**Camera Controller**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class CameraController : MonoBehaviour {

//Camera Movement

public float panSpeed = 30f;

public float panBorder = 2f;

private bool move = true;

//Scrolling

private float scrollSpeed = 2f;

public float minY = 10f;

public float maxY = 15f;

public Slider zoomSlider;

public Camera cams;

void Start() {

zoomSlider = GetComponent<Slider>();

// zoomSlider.value = zoomSlider.maxValue;

}

void Update () {

if (Input.GetKeyDown(KeyCode.Escape))

move = !move;

/\* if (Input.GetKeyDown(KeyCode.W))

move = !move;

if (Input.GetKeyDown(KeyCode.S))

move = !move;

if (Input.GetKeyDown(KeyCode.D))

move = !move;

if (Input.GetKeyDown(KeyCode.A))

move = !move;

\*/

if (!move)

return;

if (Input.GetKey("w"))// ||Input.mousePosition.y >= Screen.height - panBorder)

{

transform.Translate(Vector3.forward \* panSpeed \* Time.deltaTime, Space.World);

if (transform.position.z >= -6f) {

Debug.Log("Stop Camera Forward!");

// move = !move;

}

}

if (Input.GetKey("s"))// || Input.mousePosition.y <= panBorder)

{

Debug.Log("Move Backward");

transform.Translate(Vector3.back \* panSpeed \* Time.deltaTime, Space.World);

if (transform.position.z <= -24f) {

Debug.Log("Stop Camera Backward!");

// move = !move;

}

}

if (Input.GetKey("d"))//|| Input.mousePosition.x >= Screen.width - panBorder)

{

transform.Translate(Vector3.right \* panSpeed \* Time.deltaTime, Space.World);

if (transform.position.x >= 4f) {

Debug.Log("Stop Camera Right!");

// move = !move;

}

}

if (Input.GetKey("a"))// || Input.mousePosition.x <= panBorder)

{

transform.Translate(Vector3.left \* panSpeed \* Time.deltaTime, Space.World);

if (transform.position.x <= -6f) {

Debug.Log("Stop Camera Left!");

// move = !move;

}

}

float scroll = Input.GetAxis("Mouse ScrollWheel");

Vector3 pos = transform.position;

pos.y -= scroll \* 1000 \* scrollSpeed \* Time.deltaTime;

pos.y = Mathf.Clamp(pos.y, minY, maxY);

transform.position = pos;

//cams.fieldOfView = zoomSlider.value;

}

}

**Camera Swipe**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraSwipe : MonoBehaviour {

private Touch initTouch = new Touch();

public Camera cam;

private float rotX = 0f;

private float rotY = 0f;

private Vector3 origRot;

public float rotSpeed = 0.5f;

public float dir = -1f;

void Start () {

origRot = cam.transform.eulerAngles;

rotX = origRot.x;

rotY = origRot.y;

}

void FixedUpdate () {

foreach (Touch t in Input.touches)

{

if (t.phase == TouchPhase.Began)

{

initTouch = t;

}

else if (t.phase == TouchPhase.Moved)

{

float deltaX = initTouch.position.x - t.position.x;

float deltaY = initTouch.position.y - t.position.y;

rotX -= deltaY \* Time.deltaTime \* rotSpeed \* dir;

rotY += deltaY \* Time.deltaTime \* rotSpeed \* dir;

rotX = Mathf.Clamp(rotX, -45f, 45f);

cam.transform.eulerAngles = new Vector3(rotX, rotY, 0f);

}

else if (t.phase == TouchPhase.Ended)

{

initTouch = new Touch();

}

}

}

}

**CameraZoomTouch**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraZoomTouch : MonoBehaviour {

public float perspectiveZoomSpeed = 0.5f; // The rate of change of the field of view in perspective mode.

public float orthoZoomSpeed = 0.5f; // The rate of change of the orthographic size in orthographic mode.

public Camera cam;

void Start()

{

cam = GetComponent<Camera>();

}

void Update()

{

// If there are two touches on the device...

if (Input.touchCount == 2)

{

// Store both touches.

Touch touchZero = Input.GetTouch(0);

Touch touchOne = Input.GetTouch(1);

// Find the position in the previous frame of each touch.

Vector2 touchZeroPrevPos = touchZero.position - touchZero.deltaPosition;

Vector2 touchOnePrevPos = touchOne.position - touchOne.deltaPosition;

// Find the magnitude of the vector (the distance) between the touches in each frame.

float prevTouchDeltaMag = (touchZeroPrevPos - touchOnePrevPos).magnitude;

float touchDeltaMag = (touchZero.position - touchOne.position).magnitude;

// Find the difference in the distances between each frame.

float deltaMagnitudeDiff = prevTouchDeltaMag - touchDeltaMag;

// If the camera is orthographic...

if (cam.orthographic)

{

// ... change the orthographic size based on the change in distance between the touches.

cam.orthographicSize += deltaMagnitudeDiff \* orthoZoomSpeed;

// Make sure the orthographic size never drops below zero.

cam.orthographicSize = Mathf.Max(cam.orthographicSize, 0.1f);

}

else

{

// Otherwise change the field of view based on the change in distance between the touches.

cam.fieldOfView += deltaMagnitudeDiff \* perspectiveZoomSpeed;

// Clamp the field of view to make sure it's between 0 and 180.

cam.fieldOfView = Mathf.Clamp(cam.fieldOfView, 30f, 65f);

}

}

}

}

**Turret**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Turret : MonoBehaviour {

public Transform target;

private Enemy targetEnemy;

[Header("Attributes")]

public float turretRange = 15f;

public float fireRate = 1f;

private float fireCountdown = 0f;

[Header("Set Up Field")]

public string enemyTag = "Enemy";

public Transform partToRotate;

[Header("Shooting")]

public Animator anim;

public GameObject bullet;

public Transform firePoint;

[Header("Laser")]

public bool uselaser = false;

public LineRenderer lineRenderer;

public int damageOverTime = 30;

public float slowdown = 0.5f;

public GameObject laserImpact;

void Start () {

InvokeRepeating("UpdateTarget", 0f, 0.5f);

}

void UpdateTarget() {

GameObject[] enemies = GameObject.FindGameObjectsWithTag(enemyTag);

float shortestDistance = Mathf.Infinity;

GameObject nearestEnemy = null;

foreach (GameObject enemy in enemies)

{

float enemyDistance = Vector3.Distance(transform.position, enemy.transform.position);

if (enemyDistance < shortestDistance) {

shortestDistance = enemyDistance;

nearestEnemy = enemy;

}

}

if (nearestEnemy != null && shortestDistance <= turretRange)

{

target = nearestEnemy.transform;

targetEnemy = nearestEnemy.GetComponent<Enemy>();

}

else {

target = null;

}

}

void Update () {

if (target == null) {

anim.SetBool("Fire", false);

if (uselaser) {

if (lineRenderer.enabled)

lineRenderer.enabled = false;

}

return;

}

LockOnTarget();

if (uselaser)

{

Laser();

}

else {

if (fireCountdown <= 0f) {

ShootEnemy();

fireCountdown = 1f / fireRate;

}

fireCountdown -= Time.deltaTime;

}

}

void LockOnTarget() {

Vector3 dir = target.position - transform.position;

Quaternion lookRotation = Quaternion.LookRotation(dir);

Vector3 rotation = Quaternion.Lerp(partToRotate.rotation, lookRotation, Time.deltaTime \* 10f).eulerAngles;

partToRotate.rotation = Quaternion.Euler(0f, rotation.y, 0f);

}

void Laser() {

targetEnemy.GetComponent<Enemy>().TakeDamage(damageOverTime \* Time.deltaTime);

targetEnemy.Slow(slowdown);

if (!lineRenderer.enabled)

{

lineRenderer.enabled = true;

lineRenderer.SetPosition(0, firePoint.position);

lineRenderer.SetPosition(1, target.position);

}

lineRenderer.SetPosition(0, firePoint.position);

lineRenderer.SetPosition(1, target.position);

Vector3 dir = firePoint.position - target.position;

laserImpact.transform.position = target.position + dir.normalized;

laserImpact.transform.rotation = Quaternion.LookRotation(dir);

}

void ShootEnemy() {

GameObject bulletFire = (GameObject) Instantiate(bullet, firePoint.position, firePoint.rotation);

Bullet bullets = bulletFire.GetComponent<Bullet>();

if (bullets != null) {

anim.SetBool("Fire", true);

bullets.Seek(target);

}

}

private void OnDrawGizmosSelected()

{

Gizmos.color = Color.red;

Gizmos.DrawWireSphere(transform.position, turretRange);

}

}