using System.Collections;

using UnityEngine;

using UnityEngine.UI;

public class GunController : MonoBehaviour

{

[Header("UI")]

public Text ammoText;

public Image crossHair;

public Image heatMeter;

public Sprite weaponImg;

public Image imgToChange;

public GameObject DamageTextPrefab;

[Header("Weapon Stats")]

public float range = 100f; //The range of the bullets in meters.

public int bulletsPerMag = 30; //Bullets in magazine.

public int bullets = 250;

public static int bulletsInReserves; //The bullets in backpack.

public int currentBullets; //Bullets left in mag.

public float fireRate = 0.1f; //Delay between shots.

public float damage = 20; //Damage delt to enemys.

public float impactForce;

public float increaceRate = 1.0f;

public int bulletsFired = 1;

public float minBulletSpread = 1;

public float maxBulletSpread = 6;

[Header("Shoot Mode")]

public ShootMode shootMode; //ShootMode name.

public enum ShootMode { Auto, Semi, Burst } //ShootMode enum.

[Header("Weapon Effects")]

public ParticleSystem muzzleFlash; //Muzzle flash particle system.

public GameObject Particles; //Bullet hit effect.

[Header("Bullet Shells")]

public GameObject bulletShell;

public Transform spawnLocation;

public float ejectDelay = 0.5f;

public float minEjectForce = 3.0f;

public float maxEjectForce = 7.0f;

[Header("Other")]

public Transform shootPoint; //The point that the Raycast is cast from.

public bool isSniper;

public bool isShotgun;

public bool isPumpShotgun;

public LayerMask layerMask = -1;

[Header("Audio")]

public AudioClip shootSound;

public AudioClip reload00Audio;

public AudioClip reload01Audio;

public AudioClip inspectAudio;

private bool IsAiming;

private float ejectForce;

private float fireTimer; //Time between shots. / Pause on reload, no ammo, etc.

private float burstTime;

private float bulletSpred;

private Animator anim;

private AudioSource audioSource;

private bool isReloading; //Is the weapon reloading?

private bool shootInput;

void OnEnable()

{

//Update when active state is changed.

UpdateAmmoText();

fireTimer = fireRate;

}

void Start()

{

bulletsInReserves = bullets;

anim = GetComponent<Animator>();

audioSource = GetComponent<AudioSource>();

currentBullets = bulletsPerMag;

UpdateAmmoText();

}

void Update()

{

bullets = bulletsInReserves;

CheckBulletSpread();

CheckBurstTime();

CalculateHeatMeter();

if (isSniper == true)

{

crossHair.enabled = false;

}

if (Input.GetButtonDown("Fire2") && !isReloading)

{

IsAiming = true;

}

if (Input.GetButtonUp("Fire2") && !isReloading)

{

IsAiming = false;

}

switch (shootMode)

{

case ShootMode.Auto:

shootInput = Input.GetButton("Fire1");

break;

case ShootMode.Semi:

shootInput = Input.GetButtonDown("Fire1");

break;

case ShootMode.Burst:

shootInput = Input.GetButtonDown("Fire1");

break;

}

if (shootInput)

{

if (currentBullets > 0)

{

burstTime += Time.deltaTime;

Fire(); //Execute Fire.

}

}

if (Input.GetKeyDown(KeyCode.R))

{

if (currentBullets == 0 && bulletsInReserves > 0)

DoReload();

if (currentBullets < bulletsPerMag && currentBullets >= 1 && bulletsInReserves > 0)

DoReload02();

if (currentBullets == bulletsPerMag || bulletsInReserves == 0)

{

anim.CrossFadeInFixedTime("Inspect", 0.01f);

audioSource.PlayOneShot(inspectAudio);

}

UpdateAmmoText();

}

if (fireTimer < fireRate)

fireTimer += Time.deltaTime;

UpdateAmmoText();

}

void FixedUpdate()

{

AnimatorStateInfo info = anim.GetCurrentAnimatorStateInfo(0);

isReloading = info.IsName("Reload");

anim.SetBool("Aim", IsAiming);

anim.SetBool("Aim", IsAiming);

ChangeUI();

}

private void Fire()

{

// If this or this = false.

if (fireTimer < fireRate || currentBullets <= 0 || isReloading)

return;

anim.CrossFadeInFixedTime("Fire", 0.07f); //Play the fire animation.

muzzleFlash.Play(); //Play muzzle flash.

StartCoroutine(BulletShellDelay());

audioSource.PlayOneShot(shootSound);

currentBullets--;

UpdateAmmoText();

fireTimer = 0.0f; //Reset fireTimer.

for (int i = 0; i < bulletsFired; i++)

{

RaycastHit hit;

Vector3 fireDir = shootPoint.forward;

Quaternion fireRot = Quaternion.LookRotation(fireDir);

Quaternion randomRot = Random.rotation;

fireRot = Quaternion.RotateTowards(fireRot, randomRot, Random.Range(0.0f, bulletSpred));

if (Physics.Raycast(shootPoint.position, fireRot \* Vector3.forward, out hit, range, layerMask))

{

Debug.Log(hit.transform.name + "found!");

GameObject hitParticlesEffect = Instantiate(Particles, hit.point, Quaternion.LookRotation(hit.normal));

if (hit.rigidbody != null)

{

hit.rigidbody.AddForce(-hit.normal \* impactForce);

}

if (hit.transform.GetComponent<BossPhases>())

{

hit.transform.GetComponent<BossPhases>().TakeDamage(damage);

}

if (hit.transform.GetComponentInChildren<AIController>())

{

hit.transform.GetComponentInChildren<AIController>().TakeDamage(damage);

GameObject go = Instantiate(DamageTextPrefab, hit.point, Quaternion.LookRotation(hit.normal));

go.GetComponentInChildren<TextMesh>().text = damage.ToString();

}

}

}

}

public void Reload()

{

if (bulletsInReserves <= 0)

return;

if (isPumpShotgun)

{

int bulletsToDeduct = 1;

bulletsInReserves -= bulletsToDeduct;

currentBullets += bulletsToDeduct;

}

else

{

int bulletsToLoad = bulletsPerMag - currentBullets;

int bulletsToDeduct = (bulletsInReserves >= bulletsToLoad) ? bulletsToLoad : bulletsInReserves;

bulletsInReserves -= bulletsToDeduct;

currentBullets += bulletsToDeduct;

}

}

private void DoReload()

{

AnimatorStateInfo info = anim.GetNextAnimatorStateInfo(0);

if (isReloading)

return;

anim.CrossFadeInFixedTime("Reload02", 0.01f);

audioSource.PlayOneShot(reload01Audio);

}

private void DoReload02()

{

AnimatorStateInfo info = anim.GetNextAnimatorStateInfo(0);

if (isReloading)

return;

anim.CrossFadeInFixedTime("Reload01", 0.01f);

audioSource.PlayOneShot(reload00Audio);

}

private void UpdateAmmoText()

{

ammoText.text = currentBullets + "/" + bulletsInReserves;

}

void CheckBulletSpread()

{

if (bulletSpred > maxBulletSpread)

{

bulletSpred = maxBulletSpread;

}

if (bulletSpred < minBulletSpread)

{

bulletSpred = minBulletSpread;

}

}

void CheckBurstTime()

{

if (!IsAiming)

{

bulletSpred = minBulletSpread + burstTime \* increaceRate;

}

else if (!isShotgun)

{

bulletSpred = minBulletSpread + burstTime \* increaceRate / 2;

}

if (!shootInput || currentBullets == 0)

{

burstTime -= Time.deltaTime;

}

if (burstTime < 0)

{

burstTime = 0;

}

}

void CalculateHeatMeter()

{

if (!isShotgun)

{

heatMeter.fillAmount = bulletSpred / maxBulletSpread;

heatMeter.enabled = true;

}

else

{

heatMeter.enabled = false;

}

}

void ChangeUI()

{

imgToChange.sprite = weaponImg;

}

IEnumerator BulletShellDelay()

{

yield return new WaitForSeconds(ejectDelay);

ejectForce = Random.Range(minEjectForce, maxEjectForce);

GameObject bullet = Instantiate(bulletShell, spawnLocation.position, Random.rotation);

Rigidbody rb = bullet.GetComponent<Rigidbody>();

rb.AddForce(transform.forward \* ejectForce, ForceMode.VelocityChange);

}

}