using System;

using System.Collections;

using UnityEngine;

namespace UnityStandardAssets.Utility

{

[Serializable]

public class FOVKick

{

public Camera Camera; // optional camera setup, if null the main camera will be used

[HideInInspector] public float originalFov; // the original fov

public float FOVIncrease = 3f; // the amount the field of view increases when going into a run

public float TimeToIncrease = 1f; // the amount of time the field of view will increase over

public float TimeToDecrease = 1f; // the amount of time the field of view will take to return to its original size

public AnimationCurve IncreaseCurve;

public void Setup(Camera camera)

{

CheckStatus(camera);

Camera = camera;

originalFov = camera.fieldOfView;

}

private void CheckStatus(Camera camera)

{

if (camera == null)

{

throw new Exception("FOVKick camera is null, please supply the camera to the constructor");

}

if (IncreaseCurve == null)

{

throw new Exception(

"FOVKick Increase curve is null, please define the curve for the field of view kicks");

}

}

public void ChangeCamera(Camera camera)

{

Camera = camera;

}

public IEnumerator FOVKickUp()

{

float t = Mathf.Abs((Camera.fieldOfView - originalFov)/FOVIncrease);

while (t < TimeToIncrease)

{

Camera.fieldOfView = originalFov + (IncreaseCurve.Evaluate(t/TimeToIncrease)\*FOVIncrease);

t += Time.deltaTime;

yield return new WaitForEndOfFrame();

}

}

public IEnumerator FOVKickDown()

{

float t = Mathf.Abs((Camera.fieldOfView - originalFov)/FOVIncrease);

while (t > 0)

{

Camera.fieldOfView = originalFov + (IncreaseCurve.Evaluate(t/TimeToDecrease)\*FOVIncrease);

t -= Time.deltaTime;

yield return new WaitForEndOfFrame();

}

//make sure that fov returns to the original size

Camera.fieldOfView = originalFov;

}

}

}