Realistic Eye Movements

Version 1.1

Thanks for purchasing Realistic Eye Movements! I hope this asset helps you bring your characters to life. If you have any questions or suggestions, please drop me at line at tamulur@yahoo.com!

For discussions of this asset, visit the Unity forum at http://forum.unity3d.com/threads/released-realistic-eye-movements.297610/.

The webpage for this asset is http://tore-knabe.com/unity-asset-realistic-eye-movements

A quickstart video showing a simple use case is here: https://www.youtube.com/watch?v=BP7kLye_XxU

What this Asset Does

Realistic Eye Movements can control your character's eyes, head and eyelid movements to make them look around, at the player, or at objects, in a lifelike way. You can use it with characters that have eyes rigged to a Mecanim humanoid bone rig or characters that just have two separate eye gameobjects. You can assign points of interest in the character's environment for the character to look at, or let them just look around idly, or let them look at the player when the player comes into view or keeps staring at them.

The animations use data from published research papers. For more information, see my blog entry http://tore-knabe.com/experiments-with-head-animation.

This asset only provides scripts to control animation. For realistic looking eye meshes and shaders, I recommend Scruvystorm Studios' RealEyes asset:
https://www.assetstore.unity3d.com/en/#!/content/13847

How to Use

There are two main scripts in the folder RealisticEyeMovements/Scripts:

`LookTargetController.cs`

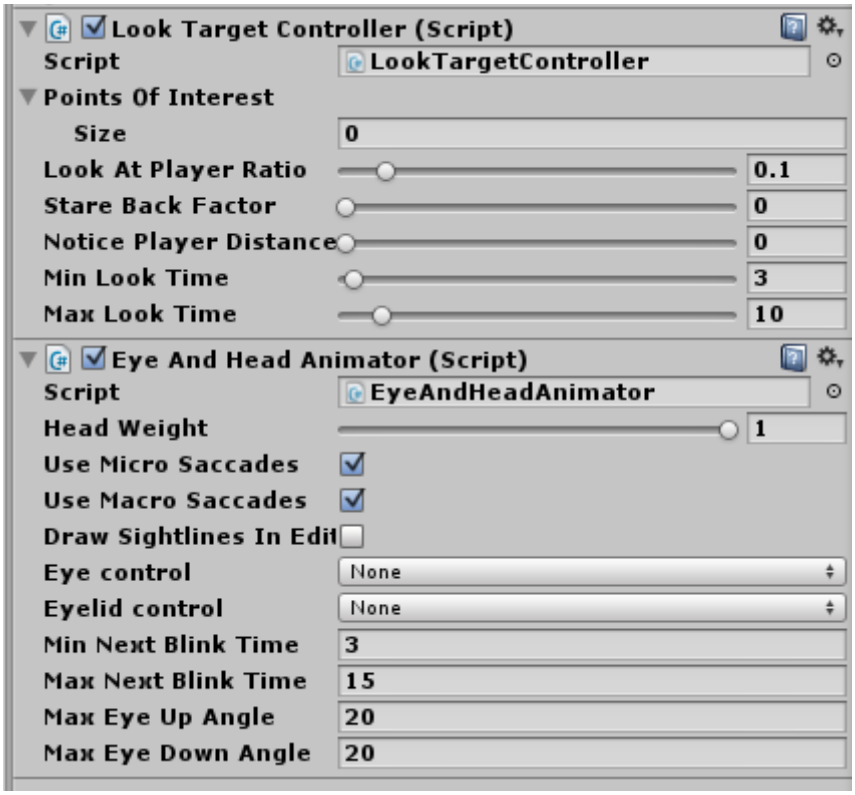
Chooses what to look at and when to switch look targets.

`EyeAndHeadAnimator.cs`

Controls the animation of eyes, head and eyelids for a given look target.

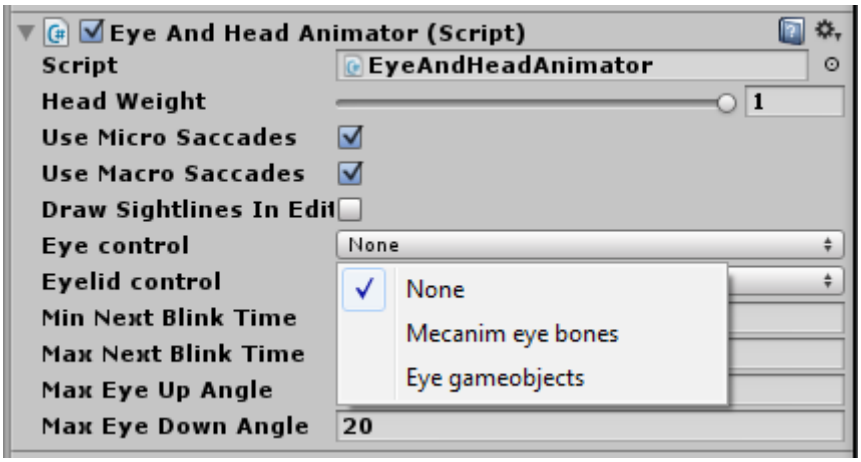
If you want more control over where and when to look you can modify LookTargetController.cs or replace it with your own script that calls functions from EyeAndHeadAnimator.cs. You probably don't neet to modify EyeAndHeadAnimator.cs.

Drag the two scripts onto your character game object in the scene hierarchy. The new components should look like that:



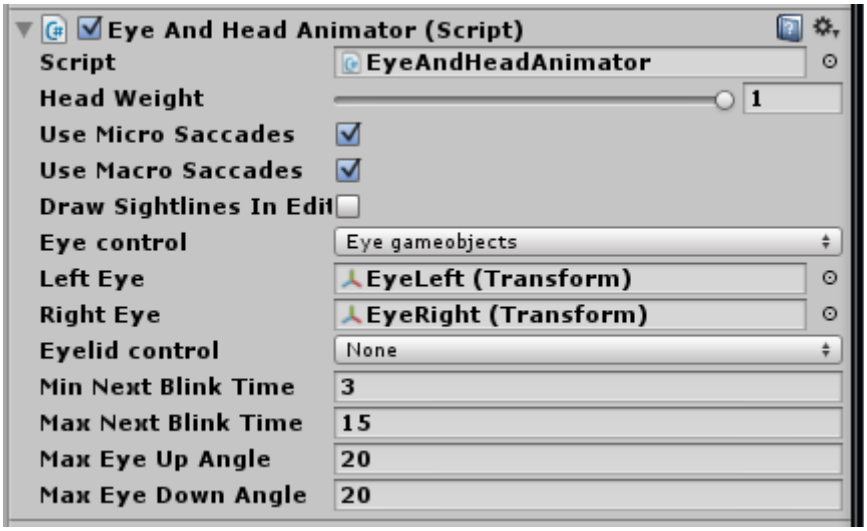
Eyes

To tell the scripts how to control the eyes, select either *Mecanim eye bones* or *Eye gameobjects* from the box **Eye control**.



If you choose *Mecanim eye bones*, your character must have a Mecanim humanoid rig and an Animator component. If your Mecanim rig has the eye bones assigned correctly, the script will find and use them to animate the eyes. Choose this option if your character's eyes are controlled by Mecanim eye bones.

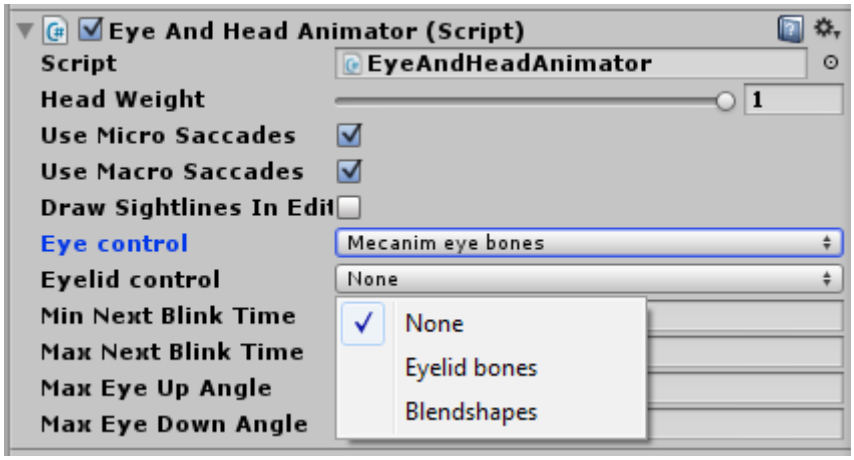
If you choose *Eye gameobjects*, you can select game objects in your character's object hierarchy to be controlled by the script. Find them in your character's hierarchy and drag each into its corresponding slot in the component:



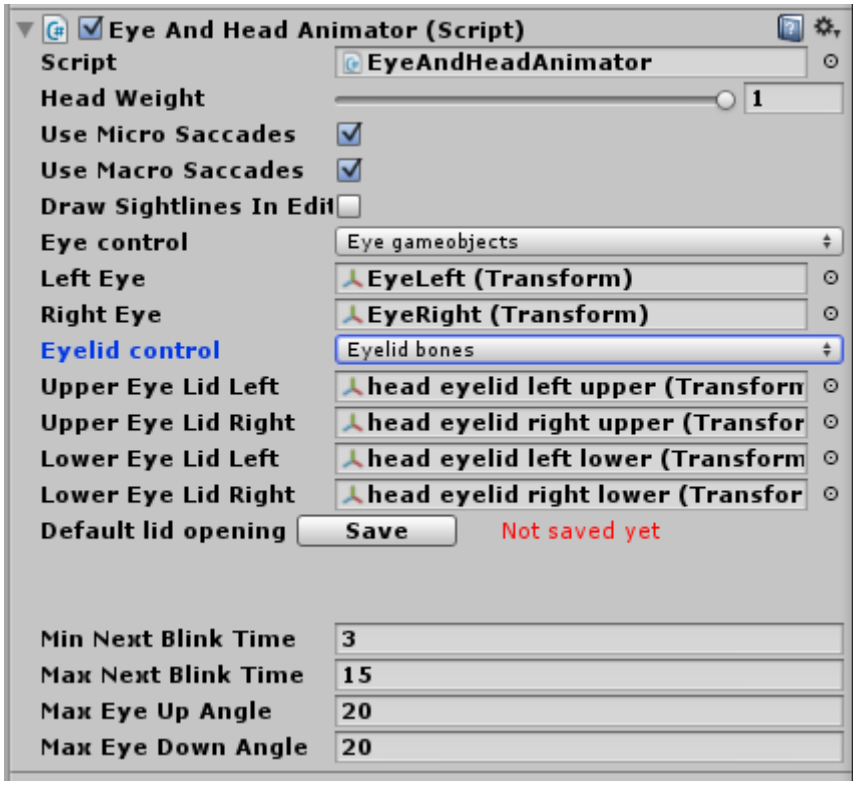
The script assumes the eyes look straight ahead at the start. If they don't, rotate them (outside of Play mode) to look straight ahead relative to the character.

Eyelids

If your character has eyelids that can be controlled by bones or blendshapes, the EyeAndHeadAnimator component can control them to add realism. If not, leave the box **Eyelid control** set to *None*. If your character does have eyelids, set it to either *Eyelid bones* or *Blendshapes*.

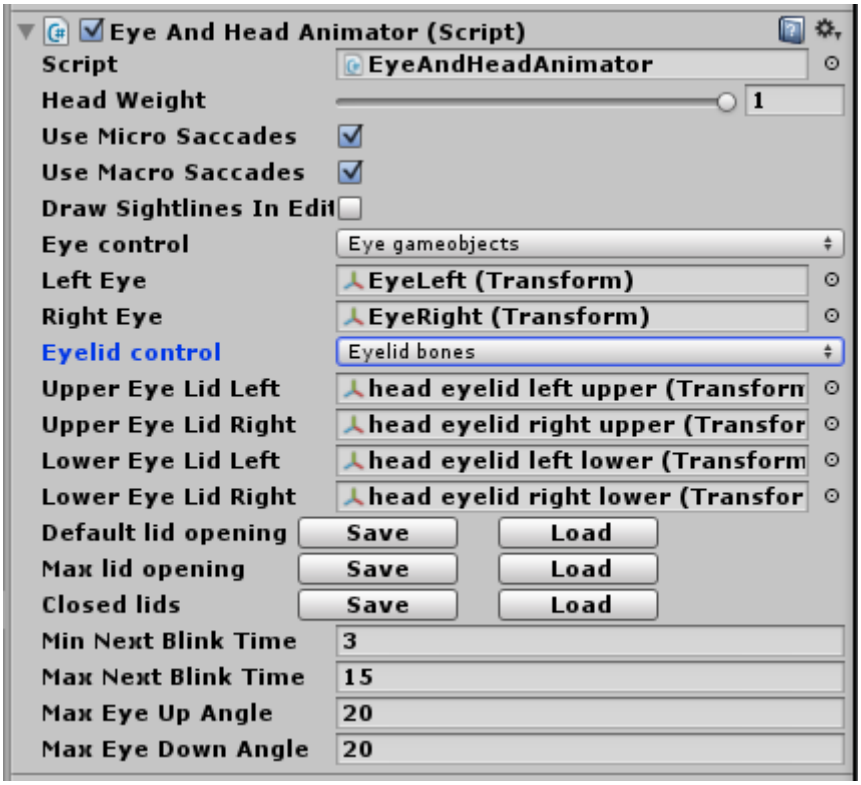


If you choose *Eyelid bones*, find the eyelid bones in your character's hierarchy and drag them into the corresponding slots:



The slots for the lower eyelids can be left empty if your character doesn't have lower eyelids to animate.

After filling the slots, click the **Save** button to save the eyelids' current position as the default open position.



The script also needs to know about how far the eyelids can open and what their position is when the eyes are closed. First, rotate the eyelid bones such that the eyes are maximally open (so rotate the upper eyelid up as far as possible such that it still looks natural and the lower eyelid down) and press **Save** next to *Max lid opening*.

To save the position for closed eyes, it is usually easiest to first rotate both upper eyelids down until only part of the iris is visible on both sides:

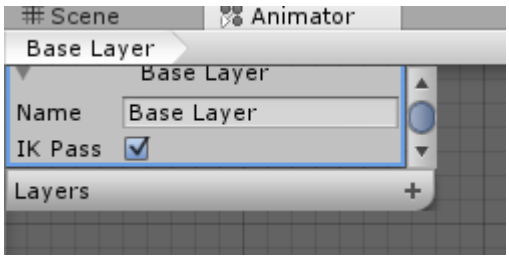
Then rotate the lower eyelids up to close the eyes:

Then press **Save** next to *Closed lids*.

If you chose *Blendshapes* for eyelid control, you have to Save the default open position as well by clicking the **Save** button. After that, close your character's eyes by setting the slider values of the corresponding blendshapes (blendshapes can be found in the SkinnedMeshRenderer components) and then press **Save** next to the *Closed lids* line. The component will determine which blendshapes have been changed and by how much to know how to manipulate eyelids via blendshapes.

This is all the component needs to know to animate the lids. If you want you can click **Load** next to *Default lid opening* to open your character's eyes again.

There are two more controls in the EyeAndHeadAnimator component. The **Head weight** slider value determines how much the script controls the character's head movement when looking at targets. For head animation to work, your character needs to have a Mecanim human rig and the Animator Controller must have „IK Pass“ checked:



Note: Mecanim head IK only works in Unity Pro.

The **Use Micro Saccades** checkbox determines whether the eyes do those little darts from time to time that human eyes usually do and that add to how lifelike the eyes seem.

The **Use Macro Saccades** checkbox determines whether the eyes do larger darts from time to time (similar to micro saccades, but less frequent and larger angles). Macro saccades are not used when the character is looking at the player's face or when you call the **LookAtPoiDirectly** function.

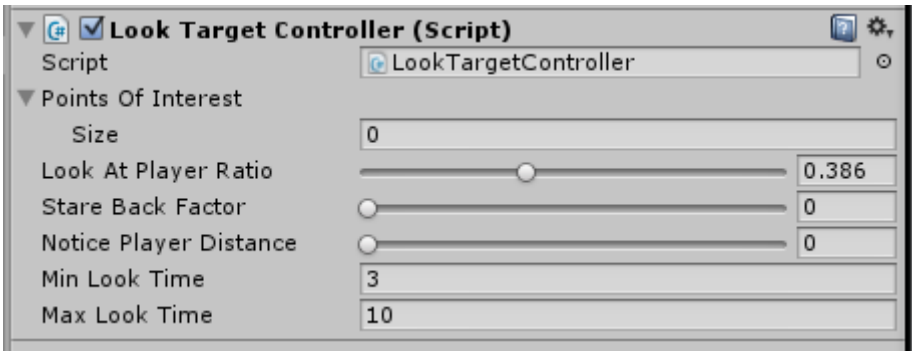
The default is to use both micro and macro saccades for most lifelike animation.

Checking **Draw Sightlines In Editor** lets you see where the eyes are looking exactly during Play mode in the editor window.

You can control the frequency of blinking by setting **Min Next Blink Time** and **Max Next Blink Time**. After a blink, the time until the next blink is a random number of seconds betwen min blink time and max blink time.

Max Eye Up Angle and **Max Eye Down Angle** determine how far the eyes can move vertically relative to the head.

Look Targets



The **LookTargetController** component lets you control where your character is looking. By default, the character looks around idly in random directions (close to straight ahead), and sometimes at the player if the player is in view. If you have specific objects in the character's environment that you want the character to choose as look targets, drag them into the array **Points of Interest**. The character will look at a random object from that list for some time, then choose another object from the list to look at.

If the player is in the character's view, the slider value of **Loot at Player Ratio** determines how often the character chooses the player as next target to look at: for example, a slider value of 0.1 means the character chooses a random direction or an object from the Points of Interest list 90% of the time, and the player as look target 10% of the time.

The slider **Stare Back Factor** determines how quickly the character looks back at the player if the characters sees the player keep staring at him or her (so how sensitive the character is to being stared at).

The slider **Notice Player Distance** determines what distance the player has to come nearer than for the character to start looking at him. A value of 0 means the player coming closer has no effect. A value of 2 means if the character hasn't noticed the player before and the player comes into view and is closer than 2 units, the character starts looking at the player.

The time the character looks at a target before choosing another target is a random number of seconds between **Min Look Time** and **Max Look Time**.

Oculus Rift Virtual Reality Headset

By default, the script uses Camera.main to determine the player's position. If your scene is set up for Oculus VR, the script uses the Oculus camera prefabs instead to find out where the player's left and right eyes are. When characters look at the player, they will cycle randomly among the left and right eye and the mouth as look target (the so called „social triangle“ people use when looking at someone's face). You don't need to change anything for the script to work in a VR scene.

Script API

You can call these functions on the LookTargetController component for more control:

LookTargetController.cs

void **Blink**(bool isShortBlink)

Makes the character blink.

void **ClearLookTarget**()

Clears the current look target and makes the character look straight ahead.

bool **IsPlayerInView**()

Returns whether the character can see the player (only checks viewing angles, doesn't check for visual obstacles in between character and player)

void **LookAtPlayer**()

Looks at the player.

void **LookAroundIdly**()

Starts looking around in random directions (close to straight ahead) or at points of interest if the list Points of Interest has objects.

void **LookAtPoiDirectly**(Transform targetTransform)

Looks at a specific transform. Keeps following the transform with the eyes if the transform moves.

void **LookAtPoiDirectly**(Vector3 targetPoint)

Looks at a specific point.

If you need more control over the animation than the components can provide in their current state, please let me know at tamulur@yahoo.com.

Good luck with your projects!

Tore Knabe

Changelog

1.1

- new blink control: minBlinkTime and maxBlinkTime
- checkbox to disable macro saccades
- checkbox to show sightlines in editor
- settings for maximum up and down eye angles
- adjusted head and eye tracking after they jumped to a new target