**CameraFollow**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraFollow : MonoBehaviour {

public Transform target;

public Transform leftBounds;

public Transform rightBounds;

public float smoothDampTime = 0.15f;

private Vector3 smoothDampVelocity = Vector3.zero;

private float camWidth, camHeight, levelMinX, levelMaxX;

// Use this for initialization

void Start ()

{

camHeight = Camera.main.orthographicSize \* 2;

camWidth = camHeight \* Camera.main.aspect;

float leftBoundWidth = leftBounds.GetComponentInChildren<SpriteRenderer> ().bounds.size.x / 2;

float rightBoundWidth = rightBounds.GetComponentInChildren<SpriteRenderer>().bounds.size.x / 2;

levelMinX = leftBounds.position.x + leftBoundWidth + (camWidth / 2);

levelMaxX = rightBounds.position.x - leftBoundWidth - (camWidth / 2);

}

// Update is called once per frame

void Update ()

{

if (target)

{

float targetX = Mathf.Max(levelMinX, Mathf.Min(levelMaxX, target.position.x));

float x = Mathf.SmoothDamp(target.position.x, targetX, ref smoothDampVelocity.x, smoothDampTime);

transform.position = new Vector3(x, transform.position.y, transform.position.z);

}

}

}

**Player**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Player : MonoBehaviour {

public float jumpVelocity;

public float bounceVelocity;

public Vector2 velocity;

public float gravity;

public LayerMask wallMask;

public LayerMask floorMask;

private bool walk, walk\_left, walk\_right, jump;

public enum PlayerState

{

jumping,

idle,

walking,

bouncing

}

private PlayerState playerState = PlayerState.idle;

private bool grounded = false;

private bool bounce = false;

// Use this for initialization

void Start ()

{

//Fall();

}

// Update is called once per frame

void Update ()

{

CheckPlayerInput();

UpadatePlayerPosition();

UpdateAnimationStates();

}

void UpadatePlayerPosition()

{

Vector3 pos = transform.localPosition;

Vector3 scale = transform.localScale;

if (walk)

{

if (walk\_left)

{

pos.x -= velocity.x \* Time.deltaTime;

scale.x = -1;

}

if (walk\_right)

{

pos.x += velocity.x \* Time.deltaTime;

scale.x = 1;

}

pos = CheckWallRays(pos, scale.x);

}

if (jump && playerState != PlayerState.jumping)

{

playerState = PlayerState.jumping;

velocity = new Vector2(velocity.x, jumpVelocity);

}

if (playerState == PlayerState.jumping)

{

pos.y += velocity.y \* Time.deltaTime;

velocity.y -= gravity \* Time.deltaTime;

}

if (bounce && playerState != PlayerState.bouncing)

{

playerState = PlayerState.bouncing;

velocity = new Vector2(velocity.x, bounceVelocity);

}

if (playerState == PlayerState.bouncing)

{

pos.y += velocity.y \* Time.deltaTime;

velocity.y -= gravity \* Time.deltaTime;

}

if (velocity.y <= 0)

{

pos = CheckFloorRays(pos);

}

if (velocity.y >= 0)

{

pos = CheckCeilingRays(pos);

}

transform.localPosition = pos;

transform.localScale = scale;

}

void UpdateAnimationStates()

{

if (grounded && !walk && !bounce)

{

GetComponent<Animator>().SetBool("isJumping", false);

GetComponent<Animator>().SetBool("isRunning", false);

}

if (grounded && walk)

{

GetComponent<Animator>().SetBool("isJumping", false);

GetComponent<Animator>().SetBool("isRunning", true);

}

if (playerState == PlayerState.jumping)

{

GetComponent<Animator>().SetBool("isJumping", true);

GetComponent<Animator>().SetBool("isRunning", false);

}

}

void CheckPlayerInput()

{

bool input\_left = Input.GetKey(KeyCode.LeftArrow);

bool input\_right = Input.GetKey(KeyCode.RightArrow);

bool input\_space = Input.GetKeyDown(KeyCode.Space);

walk = input\_left || input\_right;

walk\_left = input\_left && !input\_right;

walk\_right = !input\_left && input\_right;

jump = input\_space;

}

Vector3 CheckWallRays (Vector3 pos, float direction)

{

Vector2 originTop = new Vector2(pos.x + direction \* .4f, pos.y + 1f - 0.2f);

Vector2 originMiddle = new Vector2(pos.x + direction \* .4f, pos.y);

Vector2 originBottom = new Vector2(pos.x + direction \* .4f, pos.y - 1f + 0.2f);

RaycastHit2D wallTop = Physics2D.Raycast(originTop, new Vector2(direction, 0), velocity.x \* Time.deltaTime, wallMask);

RaycastHit2D wallMiddle = Physics2D.Raycast(originMiddle, new Vector2(direction, 0), velocity.x \* Time.deltaTime, wallMask);

RaycastHit2D wallBottom = Physics2D.Raycast(originBottom, new Vector2(direction, 0), velocity.x \* Time.deltaTime, wallMask);

if (wallTop.collider != null || wallMiddle.collider != null || wallBottom.collider != null)

{

pos.x -= velocity.x \* Time.deltaTime \* direction;

}

return pos;

}

Vector3 CheckFloorRays (Vector3 pos)

{

Vector2 originLeft = new Vector2(pos.x - 0.5f + 0.2f, pos.y - 1f);

Vector2 originMiddle = new Vector2(pos.x, pos.y - 1f);

Vector2 originRight = new Vector2(pos.x + 0.5f - 0.2f, pos.y - 1f);

RaycastHit2D floorLeft = Physics2D.Raycast(originLeft, Vector2.down, velocity.y \* Time.deltaTime, floorMask);

RaycastHit2D floorMiddle = Physics2D.Raycast(originMiddle, Vector2.down, velocity.y \* Time.deltaTime, floorMask);

RaycastHit2D floorRight = Physics2D.Raycast(originRight, Vector2.down, velocity.y \* Time.deltaTime, floorMask);

if(floorLeft.collider != null || floorMiddle.collider != null || floorRight.collider != null)

{

RaycastHit2D hitRay = floorRight;

if (floorLeft)

{

hitRay = floorLeft;

}

else if (floorMiddle)

{

hitRay = floorMiddle;

}

else if (floorRight)

{

hitRay = floorRight;

}

if (hitRay.collider.tag == "Enemy")

{

bounce = true;

hitRay.collider.GetComponent<EnemyAI>().Crush();

}

playerState = PlayerState.idle;

grounded = true;

velocity.y = 0;

pos.y = hitRay.collider.bounds.center.y + hitRay.collider.bounds.size.y / 2 + 1;

}

else

{

if (playerState != PlayerState.jumping)

{

Fall();

}

}

return pos;

}

Vector3 CheckCeilingRays(Vector3 pos)

{

Vector2 originLeft = new Vector2(pos.x - 0.5f + 0.2f, pos.y + 1f);

Vector2 originMiddle = new Vector2(pos.x, pos.y + 1f);

Vector2 originRight = new Vector2(pos.x + 0.5f - 0.2f, pos.y + 1f);

RaycastHit2D ceilLeft = Physics2D.Raycast(originLeft, Vector2.up, velocity.y \* Time.deltaTime, floorMask);

RaycastHit2D ceilMiddle = Physics2D.Raycast(originMiddle, Vector2.up, velocity.y \* Time.deltaTime, floorMask);

RaycastHit2D ceilRight = Physics2D.Raycast(originRight, Vector2.up, velocity.y \* Time.deltaTime, floorMask);

if (ceilLeft.collider != null || ceilMiddle.collider != null || ceilRight.collider != null)

{

RaycastHit2D hitRay = ceilLeft;

if (ceilLeft)

{

hitRay = ceilLeft;

}

else if (ceilMiddle)

{

hitRay = ceilMiddle;

}

else if (ceilRight)

{

hitRay = ceilRight;

}

if (hitRay.collider.tag == "QuestionBlock")

{

hitRay.collider.GetComponent<QuestionBlock>().QuestionBounce();

}

pos.y = hitRay.collider.bounds.center.y - hitRay.collider.bounds.size.y / 2 - 1;

Fall();

}

return pos;

}

void Fall()

{

velocity.y = 0;

playerState = PlayerState.jumping;

bounce = false;

grounded = false;

}

}

**EnemyAI**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemyAI : MonoBehaviour {

public float gravity;

public Vector2 velocity;

public bool isWalkingLeft = true;

public LayerMask floorMask;

public LayerMask wallMask;

private bool grounded = false;

private bool shouldDie = false;

private float deathTimer = 0;

public float timeBeforeDestroy = 1.0f;

private enum EnemyState

{

walking,

falling,

dead

}

private EnemyState state = EnemyState.falling;

// Use this for initialization

void Start ()

{

enabled = false;

Fall();

}

// Update is called once per frame

void Update ()

{

UpdateEnemyPosition();

CheckCrushed();

}

public void Crush()

{

state = EnemyState.dead;

GetComponent<Animator>().SetBool("isCrushed", true);

GetComponent<Collider2D>().enabled = false;

shouldDie = true;

}

void CheckCrushed()

{

if (shouldDie)

{

if (deathTimer <= timeBeforeDestroy)

{

deathTimer += Time.deltaTime;

}

else

{

shouldDie = false;

Destroy(this.gameObject);

}

}

}

void UpdateEnemyPosition()

{

if (state != EnemyState.dead)

{

Vector3 pos = transform.localPosition;

Vector3 scale = transform.localScale;

if (state == EnemyState.falling)

{

pos.y += velocity.y \* Time.deltaTime;

velocity.y -= gravity \* Time.deltaTime;

}

if (state == EnemyState.walking)

{

if (isWalkingLeft)

{

pos.x -= velocity.x \* Time.deltaTime;

scale.x = -1;

}

else

{

pos.x += velocity.x \* Time.deltaTime;

scale.x = 1;

}

}

if (velocity.y <= 0)

{

pos = CheckGround(pos);

}

CheckWalls(pos, scale.x);

transform.localPosition = pos;

transform.localScale = scale;

}

}

Vector3 CheckGround(Vector3 pos)

{

Vector2 originLeft = new Vector2(pos.x - 0.5f + 0.2f, pos.y - 0.5f);

Vector2 originMiddle = new Vector2(pos.x, pos.y - 0.5f);

Vector2 originRight = new Vector2(pos.x + 0.5f - 0.2f, pos.y - 0.5f);

RaycastHit2D groundLeft = Physics2D.Raycast(originLeft, Vector2.down, velocity.y \* Time.deltaTime, floorMask);

RaycastHit2D groundMiddle = Physics2D.Raycast(originMiddle, Vector2.down, velocity.y \* Time.deltaTime, floorMask);

RaycastHit2D groundRight = Physics2D.Raycast(originRight, Vector2.down, velocity.y \* Time.deltaTime, floorMask);

if (groundLeft.collider != null || groundMiddle.collider != null || groundRight.collider != null)

{

RaycastHit2D hitRay = groundRight;

if (groundLeft)

{

hitRay = groundLeft;

}

else if (groundMiddle)

{

hitRay = groundMiddle;

}

else if (groundRight)

{

hitRay = groundRight;

}

if (hitRay.collider.tag == "Player")

{

Application.LoadLevel("GameOver");

}

state = EnemyState.walking;

grounded = true;

velocity.y = 0;

pos.y = hitRay.collider.bounds.center.y + hitRay.collider.bounds.size.y / 2 + 0.5f;

}

else

{

if (state != EnemyState.falling)

{

Fall();

}

}

return pos;

}

void CheckWalls (Vector3 pos, float direction)

{

Vector2 originTop = new Vector2(pos.x + direction \* .4f, pos.y + 0.5f - 0.2f);

Vector2 originMiddle = new Vector2(pos.x + direction \* .4f, pos.y);

Vector2 originBottom = new Vector2(pos.x + direction \* .4f, pos.y - 0.5f + 0.2f);

RaycastHit2D wallTop = Physics2D.Raycast(originTop, new Vector2(direction, 0), velocity.x \* Time.deltaTime, wallMask);

RaycastHit2D wallMiddle = Physics2D.Raycast(originMiddle, new Vector2(direction, 0), velocity.x \* Time.deltaTime, wallMask);

RaycastHit2D wallBottom = Physics2D.Raycast(originBottom, new Vector2(direction, 0), velocity.x \* Time.deltaTime, wallMask);

if (wallTop.collider != null || wallMiddle.collider != null || wallBottom.collider != null)

{

RaycastHit2D hitRay = wallTop;

if (wallTop)

{

hitRay = wallTop;

}

else if (wallMiddle)

{

hitRay = wallMiddle;

}

else if (wallBottom)

{

hitRay = wallBottom;

}

if (hitRay.collider.tag == "Player")

{

Application.LoadLevel("GameOver");

}

isWalkingLeft = !isWalkingLeft;

}

}

private void OnBecameVisible()

{

enabled = true;

}

void Fall()

{

velocity.y = 0;

state = EnemyState.falling;

grounded = false;

}

}

**Castle**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Castle : MonoBehaviour {

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.tag == "Player")

{

Application.LoadLevel("Win");

}

}

}

**Hole**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Hole : MonoBehaviour {

void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "Player")

{

Application.LoadLevel("GameOver");

}

}

}

**QuestionBlock**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class QuestionBlock : MonoBehaviour {

public float bounceHeight = 0.5f;

public float bounceSpeed = 4f;

public float coinMoveSpeed = 8f;

public float coinMoveHeight = 3f;

public float coinFallDistance = 2f;

private Vector2 originalPosition;

public Sprite emptyBlockSprite;

private bool canBounce = true;

// Use this for initialization

void Start ()

{

originalPosition = transform.localPosition;

}

public void QuestionBounce()

{

if (canBounce)

{

canBounce = false;

StartCoroutine(Bounce());

}

}

// Update is called once per frame

void Update ()

{

}

void ChangeSprite()

{

GetComponent<Animator>().enabled = false;

GetComponent<SpriteRenderer>().sprite = emptyBlockSprite;

}

void PresentCoin()

{

GameObject spinningCoin = (GameObject)Instantiate(Resources.Load("Prefabs/Spinning\_Coin", typeof(GameObject)));

spinningCoin.transform.SetParent(this.transform.parent);

spinningCoin.transform.localPosition = new Vector2(originalPosition.x, originalPosition.y + 1);

StartCoroutine(MoveCoin(spinningCoin));

}

IEnumerator Bounce()

{

ChangeSprite();

PresentCoin();

while (true)

{

transform.localPosition = new Vector2(transform.localPosition.x, transform.localPosition.y + bounceSpeed \* Time.deltaTime);

if (transform.localPosition.y >= originalPosition.y + bounceHeight)

{

break;

}

yield return null;

}

while (true)

{

transform.localPosition = new Vector2(transform.localPosition.x, transform.localPosition.y - bounceSpeed \* Time.deltaTime);

if (transform.localPosition.y <= originalPosition.y)

{

transform.localPosition = originalPosition;

break;

}

yield return null;

}

}

IEnumerator MoveCoin(GameObject coin)

{

while (true)

{

coin.transform.localPosition = new Vector2(coin.transform.localPosition.x, coin.transform.localPosition.y + coinMoveSpeed \* Time.deltaTime);

if (coin.transform.localPosition.y >= originalPosition.y + coinMoveHeight + 1)

{

break;

}

yield return null;

}

while (true)

{

coin.transform.localPosition = new Vector2(coin.transform.localPosition.x, coin.transform.localPosition.y - coinMoveSpeed \* Time.deltaTime);

if (coin.transform.localPosition.y <= originalPosition.y + coinFallDistance + 1)

{

Destroy(coin.gameObject);

break;

}

yield return null;

}

}

}