using System;

using UnityEngine;

using UnityStandardAssets.CrossPlatformInput;

using UnityStandardAssets.Utility;

using Random = UnityEngine.Random;

namespace UnityStandardAssets.Characters.FirstPerson

{

[RequireComponent(typeof (CharacterController))]

[RequireComponent(typeof (AudioSource))]

public class FirstPersonController : MonoBehaviour

{

[SerializeField] private bool m\_IsWalking;

[SerializeField] private float m\_WalkSpeed;

[SerializeField] private float m\_RunSpeed;

[SerializeField] [Range(0f, 1f)] private float m\_RunstepLenghten;

[SerializeField] private float m\_JumpSpeed;

[SerializeField] private float m\_StickToGroundForce;

[SerializeField] private float m\_GravityMultiplier;

[SerializeField] private MouseLook m\_MouseLook;

[SerializeField] private bool m\_UseFovKick;

[SerializeField] private FOVKick m\_FovKick = new FOVKick();

[SerializeField] private bool m\_UseHeadBob;

[SerializeField] private CurveControlledBob m\_HeadBob = new CurveControlledBob();

[SerializeField] private LerpControlledBob m\_JumpBob = new LerpControlledBob();

[SerializeField] private float m\_StepInterval;

[SerializeField] private AudioClip[] m\_FootstepSounds; // an array of footstep sounds that will be randomly selected from.

[SerializeField] private AudioClip m\_JumpSound; // the sound played when character leaves the ground.

[SerializeField] private AudioClip m\_LandSound; // the sound played when character touches back on ground.

private Camera m\_Camera;

private bool m\_Jump;

private float m\_YRotation;

private Vector2 m\_Input;

private Vector3 m\_MoveDir = Vector3.zero;

private CharacterController m\_CharacterController;

private CollisionFlags m\_CollisionFlags;

private bool m\_PreviouslyGrounded;

private Vector3 m\_OriginalCameraPosition;

private float m\_StepCycle;

private float m\_NextStep;

private bool m\_Jumping;

private AudioSource m\_AudioSource;

// Use this for initialization

private void Start()

{

m\_CharacterController = GetComponent<CharacterController>();

m\_Camera = Camera.main;

m\_OriginalCameraPosition = m\_Camera.transform.localPosition;

m\_FovKick.Setup(m\_Camera);

m\_HeadBob.Setup(m\_Camera, m\_StepInterval);

m\_StepCycle = 0f;

m\_NextStep = m\_StepCycle/2f;

m\_Jumping = false;

m\_AudioSource = GetComponent<AudioSource>();

m\_MouseLook.Init(transform , m\_Camera.transform);

}

// Update is called once per frame

private void Update()

{

RotateView();

// the jump state needs to read here to make sure it is not missed

if (!m\_Jump)

{

m\_Jump = CrossPlatformInputManager.GetButtonDown("Jump");

}

if (!m\_PreviouslyGrounded && m\_CharacterController.isGrounded)

{

StartCoroutine(m\_JumpBob.DoBobCycle());

PlayLandingSound();

m\_MoveDir.y = 0f;

m\_Jumping = false;

}

if (!m\_CharacterController.isGrounded && !m\_Jumping && m\_PreviouslyGrounded)

{

m\_MoveDir.y = 0f;

}

m\_PreviouslyGrounded = m\_CharacterController.isGrounded;

}

private void PlayLandingSound()

{

m\_AudioSource.clip = m\_LandSound;

m\_AudioSource.Play();

m\_NextStep = m\_StepCycle + .5f;

}

private void FixedUpdate()

{

float speed;

GetInput(out speed);

// always move along the camera forward as it is the direction that it being aimed at

Vector3 desiredMove = transform.forward\*m\_Input.y + transform.right\*m\_Input.x;

// get a normal for the surface that is being touched to move along it

RaycastHit hitInfo;

Physics.SphereCast(transform.position, m\_CharacterController.radius, Vector3.down, out hitInfo,

m\_CharacterController.height/2f);

desiredMove = Vector3.ProjectOnPlane(desiredMove, hitInfo.normal).normalized;

m\_MoveDir.x = desiredMove.x\*speed;

m\_MoveDir.z = desiredMove.z\*speed;

if (m\_CharacterController.isGrounded)

{

m\_MoveDir.y = -m\_StickToGroundForce;

if (m\_Jump)

{

m\_MoveDir.y = m\_JumpSpeed;

PlayJumpSound();

m\_Jump = false;

m\_Jumping = true;

}

}

else

{

m\_MoveDir += Physics.gravity\*m\_GravityMultiplier\*Time.fixedDeltaTime;

}

m\_CollisionFlags = m\_CharacterController.Move(m\_MoveDir\*Time.fixedDeltaTime);

ProgressStepCycle(speed);

UpdateCameraPosition(speed);

}

private void PlayJumpSound()

{

m\_AudioSource.clip = m\_JumpSound;

m\_AudioSource.Play();

}

private void ProgressStepCycle(float speed)

{

if (m\_CharacterController.velocity.sqrMagnitude > 0 && (m\_Input.x != 0 || m\_Input.y != 0))

{

m\_StepCycle += (m\_CharacterController.velocity.magnitude + (speed\*(m\_IsWalking ? 1f : m\_RunstepLenghten)))\*

Time.fixedDeltaTime;

}

if (!(m\_StepCycle > m\_NextStep))

{

return;

}

m\_NextStep = m\_StepCycle + m\_StepInterval;

PlayFootStepAudio();

}

private void PlayFootStepAudio()

{

if (!m\_CharacterController.isGrounded)

{

return;

}

// pick & play a random footstep sound from the array,

// excluding sound at index 0

int n = Random.Range(1, m\_FootstepSounds.Length);

m\_AudioSource.clip = m\_FootstepSounds[n];

m\_AudioSource.PlayOneShot(m\_AudioSource.clip);

// move picked sound to index 0 so it's not picked next time

m\_FootstepSounds[n] = m\_FootstepSounds[0];

m\_FootstepSounds[0] = m\_AudioSource.clip;

}

private void UpdateCameraPosition(float speed)

{

Vector3 newCameraPosition;

if (!m\_UseHeadBob)

{

return;

}

if (m\_CharacterController.velocity.magnitude > 0 && m\_CharacterController.isGrounded)

{

m\_Camera.transform.localPosition =

m\_HeadBob.DoHeadBob(m\_CharacterController.velocity.magnitude +

(speed\*(m\_IsWalking ? 1f : m\_RunstepLenghten)));

newCameraPosition = m\_Camera.transform.localPosition;

newCameraPosition.y = m\_Camera.transform.localPosition.y - m\_JumpBob.Offset();

}

else

{

newCameraPosition = m\_Camera.transform.localPosition;

newCameraPosition.y = m\_OriginalCameraPosition.y - m\_JumpBob.Offset();

}

m\_Camera.transform.localPosition = newCameraPosition;

}

private void GetInput(out float speed)

{

// Read input

float horizontal = CrossPlatformInputManager.GetAxis("Horizontal");

float vertical = CrossPlatformInputManager.GetAxis("Vertical");

bool waswalking = m\_IsWalking;

#if !MOBILE\_INPUT

// On standalone builds, walk/run speed is modified by a key press.

// keep track of whether or not the character is walking or running

m\_IsWalking = !Input.GetKey(KeyCode.LeftShift);

#endif

// set the desired speed to be walking or running

speed = m\_IsWalking ? m\_WalkSpeed : m\_RunSpeed;

m\_Input = new Vector2(horizontal, vertical);

// normalize input if it exceeds 1 in combined length:

if (m\_Input.sqrMagnitude > 1)

{

m\_Input.Normalize();

}

// handle speed change to give an fov kick

// only if the player is going to a run, is running and the fovkick is to be used

if (m\_IsWalking != waswalking && m\_UseFovKick && m\_CharacterController.velocity.sqrMagnitude > 0)

{

StopAllCoroutines();

StartCoroutine(!m\_IsWalking ? m\_FovKick.FOVKickUp() : m\_FovKick.FOVKickDown());

}

}

private void RotateView()

{

m\_MouseLook.LookRotation (transform, m\_Camera.transform);

}

private void OnControllerColliderHit(ControllerColliderHit hit)

{

Rigidbody body = hit.collider.attachedRigidbody;

//dont move the rigidbody if the character is on top of it

if (m\_CollisionFlags == CollisionFlags.Below)

{

return;

}

if (body == null || body.isKinematic)

{

return;

}

body.AddForceAtPosition(m\_CharacterController.velocity\*0.1f, hit.point, ForceMode.Impulse);

}

}

}