Realistic Head Controller V1.0

Scripting Documentation

Support: inanevin@gmail.com & https://www.inanevin.com/realistic-head-controller

Video Tutorial of This Documentation:

Intro

Welcome, thanks for purchasing Realistic Head Controller! This documentation covers the programming reference of RHC. If you haven't already, it is advised to check the General Documentation first! Now, assuming you know how the general flow of the system works, let's start!

Usage

After you've added the RHC prefab to your player(mentioned in General Documentation), andb tweaked the components to your needs, now you can start calling the methods from your player controller in order to run the system. Before explaining, need to mention that example calls are included in the Canvas object in ExampleScene, inside the Scenes folder.

State Calls

First thing to mention is the methods in *RHC_EventManager*. *CameraStatePositioner* and *BobController* depend on the player states that are included in *RHC_EventManager*, they listen to the events that are called when states changed. So it means that in order to achieve State Positioning & Head Bobbing, you need to change the states defined in *RHC_EventManager*. As default, the states are like this:

- **PlayerStanceStates:** Standing, Crouching, Crawling, OnJump
- **PlayerMovementStates:** Idling, Walking, Running

So the system checks for the current state of the player from *RHC_EventManager*, and acts accordingly. These 2 states are defined as:

```
public enum MovementStates { Idling, Walking, Running };
public enum StanceStates { Standing, Crouching, Crawling, OnAir };
public static MovementStates en_CurrentMovementState;
public static StanceStates en_CurrentStanceState;
```

As you can see, the other components would check for the current values of `en_CurrentMovementState` and `en_CurrentStanceState` variables.

In order to get the system to work, you have to call the necessary methods to set the player states in *RHC_EventManager*. There are 2 static methods to do this:

```
public static void PSVO_ChangeMovementState(MovementState changeTo)
public static void PSVO_ChangeStanceState(StanceState changeTo)
```

PSVO_ChangeMovementState(MovementState changeTo)

This one changes the current registered movement state, it takes a `MovementState` enum variable as input, and changes the `en_CurrentMovementState` variable to that input. Then it calls the associated delegate event, to notify the other components of the change.

Example Call:

```
RHC_EventManager.PSVO_ChangeMovementState(RHC_EventManager.MovementState.Walking);
```

PSVO_ChangeStanceState(StanceState changeTo)

This one changes the current registered stance state, it takes a StanceState enum variable as input, and changes the `en_CurrentStanceState` variable to that input. Then it calls the associated delegate event, to notify the other components of the change.

Example Call:

```
RHC_EventManager.PSVO_ChangeStanceState(RHC_EventManager.StanceState.Crouching);
```

So as you'd guess, when your player jumps, call the `PSVO_ChangeStanceState` and give `RHC_EventManager.StanceState.OnAir` as parameter, when your player starts to run, call the `PSVO_ChangeMovementState` and give `RHC_EventManager.MovementState.Running` as parameter, and so on. You only need to call the static methods once on each state change, and the components will do the rest!

Camera Shake Calls

Camera shakes are controlled through RHCShakeController object. There are 2 types of shakes:

- **Hit Shake:** Can be referred as camera recoil, you would use this when you want to simulate player being hit, or even player firing. It recoils the camera.

- **Generic Shakes(Explosion Shakes):** These shakes are called based on distance, and they are manipulated by time. If you want to simulate Explosion Shakes, Earthquakes etc. Use these.

Don't forget to customize the shake lists in *RHCShakeController* object's inspector. In order to use the shakes, you have 3 methods to call in *RHC_EventManager*.

```
public static void PSVO_CallGenericShake(Vector3 sourcePosition)
public static void PSVO_CallCustomShake(Vector3 posShakeAmount, Vector3 posShakeSpeed,
 Vector3 rotShakeAmount, Vector3 rotShakeSpeed, float shakeTime)
public static void PSVO_CallRecoil()
```

PSVO_CallGenericShake(Vector3 sourcePosition)

It takes a Vector3 as input, referred as the source position of the event that causes the shake, e.g Explosion. You would call this, and the *RHCShakeController* object will determine which shake to process according to the distance from sourcePosition. Please keep in mind that you need to customize the *RHCShakeController* object's inspector to fit your needs. More details about customization is in General Documentation.
Example Call:

```
RHC_EventManager.PSVO_CallGenericShake(player.transform.position + new
Vector3(5,10,5));
```

PSVO_CallCustomShake(Vector3 posShakeAmount, Vector3 posShakeSpeed, Vector3 rotShakeAmount, Vector3 rotShakeSpeed, float shakeTime)

If you want to shake to camera, without being dependent on the generic shake list that you've adjusted on the inspector, you can use this function. Basically, use it whenever you want to shake the camera according to any input.

Variable	Description
posShakeAmount	Positional shake amount.
posShakeSpeed	Positional shake speed.
rotShakeAmount	Rotation shake amount.
rotShakeSpeed	Rotational shake speed.
shakeTime	Total time that the camera would shake.

Example Call:

```
RHC_EventManager.PSVO_CallCustomShake(new Vector3(0.2f, 0.5f, 0.0f), 3.0f, new Vector3
(0.0f, 0.0f, 10.0f), 5.0f, 2.5f);
```

PSVO_CallRecoil()

Call this whenever you want to recoil the camera. Settings are tweakable through the inspector of *RHCShakeController* object.

Example Call:

```
RHC_EventManager.PSVO_CallRecoil();
```

Bobbing Calls

Additional to editing headbobs over the inspector, you can also edit a particular headbobbing setting's values on runtime!

In order to do this, we have two methods in *RHC_EventManager*.

```
public static void PSVO_ChangeBobPreset(RHC_BobController.BobSettings settingToChange,
 RHC_BobController.Preset presetToChange)
public static void PSVO_ChangeBobCustom(RHC_BobController.BobSettings settingToChange,
 RHC_BobController.BobStyle bobStyle, Vector3 posAmount, Vector3 posSpeed, Vector3 rot
Amount, Vector3 rotSpeed, float posSmooth, float rotSmooth)
```

PSVO_ChangeBobPreset(RHC_BobController.BobSettings settingToChange, RHC_BobController.Preset presetToChange)

This method sets a particular bob setting and sets it's values according to a particular preset. It takes `RHC_BobController.BobSetting` type variable and `RHC_BobController.Preset` type variable as parameters.

Variable	Description
settingToChange	Which bob setting to edit. (Walk, Run, CrouchWalk, CrouchRun, CrawlSlow, CrawlFast)
presetToChange	Which preset shall be set to the particular bob setting.

PSVO_ChangeBobCustom(RHC_BobController.BobSettings settingToChange, RHC_BobController.BobStyle bobStyle, Vector3 posAmount, Vector3 posSpeed, Vector3 rotAmount, Vector3 rotSpeed, float posSmooth, float rotSmooth)

This method sets a particular bob setting and sets it's values according to the given inputs.

Variable	Description
settingToChange	Which bob setting to edit? (Walk, Run, CrouchWalk, CrouchRun, CrawlSlow, CrawlFast)
bobStyle	Which bob style should be applied? (Positional, Rotational, Both)
posAmount	Positional amount of bobbing.
posSpeed	How fast will the positional amount be applied?
rotAmount	Rotational amount of bobbing.
rotSpeed	How fast will the rotational amount be applied?
posSmooth	Smoothing value to be applied on positional bobbing.
rotSmooth	Smoothing value to be applied on rotational bobbing.

Extending The System

If the default player states are not enough for your needs, let's say, you have a player controller who can obtain the state "sliding", you can easily add new states for *RHCStatePositioner* & *RHCBobController* objects to check. Please check the Youtube page given above, and the tutorial playlist in order to see how to extend the system!