Basic Camera Controller (Script)

Script	BasicCameraController
Camera	CAMERA (Camera)
Settings	
Desired Distance	5
Min Zoom Distance	0
Components	
Target	
Target	TARGET (Transform)
Rotation	
Zoom	
Minimum Distance	0
Snap In	☐
Snap Out	☐
Uniform Speed	☐
Zoom In Speed	0.6
Zoom Out Speed	0.6
Enable Add Percentage On Scroll	✓
Percentage To Add	
View Collision	
Enabled	✓
Line Of Sight Layers	Default
Ignore These Game Objects	
Size	0
Sampling Points	

Third Person Camera(s) documenation of classes and functions.

Date: 8/3/2016 7:26:12 PM

Table of Contents

Introduction

How to use:

The basics of using Third Person Camera(s) is easy. All you need to make it run is the ThirdPersonCameras folder. That's it.

From there, simply drag one of the pre-created CameraController scripts onto any object. Set the Camera field and the Target field, and you're good to go.Try playing around with the settings to get the camera just the way you want it.

The prescripted camera controller you have to choose from are:
1) BasicCameraController.cs
2) HoldiPositionCameraController.cs
3) LockTargetCameraController.cs
4) MultiCameraController.cs
5) FollowCameraController.cs
6) SideScrollingCameraController.cs

I recommend starting with the BasicCameraController as it is closest to what most people would need.

Check out my YouTube channel at http://www.youtube.com/c/SaveMeOniiChan to see videos of works I've done.

Also, check out https://www.youtube.com/watch?v=4nkArpeCIxE for a video detailing every feature's usage in depth.

AdvancedUtilities

See more Advanced Utilities @
https://www.assetstore.unity3d.com/en/#!/search/page=1/sortby=popularity/query=publisher:18832

This class' purpose is to represent a Transform and to enable the use of its functions without being attached to an object. Transforms must be attached to a Game Object and cannot be new'd up, and they also affect the state of the world in Unity when altered.

This class will act as a detached Transform allowing you to use the normal functions of a transform without affecting the state of the world, or needing to use a hidden transform to get a similar effect.

public class VirtualTransform

Properties

The position of the transform stored as a Vector3 in world space.
Vector3 Position
public GET
public SET

The rotation of the transform stored as a Quaternion in world space.
Quaternion Rotation
public GET
public SET

The scale of the transform relative to the parent.
Vector3 LocalScale
public GET
public SET

The rotation as Euler angles in degrees.
Vector3 EulerAngles
public GET
public SET

The forward vector of the transform.
The blue axis of the transform in world space.
Vector3 Forward
public GET

The right vector of the transform.
The red axis of the transform in world space.
Vector3 Right
public GET

The upwards vector of the transform.
The green axis of the transform in world space.
Vector3 Up
public GET

Constructors

public VirtualTransform

No parameters.
Constructs a VirtualTransform with the position at the world origin, the rotation as the quaternion identity, and the local scale as 1.

public VirtualTransform

(UnityEngine.Vector3 position)
Constructs a VirtualTransform with the position at given position, the rotation as the quaternion identity, and the local scale as 1.

public VirtualTransform

(UnityEngine.Quaternion rotation)
Constructs a VirtualTransform with the position at the world origin, the rotation as the given quaternion, and the local scale as 1.

public VirtualTransform

(UnityEngine.Vector3 position, UnityEngine.Quaternion rotation)
Constructs a VirtualTransform with the position at the given position, the rotation as the given quaternion, and the local scale as 1.

public VirtualTransform

(UnityEngine.Vector3 position, UnityEngine.Quaternion rotation, UnityEngine.Vector3 localScale)
Constructs a VirtualTransform with the position at the given position, the rotation as the given quaternion, and the local scale as the given local scale.

public VirtualTransform

(UnityEngine.Transform transform)
Constructs a VirtualTransform setting the position, rotation, and local scale to the values of the given Transform.
transform: Transform whose properties you want to copy.

public VirtualTransform

(UnityEngine.GameObject gameObject)
Constructs a VirtualTransform setting the position, rotation, and local scale to the values of the given GameObject's transform.
gameObject: GameObject whose properties you want to copy.

public VirtualTransform

(UnityEngine.MonoBehaviour monoBehaviour)
Constructs a VirtualTransform setting the position, rotation, and local scale to the values of the given MonoBehaviour's transform.
monoBehaviour: MonoBehaviour whose properties you want to copy.

public VirtualTransform

(UnityEngine.Camera camera)
Constructs a VirtualTransform setting the position, rotation, and local scale to the values of the given Camera's transform.
camera: Camera whose properties you want to copy.

public VirtualTransform

(AdvancedUtilities.VirtualTransform virtualTransform)
Constructs a VirtualTransform setting the position, rotation, and local scale to the values of the given VirtualTransform.
virtualTransform: VirtualTransform whose properties you want to copy.

Methods

Applies the position, rotation, and localScale properties of the Transform to this VirtualTransform.

transform: The Transform whose properties you want to apply to this VirtualTransform.
public Void Apply(UnityEngine.Transform transform)

Applies the position, rotation, and localScale properties of the VirtualTransform to this VirtualTransform.

virtualTransform: The VirtualTransform whose properties you want to apply to this VirtualTransform.
public Void Apply(AdvancedUtilities.VirtualTransform virtualTransform)

Applies the position, rotation, and localScale properties of the GameObject's transform to this VirtualTransform.

gameObject: The GameObject whose transform's properties you want to apply to this VirtualTransform.
public Void Apply(UnityEngine.GameObject gameObject)

Applies the position, rotation, and localScale properties of the MonoBehaviour's transform to this VirtualTransform.

monoBehaviour: The MonoBehaviour whose transform's properties you want to apply to this VirtualTransform.
public Void Apply(UnityEngine.MonoBehaviour monoBehaviour)

Applies the position, rotation, and localScale properties of the Camera's transform to this VirtualTransform.

camera: The Camera whose transform's properties you want to apply to this VirtualTransform.
public Void Apply(UnityEngine.Camera camera)

Applies the properties of this VirtualTransform to the transform of the given Game Object. Scale isn't applied unless set to true.

gameObject: GameObject whose transform you want to apply this VirtualTransform's properties to.
applyScale: If you also want to apply scale.
public Void ApplyTo(UnityEngine.GameObject gameObject, System.Boolean applyScale)

Applies the properties of this VirtualTransform to the transform of the given MonoBehaviour. Scale isn't applied unless set to true.

monoBehaviour: MonoBehaviour whose transform you want to apply this VirtualTransform's properties to.
applyScale: If you also want to apply scale.
public Void ApplyTo(UnityEngine.MonoBehaviour monoBehaviour, System.Boolean applyScale)

Applies the properties of this VirtualTransform to the transform of the given Camera. Scale isn't applied unless set to true.

camera: Camera whose transform you want to apply this VirtualTransform's properties to.
applyScale: If you also want to apply scale.
public Void ApplyTo(UnityEngine.Camera camera, System.Boolean applyScale)

Applies the properties of this VirtualTransform to the transform of the given Transform. Scale isn't applied unless set to true.

transform: The transform you want to apply this VirtualTransform's properties to.
applyScale: If you also want to apply scale.
public Void ApplyTo(UnityEngine.Transform transform, System.Boolean applyScale)

Applies the properties of this VirtualTransform to the given VirtualTransform. Scale isn't applied unless set to true.

virtualTransform: VirtualTransform you want to apply this VirtualTransform's properties to.
applyScale: If you also want to apply scale.
public Void ApplyTo(AdvancedUtilities.VirtualTransform virtualTransform, System.Boolean applyScale)

Rotates around the position by the given amount of degrees on the Vector3.Up axis.

position: Position to rotate around.
angle: Degrees.
public Void RotateAround(UnityEngine.Vector3 position, System.Single angle)

Rotates around the position by the given amount of degrees on the given axis.

position: Position to rotate around.
angle: Degrees.
axis: Axis to rotate on.
public Void RotateAround(UnityEngine.Vector3 position, System.Single angle, UnityEngine.Vector3 axis)

Rotates around the position of the Transform by the given amount of degrees on the Vector3.Up axis.

transform: Transform whose position to rotate around.
angle: Degrees.
public Void RotateAround(UnityEngine.Transform transform, System.Single angle)

Rotates around the position of the Transform by the given amount of degrees on the given axis.

transform: Transform whose position to rotate around.
angle: Degrees.
axis: Axis to rotate on.
public Void RotateAround(UnityEngine.Transform transform, System.Single angle, UnityEngine.Vector3 axis)

Rotates around the position of the VirtualTransform by the given amount of degrees on the Vector3.Up axis.

virtualTransform: VirtualTransform whose position to rotate around.
angle: Degrees.

public Void RotateAround(AdvancedUtilities.VirtualTransform virtualTransform, System.Single angle)

Rotates around the position of the VirtualTransform by the given amount of degrees on the Vector3.Up axis.

virtualTransform: VirtualTransform whose position to rotate around.
angle: Degrees.
axis: Axis to rotate on.
public Void RotateAround(AdvancedUtilities.VirtualTransform virtualTransform, System.Single angle, UnityEngine.Vector3 axis)

Rotates around the position of the MonoBehaviour by the given amount of degrees on the Vector3.Up axis.

monoBehaviour: MonoBehaviour whose position to rotate around.
angle: Degrees.
public Void RotateAround(UnityEngine.MonoBehaviour monoBehaviour, System.Single angle)

Rotates around the position of the MonoBehaviour by the given amount of degrees on the given axis.

monoBehaviour: MonoBehaviour whose position to rotate around.
angle: Degrees.
axis: Axis to rotate on.
public Void RotateAround(UnityEngine.MonoBehaviour monoBehaviour, System.Single angle, UnityEngine.Vector3 axis)

Rotates around the position of the GameObject by the given amount of degrees on the Vector3.Up axis.

gameObject: GameObject whose position to rotate around.
angle: Degrees.
public Void RotateAround(UnityEngine.GameObject gameObject, System.Single angle)

Rotates around the position of the GameObject by the given amount of degrees on the given axis.

gameObject: GameObject whose position to rotate around.
angle: Degrees.
axis: Axis to rotate on.
public Void RotateAround(UnityEngine.GameObject gameObject, System.Single angle, UnityEngine.Vector3 axis)

Rotates around the position of the Camera by the given amount of degrees on the given axis.

camera: Camera whose position to rotate around.
angle: Degrees.
axis: Axis to rotate on.
public Void RotateAround(UnityEngine.Camera camera, System.Single angle, UnityEngine.Vector3 axis)

Rotates the VirtualTransform to look at the point. Uses Vector3.up as the worldUp vector.

position: Position in world space to look at.
public Void LookAt(UnityEngine.Vector3 position)

Rotates the VirtualTransform to look at the point.

position: Position in world space to look at.
worldUp: Upwards vector.
public Void LookAt(UnityEngine.Vector3 position, UnityEngine.Vector3 worldUp)

Rotates the VirtualTransform to look at the position of the given Transform. Uses Vector3.up as the worldUp vector.

transform: The Transform whose position in world space to look at.
public Void LookAt(UnityEngine.Transform transform)

Rotates the VirtualTransform to look at the position of the given Transform.

transform: The Transform whose position in world space to look at.
worldUp: Upwards vector.
public Void LookAt(UnityEngine.Transform transform, UnityEngine.Vector3 worldUp)

Rotates the VirtualTransform to look at the position of the given VirtualTransform. Uses Vector3.up as the worldUp vector.

virtualTransform: The VirtualTransform whose position in world space to look at.

public Void LookAt(AdvancedUtilities.VirtualTransform virtualTransform)

Rotates the VirtualTransform to look at the position of the given VirtualTransform.

virtualTransform: The VirtualTransform whose position in world space to look at.
worldUp: Upwards vector.
public Void LookAt(AdvancedUtilities.VirtualTransform virtualTransform, UnityEngine.Vector3 worldUp)

Rotates the VirtualTransform to look at the position of the game object's transform. Uses Vector3.up as the worldUp vector.

gameObject: The GameObject whose position in world space to look at.
public Void LookAt(UnityEngine.GameObject gameObject)

Rotates the VirtualTransform to look at the position of the game object's transform.

gameObject: The GameObject whose position in world space to look at.
worldUp: Upwards vector.
public Void LookAt(UnityEngine.GameObject gameObject, UnityEngine.Vector3 worldUp)

Rotates the VirtualTransform to look at the position of the monobehaviour's transform. Uses Vector3.up as the worldUp vector.

monoBehaviour: The MonoBehaviour whose position in world space to look at.
public Void LookAt(UnityEngine.MonoBehaviour monoBehaviour)

Rotates the VirtualTransform to look at the position of the Monobehaviour's transform.

monoBehaviour: The MonoBehaviour whose position in world space to look at.
worldUp: Upwards vector.
public Void LookAt(UnityEngine.MonoBehaviour monoBehaviour, UnityEngine.Vector3 worldUp)

Rotates the VirtualTransform to look at the position of the Camera's transform.

camera: The Camera whose position in world space to look at.
worldUp: Upwards vector.
public Void LookAt(UnityEngine.Camera camera, UnityEngine.Vector3 worldUp)

Translates the position of the transform by the given value in world space.

movement: Amount to move by.
public Void Translate(UnityEngine.Vector3 movement)

Translates the position of the transform by the given value in world space.

x: Amount to move by on the x axis.
y: Amount to move by on the y axis.
z: Amount to move by on the z axis.
public Void Translate(System.Single x, System.Single y, System.Single z)

Rotates the rotation of the transform by the given quaternion value in the world space.

rotation: Amount to rotate by.
public Void Rotate(UnityEngine.Quaternion rotation)

Rotates the rotation of the transform by the given quaternion value in the given space.

rotation: Amount to rotate by.
space: The space you want to rotate in.
public Void Rotate(UnityEngine.Quaternion rotation, UnityEngine.Space space)

Rotates the rotation of the transform by the Quaternion represented by the given euler vector. Rotates within the world space.

eulerRotation: Amount to rotate by.
public Void Rotate(UnityEngine.Vector3 eulerRotation)

Rotates the rotation of the transform by the Quaternion represented by the given x, y, and z values representing a euler vector. Rotates within the world space.

x: Amount to rotate by.
y: Amount to rotate by.

z: Amount to rotate by.
public Void Rotate(System.Single x, System.Single y, System.Single z)

Rotates the rotation of the transform by the Quaternion represented by the given euler vector.
eulerRotation: Amount to rotate by.
space: The space you want to rotate in.
public Void Rotate(UnityEngine.Vector3 eulerRotation, UnityEngine.Space space)

Rotates the rotation of the transform by the Quaternion represented by the given x, y, and z values representing a euler vector.
x: Amount to rotate by.
y: Amount to rotate by.
z: Amount to rotate by.
space: The space you want to rotate in.
public Void Rotate(System.Single x, System.Single y, System.Single z, UnityEngine.Space space)

Rotates the transform on the given axis by the given angle in the given space.
axis: Space you want to rotate on.
angle: Angle you want to rotate by.
space: Space you want to rotate in.
public Void Rotate(UnityEngine.Vector3 axis, System.Single angle, UnityEngine.Space space)

Rotates the transform on the given axis by the given angle in the world space.
axis: Space you want to rotate on.
angle: Angle you want to rotate by.
public Void Rotate(UnityEngine.Vector3 axis, System.Single angle)

Finds the local position of this VirtualTransform using the given parent position.
parentPositon: The position of the parent.
[Returns: The local position relative to the given parent position.]
public Vector3 GetLocalPosition(UnityEngine.Vector3 parentPositon)

Finds the local position of this VirtualTransform using the given parent Transform's position.
parentTransform: The Transform of the parent.
[Returns: The local position relative to the given parent position.]
public Vector3 GetLocalPosition(UnityEngine.Transform parentTransform)

Finds the local position of this VirtualTransform using the given parent VirtualTransform's position.
parentTransform: The VirtualTransform of the parent.
[Returns: The local position relative to the given parent position.]
public Vector3 GetLocalPosition(AdvancedUtilities.VirtualTransform parentTransform)

Finds the local rotation of this VirtualTransform using the given parent rotation.
parentRotation: Rotation of the parent.
[Returns: Local rotation of this VirtualTransform relative to the parent rotation.]
public Quaternion GetLocalRotation(UnityEngine.Quaternion parentRotation)

Finds the local rotation of this VirtualTransform using the given parent Transform rotation.
parentTransform: Transform of the parent.
[Returns: Local rotation of this VirtualTransform relative to the parent rotation.]
public Quaternion GetLocalRotation(UnityEngine.Transform parentTransform)

Finds the local rotation of this VirtualTransform using the given parent VirtualTransform rotation.
parentTransform: Transform of the parent.
[Returns: Local rotation of this VirtualTransform relative to the parent rotation.]
public Quaternion GetLocalRotation(AdvancedUtilities.VirtualTransform parentTransform)

No XML summary.
public String ToString()

AdvancedUtilities.LerpTransformers

Allows a function to be used against lerp t delta.

public class DelegateLerpTransformer

Properties

The function used by the delegate transformer.
Func`2 Function
public GET
private SET

Constructors

public DelegateLerpTransformer

(System.Func`2[System.Single,System.Single] function)
No XML summary.

Methods

Processes the given t and returns a new t value.

t: Given t.
[Returns: Processed t.]
public Single Process(System.Single t)

AdvancedUtilities.LerpTransformers

Does nothing to the passed value and simply passes it back.

public class DoNothingLerpTransformer

Constructors

public DoNothingLerpTransformer

No parameters.
No XML summary.

Methods

Processes the given t and returns a new t value.

t: Given t.
[Returns: Processed t.]
public Single Process(System.Single t)

AdvancedUtilities.LerpTransformers

An interface that describes a way to modify the delta t passed to lerp functions.

public interface ILerpTransformer

Methods

Processes the given t and returns a new t value.

t: Given t.
[Returns: Processed t.]
public Single Process(System.Single t)

AdvancedUtilities.LerpTransformers

Produces a smoothing effect at the beginning and end of lerping.

public class SmoothInOutLerpTransformer

Properties

The function used by the delegate transformer.
Func`2 Function
public GET

Constructors

public SmoothInOutLerpTransformer

No parameters.
Constructs a SmoothInOutLerpTransformer used to smooth the beginnings and ends of lerping.

Methods

Processes the given t and returns a new t value.

t: Given t.
[Returns: Processed t.]
public Single Process(System.Single t)

AdvancedUtilities.Cameras

A basic camera controller.

public class BasicCameraController

Fields

The distance that the camera wants to position itself at from the target.
public Single DesiredDistance

The minimum that zooming will let you zoom in to.
public Single MinZoomDistance

The maximum that zooming will let you zoom out to.
public Single MaxZoomDistance

When the CameraController starts, horizontal rotation will be set to this value.
public Single InitialHorizontalRotation

When the CameraController starts, vertical rotation will be set to this value.
public Single InitialVerticalRotation

TargetComponent
public TargetComponent Target

RotationComponent
public RotationComponent Rotation

ZoomComponent
public ZoomComponent Zoom

ViewCollisionComponent
public ViewCollisionComponent ViewCollision

InputComponent
public InputComponent Input

EasyUnityInputComponent
public EasyUnityInputComponent EasyUnityInput

CursorComponent
public CursorComponent Cursor

HeadbobComponent
public HeadbobComponent Headbob

ScreenShakeComponent
public ScreenShakeComponent ScreenShake

No XML summary.
public Camera Camera

Properties

Represents the Transform of the Camera being controlled.
VirtualTransform CameraTransform
public GET

Whether or not this CameraController has been initialized yet.
If the CameraController has not been initialized, it's component's may not function properly.
Boolean IsInitialized
public GET

Returns whether or not the components for this controller have been added already or not.
Boolean ComponentsAdded

public GET

Constructors

public BasicCameraController

No parameters.
No XML summary.

Methods

No XML summary.

public Void UpdateCamera()

Add a component to this Camera Controller.

component: The component to add.
public Void AddCameraComponent(AdvancedUtilities.Cameras.Components.CameraComponent component)

For each Camera Component that is shared between this Camera Controller and the given Camera Controller, this will copy the public fields from the given controller's components onto this controller's components. This does not replace this controller's components with the given controller's components.

otherController: The controller whose component's you want to override this controller's components.
public Void CopyPublicFields(AdvancedUtilities.Cameras.CameraController otherController)

No XML summary.

public T GetCameraComponent()

AdvancedUtilities.Cameras

The base class for creating CameraControllers.

public class CameraController

Fields

No XML summary.
public Camera Camera

Properties

Represents the Transform of the Camera being controlled.
VirtualTransform CameraTransform
public GET
private SET

Whether or not this CameraController has been initialized yet.
If the CameraController has not been initialized, it's component's may not function properly.
Boolean IsInitialized
public GET
private SET

Returns whether or not the components for this controller have been added already or not.
Boolean ComponentsAdded
public GET
private SET

Methods

Performs the camera's update.
public Void UpdateCamera()

Add a component to this Camera Controller.
component: The component to add.
public Void AddCameraComponent(AdvancedUtilities.Cameras.Components.CameraComponent component)

For each Camera Component that is shared between this Camera Controller and the given Camera Controller, this will copy the public fields from the given controller's components onto this controller's components. This does not replace this controller's components with the given controller's components.
otherController: The controller whose component's you want to override this controller's components.
public Void CopyPublicFields(AdvancedUtilities.Cameras.CameraController otherController)

No XML summary.
public T GetCameraComponent()

AdvancedUtilities.Cameras

No XML summary.

public class FollowCameraController

Fields

No XML summary.
public Single DesiredDistance

No XML summary.
public Single MinZoomDistance

No XML summary.
public Single MaxZoomDistance

No XML summary.
public TargetComponent Target

No XML summary.
public FollowRotationComponent FollowRotation

No XML summary.
public ZoomComponent Zoom

No XML summary.
public ViewCollisionComponent ViewCollision

No XML summary.
public InputComponent Input

No XML summary.
public EasyUnityInputComponent EasyUnityInput

No XML summary.
public HeadbobComponent Headbob

No XML summary.
public ScreenShakeComponent ScreenShake

No XML summary.
public Camera Camera

Properties

Represents the Transform of the Camera being controlled.
VirtualTransform CameraTransform
public GET

Whether or not this CameraController has been initialized yet.
If the CameraController has not been initialized, it's component's may not function properly.
Boolean IsInitialized
public GET

Returns whether or not the components for this controller have been added already or not.
Boolean ComponentsAdded
public GET

Constructors

public FollowCameraController

No parameters.
No XML summary.

Methods

No XML summary.

public Void UpdateCamera()

Add a component to this Camera Controller.

component: The component to add.
public Void AddCameraComponent(AdvancedUtilities.Cameras.Components.CameraComponent component)

For each Camera Component that is shared between this Camera Controller and the given Camera Controller, this will copy the public fields from the given controller's components onto this controller's components. This does not replace this controller's components with the given controller's components.

otherController: The controller whose component's you want to override this controller's components.
public Void CopyPublicFields(AdvancedUtilities.Cameras.CameraController otherController)

No XML summary.

public T GetCameraComponent()

AdvancedUtilities.Cameras

A camera controller that can hold a position and look at a target.

public class HoldPositionCameraController

Fields

This is the position that the Holding Camera will be located when looking at the target.
public Transform HoldingPosition

TargetComponent
public TargetComponent Target

ZoomComponent
public ZoomComponent Zoom

ViewCollisionComponent
public ViewCollisionComponent ViewCollision

CursorComponent
public CursorComponent Cursor

HeadbobComponent
public HeadbobComponent Headbob

ScreenShakeComponent
public ScreenShakeComponent ScreenShake

No XML summary.
public Camera Camera

Properties

Represents the Transform of the Camera being controlled.
VirtualTransform CameraTransform
public GET

Whether or not this CameraController has been initialized yet.
If the CameraController has not been initialized, it's component's may not function properly.
Boolean IsInitialized
public GET

Returns whether or not the components for this controller have been added already or not.
Boolean ComponentsAdded
public GET

Constructors

public HoldPositionCameraController

No parameters.
No XML summary.

Methods

No XML summary.
public Void UpdateCamera()

Add a component to this Camera Controller.
component: The component to add.
public Void AddCameraComponent(AdvancedUtilities.Cameras.Components.CameraComponent component)

For each Camera Component that is shared between this Camera Controller and the given Camera Controller, *this will copy the public fields from the given controller's components onto this controller's components. This does not replace this controller's components with the given controller's components.*

otherController: The controller whose component's you want to override this controller's components.
public Void CopyPublicFields(AdvancedUtilities.Cameras.CameraController otherController)

No XML summary.

public T GetCameraComponent()

AdvancedUtilities.Cameras

A camera capable of locking onto a target and looking at through another target making it look like the camera is locked onto
the target from your character.

public class LockTargetCameraController

Fields

The desired distance from the Target point by the Camera.
public Single DesiredDistance

SwitchTargetComponent
public SwitchTargetComponent LockTarget

MultipleRelativeTargetComponent
public MultipleRelativeTargetComponent Target

ZoomComponent
public ZoomComponent Zoom

ViewCollisionComponent
public ViewCollisionComponent ViewCollision

CursorComponent
public CursorComponent Cursor

HeadbobComponent
public HeadbobComponent Headbob

ScreenShakeComponent
public ScreenShakeComponent ScreenShake

No XML summary.
public Camera Camera

Properties

Represents the Transform of the Camera being controlled.
VirtualTransform CameraTransform
public GET

Whether or not this CameraController has been initialized yet.
If the CameraController has not been initialized, it's component's may not function properly.
Boolean IsInitialized
public GET

Returns whether or not the components for this controller have been added already or not.
Boolean ComponentsAdded
public GET

Constructors

public LockTargetCameraController

No parameters.
No XML summary.

Methods

No XML summary.

public Void UpdateCamera()

Add a component to this Camera Controller.

component: The component to add.

public Void AddCameraComponent(AdvancedUtilities.Cameras.Components.CameraComponent component)

For each Camera Component that is shared between this Camera Controller and the given Camera Controller, this will copy the public fields from the given controller's components onto this controller's components. This does not replace this controller's components with the given controller's components.

otherController: The controller whose component's you want to override this controller's components.

public Void CopyPublicFields(AdvancedUtilities.Cameras.CameraController otherController)

No XML summary.

public T GetCameraComponent()

AdvancedUtilities.Cameras

A camera controller that can control multiple other camera controllers and switch/lerp between them.

public class MultiCameraController

Fields

The Camera Controllers that are controlled by this MultiCamera Controller.
public List`1 CameraControllers

The Camera Controller in the list that is being used.
public Int32 CurrentIndex

The number of seconds it takes to switch between cameras.
public Single SwitchSpeed

When switching, if set, rotation will be Slerp'd instead of Lerp'd.
public Boolean SwitchSlerpRotation

When the lerping begins, the distance between position will be measured and used to calculate the duration of the lerp.
The TargetSwitchSpeed will represent the units/second that camera will lerp at when lerping begins.
That speed is not guarenteed throughout the lerp unless the camera positions and rotation remain stationary.
public Boolean ConstantSwitchSpeedInit

No XML summary.
public Camera Camera

Properties

The current camera controller being controlled.
CameraController CurrentCameraController
public GET

Whether or not the target is currently switching or not.
Boolean IsSwitching
public GET

This transformer alters the way the position lersp to new positions.
ILerpTransformer PositionLerpTransformer
public GET
private SET

This transformer alters the way therotation lerps to new positions.
ILerpTransformer RotationLerpTransformer
public GET
private SET

Represents the Transform of the Camera being controlled.
VirtualTransform CameraTransform
public GET

Whether or not this CameraController has been initialized yet.
If the CameraController has not been initialized, it's component's may not function properly.
Boolean IsInitialized
public GET

Returns whether or not the components for this controller have been added already or not.
Boolean ComponentsAdded
public GET

Constructors

public MultiCameraController

No parameters.
No XML summary.

Methods

No XML summary.

public Void UpdateCamera()

Switches the active camera to the given index.

index: Index in the list of cameras.
copyComponents: Whether or not components fields will be copied from the current camera to the next.
public Void SwitchCamera(System.Int32 index, System.Boolean copyComponents)

No XML summary.

public Void SwitchCamera(System.Boolean copyComponents)

Add a component to this Camera Controller.

component: The component to add.
public Void AddCameraComponent(AdvancedUtilities.Cameras.Components.CameraComponent component)

For each Camera Component that is shared between this Camera Controller and the given Camera Controller, this will copy the public fields from the given controller's components onto this controller's components. This does not replace this controller's components with the given controller's components.

otherController: The controller whose component's you want to override this controller's components.
public Void CopyPublicFields(AdvancedUtilities.Cameras.CameraController otherController)

No XML summary.

public T GetCameraComponent()

AdvancedUtilities.Cameras

A camera controller for side scrolling.

public class SideScrollingCameraController

Fields

The distance that the camera wants to position itself at from the target.
public Single DesiredDistance

When the CameraController starts, horizontal rotation will be set to this value.
public Single InitialHorizontalRotation

When the CameraController starts, vertical rotation will be set to this value.
public Single InitialVerticalRotation

TargetComponent
public TargetComponent Target

RotationComponent
public RotationComponent Rotation

ZoomComponent
public ZoomComponent Zoom

ViewCollisionComponent
public ViewCollisionComponent ViewCollision

HeadbobComponent
public HeadbobComponent Headbob

ScreenShakeComponent
public ScreenShakeComponent ScreenShake

No XML summary.
public Camera Camera

Properties

Represents the Transform of the Camera being controlled.
VirtualTransform CameraTransform
public GET

Whether or not this CameraController has been initialized yet.
If the CameraController has not been initialized, it's component's may not function properly.
Boolean IsInitialized
public GET

Returns whether or not the components for this controller have been added already or not.
Boolean ComponentsAdded
public GET

Constructors

public SideScrollingCameraController

No parameters.
No XML summary.

Methods

No XML summary.
public Void UpdateCamera()

Add a component to this Camera Controller.

component: The component to add.
public Void AddCameraComponent(AdvancedUtilities.Cameras.Components.CameraComponent component)

For each Camera Component that is shared between this Camera Controller and the given Camera Controller, this will copy the public fields from the given controller's components onto this controller's components. This does not replace this controller's components with the given controller's components.

otherController: The controller whose component's you want to override this controller's components.
public Void CopyPublicFields(AdvancedUtilities.Cameras.CameraController otherController)

No XML summary.

public T GetCameraComponent()

AdvancedUtilities.Cameras.Components

A component that can be attached to a CameraController.

Fields that are publically accessible from the Editor should not be used to track information.
Publically accessible fields should only be used to hold settings, so that copying public fields functions properly.

public class CameraComponent

Fields

Calculations by Camera Components involving floats will use this tolerance when comparing floats.
public, static, literal, hasdefault Single FLOAT_TOLERANCE

Methods

Sets up this component for the given Camera Controller that it is attached to.

cameraController: The camera controller that this component is attached to.
public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.
public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

An error thrown by CameraComponents when accessing something that requires the CameraComponent to have been initialized first.

public class ComponentNotInitializedException

Constructors

public ComponentNotInitializedException

(System.String message)
No XML summary.

AdvancedUtilities.Cameras.Components

A component that can take control of the cursor's locking and visiblity states based off settings.

public class CursorComponent

Fields

Sets whether the controller should control locking the mouse or not.
public Boolean Enabled

The button input for LockCursorButton.
public String LockButtonInputName

The mouse will lock and unlock automatically when the provided LockButtonInputName is being pressed and released.
public Boolean LockCursorButton

The button input for HoldToReleaseLock.
public String HoldToReleaseInputName

If true, the HoldToReleaseInputName will unlock the cursor when held and will be relocked when released.
public Boolean HoldToReleaseLock

Whether or not the cursor will be visible when locked.
public Boolean VisibleWhenLocked

Whether or not the cursor will be visible when unlocked.
public Boolean VisibleWhenUnlocked

Locking will disabled when the pointer meets the conditions for the PointerOverGui object.
public PointerOverGui DisableLockingWhenOver

Constructors

public CursorComponent

No parameters.
No XML summary.

Methods

No XML summary.
public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Sets the cursor lock and visiblity based off the current input and settings.
public Void SetCursorLock()

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.
public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

A component to automatically grab input in a simple way for Camera Controllers to use without the need to create InputValues objects and pass them manually.

public class EasyUnityInputComponent

Fields

Whether or not this component will do anything when it's called to append its input.
public Boolean Enabled

Easy input will be appended over other input. If false, it will be appended under.
public Boolean AppendOver

The axis input used to rotate the camera's view horizontally.
public String HorizontalInputName

The axis input used to rotate the camera's view vertically.
public String VerticalInputName

The axis input used to scroll the distance inwards.
public String ZoomInInputName

The axis input used to scroll the distance outwards.
public String ZoomOutInputName

Horizontal input will be found using the given input name.
public Boolean EnableHorizontal

Vertical input will be found using the given input name.
public Boolean EnableVertical

ZoomIn input will be found using the given input name.
public Boolean EnableZoomIn

ZoomOut input will be found using the given input name.
public Boolean EnableZoomOut

The input name for the button to support EnableRotationOnlyWhenMousePressed.
public String MouseInputButtonName

When true, the input will only pass Horizontal and Vertical input when the mouse button specified is pressed.
public Boolean EnableRotationOnlyWhenMousePressed

When true, the input will only pass Horizontal and Vertical input when the cursor is locked.
public Boolean EnableRotationOnlyWhenCursorLocked

Input will disabled when the pointer meets the conditions for the PointerOverGui object.
public PointerOverGui DisableInputWhenOver

Constructors

public EasyUnityInputComponent

No parameters.
No XML summary.

Methods

No XML summary.

public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Appends the Easy input to the current Input component.

public Void AppendInput()

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.

public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

A camera component that handles rotation for following behind a target automatically.

public class FollowRotationComponent

Fields

Whether the camera will rotate to follow the target or not.
public Boolean Enabled

If set, the rotation will always be snapped to the nuetral position.
public Boolean SnapRotation

The offset of the Y rotation from the target's actual rotation.
If set to a positive value, then this camera will always try to rotate to look at the target from above as a nuetral state.
public Single YRotationOffset

The offset of the X rotation from the target's actual rotation.
If set to a positive value, then this camera will always try to rotate to look at the target from the right as a nuetral state.
public Single XRotationOffset

The speed that the camera renormalizes back to nuetral orientation.
public Single AnglePerSecond

The amount of movement needed to trigger the AdjustCompletelyOnMovement and the AdjustWhileMoving modes.
public Single MovementTolerance

The amount of rotation needed to trigger the AdjustCompletelyOnRotation mode.
public Single RotationTolerance

The mode of adjusting the rotation of the camera.
public FollowRotationMode Mode

Rotation will be determined by the difference in movement between updates instead of the rotation of the actual target itself.
public Boolean DetermineRotationFromMovement

When determining rotation from movement, this will flatten the rotation using the target's UP vector so that it doesn't not rotate at angles.
public Boolean FlattenRotationWhenDeterminingFromMovement

A new rotation will only be determined after the character moves this many units.
public Single DetermineTolerance

Properties

Whether or not the camera is currently adjusting it's rotation to the nuetral state of the target's rotation.
Boolean IsRotating
public GET

Lerp transformer used for rotation.
ILerpTransformer RotationLerpTransformer
public GET
public SET

Constructors

public FollowRotationComponent

No parameters.
No XML summary.

Methods

No XML summary.

public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

The camera will rotate to it's neutral position behind the target.

public Void ForceAdjustment()

Updates the rotation of the camera based on the target and

public Void UpdateRotation()

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.

public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

A component that allows the camera to perform a headbob like motion.

public class HeadbobComponent

Fields

Whether Headbob is enabled or not. Headbobbing will happen at 0 distance if enabled.
public Boolean Enabled

If activated, the Headbob Component will bob regardless of the settings.
public Boolean Forced

If true, then headbobbing will occur at distances from the target up to the given Range, otherwise it will only happen at 0.
public Boolean EnableRangedActivation

The distance from the target that the camera will perform a headbobbing motion.
public Single ActivationRange

The curve that the camera will headbob on modifying the target's position by the camera's up vector.
public AnimationCurve MagnitudeCurve

The total time it takes to complete the headbob curve.
public Single DurationOfCurve

This number will be multiplied against the number of units the camera is offset by during headbobbing.
public Single ScaleOfCurve

When you stop headbobbing, the camera will readjust itself back to normal viewing levels. This is the total time it will take in seconds to do so.
public Single ReadjustToNormalTime

Properties

Whether the camera is currently headbobbing or not.
Boolean IsHeadbobbing
public GET
private SET

That headbobbing is currently readjusting itself back to neutral after being disabled.
Boolean Reorienting
public GET
private SET

When the Headbobbing is reorienting back to normal, this transformer modifies the way it lerps back to normal. Altering this when the headbob component is reorienting may result is errors.
ILerpTransformer ReorientingLerpTransformer
public GET
public SET

Constructors

public HeadbobComponent

No parameters.
No XML summary.

Methods

No XML summary.
public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Returns the offset needed to maintain headbobbing properly when both enabled and disabled. When disabled, this method will gradually reduce the offset to Vector3.zero.

cameraDistanceFromTarget: The units the camera is currently from the target.
[Returns: The offset needed to headbob.]
public Vector3 GetHeadbobModifier(System.Single cameraDistanceFromTarget)

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.
public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

A component that can take and process input values that can then be passed to other components.

public class InputComponent

Fields

Whether or not horizontal input will be processed.
public Boolean EnableHorziontal

Whether or not vertical input will be processed.
public Boolean EnableVertical

Whether or not zooming in input will be processed.
public Boolean EnableZoomIn

Whether or not zooming out input will be processed.
public Boolean EnableZoomOut

Sensitivity settings for this InputComponent.
public InputSensitivity Sensitivity

The input inversion settings for this InputComponent
public InputInversion Invert

Properties

The current InputValues object on the InputComponent.
This is the actual InputValues object used by the component.
InputValues Input
public GET
private SET

The processed InputValue modified by the settings on this InputComponent.
This is a copy of the InputValues object used by the component and will not affect the component.
InputValues ProcessedInput
public GET

Constructors

public InputComponent

No parameters.
No XML summary.

Methods

No XML summary.
public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Processes the given input and returns a newly created object with the input sensitivities, inversions, and enablings of this component setup properly.
input: Input to process.
public InputValues ProcessInput(AdvancedUtilities.Cameras.Components.InputValues input)

Sets the input for the InputComponent. This completely overrides the current input replacing it.
input: Input you want to set it to.
public Void SetInput(AdvancedUtilities.Cameras.Components.InputValues input)

No XML summary.
public Void SetInput(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Nullable`1[System.Single] zoomIn, System.Nullable`1[System.Single] zoomOut)

Clears the input values on the InputComponent.

public Void ClearInput()

Inverts the input of each input multiplying it by -1.

public Void InvertInput()

Applies all non-null input properties in the given Input to the existing input. If appendOver is set to true, then all non-null values will replace any values currently existing. If appendOver is set to false, then all non-null values will only replace null values in the current input.

input: Input to apply properties under this objects properties.
appendOver: Appends these values over existing values.
public Void AppendInput(AdvancedUtilities.Cameras.Components.InputValues input, System.Boolean appendOver)

No XML summary.

public Void AppendInput(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Nullable`1[System.Single] zoomIn, System.Nullable`1[System.Single] zoomOut, System.Boolean appendOver)

Adds all non-null values to the current input. Null values on the current input become whatever the given input provides.

input: Input you want to add.
public Void AddInput(AdvancedUtilities.Cameras.Components.InputValues input)

No XML summary.

public Void AddInput(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Nullable`1[System.Single] zoomIn, System.Nullable`1[System.Single] zoomOut)

Multiplies all non-null values to the current input.

input: The given input containing the values to multiply by.
public Void MultiplyInput(AdvancedUtilities.Cameras.Components.InputValues input)

No XML summary.

public Void MultiplyInput(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Nullable`1[System.Single] zoomIn, System.Nullable`1[System.Single] zoomOut)

Multiplies the current input by delta time.

public Void MultiplyInputByDeltaTime()

No XML summary.

public Void SetHorizontalInput(System.Nullable`1[System.Single] horizontal)

No XML summary.

public Void SetVerticalInput(System.Nullable`1[System.Single] vertical)

No XML summary.

public Void SetZoomInInput(System.Nullable`1[System.Single] zoomIn)

No XML summary.

public Void SetZoomOutInput(System.Nullable`1[System.Single] zoomOut)

This will override any null values currently on the Input and set them to 0.

public Void SetNullInputValuesToZero()

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.

public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

A class that contains a set of values to be input into the InputComponent

public class InputValues

Properties

The input for Horizontal input.
Nullable`1 Horizontal
public GET
public SET

The input for Vertical input.
Nullable`1 Vertical
public GET
public SET

The input for zooming in input.
Nullable`1 ZoomIn
public GET
public SET

The input for zooming out input.
Nullable`1 ZoomOut
public GET
public SET

Constructors

public InputValues

(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Nullable`1[System.Single] zoomIn, System.Nullable`1[System.Single] zoomOut)
No XML summary.

public InputValues

No parameters.
Empty constructor leaving all values null.

Methods

Returns a new InputValues object with copied data from this InputValues object.
public InputValues Copy()

Applies all non-null input properties in the given Input to this object. This overrides the current properties with the given properties if they are not null.
input: Input to apply properties over this objects properties.
public Void ApplyOver(AdvancedUtilities.Cameras.Components.InputValues input)

Applies all non-null input properties in the given Input to this object's null properties. This only sets null properties with the values on the given input and will not override any of it's set properties.
input: Input to apply properties under this objects properties.
public Void ApplyUnder(AdvancedUtilities.Cameras.Components.InputValues input)

No XML summary.
public Void AddValues(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Nullable`1[System.Single] zoomIn, System.Nullable`1[System.Single] zoomOut)

No XML summary.
public Void MultiplyBy(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Nullable`1[System.Single] zoomIn, System.Nullable`1[System.Single] zoomOut)

Sets each corresponding property given a true value to null.

horizontal: Set Horizontal to null.
vertical: Set Vertical to null.
zoomIn: Set ZoomIn to null.
zoomOut: Set ZoomOut to null.
public Void NullOutValues(System.Boolean horizontal, System.Boolean vertical, System.Boolean zoomIn, System.Boolean zoomOut)

Inverts each corresponding property given a true value, multiplying it by -1.
horizontal: Invert Horizontal.
vertical: Invert Vertical.
zoomIn: Invert ZoomIn.
zoomOut: Invert ZoomOut.
public Void InvertValues(System.Boolean horizontal, System.Boolean vertical, System.Boolean zoomIn, System.Boolean zoomOut)

Multiplies this input by delta time.
public Void MultiplyByDeltaTime()

Replaces any null input values currently on this input with 0;
public Void SetNullValuesToZero()

No XML summary.
public String ToString()

AdvancedUtilities.Cameras.Components

A complex target system that can choose a target from a list of targets based off the proximity of each target to a world space point, and can lerp to the location of new targets.

public class MultipleRelativeTargetComponent

Fields

Possible targets for the component to choose from. Local space offsets will use each individual transform's rotation when factoring it in.
public List`1 Targets

When getting the target, this will determine the target. Closest will get the target closest to a given world space point.
public MultipleRelativeTargetSwitchMode SwitchMode

The total number of seconds for a switch to occur.
If ConstantSwitchSpeedInit is set, then it will be the number of units per second at the beginning of the switch.
If target position's change, that speed may end up not being linear.
public Single SwitchSpeed

The current target won't be switched regardless of the mode.
public Boolean MaintainTarget

Makes the switch speed during each switch a constant based off the switch speed.
If transforms move to different locations in relation to each other, then the speed may not remain constant.
public Boolean ConstantSwitchSpeedInit

Properties

This is the current list of offsets in the World Space that modify the target.
This is a copied list, so that modifying it does not damage the offsets.
IList`1 WorldSpaceOffsets
public GET

This is the sum of the World Space offsets
Vector3 WorldSpaceOffset
public GET

This is the current list of offsets in the Local relative Space that modify the target.
These offsets of relative to the Transform used for the Target and are modified by it's rotation.
This is a copied list, so that modifying it does not damage the offsets.
IList`1 LocalSpaceOffsets
public GET

This is the sum of the Local relative Space offsets.
This offsets is relative to the Transform used for the Target and are modified by it's rotation.
Vector3 LocalSpaceOffset
public GET

Modifies the way the lerping is transformed when switching targets.
ILerpTransformer LerpTransformer
public GET
public SET

Whether or not the target is currently switching or not.
Boolean IsSwitching
public GET

Constructors

public MultipleRelativeTargetComponent

No parameters.

No XML summary.

Methods

No XML summary.

public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Gets a random target.

[Returns: Random target.]
public Vector3 GetTarget()

Gets a random target.

[Returns: Random target.]
public Vector3 GetTarget(System.Boolean maintainTarget)

Gets a random target.

[Returns: Random target.]
public Vector3 GetTarget(UnityEngine.Vector3 worldSpacePoint, System.Boolean maintainTarget)

Gets a random target.

[Returns: Random target.]
public Vector3 GetTarget(UnityEngine.Vector3 worldSpacePoint)

Adds a World Space offset that modifies where the target is in the 3D World Space.

offset: Offset of the target to add.
public Void AddWorldSpaceOffset(UnityEngine.Vector3 offset)

Adds a Local relative Space offset that modifies where the target is in the 3D Local Space. This offset factors in the Target Transform's rotation to maintain it's relativity.

offset: Offset of the target to add.
public Void AddLocalSpaceOffset(UnityEngine.Vector3 offset)

Clears all current World Space offsets and Local Space offsets.

public Void ClearAdditionalOffsets()

Clears all World Space offsets.

public Void ClearWorldSpaceOffsets()

Clears all Local Space offsets.

public Void ClearLocalSpaceOffsets()

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.
public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

Used for determining if Unity's mouse is over a provided Rect, or over a GUI element.

public class PointerOverGui

Fields

Input will not work while the mouse is over a gui.
Uses "EventSystem.current.IsPointerOverGameObject()"
public Boolean EventSystemWhenPointerOverGuiElement

When the pointer is within any of these rects, it will not provide input.
public List`1 WhenPointerOverRects

Constructors

public PointerOverGui

No parameters.
No XML summary.

Methods

Returns whether or not the pointer is over a Gui element or within a rect provided.

[Returns: Whether the pointer is on a GUI element.]
public Boolean IsPointerOverGui()

AdvancedUtilities.Cameras.Components

A component that can rotate the view of a camera.

public class RotationComponent

Fields

Whether or not the RotationComponent will do anything when called.
public Boolean Enabled

The settings for rotation limits for this RotationComponent.
Limits the amount you can rotate in any direction based off the orientation.
public RotationLimits Limits

The settings for horizontal rotation events for this RotationComponent.
The camera controller has the ability to fire an event when you rotate it on the horizontal direction.
public RotationHorizontalDegreesEvent HorizontalDegreesEvent

The settings for vertical rotation events for this RotationComponent.
The camera controller has the ability to fire an event when you rotate it on the vertical direction.
public RotationVerticalDegreesEvent VerticalDegreesEvent

Allows you to automatically rotate the camera over a period of time
public RotationAutoRotate AutoRotation

Properties

The orientation of the Camera's rotation. This orientation is used to determine what is horizontal, vertical, and limits on rotation.
Quaternion Orientation
public GET

The up vector for the current orientation of the Camera's rotation.
Vector3 OrientationUp
public GET

The amount of degrees the camera has rotated horizontally from the nuetral position bounded by [0, 360)
Single HorizontalRotation
public GET

The amount of degrees the camera has rotated vertically from the nuetral position bounded by [0, 360)
Single VerticalRotation
public GET

The VirtualTransform used to represent the orientation.
Be careful directly modifying this as it may have unintended side effects.
Modifying rotation of the orientation directly may have uses if you want to modify it's rotation without rotating the whole system,
but manual rotation will not obey limits set up in the Rotation component.
I imagine the primary use of directly accessing this would be when you've limited the horizontal rotation and you want to change the
orientation so the limits are in different locations.
VirtualTransform OrientationTransform
public GET

Constructors

public RotationComponent

No parameters.
No XML summary.

Methods

No XML summary.

public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Adjusts the orientation of RotationComponent. This should be used to change the up vector of the Camera's rotation, as well as where the limits of rotation are for the RotationComponent. This will preserve the Camera's current location in relation to the target by moving the Camera.

eulerAngles: New orientation in euler angles.
target: OffsetTarget that is currently being Rotated around.
public Void SetRotationOrientation(UnityEngine.Vector3 eulerAngles, UnityEngine.Vector3 target)

Adjusts the orientation of RotationComponent. This should be used to change the up vector of the Camera's rotation, as well as where the limits of rotation are for the RotationComponent. This will preserve the Camera's current location in relation to the target by moving the Camera.

orientation: New orientation.
target: Target that is currently being Rotated around.
public Void SetRotationOrientation(UnityEngine.Quaternion orientation, UnityEngine.Vector3 target)

Updates the automatic rotation. This will progress the current automatic rotation.

public Void UpdateAutoRotate()

Automatically rotates the camera by the given horizontal and vertical value, in the given amount of time.

horizontal: Degrees of rotation for the horizontal.
vertical: Degrees of rotation for the vertical.
time: Total time to complete this rotation.
public Void AutoRotateBy(System.Single horizontal, System.Single vertical, System.Single time)

Automatically rotates the camera by the given horizontal and vertical value, in the given amount of time.

horizontal: Degrees of rotation for the horizontal.
vertical: Degrees of rotation for the vertical.
time: Total time to complete this rotation.
lerpTransformer: Transformers the progression of the rotation for both rotations using the lerp transformer.
public Void AutoRotateBy(System.Single horizontal, System.Single vertical, System.Single time, AdvancedUtilities.LerpTransformers.ILerpTransformer lerpTransformer)

Automatically rotates the camera by the given horizontal and vertical value, in the given amount of time.

horizontal: Degrees of rotation for the horizontal.
vertical: Degrees of rotation for the vertical.
time: Total time to complete this rotation.
horizontalLerpTransformer: Transformers the progression of the rotation for the horizontal rotation using the lerp transformer.
verticalLerpTransformer: Transformers the progression of the rotation for the vertical rotation using the lerp transformer.
public Void AutoRotateBy(System.Single horizontal, System.Single vertical, System.Single time, AdvancedUtilities.LerpTransformers.ILerpTransformer horizontalLerpTransformer, AdvancedUtilities.LerpTransformers.ILerpTransformer verticalLerpTransformer)

No XML summary.

public Void AutoRotateTo(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Single time)

No XML summary.

public Void AutoRotateTo(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Single time, AdvancedUtilities.LerpTransformers.ILerpTransformer lerpTransformer)

No XML summary.

public Void AutoRotateTo(System.Nullable`1[System.Single] horizontal, System.Nullable`1[System.Single] vertical, System.Single time, AdvancedUtilities.LerpTransformers.ILerpTransformer horizontalLerpTransformer, AdvancedUtilities.LerpTransformers.ILerpTransformer verticalLerpTransformer)

Rotates the Camera horizontally by the given degrees.

degrees: Given degrees.
public Void RotateHorizontally(System.Single degrees)

Rotates the Camera vertically by the given degrees

degrees: Given degrees.
public Void RotateVertically(System.Single degrees)

Rotates the Camera horizontally and then vertically at once by the given degrees.

horizontalDegrees: Given degrees for horizontal.
verticalDegrees: Given degrees for vertical.
public Void Rotate(System.Single horizontalDegrees, System.Single verticalDegrees)

Sets the rotation of the rotation component to the given settings.

horizontalDegrees: Horizontal degrees of rotation to set it to.
verticalDegrees: Vertical degrees of rotation to set it to.
trackRotation: Whether or not rotation is tracked for events.
public Void SetRotation(System.Single horizontalDegrees, System.Single verticalDegrees, System.Boolean trackRotation)

Sets the rotation of the rotation component to the given settings.

rotation: Represents the rotation we want.
trackRotation: Whether or not rotation is tracked for events.
public Void SetRotation(UnityEngine.Quaternion rotation, System.Boolean trackRotation)

Returns the amount of degrees you can rotate by when enforcing horizontal limits when given an amount of degrees you want to rotate by.

degrees: Degrees you want to attempt to rotate by.
[Returns: Actual degrees you're allowed to rotate by.]
public Single GetEnforcedHorizontalDegrees(System.Single degrees)

Returns the amount of degrees you can rotate by when enforcing vertical limits when given an amount of degrees you want to rotate by.

degrees: Degrees you want to attempt to rotate by.
[Returns: Actual degrees you're allowed to rotate by.]
public Single GetEnforcedVerticalDegrees(System.Single degrees)

Checks the horizontal degrees events and fires events as needed first, then checks the vertical degrees events and fires events as needed. This does not fire if the rotation component is not enabled.

public Void CheckRotationDegreesEvents()

Checks the horizontal degrees events and fires events as needed. This does not fire if the rotation component is not enabled or if the horizontal degrees event is not enabled.

public Void CheckHorizontalDegreesEvents()

Registers a listener to listen to when the camera has rotated a set number of degrees. The value will be positive or negative based on which direction the camera rotated. You shouldn't alter the camera during this event or bad things may happen.

degreesListener: The listener
public Void RegisterListener(AdvancedUtilities.Cameras.Components.Events.HorizontalDegreesListener degreesListener)

Unregisters a listener so that it no longer listens to when the camera has rotated a set number of degrees.

degreesListener:
public Void UnregisterListener(AdvancedUtilities.Cameras.Components.Events.HorizontalDegreesListener degreesListener)

Clears all HorizontalDegreeListeners from listening to this RotationComponent.

public Void ClearHorizontalListeners()

Checks the vertical degrees events and fires events as needed. This does not fire if the rotation component is not enabled or if the vertical degrees event is not enabled.

public Void CheckVerticalDegreesEvents()

Registers a listener to listen to when the camera has rotated a set number of degrees. The value will be positive or negative based on which direction the camera rotated. You shouldn't alter the camera during this event or bad things may happen.

degreesListener: The listener

public Void RegisterListener(AdvancedUtilities.Cameras.Components.Events.VerticalDegreesListener degreesListener)

Unregisters a listener so that it no longer listens to when the camera has rotated a set number of degrees.

degreesListener:

public Void UnregisterListener(AdvancedUtilities.Cameras.Components.Events.VerticalDegreesListener degreesListener)

Clears all HorizontalDegreeListeners from listening to this RotationComponent.

public Void ClearVerticalListeners()

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.

public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

A component that allows the camera to shake vertically, horizontally, or both at the same time.

public class ScreenShakeComponent

Fields

Whether the screen is shaking or not.
public Boolean Enabled

The type of shaking the Camera will do.
public ScreenShakeMode Mode

The intensity in units that the Camera will shake Horizontally.
public Single HorizontalIntensity

The intensity in units that the Camera will shake Vertically.
public Single VerticalIntensity

Uses the specified seed for Random shaking rather than a time dependant seed.
public Boolean UseSpecificSeed

A seed for Random shaking.
public Int32 Seed

Amount of seconds between each position being randomized.
public Single RandomizeTime

Constructors

public ScreenShakeComponent

No parameters.
No XML summary.

Methods

No XML summary.

public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Returns a random shaking offset Vector3.

[Returns: Offset for randomly shaking.]
public Vector3 GetShaking()

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.
public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

A target that when switched can lerp to the new target rather than snapping to it.

public class SwitchTargetComponent

Fields

The current target we want to be referencing.
public Transform Target

The total number of seconds for a switch to occur.
If ConstantSwitchSpeedInit is set, then it will be the number of units per second at the beginning of the switch.
If target position's change, that speed may end up not being linear.
public Single SwitchSpeed

Makes the switch speed during each switch a constant based off the switch speed.
If transforms move to different locations in relation to each other, then the speed may not remain constant.
public Boolean ConstantSwitchSpeedInit

Properties

This is the current list of offsets in the World Space that modify the target.
This is a copied list, so that modifying it does not damage the offsets.
IList`1 WorldSpaceOffsets
public GET

This is the sum of the World Space offsets
Vector3 WorldSpaceOffset
public GET

This is the current list of offsets in the Local relative Space that modify the target.
These offsets of relative to the Transform used for the Target and are modified by it's rotation.
This is a copied list, so that modifying it does not damage the offsets.
IList`1 LocalSpaceOffsets
public GET

This is the sum of the Local relative Space offsets.
This offsets is relative to the Transform used for the Target and are modified by it's rotation.
Vector3 LocalSpaceOffset
public GET

Modifies the way the lerping is transformed when switching targets.
ILerpTransformer LerpTransformer
public GET
public SET

Whether or not the target is currently switching or not.
Boolean IsSwitching
public GET

Constructors

public SwitchTargetComponent

No parameters.
No XML summary.

Methods

No XML summary.

public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Returns the target's location.

[Returns: Target vector3.]

public Vector3 GetTarget()

Adds a World Space offset that modifies where the target is in the 3D World Space.

offset: Offset of the target to add.
public Void AddWorldSpaceOffset(UnityEngine.Vector3 offset)

Adds a Local relative Space offset that modifies where the target is in the 3D Local Space. This offset factors in the Target Transform's rotation to maintain it's relativity.

offset: Offset of the target to add.
public Void AddLocalSpaceOffset(UnityEngine.Vector3 offset)

Clears all current World Space offsets and Local Space offsets.

public Void ClearAdditionalOffsets()

Clears all World Space offsets.

public Void ClearWorldSpaceOffsets()

Clears all Local Space offsets.

public Void ClearLocalSpaceOffsets()

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.
public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

This component describes the location in World Space where a target currently is.

public class TargetComponent

Fields

This will be the target that the camera will attempt to aim at.
public Transform Target

Properties

This is the current list of offsets in the World Space that modify the target.
This is a copied list, so that modifying it does not damage the offsets.
IList`1 WorldSpaceOffsets
public GET

This is the sum of the World Space offsets
Vector3 WorldSpaceOffset
public GET

This is the current list of offsets in the Local relative Space that modify the target.
These offsets of relative to the Transform used for the Target and are modified by it's rotation.
This is a copied list, so that modifying it does not damage the offsets.
IList`1 LocalSpaceOffsets
public GET

This is the sum of the Local relative Space offsets.
This offsets is relative to the Transform used for the Target and are modified by it's rotation.
Vector3 LocalSpaceOffset
public GET

Constructors

public TargetComponent

No parameters.
No XML summary.

Methods

No XML summary.

public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Adds a World Space offset that modifies where the target is in the 3D World Space.

offset: Offset of the target to add.
public Void AddWorldSpaceOffset(UnityEngine.Vector3 offset)

Adds a Local relative Space offset that modifies where the target is in the 3D Local Space. This offset factors in the Target Transform's rotation to maintain it's relativity.

offset: Offset of the target to add.
public Void AddLocalSpaceOffset(UnityEngine.Vector3 offset)

Clears all current World Space offsets and Local Space offsets.

public Void ClearAdditionalOffsets()

Clears all World Space offsets.

public Void ClearWorldSpaceOffsets()

Clears all Local Space offsets.

public Void ClearLocalSpaceOffsets()

Returns the target that the camera should be looking at.

[Returns: The position in 3D space that the camera should be looking at.]
public Vector3 GetTarget()

Returns the actual distance from the Camera to the current target. This is the distance between the two positions in 3D space.

[Returns: Distance from camera position to the target position]
public Single GetDistanceFromTarget()

Returns the actual distance from the Camera to the current target. This is the distance between the two positions in 3D space.

[Returns: Distance from camera position to the target position]
public Single GetDistanceFromTarget(UnityEngine.Vector3 target)

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.
public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

This component calculates positions using the Camera's rotation and provided values to determine if there are any game objects
between the target and a position outside of the target.

public class ViewCollisionComponent

Fields

The camera will take collision into account when determining how far away from the target it should be.
public Boolean Enabled

These layers will indicate layers that are considered line of sight blockers and will force the camera forward.
public LayerMask LineOfSightLayers

These GameObjects will be ignored when handling view collision even if they are apart of the LineOfSightLayers LayerMask.
public List`1 IgnoreTheseGameObjects

X value: Left or right on the screen. Greater values are further to the right.
Y value: Up or down on the screen. Greater values are further to the top of the screen.
Z value: In or out, depth of the screen. Greater values are closer to you, while negative values extend past the target.
public List`1 SamplingPoints

Allows you to ignore objects that are below a certain level of thickness.
There is an inherit inaccuracy based off the number of checks you perform.
Objects near the targeted width may not have desired results if too few checks are done, or if they are too similar to the targeted width.
public ViewCollisionThicknessChecking ThicknessChecking

Constructors

public ViewCollisionComponent

No parameters.
No XML summary.

Methods

No XML summary.
public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Calculates the maximum distance that the camera can be at from the target when factoring in view collision. This method assumes the Camera's rotation is set up properly and calculates outwards from the target based on that rotation.
target: The target the camera will calculate the distance to.
furthestDistance: The maximum distance from the target that the camera can be at.
[Returns: The maximum distance from the target when factoring in view collision, if set to true.]
public Single CalculateMaximumDistanceFromTarget(UnityEngine.Vector3 target, System.Single furthestDistance)

Calculates the maximum distance that the camera can be at from the target when factoring in view collision. This method assumes the Camera's rotation is set up properly and calculates outwards from the target based on that rotation. This method overrides the settings for Enabled with the provided value for viewCollisionEnabled.
target: The target the camera will calculate the distance to.
furthestDistance: The maximum distance from the target that the camera can be at.
viewCollisionEnabled: Whether or not view collision will be considered or not.
[Returns: The maximum distance from the target when factoring in view collision, if set to true.]
public Single CalculateMaximumDistanceFromTarget(UnityEngine.Vector3 target, System.Single furthestDistance, System.Boolean viewCollisionEnabled)

Returns whether or not the give GameObject should be considered something the camera ray casting should colide with.

collisionObject: The game object that may have view collision.
[Returns: The game object has view collision.]
public Boolean IsViewCollidable(UnityEngine.GameObject collisionObject)

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.
public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

The state of this component is persistant between calls for calculations,
however it will handle any adjustments that you make to it on the fly based off the values you provide it.

public class ZoomComponent

Fields

When the camera needs to move inward to adjust for collision, this is the minimum the camera will adjust to, regardless if collision determines the distance should be a smaller value.
public Single MinimumDistance

Whether or not readjusting inwards will be instantanous or not.
public Boolean SnapIn

Whether or not readjusting outwards will be instantanous or not.
public Boolean SnapOut

If true, the camera will travel the given Zoom Speeds in units each second,
rather than the entire distance in Speed number of seconds.
Changing this value while the camera is zooming is not supported."
public Boolean UniformSpeed

The speed the camera moves into the target. Varies by which zooming setting you have set.
public Single ZoomInSpeed

The speed the camera moves out from the target. Varies by which zooming setting you have set.
public Single ZoomOutSpeed

Each time the Desired Distance changes, a percentage of the total travel time will be added to the travel time.
public Boolean EnableAddPercentageOnScroll

Each time the Desired Distance changes, this percentage of the total travel time will be added to travel time.
public Single PercentageToAdd

Properties

Whether the camera is currently scrolling in or out at all
Boolean IsZooming
public GET

Whether the camera is currently scrolling in
Boolean IsZoomingIn
public GET
private SET

Whether the camera is currently scrolling out
Boolean IsZoomingOut
public GET
private SET

This transformer alters the way the zooming lerps to new desired distances.
ILerpTransformer ZoomLerpTransformer
public GET
private SET

Constructors

public ZoomComponent

No parameters.
No XML summary.

Methods

No XML summary.

public Void Initialize(AdvancedUtilities.Cameras.CameraController cameraController)

Calculates the distance that the camera should be from the target given. This factors in how far the camera should be from the target during the scrolling period.

currentDistance: Current distance that camera is from the target.
calculatedDistance: The maximum calculated distance from the target at this point.
desiredDistance: The distance we want to move towards.
public Single CalculateDistanceFromTarget(System.Single currentDistance, System.Single calculatedDistance, System.Single desiredDistance)

Adds an amount of time to the remaining travel time of the zoom. The amount of time added is a percentage of the speed of the camera's current zooming. The provided delta is between [0,1].

delta: Percentage of the current speed to add to the travel time of the camera.
public Void AddPercentageToTimeRemaining(System.Single delta)

Modifies the Zoom Out Speed to the given value.

value: The speed value you want to set the zoom out speed to.
public Void SetZoomOutSpeed(System.Single value)

Modifies the Zoom In Speed to the given value.

value: The speed value you want to set the zoom in speed to.
public Void SetZoomInSpeed(System.Single value)

Copies the given Camera Component's public fields to this Camera Components public fields. The Camera Component given must be the same type as this Camera Component. The purpose of this method is to allow Camera Controllers to swap all their public variables of their components with the fields from a given component. This is useful for when you need to switch camera controller's and want to preserve the settings of individual components.

component: The component whose properties you want. This must be the same type of component as the component you are setting it to.
public Void CopyPublicFields(AdvancedUtilities.Cameras.Components.CameraComponent component)

AdvancedUtilities.Cameras.Components

The mode that the follow target component uses.

public enum FollowRotationMode

Fields

Int32: value__
FollowRotationMode: AdjustCompletelyOnRotation
FollowRotationMode: AdjustCompletelyOnMovement
FollowRotationMode: AdjustWhileMoving

AdvancedUtilities.Cameras.Components

The different modes that the MultipleRelativeTargetComponent can use.

public enum MultipleRelativeTargetSwitchMode

Fields

Int32: value__
MultipleRelativeTargetSwitchMode: Closest
MultipleRelativeTargetSwitchMode: Furthest
MultipleRelativeTargetSwitchMode: Random

AdvancedUtilities.Cameras.Components.Events

An interface that allows a class to listen to the horizontal rotation event fired from a RotationComponent.

public interface HorizontalDegreesListener

Methods

Tells the listener the amount of degrees the RotationComponent has rotated by.

degrees: Amount of degrees rotated by.

public Void DegreesRotated(System.Single degrees)

AdvancedUtilities.Cameras.Components.Events

An interface that allows a class to listen to the vertical rotation event fired from a RotationComponent.

public interface VerticalDegreesListener

Methods

Tells the listener the amount of degrees the RotationComponent has rotated by.

degrees: Amount of degrees rotated by.

public Void DegreesRotated(System.Single degrees)

AdvancedUtilities.Cameras.Components.Enums

An enum representing the types of screen shaking.

public enum ScreenShakeMode

Fields

Int32: value__
ScreenShakeMode: HorizontalAndVertical
ScreenShakeMode: Horizontal
ScreenShakeMode: Vertical

AdvancedUtilities.Cameras.Components

Encapsulates input sensitivity for the InputComponent.

public class InputSensitivity

Fields

A value that adjusts the sensitivity of input meant to rotate the camera horizontally.
public Single Horizontal

A value that adjusts the sensitivity of input meant to rotate the camera veritcally.
public Single Vertical

A value that adjusts the sensitivity of input meant to zoom the camera in.
public Single ZoomIn

A value that adjusts the sensitivity of input meant to zoom the camera out.
public Single ZoomOut

Constructors

public InputSensitivity

No parameters.
No XML summary.

AdvancedUtilities.Cameras.Components

Encapsulates input inversion settings for the InputComponent.

public class InputInversion

Fields

Horizontal input values will be multiplied by -1f when processed.
public Boolean Horizontal

Vertical input values will be multiplied by -1f when processed.
public Boolean Vertical

ZoomIn input values will be multiplied by -1f when processed.
public Boolean ZoomIn

ZoomOut input values will be multiplied by -1f when processed.
public Boolean ZoomOut

Constructors

public InputInversion

No parameters.
No XML summary.

AdvancedUtilities.Cameras.Components

Encapsulates the rotation limits for the RotationComponent.

public class RotationLimits

Fields

Limits for vertical rotation will be enforced.
public Boolean EnableVerticalLimits

The limit that the Camera can rotate vertically above the target. Generally this should be a positive value.
public Single VerticalUp

The limit that the Camera can rotate vertically below the target. Generally this should be a negative value.
public Single VerticalDown

Limits for horizontal rotation will be enforced.
public Boolean EnableHorizontalLimits

The limit that the Camera can rotate horizontally to the left of the target. Generally this should be a positive value.
public Single HorizontalLeft

The limit that the Camera can rotate horizontally to the right of the target. Generally this should be a negative value.
public Single HorizontalRight

Constructors

public RotationLimits

No parameters.
No XML summary.

AdvancedUtilities.Cameras.Components

Encapsulates information on horizontal rotation degrees events.

public class RotationHorizontalDegreesEvent

Fields

Enables the rotation component to fire events every time the horizontal has rotated a given amount of degrees in either direction.
public Boolean Enabled

Everytime the camera rotates the given amount in degrees, it will fire an event to all listeners registered to it with the amount it rotated.
public Single DegreesTrigger

If true, everytime the event is fired, it will reset tracking during that update to having rotated 0 degrees.
If set to false, if you rotate 135 degrees and fire an event every 90 degrees, then you will fire an event and be left with 45 degrees.
Technically multiple events can fire in a single update if this is set to false you rotate in a multiple of your event degrees value.
Only really useful if you have a low trigger or massive rotations.
public Boolean ResetTotalAfterEachEvent

A list of all the rotation listeners currently listening for rotation events.
public IList`1 RotationEventListeners

Constructors

public RotationHorizontalDegreesEvent

No parameters.
No XML summary.

AdvancedUtilities.Cameras.Components

Encapsulates information on vertical rotation degrees events.

public class RotationVerticalDegreesEvent

Fields

Enables the rotation component to fire events every time the vertical has rotated a given amount of degrees in either direction.
public Boolean Enabled

Everytime the camera rotates the given amount in degrees, it will fire an event to all listeners registered to it with the amount it rotated.
public Single DegreesTrigger

If true, everytime the event is fired, it will reset tracking during that update to having rotated 0 degrees.
If set to false, if you rotate 135 degrees and fire an event every 90 degrees, then you will fire an event and be left with 45 degrees.
Technically multiple events can fire in a single update if this is set to false you rotate in a multiple of your event degrees value.
Only really useful if you have a low trigger or massive rotations.
public Boolean ResetTotalAfterEachEvent

A list of all the rotation listeners currently listening for rotation events.
public IList`1 RotationEventListeners

Constructors

public RotationVerticalDegreesEvent

No parameters.
No XML summary.

AdvancedUtilities.Cameras.Components

Encapsulates information related to automatically rotating the camera over a period of time.

public class RotationAutoRotate

Fields

Whether or not the auto rotation will actually happen.
public Boolean Enabled

When the component is automatically rotating, the camera will not be able to be rotated with this component from external method calls.
public Boolean DisableExternalRotationWhileAutoRotating

If set to true, then when you call auto rotate, it will begin performing a new auto rotation instead of doing nothing and waiting for the first one to complete.
public Boolean AllowOverrides

Constructors

public RotationAutoRotate

No parameters.
No XML summary.

AdvancedUtilities.Cameras.Components

Encapsulates the settings for thickness checking on the ViewCollisionComponent.

public class ViewCollisionThicknessChecking

Fields

When enabled, the camera will perform many raycasts for each sampling point to try to determine if the objects it is hitting are below a thickness threshhold. If they are below the thickness threshhold, then they won't be considered collision.
public Boolean Enabled

Any object determined to be less thick than this in units will not be considered collision. Objects of the exact size will only be noticed if they are perfectly on the original point. It is recommended that you use a smaller size than the objects you are trying to exclude, so if you want to exclude objects 1 and greater, use 0.95f, or a similar value.
public Single ThicknessThreshhold

*The number of checks performed on each side of a raycast to determine it's accuracy. Each raycast sampling point that hits will test this number * 2 times to determine thickness. The accuracy of a check is actually 2^(this value), as the binary search used to determine width increases accuracy exponentially with more checks.*
public Int32 NumberOfChecks

The number of sets of checks. One set of checks would check both left and right of the camera view. Two sets of checks would check both left and right, and up and down. The more angles, the more accurately thickness can be determined at different camera angles.
public Int32 NumberOfAngles

Constructors

public ViewCollisionThicknessChecking

No parameters.
No XML summary.