```cpp

// Fill out your copyright notice in the Description page of Project Settings.


#include "MyCharacter4.h"
#include "UObject/ConstructorHelpers.h"
#include "Camera/CameraComponent.h"
#include "Components/StaticMeshComponent.h"
#include "Components/InputComponent.h"
#include "GameFramework/SpringArmComponent.h"
#include "Engine/World.h"
#include "Engine/StaticMesh.h"
// Sets default values
AMyCharacter4::AMyCharacter4()
{
    // Set this character to call Tick() every frame.  You can turn this off to improve performance if you don't need it.
    PrimaryActorTick.bCanEverTick = true;

    AutoPossessPlayer = EAutoReceiveInput::Player0;

    bUseControllerRotationYaw = false;

    cam = CreateDefaultSubobject<UCameraComponent>(TEXT("Camera"));
    cam->AttachTo(RootComponent);
    cam->SetRelativeLocation(FVector(0, 0, 40));



}

// Called when the game starts or when spawned
void AMyCharacter4::BeginPlay()
{
    Super::BeginPlay();
    grounded = true;

    CurrentVelocity;

    YawRotation;    //Y axis rotation

    PitchRotation;  //Z axis rotation

    RollRotation;   //X axis rotation

    Gravity = -9.8f;

    Acceleration = 200.0f;

    Mass = 20400.0f;

    Volume = 60.0f;
```

```
52
53      Density = Mass/Volume;
54
55      JetReferenceArea = 2.5f;
56
57      JetWingArea = 84.0f;
58
59      JetFighterDragCoeffecient = 0.016f;
60
61      XRotation;
62
63      YRotation;
64
65      ZRotation;
66  }
67
68  // Called every frame
69  void AMyCharacter4::Tick(float DeltaTime)
70  {
71      Super::Tick(DeltaTime);
72
73
74      CalculateYaw();
75      CalculatePitch();
76
77      //Calculations
78      CalculateLift();
79      CalculateGravity();
80      CalculateDrag();
81
82      UE_LOG(LogTemp, Warning, TEXT("%f"), YLength);
83      UE_LOG(LogTemp, Warning, TEXT("%f"), XLength);
84
85
86      GetVelocity();
87      const FVector ThrustVector = FVector(XLength*(Thrust*20),YLength*(Thrust
            * 20), ZLength * (Thrust * 20));
88      const FVector DragVector = FVector(Drag, 0, 0);
89      const FVector LiftVector = FVector(0, 0, FMath::Sqrt(Lift));
90      const FVector GravityVector = FVector(0, 0, -Gravity);
91      FVector changeinrotation(0,0,0);
92      const FVector LocalMove = FVector(XAxisSpeed * DeltaTime, 0.f,
          ZAxisSpeed*DeltaTime);
93      //FRotationMatrix(GetActorRotation()).GetScaledAxis(EAxis::X);
94     // AddMovementInput(DragVector, 1);
95      AddMovementInput(ThrustVector, 1);
96      AddMovementInput(GravityVector, 1);
97      AddMovementInput(LiftVector, 1);
98
99
100
101     XAxisSpeed = GetVelocity().X;
102     YAxisSpeed = GetVelocity().Y;
```

```
103     ZAxisSpeed = GetVelocity().Z;
104     ZRotation = GetActorRotation().Yaw;
105     YRotation = GetActorRotation().Pitch;
106     Velocity = XAxisSpeed;
107     //UE_LOG(LogTemp, Warning, TEXT("THIS SPEED: %f"), CurrentVelocity);
108 //  AddMovementInput(LocalMove, 1000);
109
110 }
111
112 // Called to bind functionality to input
113 void AMyCharacter4::SetupPlayerInputComponent(UInputComponent*
      PlayerInputComponent)
114 {
115     Super::SetupPlayerInputComponent(PlayerInputComponent);
116     //Y axis
117     InputComponent->BindAxis("Yaw", this, &AMyCharacter4::Yaw);
118     //X axis
119     InputComponent->BindAxis("XAxisMovement", this,
        &AMyCharacter4::ForwardThrust);
120     InputComponent->BindAxis("Roll", this, &AMyCharacter4::Roll);
121     //Z axis
122     InputComponent->BindAxis("Pitch", this, &AMyCharacter4::Pitch);
123     //Additional thrust
124     InputComponent->BindAxis("JetBoost", this,
        &AMyCharacter4::JetEnginePush);
125     //Camera Rotation
126     InputComponent->BindAxis("VerticalViewRotation", this,
        &AMyCharacter4::VerticalRot);
127     InputComponent->BindAxis("HorizontalViewRotation", this,
        &AMyCharacter4::HorizontalRot);
128
129     //Changes Camera View
130     InputComponent->BindAction("TestButton", IE_Pressed,this,
        &AMyCharacter4::ChangeCamView);
131     InputComponent->BindAction("TestButton", IE_Released, this,
        &AMyCharacter4::ChangeCamViewBack);
132     //InputComponent->BindAxis("XAxisRot", this, &AMyCharacter4::XRot);
133 }
134
135 //Y axis
136 void AMyCharacter4::Yaw(float value) {
137     //Y axis rotation  A,D
138
139         AddActorLocalRotation(FRotator(0, value, 0));
140
141 }
142 void AMyCharacter4::ForwardThrust(float value) {
143         CalculateThrust(value);
144         CalculateAcceleration(value);
145         //const FVector SPEED = FVector(5000.f, 0.f, 0.f);
146         //AddMovementInput(GetActorForwardVector(), value);
147     //  AddMovementInput(GetActorRightVector(), value);
148         //CurrentForwardSpeed = CurrentForwardSpeed + GetActorForwardVector
```

```
            ().X;
149         speed = speed + value;
150 }
151 //X axis
152 void AMyCharacter4::Pitch(float value) {
153         //Z axis rotation    Up,Down
154         AddActorLocalRotation(FRotator(value, 0, 0));
155 }
156 //Z axis
157 void AMyCharacter4::Roll(float value) {
158         //X axis rotation    Right,Left
159         AddActorLocalRotation(FRotator(0, 0, value));
160 }
161 void AMyCharacter4::JetEnginePush(float value) {
162         AddMovementInput(GetActorUpVector(), value);
163     //  UE_LOG(LogTemp, Warning, TEXT("PUSHED"));
164 }
165 //Camera functions
166 void AMyCharacter4::HorizontalRot(float value) {
167         cam->AddLocalRotation(FRotator(0, value, 0));
168 }
169 void AMyCharacter4::VerticalRot(float value) {
170     //  float temp = cam->GetRelativeRotation().Pitch + value;
171
172 }
173 void AMyCharacter4::HorizontalMove(float value) {
174     if (value) {
175     //  AddMovementInput(GetActorRightVector(), value);
176     }
177 }
178 void AMyCharacter4::ChangeCamView(void) {
179     cam->AddLocalRotation(FRotator(0, 180, 0));
180     cam->SetRelativeLocation(FVector(1200,0, 200));
181 }
182 void AMyCharacter4::ChangeCamViewBack(void) {
183     cam->AddLocalRotation(FRotator(0, 180, 0));
184     cam->SetRelativeLocation(FVector(-1750.0, 0, 390));
185 }
186 void AMyCharacter4::CalculateThrust(float value) {
187
188     Thrust = Mass * value;
189 }
190 void AMyCharacter4::CalculateDrag() {
191
192
193     Drag = JetFighterDragCoeffecient* ((Mass / Volume) * ( FMath::Square
          (Velocity)) / 2)* JetReferenceArea;
194
195 }
196
197 void AMyCharacter4::CalculateLift() {
198     Lift = JetFighterDragCoeffecient* ((Mass / Volume) * ( FMath::Square
          (Velocity)) / 2)* JetWingArea;
```

```
199
200 }
201 void AMyCharacter4::CalculateGravity() {
202
203     Gravity = Mass * 9.8;
204 }
205
206 void AMyCharacter4::CalculateAcceleration(float value) {
207
208 //  Acceleration = 0;
209 //  Acceleration = value;
210
211 }
212 void AMyCharacter4::CalculateYaw() {
213     //Returns a value between 0 and 1 that determines player vector based on ↵
        the rotation
214     XLength = FMath::Cos(FMath::DegreesToRadians(ZRotation));
215     YLength = FMath::Sin(FMath::DegreesToRadians(ZRotation));
216 }
217 void AMyCharacter4::CalculatePitch() {
218     //Returns a value between 0 and 1 that determines player vector based on ↵
        the rotation
219     //XLength = FMath::Cos(FMath::DegreesToRadians(YRotation));
220     ZLength = FMath::Sin(FMath::DegreesToRadians(YRotation));
221
222 }
223 void AMyCharacter4::CalculateRoll() {
224     //Returns a value between 0 and 1 that determines player vector based on ↵
        the rotation
225     ZLength = FMath::Cos(FMath::DegreesToRadians(XRotation));
226     YLength = FMath::Sin(FMath::DegreesToRadians(XRotation));
227 }
228
229
230
```