using System.Linq;

using UnityEngine;

public class PlayerController : MonoBehaviour

{

    // Attached to player object in Luksong Baka Scene

    // Manages controls of the player such as jumping, throws, animations, etc

    // Most variables here are assigned via reference

    // References from other scripts

    public PlayerPowerUp powerUpScript;

    public PlayerHealth healthScript;

    public BackgroundMusic backgroundMusicScript;

    public SpawnManager spawnManagerScript;

    public UIManagerInGame UIManagerScript;

    public GameManager gameManagerScript;

    public ShakeScreen shakeScreenScript;

    // Effects and Sounds

    public ParticleSystem explosionParticle;

    public ParticleSystem dirtParticle;

    public GameObject heartPickUpParticle;

    public AudioClip jumpSound;

    public AudioClip crashSound;

    public AudioClip damageSound;

    public AudioClip healSound;

    public AudioClip throwCowsSound;

    public AudioClip throwObstacleSound;

    private AudioSource sfxPlayer;

    // Animation

    public Animator playerAnim;

    private string[] idleAnimations = { "Idle\_WipeMouth", "Salute", "Idle\_CheckWatch" };

    private float endTimeOfAnimation;

    public float idleAnimationDelay;

    public float runningAnimationSpeed;

    public float modifiedRunningAnimationSpeed;

    public float runningSpeedMultiplier;

    public float dirtAnimationSpeed;

    public float modifiedDirtAnimationSpeed;

    public float dirtSpeedMultiplier;

    public float jumpingAnimationSpeed;

    public float deathAnimationSpeed;

    // Movement

    private Rigidbody playerRb;

    public float jumpForce;

    public Vector3 gravityConstant;

    public Vector3 introStartPosition;

    private bool isInIntro;

    private bool isOnGround;

    private bool hasDoubleJumped;

    // Projectiles

    public GameObject bombPrefab;

    public GameObject daggerPrefab;

    private Vector3 projectileSpawnOffset = new Vector3(1.5f, 1.5f);

-----SIEGFRED------------------------------

    void Start()

    {

        // Components of other objects

        spawnManagerScript = GameObject.Find("SpawnManager").GetComponent<SpawnManager>();

        UIManagerScript = GameObject.Find("UIManager").GetComponent<UIManagerInGame>();

        backgroundMusicScript = GameObject.FindGameObjectWithTag("Background Music").GetComponent<BackgroundMusic>();

        gameManagerScript = GameObject.Find("GameManager").GetComponent<GameManager>();

        shakeScreenScript = GameObject.Find("Main Camera").GetComponent<ShakeScreen>();

        sfxPlayer = GameObject.Find("SfxPlayer").GetComponent<AudioSource>();

        // Components of player

        playerRb = GetComponent<Rigidbody>();

        playerAnim = GetComponent<Animator>();

        powerUpScript = GetComponent<PlayerPowerUp>();

        healthScript = GetComponent<PlayerHealth>();

        // Set up gravity constant, then start the start the intro

        Physics.gravity = gravityConstant;

        SetupIntro();

    }

-----SHIN-------------------------

    // Called when on dash (either by left shift or strength powerup)

    // Speed up the running and dirt animation of the player if player is on ground

    public void SpeedUp()

    {

        gameManagerScript.playerIsDashing = true;

        modifiedRunningAnimationSpeed = runningAnimationSpeed \* runningSpeedMultiplier;

        modifiedDirtAnimationSpeed = dirtAnimationSpeed \* dirtSpeedMultiplier;

        if (isOnGround)

        {

            playerAnim.speed = modifiedRunningAnimationSpeed;

            var dirtParticleMain = dirtParticle.main;

            dirtParticleMain.simulationSpeed = modifiedDirtAnimationSpeed;

        }

    }

    // Called in start method via setup intro method, in transition to death method when game is over,

    // Called when releaseing left shift without strength powerup, unhold left shift when strength powerup is about to run out

    // Slows down the running and dirt animation of the player to normal if player is on ground

    public void SlowDown()

    {

        gameManagerScript.playerIsDashing = false;

        modifiedRunningAnimationSpeed = runningAnimationSpeed;

        modifiedDirtAnimationSpeed = dirtAnimationSpeed;

        if (isOnGround)

        {

            playerAnim.speed = modifiedRunningAnimationSpeed;

            var dirtParticleMain = dirtParticle.main;

            dirtParticleMain.simulationSpeed = modifiedDirtAnimationSpeed;

        }

    }

-----JED------------------------------

    // Called at start method

    // Place the player in the its starting position, slow down the player's animations, start intro animation and play bg music

    public void SetupIntro()

    {

        transform.position = introStartPosition;

        SlowDown();

        PerformIntro();

        isInIntro = true;

        backgroundMusicScript.PlayBackgroundMusic();

    }

    // Perform a random idle animation

    private void PerformIntro()

    {

        playerAnim.SetBool(GameManager.ANIM\_DEATH\_B, false);

        playerAnim.SetFloat(GameManager.ANIM\_SPEED\_F, 0);

        int index = Random.Range(0, idleAnimations.Length);

        playerAnim.Play(idleAnimations[index]);

        endTimeOfAnimation = Time.time + idleAnimationDelay;

    }

-----KENJI------------------------------

    // Called in update, checks if intro animation is finished

    private void CheckIntroProgress()

    {

        // If the idle animations is finished playing

        if (!playerAnim.GetCurrentAnimatorStateInfo(2).IsTag("Idle") && Time.time > endTimeOfAnimation)

        {

            // Start running animation

            playerAnim.SetFloat(GameManager.ANIM\_SPEED\_F, 1.0f);

            playerAnim.SetInteger(GameManager.ANIM\_INT, 0);

            // If already on running animation and its transitions are finished,

            // Then officially start the game by activating the player,

            // Start spawning, and set game state to not stopped and intro to be done

            if (playerAnim.GetCurrentAnimatorStateInfo(0).IsName("Run\_Static"))

            {

                ActivatePlayer();

                spawnManagerScript.StartSpawner();

                gameManagerScript.isGameStopped = false;

                isInIntro = false;

            }

        }

    }

    // Called when player is about to run after its idle animation

    // Set running and dirt animation to be normal, start running

    private void ActivatePlayer()

    {

        modifiedRunningAnimationSpeed = runningAnimationSpeed;

        modifiedDirtAnimationSpeed = dirtAnimationSpeed;

        dirtParticle.Play();

        TransitionToRunning();

    }

-----SIEGFRED------------------------------

    // Move the player if intro is done

    void Update()

    {

        if (isInIntro)

        {

            CheckIntroProgress();

            return;

        }

        MovePlayer();

    }

    // Called in update method, this is practically the update method if intro is done

    // Preforms commands base on user input

    private void MovePlayer()

    {

        // If user presses spacebar while on ground,

        // Or user presses spacebar, and is not on ground, and has has not yet double jumped

        // And game is not paused and not over

        if (((Input.GetKeyDown(KeyCode.Space) && isOnGround)

            || (Input.GetKeyDown(KeyCode.Space) && !isOnGround && !hasDoubleJumped))

            && !gameManagerScript.isGamePaused && !gameManagerScript.isGameStopped)

        {

            // Since conditionas are satisfied then let user jump

            if (!isOnGround)

            {

                PerformDoubleJump();

            }

            else

            {

                PerformJump();

            }

        }

        // If user press left shift and game is not over,

        // Or strength is activated and it is not yet about to run out

        // Then start dashing by calling speed up method

        if ((Input.GetKeyDown(KeyCode.LeftShift) && !gameManagerScript.isGameStopped)

            || (powerUpScript.powerUps["Strength"].isActivated && !powerUpScript.powerUpAboutToRunOut))

        {

            SpeedUp();

        }

        // If user releases left shift, then stop dashing by slowing down

        else if ((Input.GetKeyUp(KeyCode.LeftShift) && !gameManagerScript.isGameStopped))

        {

            SlowDown();

        }

        // If user press E while having powerup, then throw projectiles based on what powerup is activated

        if (Input.GetKeyDown(KeyCode.E) && powerUpScript.hasPowerUp && !gameManagerScript.isGameStopped && !gameManagerScript.isGamePaused)

        {

            // Throw bomb if bomb powerup is activated

            if (powerUpScript.powerUps["Bomb"].isActivated)

            {

                ThrowBomb();

            }

            // Throw dagger if dagger powerup is enabled

            else if (powerUpScript.powerUps["Dagger"].isActivated)

            {

                ThrowDagger();

            }

        }

    }

-----SHIN------------------------------

    // Called when user press spacebar while on ground

    // Use force impluse upward to simulate a movement

    // (Note: jump animation is played on on collision exit method)

    private void PerformJump()

    {

        playerRb.AddForce(Vector3.up \* jumpForce, ForceMode.Impulse);

    }

    // Called when user press spacebar when not ground, and is not yet on second jump

    // Jump and reset jump animation

    private void PerformDoubleJump()

    {

        PerformJump();

        hasDoubleJumped = true;

        playerAnim.ResetTrigger(GameManager.ANIM\_JUMP\_TRIG);

        TransitionToJumping();

    }

    // Called when doing jump from perform double jump method and on exit collision method

    // Handles animation, sfx, effects, and play state when jumping

    private void TransitionToJumping()

    {

        playerAnim.SetBool(GameManager.STATIC\_B, true);

        playerAnim.speed = jumpingAnimationSpeed;

        isOnGround = false;

        dirtParticle.Stop();

        playerAnim.SetTrigger(GameManager.ANIM\_JUMP\_TRIG);

        sfxPlayer.PlayOneShot(jumpSound);

    }

-----JED------------------------------

    // Called when activating player at start of the game, and colliding with walkable objects

    // Handles animation, speed, and play state when running

    private void TransitionToRunning()

    {

        playerAnim.SetBool(GameManager.STATIC\_B, true);

        playerAnim.ResetTrigger(GameManager.ANIM\_JUMP\_TRIG);

        playerAnim.SetFloat(GameManager.ANIM\_SPEED\_F, 1);

        playerAnim.SetBool(GameManager.ANIM\_DEATH\_B, false);

        playerAnim.speed = modifiedRunningAnimationSpeed;

        isOnGround = true;

        hasDoubleJumped = false;

    }

    // Called when player lose all lives/game is over

    // Handles animation, speed, effects, sfx when dying/game over

    private void TransitionToDeath()

    {

        SlowDown();

        playerAnim.speed = deathAnimationSpeed;

        playerAnim.SetBool(GameManager.ANIM\_DEATH\_B, true);

        dirtParticle.Stop();

        explosionParticle.Play();

        sfxPlayer.PlayOneShot(crashSound);

    }

    // Called when pressing E while on bomb powerup, throws bomb

    private void ThrowBomb()

    {

        Instantiate(bombPrefab, transform.position + projectileSpawnOffset, Quaternion.identity);

    }

    // Called when pressing E while on dagger powerup, throws bomb, and reduce number of daggers left

    private void ThrowDagger()

    {

        Instantiate(daggerPrefab, transform.position + projectileSpawnOffset, daggerPrefab.transform.rotation);

        powerUpScript.ReduceDagger();

    }

-----KENJI------------------------------

private void OnCollisionEnter(Collision other)

    {

        // if player collided with walkable objects (ground and top of trailers), then start running

        if (other.gameObject.tag == GameManager.TAG\_WALKABLE && !gameManagerScript.isGameStopped && !isInIntro)

        {

            TransitionToRunning();

            // Only play the dirt particles when on ground (not when running on top of trucks)

            if (other.gameObject.name == GameManager.NAME\_GROUND)

            {

                dirtParticle.Play();

            }

        }

        // If player collided with an obstacle, then call colliding with obstacle method

        // (Note: Must not be truck, truck collision has a seperate script for its colider)

        if (other.gameObject.CompareTag(GameManager.TAG\_OBSTACLE) && other.gameObject.name != "TruckCollider")

        {

            CollidingWithObstacle(other.gameObject);

        }

        // If player hit the height limit then reset its velocity to let it free fall

        // (Note: jumping animation is kept due to on collision exit method)

        if (other.gameObject.CompareTag(GameManager.TAG\_HEIGHTLIMIT))

        {

            playerRb.velocity = Vector3.zero;

        }

    }

    // Called when player collided with a obstacle (including trucks from truck collision script)

    public void CollidingWithObstacle(GameObject obstacle)

    {

        // If player has strength powerup upon collision

        if (powerUpScript.powerUps["Strength"].isActivated)

        {

            // Get the script of the obstacle and set isThrown to true

            // (setting it to true will stop move left movement and enable move top right movement)

            MoveLeft moveLeftScript = obstacle.gameObject.GetComponent<MoveLeft>();

            moveLeftScript.isThrown = true;

            // Add score

            gameManagerScript.IncreaseScore(1);

            // Play different sfx depending if it the object thrown is a cow or not

            if (gameManagerScript.NAME\_COWS.Concat(gameManagerScript.NAME\_CALVES).Contains(obstacle.gameObject.name))

            {

                sfxPlayer.PlayOneShot(throwCowsSound);

            }

            else

            {

                sfxPlayer.PlayOneShot(throwObstacleSound);

            }

        }

        // If player didn't have strength powerup on collision, then take damage, decrease score, and other effects

        else

        {

            healthScript.TakeDamage();

            shakeScreenScript.StartShaking();

            sfxPlayer.PlayOneShot(damageSound);

            gameManagerScript.IncreaseScore(-3);

            if (healthScript.currentLives == 0)

            {

                GameOver();

            }

        }

    }

    // Called when player has 0 lives

    // Manages game states, call game over from other script, stop bg music, start player death animation

-----SIEGFRED-------------------------

    public void GameOver()

    {

        gameManagerScript.isGameStopped = true;

        TransitionToDeath();

        powerUpScript.TurnOffPowerUp();

        spawnManagerScript.GameOver();

        UIManagerScript.GameOver();

        backgroundMusicScript.StopBackgroundMusic();

        playerAnim.SetFloat(GameManager.ANIM\_SPEED\_F, 0);

    }

    private void OnCollisionExit(Collision other)

    {

        // If player exits collision with a walkable object, then play jump animation

        if ((other.gameObject.tag == GameManager.TAG\_WALKABLE) &&

            !gameManagerScript.isGameStopped)

        {

            TransitionToJumping();

        }

    }

-----SHIN-------------------------

    private void OnTriggerEnter(Collider other)

    {

        // If player picked up a powerup, then enable it

        if (other.gameObject.CompareTag(GameManager.TAG\_POWERUP))

        {

            powerUpScript.EnablePowerUp(other.gameObject);

        }

        // if player picked up a heart, then add heal player, and play other effects

        else if (other.CompareTag(GameManager.TAG\_HEART))

        {

            Instantiate(heartPickUpParticle, other.gameObject.transform.position, Quaternion.identity);

            Destroy(other.gameObject);

            healthScript.Heal();

            sfxPlayer.PlayOneShot(healSound, 2.0f);

        }

    }

}