using UnityEngine;  
using System.Collections;  
using UnityEngine.UI;  
  
public class WalkingFilter : MonoBehaviour {  
  
 *// Use this for initialization* public Text x,y,z,vT;  
 public float lowerLiM=-1.3f;  
 public float upperLiM=-0.7f;  
 public float walkMinTime=1.5f;  
 public float speedFactor=0.25f;  
 public float jumpTresHold=-0.5f;  
 public Transform head;  
 public float JumpSpeed=2.5f;  
 public float timeJump=0.7f;  
 public float deltaDerivative=0.05f;  
 public float derivativeThreshold=15;  
  
 float acc\_j\_1,acc\_j;  
 public float Dacc;  
 public bool jumping;  
  
 float accX,accY,accZ;  
 Rigidbody RB;  
  
 float LOWtime;  
 float UPtime;  
 float elapsed;  
  
 float max=0;  
  
  
 void Start ()  
 {  
 UPtime=1000;  
 LOWtime=500;  
 RB=GetComponent<Rigidbody>();  
 elapsed=0;  
  
 Invoke("restartMax",2);  
  
  
 }  
   
 *// Update is called once per frame* void FixedUpdate ()   
 {  
  
 *//obtain accelerations* accX=Input.acceleration.x;  
 accY=Input.acceleration.y;  
 accZ=Input.acceleration.z;  
  
 *//display accelerations if wanted* x.text="ACC X="+accX;  
 y.text="ACC Y="+accY;  
 z.text="ACC Z="+accZ;  
  
  
 *//obtain derivative:* elapsed+=Time.fixedDeltaTime;  
  
  
 if(elapsed > deltaDerivative)   
 {  
 acc\_j=accY;  
  
 *//dericative* Dacc=Mathf.Abs( (acc\_j-acc\_j\_1)/elapsed);  
  
  
 max=Mathf.Max(Dacc,max);  
  
 vT.text="VAR Y="+max;  
  
 acc\_j\_1=accY;  
  
 elapsed=0;  
 }  
  
 *//check downstep* if(accY>upperLiM)  
 {   
 UPtime=Time.fixedTime;  
 }  
 else if(Time.fixedTime-UPtime>walkMinTime)  
 {  
 UPtime=1000;  
 }  
   
 *//check upstep* if(accY<lowerLiM)  
 {   
 LOWtime=Time.fixedTime;  
 }  
 else if(Time.fixedTime-UPtime>walkMinTime)  
 {  
 UPtime=500;  
 }  
  
  
 *//check jump* if( Mathf.Abs(UPtime-LOWtime)<walkMinTime && Dacc< derivativeThreshold)  
 {  
 move(1/Mathf.Abs(UPtime-LOWtime));  
 }  
  
 if(Dacc > derivativeThreshold && jumping==false)  
 {  
 jump();  
 jumping=true;  
 }  
  
  
 }  
  
  
 public void move(float v)  
 {  
 RB.MovePosition(transform.position+head.transform.forward\*v\*speedFactor);  
 }  
  
 public void jump()  
 {  
  
 RB.AddForce(new Vector3(0,JumpSpeed\*RB.mass,0));  
 Invoke("stopJump",timeJump);  
  
 }  
  
 public void stopJump()  
 {  
  
 *//RB.AddForce(JumpSpeed\*RB.mass/10\*head.forward);* jumping=false;  
  
 }  
  
  
 public void restartMax()  
 {  
 max=0;  
 }  
  
}