

```

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 =
18         (AllowPrivateAccess = "true"))
19     class UCameraComponent* FollowCamera;
20
21     /** Top Down camera */
22     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
23         (AllowPrivateAccess = "true"))
24     class UCameraComponent* SkyViewCamera;
25
26     /** tracks the position of the right tire */
27     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
28         (AllowPrivateAccess = "true"))
29     class UBoxComponent* RightTirePos;
30
31     /** tracks the position of the left tire */
32     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
33         (AllowPrivateAccess = "true"))
34     class UBoxComponent* LeftTirePos;
35
36     /** Allows for a sound to be placed on the mower */
37     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera, meta =
38         (AllowPrivateAccess = "true"))
39     class UAudioComponent* MowerSounds;
40
41 public:
42     AKeepMowingALawnCharacter();
43
44     /** Base turn rate, in deg/sec. Other scaling may affect final turn rate. */
45     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category=Camera)
46     float BaseTurnRate;
47
48     /** Base look up/down rate, in deg/sec. Other scaling may affect final rate. */
49     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category=Camera)
50     float BaseLookUpRate;
51
52     /** Designed to track how much pressure is put on the left trigger */
53     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera)
54     float LookLeftValue;
55
56     /** Designed to track how much pressure is put on the right trigger*/
57     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera)
58     float LookRightValue;
59
60     /** Global Variable for the value of the left joystick when pressed */
61     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = MotionPhysics)
62     float LeftJoystickValue;
63
64     /** Global variable for the value of the right joystick when pressed */
65     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = MotionPhysics)
66     float RightJoystickValue;

```

```

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
83     inbetween function calls as need be
84     */
85     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Distance)
86         float PotentialDistanceTravelled;
87
88     virtual void Tick(float Delta) override;
89
90     /** Below are functions for the HUD
91     * They allow Blueprints to access data from the C++ mower
92     * The HUD is a widget with blueprint functions.
93     */
94
95     /** Get the time of the current Tick in seconds*/
96     UFUNCTION(BlueprintCallable, Category = "Time")
97         float GetElapsedTime();
98
99     /** Gets the time at a specific interval*/
100    UFUNCTION(BlueprintCallable, Category = "Time")
101        float GetAbsoluteTime();
102
103    UFUNCTION(BlueprintCallable, Category = "Distance")
104        float GetTravelledDistance();
105
106    /** Sets the time at a specific interval
107    * This is used to keep from constantly changing the time of the HUD during the
108    endgame HUD
109    * Note: True time still runs in the background while parked in the garage
110    */
111    UFUNCTION(BlueprintCallable, Category = "Time")
112        void SetAbsoluteTime();
113
114    UFUNCTION(BlueprintCallable, Category = "MowerSounds")
115        void StopMowerSound();
116
117    UFUNCTION(BlueprintCallable, Category = "MowerSpeed")
118        float GetGearStat(); //Returns SpeedMultiplier
119
120    /** This function calculates change of coordinates followed by rotation about an
121    angle, and then changes the coordinantes back appropriately */
122    FVector CalculateRotation(FVector CurrentWorldLoc, FVector PivotPoint, float
123    AngleofChangeDegree);
124
125    /** Calculates the new position for the lawn mower at a single instance */
126    FVector CalculatePosition(float rightangle, float leftangle);
127
128    /**
129    * Calculates the position of a point between two vectors about a given world
130    coordinant (originpoint)
131    * multiplier is for increasing the magnitude via scalar multiplication
132    * Note: Uses the (LocationTwo) variable for the magnitude of the new point
133    */

```

```

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

```