```cpp
//KeepMowingALawnCharacter.cpp

// Copyright 1998-2018 Epic Games, Inc. All Rights Reserved.

#include "KeepMowingALawnCharacter.h"
#include "HeadMountedDisplayFunctionLibrary.h"
#include "Camera/CameraComponent.h"
#include "Components/CapsuleComponent.h"
#include "Components/InputComponent.h"
#include "Components/BoxComponent.h"
#include "Components/AudioComponent.h"
#include "Engine.h"
#include "TimerManager.h"
#include "GameFramework/CharacterMovementComponent.h"
#include "GameFramework/Controller.h"
#include "GameFramework/SpringArmComponent.h"

//////////////////////////////////////////////////////////////////////////
// AKeepMowingALawnCharacter

AKeepMowingALawnCharacter::AKeepMowingALawnCharacter()
{
    // Set size for collision capsule
    GetCapsuleComponent()->InitCapsuleSize(42.f, 96.0f);


    // Don't rotate when the controller rotates. Let that just affect the camera.
    bUseControllerRotationPitch = false;
    bUseControllerRotationYaw = false;
    bUseControllerRotationRoll = false;

    // Create a follow camera
    FollowCamera = CreateDefaultSubobject<UCameraComponent>(TEXT("FollowCamera"));
    FollowCamera->SetupAttachment(RootComponent, USpringArmComponent::SocketName);


    /*/Code I Made/Edited /*/

    /** Below are camera settings **/
    FVector NewCameraPosition(-27.8, 0, -30.7886);
    FRotator NewCameraRotation(-10, 0, 0);
    FVector NewCameraScale(0, 0, 0);

    FTransform NewCameraTransform(NewCameraRotation, NewCameraPosition, NewCameraScale);
    FollowCamera->SetRelativeTransform(NewCameraTransform);


    SkyViewCamera = CreateDefaultSubobject<UCameraComponent>(TEXT("SkyViewCamera"));
    SkyViewCamera->SetupAttachment(RootComponent, USpringArmComponent::SocketName);

    NewCameraPosition.Set(-12.5545, 0, 30);
    SkyViewCamera->SetRelativeTransform(FTransform(FRotator(-90, 0,
    0),FVector(-12.5445,0,1346),FVector(0,0,0)));
    SkyViewCamera->Deactivate();


    /** Below are box collision objects which handle tire tracking **/
    LeftTirePos = CreateDefaultSubobject<UBoxComponent>(TEXT("LeftTirePos"));
    RightTirePos = CreateDefaultSubobject<UBoxComponent>(TEXT("RightTirePos"));

    LeftTirePos->SetupAttachment(RootComponent);
    RightTirePos->SetupAttachment(RootComponent);

    LeftTirePos->SetRelativeLocation(FVector(-50, -40, -80));
    RightTirePos->SetRelativeLocation(FVector(-50, 40, -80));

    LeftTirePos->SetRelativeScale3D(FVector(.5, .5, .5));
    RightTirePos->SetRelativeScale3D(FVector(.5, .5, .5));

```

```
69      /** below are components for the sound of the mower **/
70      MowerSounds = CreateDefaultSubobject<UAudioComponent>(TEXT("SoundEffects"));
71
72      MowerSounds->SetupAttachment(RootComponent);
73
74
75      // Note: The skeletal mesh and anim blueprint references on the Mesh component
        (inherited from Character)
76      // are set in the derived blueprint asset named MyCharacter (to avoid direct
        content references in C++)
77  }
78
79  //////////////////////////////////////////////////////////////////////////
80  // Input
81
82  void AKeepMowingALawnCharacter::SetupPlayerInputComponent(class UInputComponent*
    PlayerInputComponent)
83  {
84      // Set up gameplay key bindings
85      check(PlayerInputComponent);
86
87      PlayerInputComponent->BindAxis("MoveLeft", this,
        &AKeepMowingALawnCharacter::MoveLeft);
88      PlayerInputComponent->BindAxis("MoveRight", this,
        &AKeepMowingALawnCharacter::MoveRight);
89
90      //Control Mower Speeds
91      PlayerInputComponent->BindAction("ShiftUp",IE_Pressed, this,
        &AKeepMowingALawnCharacter::ShiftUp);
92      PlayerInputComponent->BindAction("ShiftDown", IE_Pressed, this,
        &AKeepMowingALawnCharacter::ShiftDown);
93
94      //Allow Mower Sound to be turnd on/off
95      PlayerInputComponent->BindAction("Enable/Disable Mower Sound", IE_Pressed, this,
        &AKeepMowingALawnCharacter::ChangeMowerSoundState);
96
97      //Allows for the camera to be moved when the mower is stopped
98      PlayerInputComponent->BindAxis("LookRight", this,
        &AKeepMowingALawnCharacter::LookRight);
99      PlayerInputComponent->BindAxis("LookLeft", this,
        &AKeepMowingALawnCharacter::LookLeft);
100
101     //Allows for the Camera to toggle between Skyview and First Person
102     PlayerInputComponent->BindAction("ChangeCameraView", IE_Pressed, this,
        &AKeepMowingALawnCharacter::ChangeCameraView);
103
104     // VR headset functionality
105     PlayerInputComponent->BindAction("ResetVR", IE_Pressed, this,
        &AKeepMowingALawnCharacter::OnResetVR);
106 }
107
108
109 void AKeepMowingALawnCharacter::OnResetVR()
110 {
111     UHeadMountedDisplayFunctionLibrary::ResetOrientationAndPosition();
112 }
113
114
115 /** Returns the value of the left joystick **/
116 void AKeepMowingALawnCharacter::MoveLeft(float Value)
117 {
118     LeftJoystickValue = Value;
119 }
120
121 /** Returns the value of the right joystick **/
122 void AKeepMowingALawnCharacter::MoveRight(float Value)
123 {
124     RightJoystickValue = Value;
125 }
```

```
126
127
128 void AKeepMowingALawnCharacter::Tick(float Delta)
129 {
130     Super::Tick(Delta);
131
132     FRotator NewCharacterRotation = GetActorRotation();
133
134     //Left Joystick Press
135     float leftangle = 360 + (SpeedMultiplier * LeftJoystickValue);
136     float rightangle = 360 + (SpeedMultiplier * RightJoystickValue);
137     float distancechange = (SpeedMultiplier * LeftJoystickValue) + (SpeedMultiplier *
        RightJoystickValue);
138
139
140     if (LeftJoystickValue != 0 || RightJoystickValue != 0)
141     {
142
143         /** this block of if statements forces the driver to not be able to overturn at
            higher speeds **/
144         if (distancechange > 3 && RightJoystickValue < -.4)
145             distancechange = 3;
146         else if (distancechange < -3 && LeftJoystickValue > .4)
147             distancechange = -3;
148         else if (distancechange < -3 && RightJoystickValue > .4)
149             distancechange = -3;
150         else if (distancechange > 3 && LeftJoystickValue < -.4)
151             distancechange = 3;
152
153         NewCharacterRotation.Yaw += distancechange;
154
155         FVector NewLocation = CalculatePosition(rightangle, leftangle);
156
157         /** This checks if the object hit something while moving and tracks distance
            accordingly**/
158         FHitResult hitresult;
159         if (!SetActorLocation(NewLocation, true, &hitresult))
160         {   UE_LOG(LogClass, Log, TEXT("Blocked"));}
161         else if(LeftJoystickValue != 0 && RightJoystickValue != 0)
162         {   DistanceTravelled += PotentialDistanceTravelled;}
163
164         SetActorRotation(NewCharacterRotation);
165     }
166
167         FRotator RotateCamera = GetActorRotation();
168
169         FHitResult hitresult;
170         RotateCamera.Add(0, LookLeftValue + LookRightValue, 0);
171         FollowCamera->SetWorldRotation(RotateCamera,true, &hitresult);
172
173         //UE_LOG(LogClass, Log, TEXT("PositionTesting %f, %f"), GetActorLocation().X,
            GetActorLocation().Y);
174
175 }
176
177
178 /* CalculateRotation
179 *This function shall take in two FVectors containing location points. One to be rotated
    (currentworldloc) and one to be the pivot (PivotPoint) about an angle
    (AngleofChangeDegree)
180 */
181 FVector AKeepMowingALawnCharacter::CalculateRotation(FVector CurrentWorldLoc, FVector
    PivotPoint, float AngleofChangeDegree)
182 {
183
184     float ShiftCoordinatesX = CurrentWorldLoc.X - PivotPoint.X;
185     float ShiftCoordinatesY = CurrentWorldLoc.Y - PivotPoint.Y;
186
187     /*Mat is short for Matrix, as these are the values for the matrix rotation*/
```

```
188         float MatSinDeg = FMath::Sin(FMath::DegreesToRadians(AngleofChangeDegree));
189         float MatCosDeg = FMath::Cos(FMath::DegreesToRadians(AngleofChangeDegree));
190 
191         float RotatedCoordinatesX = (ShiftCoordinatesX * MatCosDeg) - (ShiftCoordinatesY *
            MatSinDeg);
192         float RotatedCoordinatesY = (ShiftCoordinatesX * MatSinDeg) + (ShiftCoordinatesY *
            MatCosDeg);
193 
194         FVector UpdatedCoordinatesVector(RotatedCoordinatesX + PivotPoint.X,
            RotatedCoordinatesY + PivotPoint.Y, CurrentWorldLoc.Z);
195 
196         //UE_LOG(LogClass, Log, TEXT("SinFloat: %f"), UpdatedCoordinatesVector.X);
197         //UE_LOG(LogClass, Log, TEXT("CosFloat: %f"), UpdatedCoordinatesVector.Y);
198 
199         return UpdatedCoordinatesVector;
200 
201     }
202 
203     /**
204     * Calculates the position of a point between two vectors about a given world coordinant
        (originpoint)
205     * Note: Uses the (LocationTwo) variable for the magnitude of the new point
206     **/
207     FVector AKeepMowingALawnCharacter::LocationCombiner(FVector LocationOne, FVector
        LocationTwo, FVector OriginPoint, float multiplier)
208     {
209 
210         //Calculate new true vector direction
211         FVector leftlocalvector = FVector(LocationOne.X - OriginPoint.X, LocationOne.Y -
            OriginPoint.Y, 0);
212         FVector rightlocalvector = FVector(LocationTwo.X - OriginPoint.X, LocationTwo.Y -
            OriginPoint.Y, 0);
213 
214         float distancetotravel = rightlocalvector.Size() * multiplier;
215         PotentialDistanceTravelled = distancetotravel;
216 
217         FVector newplacement = leftlocalvector + rightlocalvector;
218         newplacement.Normalize(20);
219 
220         //Updates the normalized newplacement with the needed magnitude and put it in world
            coordiantes.
221         newplacement.Set((newplacement.X * distancetotravel) + OriginPoint.X,
            (newplacement.Y * distancetotravel) + OriginPoint.Y, OriginPoint.Z);
222 
223         return newplacement;
224 
225     }
226 
227     /*
228     * This will calculate the new position for the lawnmower based upon the given angles.
229     * Rightangle is for the rightjoysticks value yeild.
230     * Leftangle is for the leftjoysticks value yeild.
231     *
232     * Note: This code is wasteful in order to increase readability, as it is already
        difficult to read
233     */
234 
235     FVector AKeepMowingALawnCharacter::CalculatePosition(float rightangle, float leftangle)
236     {
237 
238         FVector currentlocation = GetActorLocation();
239         FVector currentlefttire = LeftTirePos->GetComponentLocation();
240         FVector currentrighttire = RightTirePos->GetComponentLocation();
241         //Calculate Right First Rotation
242 
243         FVector newlefttire = CalculateRotation(currentlefttire, currentrighttire,
            leftangle);
244         FVector newrightfirstcenter = CalculateRotation(currentlocation, currentrighttire,
            leftangle);
```

```
245
246         newrightfirstcenter = CalculateRotation(newrightfirstcenter, newlefttire,
            rightangle);
247
248         //Calculate Left First Rotation
249         FVector newrighttire = CalculateRotation(currentrighttire, currentlefttire,
            rightangle);
250         FVector newleftfirstcenter = CalculateRotation(currentlocation, currentlefttire,
            rightangle);
251
252         newleftfirstcenter = CalculateRotation(newleftfirstcenter, newrighttire, leftangle);
253
254         //Calculate new true vector direction
255         return LocationCombiner(newleftfirstcenter, newrightfirstcenter, currentlocation, 1);
256
257
258     }
259
260     void AKeepMowingALawnCharacter::ShiftUp()
261     {
262         if (SpeedMultiplier < 10)
263             SpeedMultiplier++;
264     }
265
266     void AKeepMowingALawnCharacter::ShiftDown()
267     {
268         if (SpeedMultiplier > 3)
269             SpeedMultiplier--;
270     }
271
272     void AKeepMowingALawnCharacter::LookRight(float value)
273     {
274         LookRightValue = value * 90;
275     }
276
277
278     void AKeepMowingALawnCharacter::LookLeft(float value)
279     {
280         LookLeftValue = value * 90;
281     }
282
283     void AKeepMowingALawnCharacter::BeginPlay()
284     {
285         Super::BeginPlay();
286         SpeedMultiplier = 5;
287         FRotator NewCameraRotation = GetActorRotation();
288         FollowCamera->SetWorldRotation(NewCameraRotation, true);
289         TimeElapsed = 0;
290         DistanceTravelled = 0;
291         GetWorldTimerManager().SetTimer(TimeMangementHandler, this,
            &AKeepMowingALawnCharacter::IncreaseTime, 1.0f, true, 0.0f);
292
293     }
294
295     void AKeepMowingALawnCharacter::ChangeMowerSoundState()
296     {
297         if (MowerSounds->IsPlaying())
298         {
299             MowerSounds->Stop();
300         }
301         else
302         {
303             MowerSounds->Play();
304         }
305     }
306
307
308     void AKeepMowingALawnCharacter::StopMowerSound()
309     {
```

```
310         if (MowerSounds->IsPlaying())
311         {
312             MowerSounds->Stop();
313         }
314 
315         return;
316     }
317 
318     float AKeepMowingALawnCharacter::GetGearStat()
319     {
320 
321         return SpeedMultiplier;
322     }
323 
324 
325     void AKeepMowingALawnCharacter::ChangeCameraView()
326     {
327         if (FollowCamera->IsActive())
328         {
329             FollowCamera->Deactivate();
330             SkyViewCamera->Activate();
331         }
332         else
333         {
334             SkyViewCamera->Deactivate();
335             FollowCamera->Activate();
336         }
337     }
338 
339     void AKeepMowingALawnCharacter::IncreaseTime()
340     {
341         TimeElapsed += 1;
342     }
343 
344     float AKeepMowingALawnCharacter::GetElapsedTime()
345     {
346         //if (GEngine)
347         //  GEngine->AddOnScreenDebugMessage(-1, 15.0f, FColor::Yellow,
        FString::Printf(TEXT("Distance: %f"), TimeElapsed));
348 
349         return TimeElapsed;
350     }
351 
352     float AKeepMowingALawnCharacter::GetAbsoluteTime()
353     {
354         return AbsoluteTime;
355     }
356 
357 
358     float AKeepMowingALawnCharacter::GetTravelledDistance()
359     {
360         return DistanceTravelled;
361     }
362 
363     void AKeepMowingALawnCharacter::SetAbsoluteTime()
364     {
365         AbsoluteTime = TimeElapsed;
366         return;
367     }
```