using System;

using UnityEngine;

namespace UnityStandardAssets.Utility

{

[Serializable]

public class CurveControlledBob

{

public float HorizontalBobRange = 0.33f;

public float VerticalBobRange = 0.33f;

public AnimationCurve Bobcurve = new AnimationCurve(new Keyframe(0f, 0f), new Keyframe(0.5f, 1f),

new Keyframe(1f, 0f), new Keyframe(1.5f, -1f),

new Keyframe(2f, 0f)); // sin curve for head bob

public float VerticaltoHorizontalRatio = 1f;

private float m\_CyclePositionX;

private float m\_CyclePositionY;

private float m\_BobBaseInterval;

private Vector3 m\_OriginalCameraPosition;

private float m\_Time;

public void Setup(Camera camera, float bobBaseInterval)

{

m\_BobBaseInterval = bobBaseInterval;

m\_OriginalCameraPosition = camera.transform.localPosition;

// get the length of the curve in time

m\_Time = Bobcurve[Bobcurve.length - 1].time;

}

public Vector3 DoHeadBob(float speed)

{

float xPos = m\_OriginalCameraPosition.x + (Bobcurve.Evaluate(m\_CyclePositionX)\*HorizontalBobRange);

float yPos = m\_OriginalCameraPosition.y + (Bobcurve.Evaluate(m\_CyclePositionY)\*VerticalBobRange);

m\_CyclePositionX += (speed\*Time.deltaTime)/m\_BobBaseInterval;

m\_CyclePositionY += ((speed\*Time.deltaTime)/m\_BobBaseInterval)\*VerticaltoHorizontalRatio;

if (m\_CyclePositionX > m\_Time)

{

m\_CyclePositionX = m\_CyclePositionX - m\_Time;

}

if (m\_CyclePositionY > m\_Time)

{

m\_CyclePositionY = m\_CyclePositionY - m\_Time;

}

return new Vector3(xPos, yPos, 0f);

}

}

}