStateChanged() { return StateChangedEvent; }

private:

UPROPERTY(BlueprintAssignable, BlueprintCallable)

FEnemyStateChangedEvent StateChangedEvent;

Create implementation of Hunt and Idle

In CPP file delete event tick, begin play SetupPlayerInputComponent and contructor function defenitions

Add following code to Hunt and Idle functions

void AEnemyPawn::Hunt()

{

State = EEnemyState::Default;

StateChangedEvent.Broadcast(State);

}

void AEnemyPawn::Idle()

{

State = EEnemyState::Idle;

StateChangedEvent.Broadcast(State);

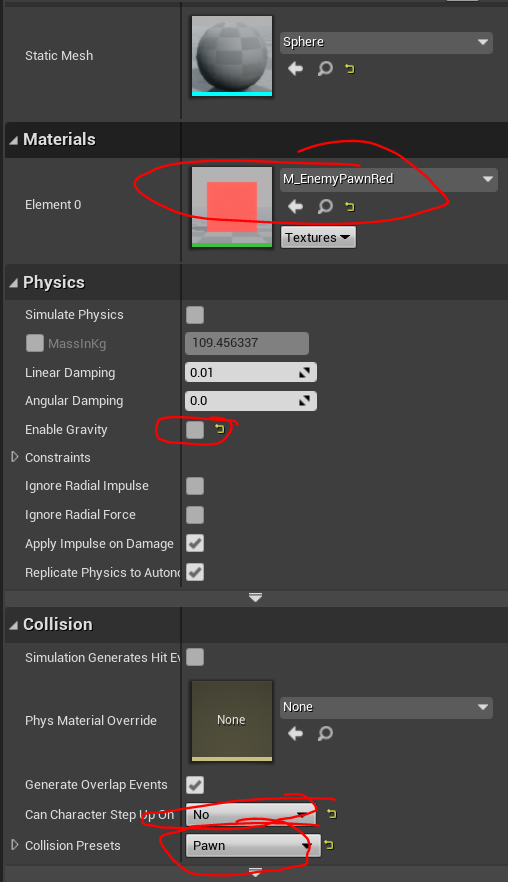
}

Sometimes restarting VS required

Create a BP version of Enemy

Add a sphere to the components

Set enemy material, Disable gravity, set collision to Pawn and step on NO

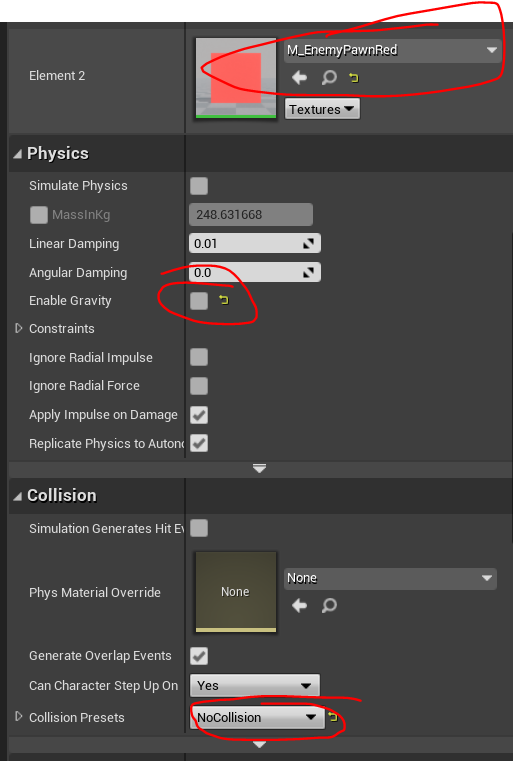


Make Visibility off in rendering section

Add another static mesh

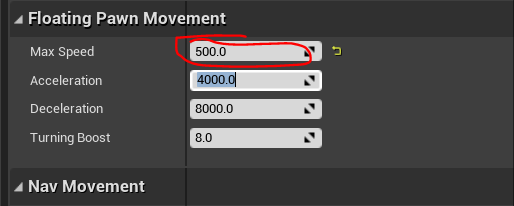
Select Enemy default in mesh list in details panel

Set collision to NoCollision and disable gravity



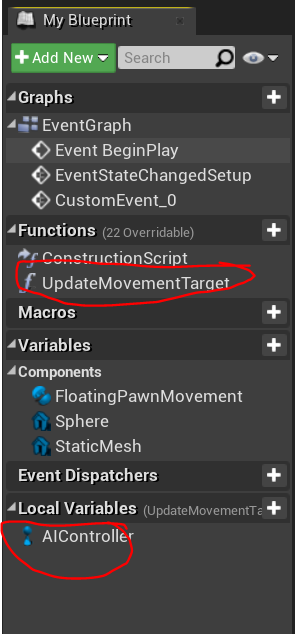
Add floating pawn movement in components

Add 500 max speed

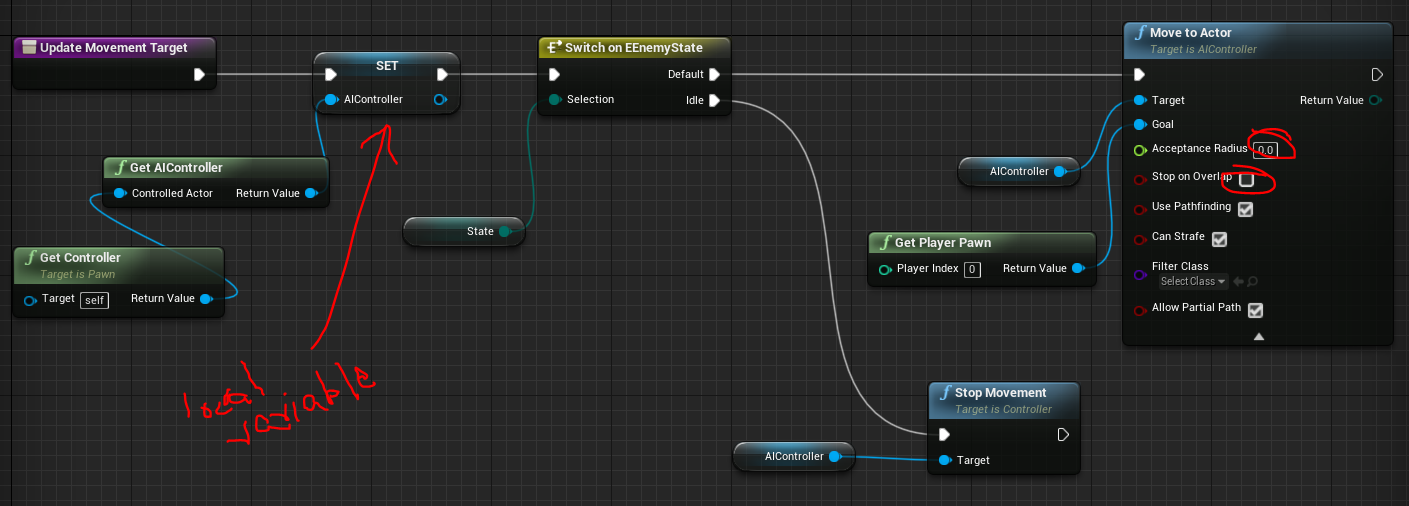


Open Enemy BP and add function UpdateMovementTarget

Create a local variable called AIController of the type AIController



Add the following BP to the new function

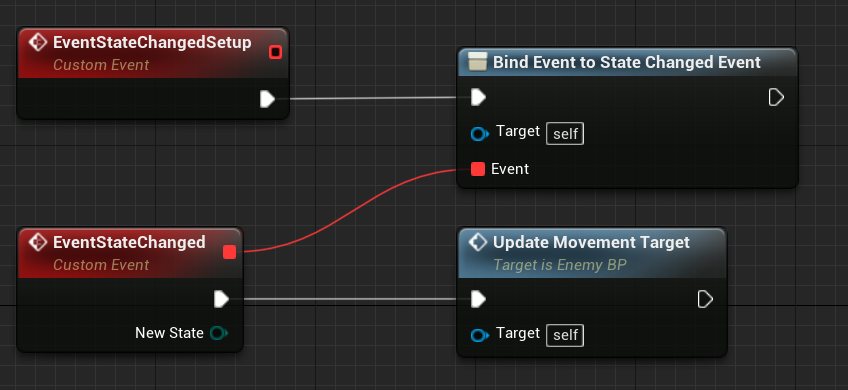


Now in the event graph add custom event EventStateChangedSetup

Add another Custom event called EventStateChanged

Add a parameter to it “NewState” of the type EEnemyState

Add the following BP to the events

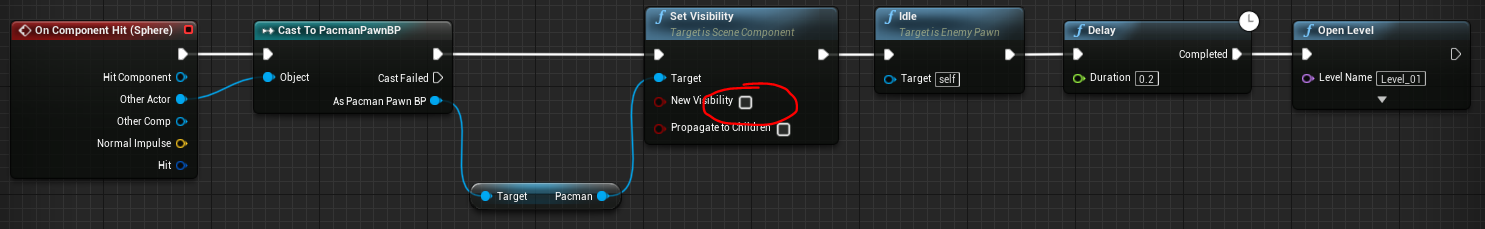


Add following BP to begin play of Enemy



Enemy should be working now.

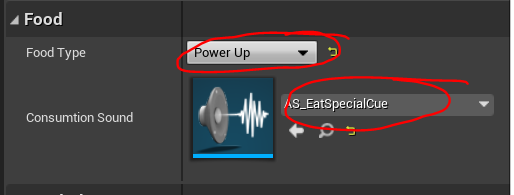
To handle hit add the following bp



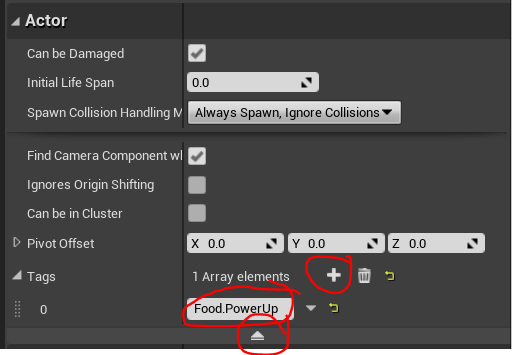
**PowerUp Food**

Create another BP version of Food called Food\_PowerUp

In the details panel change sound and food type

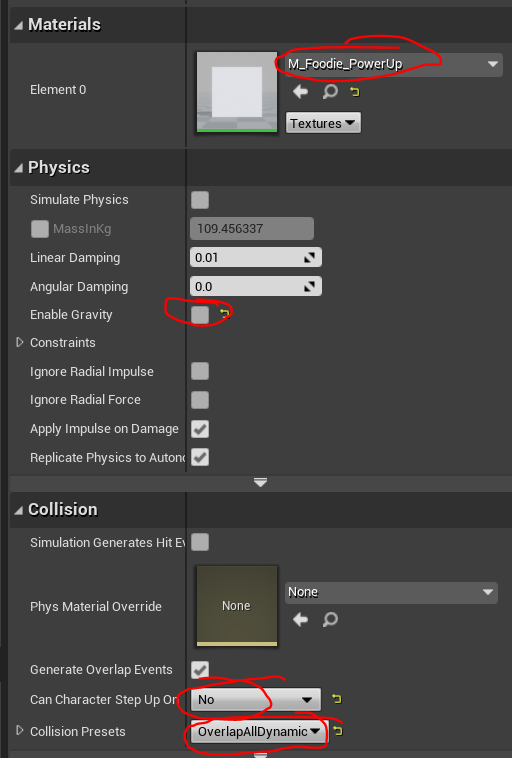


In the Actor section add a tag called “Food.PowerUp”



Add a sphere to the actor and

Make the following changes to the sphere in details panel



Add and if to the OnOverlapBegin function of PacmanPawn as follows

if (OtherActor->ActorHasTag("Food.PowerUp")) {

Cast<AFood>(OtherActor)->Consume();

}

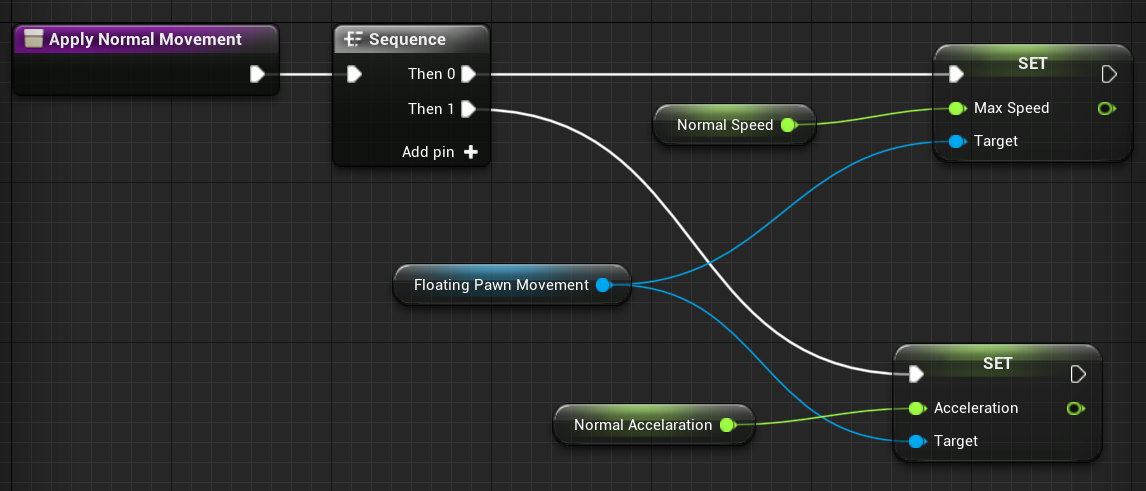
In PacmanPawnBP add 5 float variables with default values as follows

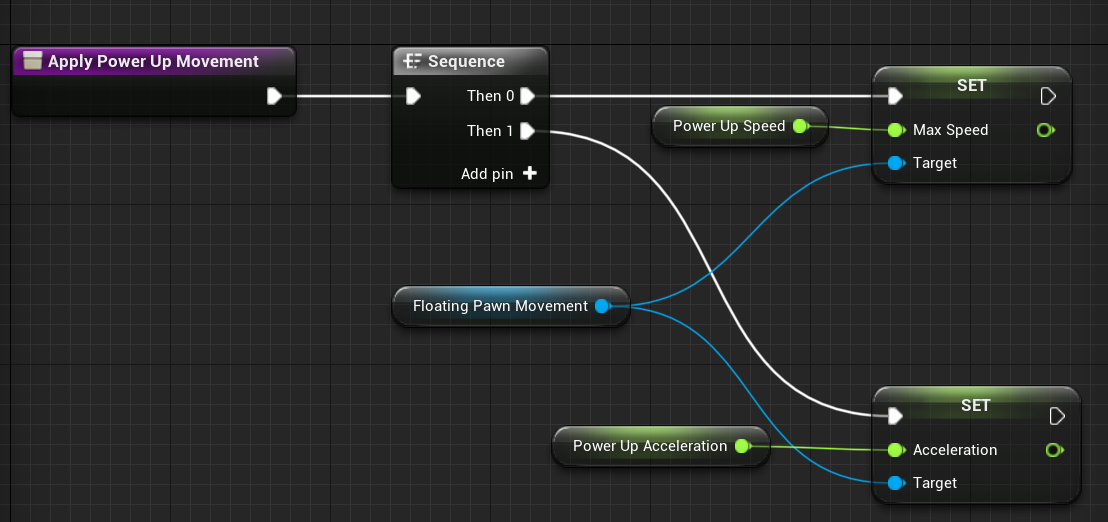
1. Normal Speed = 800
2. Normal Acceleration = 4000
3. PowerUp Speed = 1600
4. PowerUp Acceleration = 8000
5. PowerUp Duration = 3

Create 2 functions

1. ApplyNormalMovement
2. ApplyPowerUpMovement

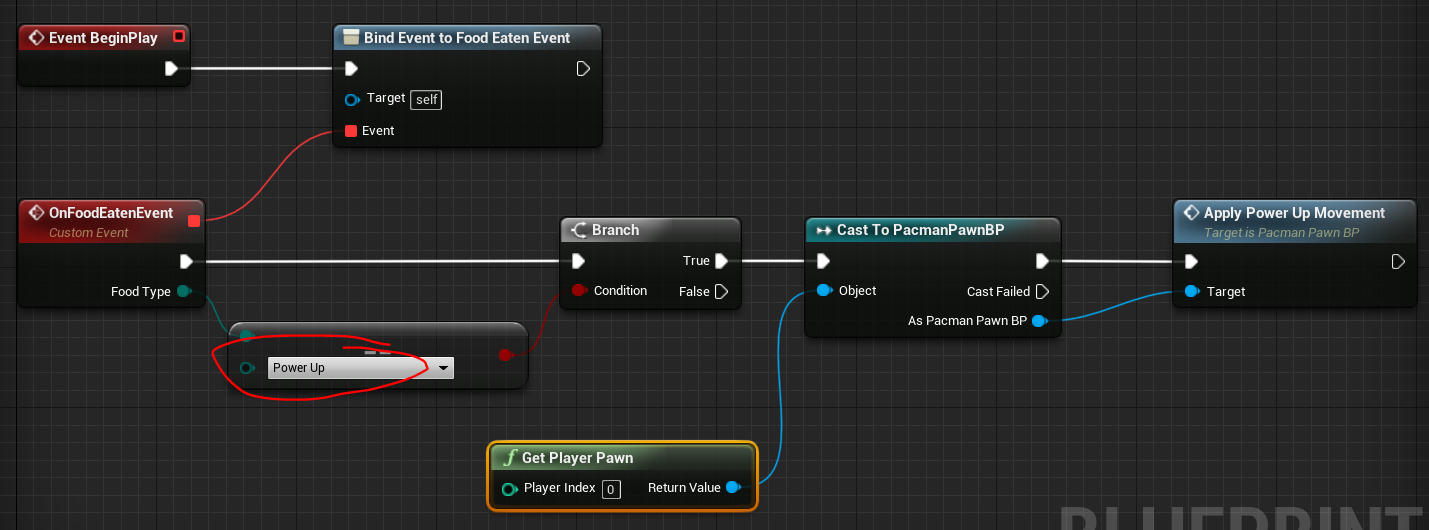
Add following BP to functions





In the Power up event graph create a custom event called OnFoodEatenEvent with an input parameter called FoodType of the type EFoodType Enum

Add the following BP

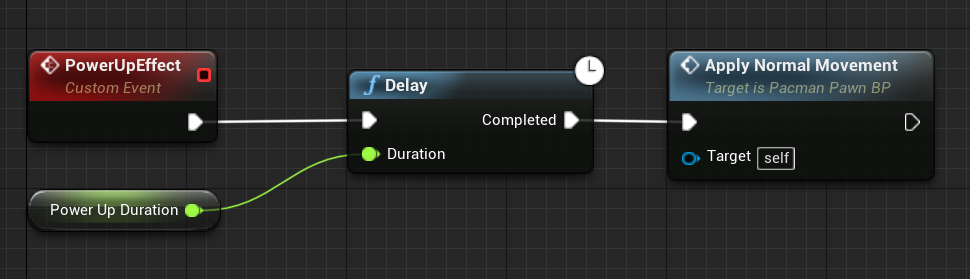


Place the actor in level.

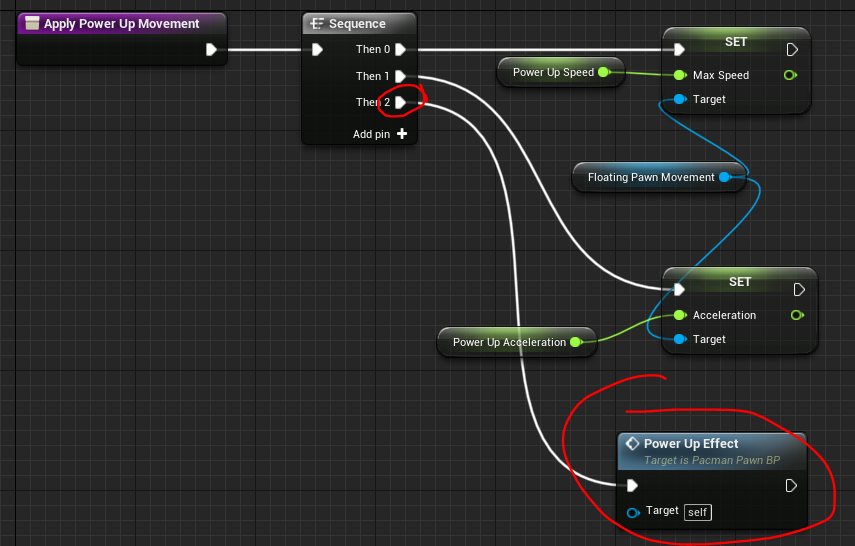
Player should be able to do power up

Now need to delay and change back to normal speed.

Create a custom event in PacmanPawnBP event graph called PowerUpEffect and add the following BP



Add another pin to sequence in ApplyPowerUpMovement and add new even PowerUpEffect as follows



**Teleporting**

Create and Actor C++ class called TeleporterActor

Add following code to header file

void TeleportToTarget(AActor Actor);

UPROPERTY(EditAnywhere)

ATeleporterActor\* Target = nullptr;

UPROPERTY(EditAnywhere)

USoundCue\* TeleporterSound;

UFUNCTION()

void OnOverlapBegin(AActor\* TeleporterActor, AActor\* OtherActor);

Forward decleration of USoundCue required

Create definition for TeleportToTarget and OnOverlpaBegin functions

In the C++ file add the following includes

#include "Engine/Public/TimerManager.h"

#include "Kismet/GameplayStatics.h"

#include "Sound/SoundCue.h"

#include "Components/SceneComponent.h"

Add following code to OnOverlapBegin function

void ATeleporterActor::OnOverlapBegin(AActor\* TeleporterActor, AActor\* OtherActor)

{

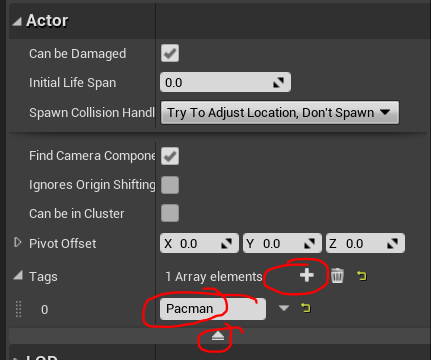
if (OtherActor->ActorHasTag("Pacman")) {

GetWorldTimerManager().SetTimerForNextTick([OtherActor, this]() {TeleportToTarget(OtherActor); });

}

}

Add the tag Pacman to the PacmanPawnBP



Add the following code for TeleportToTarget function

void ATeleporterActor::TeleportToTarget(AActor\* Actor)

{

USceneComponent\* TargetSpawn = Cast<USceneComponent>(Target->GetDefaultSubobjectByName("Spawn"));

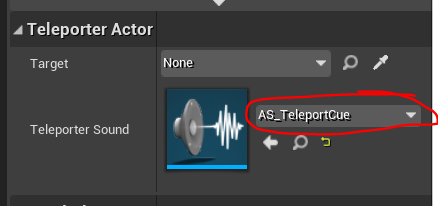
UGameplayStatics::PlaySound2D(this, TeleporterSound);

Actor->SetActorLocation(TargetSpawn->GetComponentLocation());

}

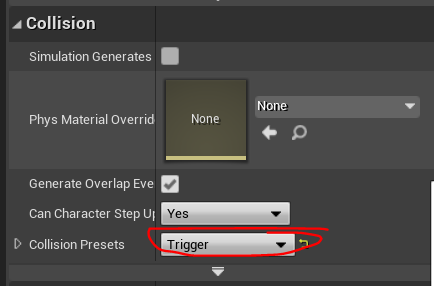
Create a BP version of the Teleporter

Add sound to the sound variable



Add a BoxCollision component

Change the collision present to trigger

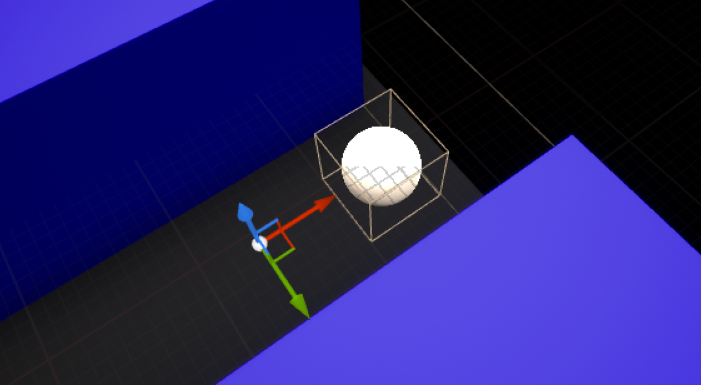


Add a scene component and rename it to “Spawn”

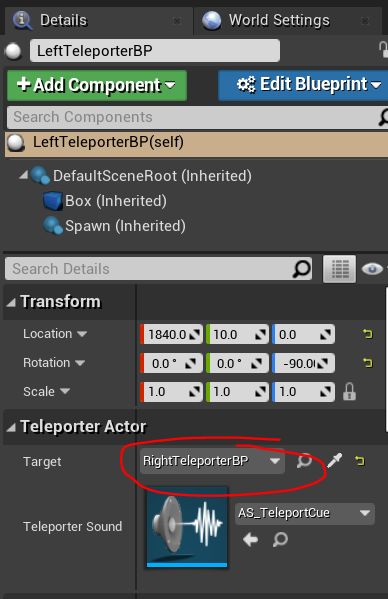
Set Translate Y to -100.

Place 2 TeleporterBP on either side of the map.

Rotate so that the spawn (scene component) is facing inside



Select the LeftTeleporter and in the details panel select RightTeleporter from dropdown list



Might need to adjust the size of Collision Box and shift Spawn (SceneComponent) to work properly

**Broadcasting summery**

DECLARE\_DYNAMIC\_MULTICAST\_DELEGATE\_OneParam (FFoodEatenEvent, EFoodType, FoodType);

DECLARE\_DYNAMIC\_MULTICAST\_DELEGATE\_OneParam

MULTICAST – to access in BP

FFoodEatenEvent – Event declaration

EFoodType – Parameter type passed to the event

FoodType – Name of Parameter