**Brick Game**

Start a new blank project.

Delete all components in scene and save the level.

Copy assests such as Materials and Mesh folders

Drag and drop SM\_Backdrop and scale to 10.

Add wall and position to the origin

Add a Pawn C++ class of the name “Paddle”

Add 2 variables in header file

UPROPERTY(VisibleAnywhere, BlueprintReadOnly)

UStaticMeshComponent\* SM\_Paddle;

UPROPERTY(VisibleAnywhere, BlueprintReadOnly)

UFloatingPawnMovement\* FloatingMovement;

Forward declaration required

class UFloatingPawnMovement;

Add 2 includes in the C++ file.

#include "GameFramework/FloatingPawnMovement.h"

#include "Components/StaticMeshComponent.h"

In the constructor in C++ file, do the following setup

SM\_Paddle = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("SMPaddle"));

RootComponent = SM\_Paddle;

SM\_Paddle->SetEnableGravity(false);

SM\_Paddle->SetConstraintMode(EDOFMode::XZPlane);

SM\_Paddle->SetCollisionEnabled(ECollisionEnabled::QueryAndPhysics);

SM\_Paddle->SetCollisionProfileName(TEXT("PhysicsActor"));

FloatingMovement = CreateDefaultSubobject<UFloatingPawnMovement>(TEXT("Floating Pawn Movement"));

Add a functions in header file

virtual void MoveHorizontal(float AxisValue);

Create its definition and add the following in its definition.

AddMovementInput(FVector(AxisValue, 0.0f, 0.0f), 1.0f, false);

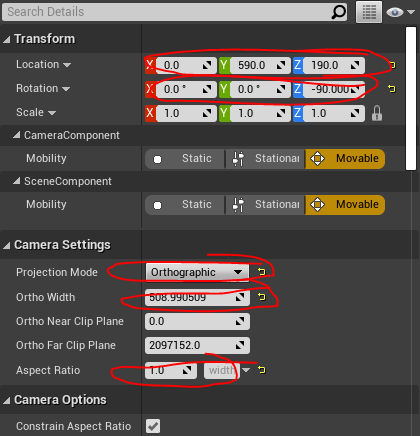
Create a BP version of the pawn in the editor.

Select the SM\_Paddle component in the BP and select the Paddle static mesh from the details panel.

Create a Game mode in the the BP folder and name it BrickGameMode

Select the Paddle pawn as the default pawn in the game mode and set the game mode in the maps and modes in project settings.

Add a camera and adjust the following settings to get a full view of the scene.



Create a PlayerController C++ class and name it Brick\_PC

Add following code in the header file

ABrick\_PC();

UFUNCTION()

virtual void SetupInputComponent() override;

protected:

virtual void BeginPlay() override;

void MoveHorizontal(float AxisValue);

Create definition for all 4 methods

Add following includes in the C++ file

First we set the camera added as target camera to view the game.

For that do this in BeginPlay

TArray<AActor\*> CameraActor;

UGameplayStatics::GetAllActorsOfClass(GetWorld(), ACameraActor::StaticClass(), CameraActor);

FViewTargetTransitionParams Params;

SetViewTarget(CameraActor[0], Params);

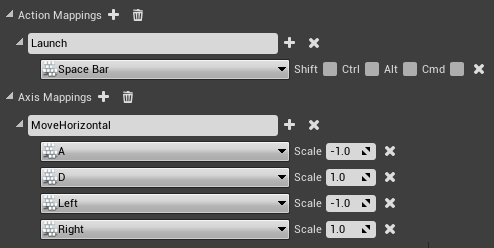
Compile and test

Create a BP version of the player controller

Set it as the PC in the game mode

Play to see the result

Add the following input binding



In the C++ file go to setup input component method and add the following

Super::SetupInputComponent();

EnableInput(this);

InputComponent->BindAxis("MoveHorizontal", this, &ABrick\_PC::MoveHorizontal);

The paddle should be working now.

**Creating Ball**

Create and actor class called “Ball”

In the header file of Ball add the following in Public

virtual void LaunchBall();

bool BallLaunched;

In protect under the begin play method add the following

UPROPERTY(VisibleAnywhere, BlueprintReadOnly)

UStaticMeshComponent\* SM\_Ball;

UPROPERTY(VisibleAnywhere, BlueprintReadOnly)

UProjectileMovementComponent\* ProjectileMovement;

Forward declaration required

class UProjectileMovementComponent;

In public add the following method deceleration

UFUNCTION()

UStaticMeshComponent\* GetBall();

Create definition of both functions LaunchBall and GetBall.

In C++ file add following include files

#include "Components/StaticMeshComponent.h"

#include "GameFramework/ProjectileMovementComponent.h"

Add the following in the constructor of the C++ file

SM\_Ball = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("Ball Mesh"));

RootComponent = SM\_Ball;

SM\_Ball->SetSimulatePhysics(true);

SM\_Ball->SetEnableGravity(false);

SM\_Ball->SetConstraintMode(EDOFMode::XZPlane);

SM\_Ball->SetCollisionEnabled(ECollisionEnabled::QueryAndPhysics);

SM\_Ball->SetCollisionProfileName(TEXT("PhysicsActor"));

ProjectileMovement = CreateDefaultSubobject<UProjectileMovementComponent>(TEXT("Projectile Movement"));

ProjectileMovement->bShouldBounce = true;

ProjectileMovement->Bounciness = 1.1f;

ProjectileMovement->Friction = 0.0f;

ProjectileMovement->Velocity.X = 0.0f;

Add the folloing code to LaunchBall

if (!BallLaunched)

{

SM\_Ball->AddImpulse(FVector(140.0f, 0.0f, 130.0f), FName(), true);

BallLaunched = true;

}

Add the following to GetBall

return SM\_Ball;

Create a BP version of the Ball class called “Ball\_BP”

Select the ball mesh in the static mesh component in the BP version

Place the ball in the world and resize accordingly.

**2:03:20 time to start next**

adfaf