```
1   //KeepMowingALawnCharacter.h
2
3   // Copyright 1998-2018 Epic Games, Inc. All Rights Reserved.
4
5   #pragma once
6
7   #include "CoreMinimal.h"
8   #include "GameFramework/Character.h"
9   #include "KeepMowingALawnCharacter.generated.h"
10
11  UCLASS(config=Game)
12  class AKeepMowingALawnCharacter : public ACharacter
13  {
14      GENERATED_BODY()
15
16      /** FirstPerson camera */
17      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
        (AllowPrivateAccess = "true"))
18      class UCameraComponent* FollowCamera;
19
20      /** Top Down camera */
21      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
        (AllowPrivateAccess = "true"))
22          class UCameraComponent* SkyViewCamera;
23
24      /** tracks the position of the right tire **/
25      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
        (AllowPrivateAccess = "true"))
26          class UBoxComponent* RightTirePos;
27
28      /** tracks the position of the left tire **/
29      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
        (AllowPrivateAccess = "true"))
30          class UBoxComponent* LeftTirePos;
31
32      /** Allows for a sound to be placed on the mower **/
33      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
        (AllowPrivateAccess = "true"))
34          class UAudioComponent* MowerSounds;
35
36
37
38  public:
39      AKeepMowingALawnCharacter();
40
41      /** Base turn rate, in deg/sec. Other scaling may affect final turn rate. */
42      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category=Camera)
43      float BaseTurnRate;
44
45      /** Base look up/down rate, in deg/sec. Other scaling may affect final rate. */
46      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category=Camera)
47      float BaseLookUpRate;
48
49      /** Designed to track how much pressure is put on the left trigger **/
50      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera)
51          float LookLeftValue;
52
53      /** Designed to track how much pressure is put on the right trigger**/
54      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera)
55          float LookRightValue;
56
57      /** Global Variable for the value of the left joystick when pressed **/
58      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = MotionPhysics)
59          float LeftJoystickValue;
60
61      /** Global variable for the value of the right joystick when pressed **/
62      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = MotionPhysics)
63          float RightJoystickValue;
64
```

```
65      /** A value editable by the shift(up/down) functions for speed **/
66      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = MotionPhysics)
67          float SpeedMultiplier;
68
69      /** A value for the time ticked in seconds **/
70      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Time)
71          float TimeElapsed;
72
73      /** Gets the time at a single tick without frequently updating**/
74      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Time)
75          float AbsoluteTime;
76
77      /** The distance the mower travelled **/
78      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Distance)
79          float DistanceTravelled;
80
81      /** Due to my functions, I need to get magnitude in the middle of a function.
82      * To allow reusability of the function, I have this variable to get the magnitude
        inbetween function calls as need be
83      **/
84      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Distance)
85          float PotentialDistanceTravelled;
86
87      virtual void Tick(float Delta) override;
88
89      /** Below are functions for the HUD
90      * They allow Blueprints to access data from the C++ mower
91      * The HUD is a widget with blueprint functions.
92      **/
93
94      /** Get the time of the current Tick in seconds**/
95      UFUNCTION(BlueprintCallable, Category = "Time")
96          float GetElapsedTime();
97
98      /** Gets the time at a specific interval**/
99      UFUNCTION(BlueprintCallable, Category = "Time")
100         float GetAbsoluteTime();
101
102     UFUNCTION(BlueprintCallable, Category = "Distance")
103         float GetTravelledDistance();
104
105     /** Sets the time at a specific interval
106     * This is used to keep from constantly changing the time of the HUD during the
        endgame HUD
107     * Note: True time still runs in the background while parked in the garage
108     **/
109     UFUNCTION(BlueprintCallable, Category = "Time")
110         void SetAbsoluteTime();
111
112     UFUNCTION(BlueprintCallable, Category = "MowerSounds")
113         void StopMowerSound();
114
115     UFUNCTION(BlueprintCallable, Category = "MowerSpeed")
116         float GetGearStat(); //Returns SpeedMultiplier
117
118     /** This function calculates change of coordinates followed by rotation about an
        angle, and then changes the coordiantes back appropriately **/
119     FVector CalculateRotation(FVector CurrentWorldLoc, FVector PivotPoint, float
        AngleofChangeDegree);
120
121     /** Calculates the new position for the lawn mower at a single instance **/
122     FVector CalculatePosition(float rightangle, float leftangle);
123
124     /**
125     * Calculates the position of a point between two vectors about a given world
        coordinant (originpoint)
126     * multiplier is for increasing the magnitude via scalar multiplication
127     * Note: Uses the (LocationTwo) variable for the magnitude of the new point
128     **/
```

```
129         FVector LocationCombiner(FVector LocationOne, FVector LocationTwo, FVector
            OriginPoint, float multiplier);
130
131     protected:
132
133         virtual void BeginPlay() override;
134
135         /** Resets HMD orientation in VR. */
136         void OnResetVR();
137
138         /** Called for forwards/backward input **/
139         void MoveLeft(float Value);
140
141         /** Called for side to side input **/
142         void MoveRight(float Value);
143
144         /** Changes SpeedMultiplier by +1 **/
145         void ShiftUp();
146
147         /** Changes SpeedMultiplier by -1 **/
148         void ShiftDown();
149
150         /** used to enable/disable mower sounds **/
151         void ChangeMowerSoundState();
152
153         /** allows the camera to pan right while the mower is stopped **/
154         void LookRight(float value);
155
156         /** allows the camera to pan left while the mower is stopped **/
157         void LookLeft(float value);
158
159         /** Switches between the two camera components**/
160         void ChangeCameraView();
161
162         /** Increases a time variable **/
163         void IncreaseTime();
164
165         FTimerHandle TimeMangementHandler;
166
167
168     protected:
169         // APawn interface
170         virtual void SetupPlayerInputComponent(class UInputComponent* PlayerInputComponent)
            override;
171         // End of APawn interface
172
173     public:
174
175         /** Returns FollowCamera subobject **/
176         FORCEINLINE class UCameraComponent* GetFollowCamera() const { return FollowCamera; }
177
178     };
179
180
```