Tabla de contenido

Conceptos Unity

1. Hoja de sprites

Al agregar una hoja de sprites en la parte de edición se puede indicarle de cuantos pixeles está conformada cada sub-imagen; las imágenes también se pueden recortar de modo automático en la herramienta de “Sprite Editor” además es posible agregar márgenes y un padding al rededor. Si la imagen es borrosa se debe cambiar el “Filter Mode” a point en lugar de bilinear.

2. Project Settings

2.1. Input

GetAxisRaw permite obtener el valor que el input por defecto cambia según presionemos las flechas

Todo esto definido en input

2.1.1. Eventos de teclado

if(Input.GetKey (KeyCode.UpArrow))

3. Interactuar con las partes de un objeto: GetComponent

GetComponent<>()

```
public Color color1 = Color.red;
    public Color color2 = Color.blue;
    public float duration = 3.0F;
```

```
    Camera cam;
    bool ortho = false;

    public Transform objetivo;

    void Start()
    {
        cam = GetComponent<Camera>();
        cam.clearFlags = CameraClearFlags.SolidColor;

        //distancia del renderizado
        //cam.farClipPlane = 20.0f;
    }

    void Update()
    {

        //sigue con la mirada al objetivo debe ser asignado luego
        transform.LookAt (objetivo);



        if (Input.GetKeyDown (KeyCode.Alpha3))
        {
            ortho=!ortho; // este es un flag
            cam.orthographic = ortho;

        }

        if (Input.GetKeyDown (KeyCode.Alpha4))
        {
            //distancia del renderizado aumenta
            cam.farClipPlane++;
        }

        if (Input.GetKeyDown (KeyCode.Alpha5))
        {
            cam.farClipPlane--;
        }

        if (Input.GetKeyDown (KeyCode.KeypadPlus))
        {
            //la suma va en grados para la apertura de la vista de
la caamara
```

```
            cam.fieldOfView += 5;
        }

        if (Input.GetKeyDown (KeyCode.KeypadMinus))
        {
            cam.fieldOfView -= 5;
        }


        /*
        if (Input.GetKeyDown (KeyCode.Alpha1))
        {
            cam.backgroundColor = color1;
        }

        if (Input.GetKeyDown (KeyCode.Alpha2))
        {
            cam.backgroundColor = color2;
        }
        //float t = Mathf.PingPong(Time.time, duration) / duration
;
        //cam.backgroundColor = Color.Lerp(color1, color2, t);*/
```

4. Animator

Controla los clips de animaciones con máquinas de estado.

- animator.SetFloat ("moveX",Input.GetAxisRaw("Horizontal"));

Según las variables se producen transiciones ya que desde el código se puede cambiar la máquina de estado a ejecutarse

- "Fixed duration" marcado indica que no sea en bucle el movimiento, es necesario desmarcarlo en la mayoría de ocasiones.
- "Transition duration" es el tiempo que se demora en pasar de una animacion a otra, de preferencia para el 2d en movimiento ese valor debe ser 0

Animator animator = GetComponent<Animator> ();
animator.Play ("saltar",-1,0f);

4.1. Blend Trees

Es una forma organizacional de estados

5. Colisiones

Para que los objetos colisionen no basta con las “box colider 2D” es necesario “Rigidbody 2D”

- void OnTriggerEntered2D(Collider2D other){}
- void OnCollisionEntered2D(Collision2D other){}

También hay la versión Exit, Enter y la persistida

```
public void OnCollisionEnter(Collision obj_col){
        if (obj_col.gameObject.name == objetivo_desaparicion)
                Destroy (obj_paracaidas.gameObject, 2f);
        if (obj_col.gameObject.name == "barco_juego")
                Destroy (obj_paracaidas.gameObject); }
```

```
void OnTriggerEnter(Collider obj) //OnTriggerExit,OnTriggerStay
    {
        //transform.Translate (10, 0, 0);
        //Debug.Log ("Trigger {0}"+obj.gameObject.name);
    }

    void OnCollisionEnter(Collision obj)
    {

        if (obj.gameObject.name == "cubo3")
        {
            // para muchos objetos es mejor usar tags
            Debug.Log ("Colisiono con el tercero");
            Destroy (obj.gameObject);
        }

    }

    void OnControllerColliderHit(ControllerColliderHit obj_col)
    {
        if (obj_col.gameObject.name == "cubo1")
            Destroy (obj_col.gameObject);
    }
```

6. Capas

“Sorting layers” indica que objeto irá encima de otro.

Podemos apoyarnos en el vector z para esto pero no es muy aconsejable.

1. Instanciación de un objeto especifico a partir de uno genérico (prefab)
   - var clone = (GameObject) Instantiate (danio_flotante, col.gameObject.transform.position, Quaternion.Euler (Vector3.zero));
     clone.GetComponent<NumerosFlotantes> ().numero = damage;

El código de arriba clona el prefab para permitir la modificación de sus atributos

7. UI interfaces y Canvas

7.1. Texto flotante

Primero se lo aparta del canvas agregándole la opción “world space” de esa manera no se ata al UI. El texto se lo crea en una posición y se aumenta su eje y.

7.2. Barra de vida

Con un slider se puede crear una barra de vida, eliminando las partes innecesarias

Se necesitan dos sprites:

1.-Marco

2.-Barra con los colores

Si son imágenes necesitan ser convertidas a sprites de manera interna

Se agrega la jerarquía:

Canvas

Imagen (con ancho y alto del marco/Barra, componente -> scrollbar y agregamos la mascara de abajo en handle rect y modificamos Size y Value)

Imagen(Marco)

Mascara (con el componente Mask y sin la casilla marcada, center -> stretch)

Imagen(Imagen de la vida llena)

Lo que está más abajo es loque se muestra al frente, es por eso que marco debe ir debajo de la más cara asi se muestra de frente

```
private float vida_base = 100;
private float vida_actual;
public Scrollbar barra_vida;
```

```
public void reducirVida(float reducción){
        if (vida_actual >= reducción) {
            vida_actual -= reducción;
            barra_vida.size = vida_actual / vida_base;}}
```

8. Sistema de partículas

8.1. Sangre

Se usó la instanciación esférica, de color rojo, con tamaño reducido y creación de cuadros

9. NPC

9.1. Movimiento

Para limitar el movimiento de un npc en 2d lo encerramos en un box collider que solo detectará el mismo. También se puede usar la mecánica del 3D donde se usa EmptyObjects como target de una ruta (mirar el script del oso)

10. Composición (POO) en MonoBehaviour

Para realizar composición en lugar de "new" debemos agregar componentes

- PlayerStats estadisticas = gameObject.AddComponent <PlayerStats>();

11. Triggers

Los objetos vacíos pueden servir para activar cosas en el juego, esto es importante.

12. Audio

Para agregar un sonido rápidamente se puede usar un prefab que cree algo y arrastrando el clip de sonido al mismo sonara al ser instanciado.

Otra manera es crear un controlador (con varios audios sources) con la música ya cargada. Agregamos esos controladores a nuestra clase donde habrá sonido y simplemente con la función .Play podemos reproducir el efecto cuando deba suceder

13. Escenas

13.1. Cambio de escenas

Primero se cargan en BuildSettings todas las escenas a utilizar

Luego en código dentro de un objeto trigger se importa using UnityEngine.SceneManagement; y se usa SceneManager.LoadScene (level2load); siendo level2load el nombre de la escena

14. Persistencia de datos

DontDestroyOnLoad (transform.gameObject);

15. Comunicación entre scripts con SendMessage

```
//el objeto dañino


if (obj_col.gameObject.name == "jugador") {


                        obj_col.gameObject.SendMessageUpwards ("reducirVida",
10,SendMessageOptions.DontRequireReceiver);

                }
```

16. Desactivación en lugar de destrucción

Con esto evitamos errores y simplemente lo hacemos no jugable

```
col.gameObject.SetActive (false);
```

17. Transform

Para cambiar la posición local o rotación usamos

- this.transform.localPosition = new Vector3 (0.004f,-0.292f,0f);
- this.transform.eulerAngles = new Vector3(0f, 180f, -179.13f);

Importante en el rotation no es necesario usar Quaterniones

18. Búsqueda de objetos dentro del juego

```
GameObject.Find ("plataforma");
```

19. Timers

Uso de Time.deltaTime

20. Números aleatorios

```
Random.Range(a,b);
```

21. Instanciación de objetos

```
Instantiate (objeto, new Vector3(x,y,z), transform.rotation);
```

22. Rigidbody y fuerzas que interactúan con él

```
RigidBody rb = GetComponent<Rigidbody> ();

//Hay 4 distintas fuerzas que se pueden agregar a un rigidbody

rb.AddForce (new Vector3 (x*Time.deltaTime,0,0), ForceMode.Force);

relativeForce
```

23. GUI

```
//mensajes en pantalla

void OnGUI(){
        GUI.Label (new Rect (10, 60, 140, 20), "Cajas recogidas: " + cajas_recogidas.ToString ());
        GUI.Label (new Rect (150, 60, 150, 20), "Vidas: " + vidas_barco.ToString ());}


void OnGUI()    {
float x = transform.position.x, y = transform.position.y, z = transform.position.z;
float xr = transform.rotation.eulerAngles.x, yr = transform.rotation.eulerAngles.y, zr =
transform.rotation.eulerAngles.z;

GUI.Label (new Rect (10, 10, 200, 30), "Hola " + texto);
GUI.Label (new Rect (10, 50, 120, 30), x.ToString () + "," + y.ToString () + "," + z.ToString ());

GUI.Box (new Rect (10, 90, 120, 30), xr.ToString () + "," + (Mathf.Round (yr)).ToString () + "," +
zr.ToString ());
```

```
if (GUI.Button (new Rect (40, 140, 60, 30), "Arriba"))
        transform.Translate (0, avance, 0);
if (GUI.Button (new Rect (10, 170, 60, 30), "Izquierda"))
        transform.Translate (-avance, 0, 0);

if (GUI.Button (new Rect (70, 170, 60, 30), "Derecha"))
        transform.Translate (avance, 0, 0);

if (GUI.Button (new Rect (40, 200, 60, 30), "Abajo"))
        transform.Translate (0, -avance, 0);
texto = GUI.TextField (new Rect (250, 10, 200, 30), texto, 25);
rot = GUI.HorizontalSlider (new Rect (10, 230, 100, 30), rot, -45.0f, 45.0f);
transform.Rotate (0, rot, 0);
movimiento = GUI.Toggle (new Rect (10, 250, 100, 30), movimiento, "Permitir avance");
if (movimiento)
        avance = 1;
else
        avance = 0;
//RepeatButton() Window()}
```

24. Disparos con raycast

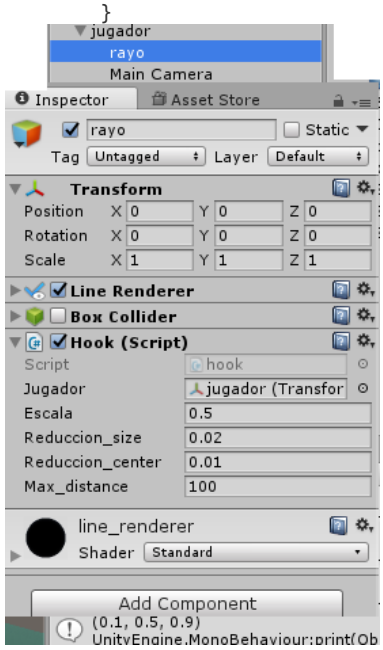
```
//mecánica de disparo

if (Input.GetKeyDown (KeyCode.Alpha1)) {
        if(Physics.Raycast( new Vector3(transform.position.x, pos_y , transform.position.z), -
transform.forward , out fin_rayo, max_distance)){
                line.enabled = true;
                line.SetPosition (1, fin_rayo.point);
                if (fin_rayo.collider.gameObject.name == "tiburon_prefab(Clone)") {
                        fin_rayo.collider.transform.Translate (transform.forward);
                        Destroy (fin_rayo.collider.gameObject);
                        line.enabled = false;}}}
```

24.1. Grappling hook basic

```
private LineRenderer line_renderer;
    private float counter, objetivo_dist;
    GameObject cam;

    public Transform jugador;
```

```
public float escala = .5f, reduccion_size = .3f, reduccion_center = .3f, max_distance = 20f;



//BoxCollider box_collider;
public static Vector3 hand, objetivo_dir, objetivo_punto;
public static RaycastHit hit;
public static bool enganche, press=false;

void Awake()
{
    line_renderer = GetComponent<LineRenderer> ();
    cam = GameObject.Find ("Main Camera");
}


void Start () {


    enganche = false;

}


void Update () {
    print (cam.transform.forward);

    if(Input.GetMouseButton(0))
    {
        if(enganche == false)
            buscar_punto ();
        if(enganche == true)
            mostrar_linea ();
        press = true;

    }
    else
    {
        press = false;
        mostrar_linea ();
    }






}

float x;

void buscar_punto()
{
    print ("buscando punto");

    if ( Physics.Raycast(jugador.position, cam.transform.forward, out hit, max_distance))
    {
        objetivo_dir = cam.transform.forward;
        objetivo_punto = hit.point;
        enganche = true;
    }
    else
    {
        print ("gancho perdido");
    }
```

```
}

void mostrar_linea()
{
    if(press)
    {
        hand = new Vector3 (jugador.position.x, jugador.position.y,
            jugador.position.z + jugador.localScale.z );

        //box_collider.center = new Vector3(box_collider.center.x, box_collider.center.y,
        //    box_collider.center.z+(x * reduccion_center));
        //box_collider.size = new Vector3(box_collider.size.x, box_collider.size.y,
        //    box_collider.size.z+(x * reduccion_size));

        line_renderer.enabled = true;
        line_renderer.SetPosition (0, jugador.position);
        line_renderer.startWidth = .1f;
        line_renderer.endWidth = .1f;


        counter += (escala * Time.deltaTime);


        //x = Mathf.Lerp (0.1f,  Vector3.Distance( jugador.position , hit.point), counter);

        line_renderer.SetPosition (1, objetivo_punto );



        //Physics.Raycast (hand, objetivo_dir, max_distance);
    }

    if(Vector3.Distance( jugador.position , hit.point) < 5f || !press)
    {
        enganche = false;
        line_renderer.enabled = false;
        counter = 0.1f;
    }
```

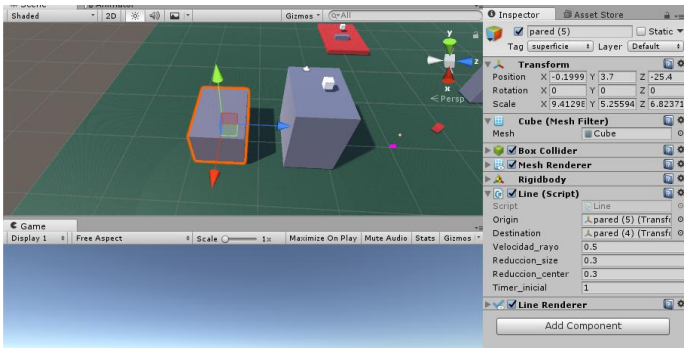


25. Line renderer 3d

CREAR ANIMACION DE linea script

```
private LineRenderer line_renderer;
    private float counter,dist;

    public Transform origin, destination;

    private float line_draw_speed;
    public float velocidad_rayo = .5f, reduccion_size = .3f, reduccion_center = .3f;
    float x;
    bool linea_visible = false;
    BoxCollider caja;
    Vector3 origen_collider_center;

    float timer;
    public float timer_inicial = 6f;
    void Start () {
        line_renderer = GetComponent<LineRenderer> ();

        line_draw_speed = velocidad_rayo * Time.deltaTime;
        caja = GetComponent<BoxCollider> ();
        timer = timer_inicial;
    }


    void Update () {
        if(timer > 0)
            timer -= Time.deltaTime;
        if(timer < 0 )//Input.GetMouseButtonDown(0))
        {
            timer = 0;
            origen_collider_center = caja.center;
            linea_visible = true;
        }



        if (linea_visible)
            mostrar_linea ();

    }






    void mostrar_linea()
    {

        caja.center = new Vector3(caja.center.x, caja.center.y, caja.center.z+(x * reduccion_center)
);
        caja.size = new Vector3(caja.size.x, caja.size.y, caja.size.z+(x * reduccion_size));

        line_renderer.enabled = true;
        line_renderer.SetPosition (0, origin.position);
        line_renderer.startWidth = .1f;
        line_renderer.endWidth = .1f;
        dist = Vector3.Distance (origin.position, destination.position);

        if (counter <= 1f)
        {
            counter += (line_draw_speed);
```

```
            x = Mathf.Lerp (0, dist, counter);

            Vector3 point_a = origin.position;
            Vector3 point_b = destination.position;

            Vector3 point_a_long_line = x * Vector3.Normalize (point_b - point_a) + point_a;

            line_renderer.SetPosition (1, point_a_long_line);
        }
        else
        {
            linea_visible = false;
            line_renderer.enabled = false;
            counter = 0;
            caja.center = origen_collider_center;
            caja.size = new Vector3(0.5f,0.5f,0f);
            timer = timer_inicial;
        }


    }
```



26. Cursor

```
CursorLockMode wantedMode;

    // Apply requested cursor state
    void SetCursorState()
    {
        Cursor.lockState = wantedMode;
        // Hide cursor when locking
        Cursor.visible = (CursorLockMode.Locked != wantedMode);
    }

    void OnGUI()
    {
        GUILayout.BeginVertical();
        // Release cursor on escape keypress
        if (Input.GetKeyDown(KeyCode.Escape))
            Cursor.lockState = wantedMode = CursorLockMode.None;

        switch (Cursor.lockState)
        {
        case CursorLockMode.None:
            GUILayout.Label("Cursor is normal");
            if (GUILayout.Button("Lock cursor"))
                wantedMode = CursorLockMode.Locked;
```

```
            if (GUILayout.Button("Confine cursor"))
                wantedMode = CursorLockMode.Confined;
            break;
        case CursorLockMode.Confined:
            GUILayout.Label("Cursor is confined");
            if (GUILayout.Button("Lock cursor"))
                wantedMode = CursorLockMode.Locked;
            if (GUILayout.Button("Release cursor"))
                wantedMode = CursorLockMode.None;
            break;
        case CursorLockMode.Locked:
            GUILayout.Label("Cursor is locked");
            if (GUILayout.Button("Unlock cursor"))
                wantedMode = CursorLockMode.None;
            if (GUILayout.Button("Confine cursor"))
                wantedMode = CursorLockMode.Confined;
            break;
        }

        GUILayout.EndVertical();

        SetCursorState();
```

```
Cursor.lockState = CursorLockMode.Locked;
        print ("width: "+ Screen.width.ToString()+", height: "+Scr
een.height.ToString());
```

27. Luces

```
public Light luz_point;
    Color mi_color1 = Color.red;
    Color mi_color2 = Color.green;
    public float duration = 3.0F;
    // Use this for initialization
    void Start ()
    {

    }

    // Update is called once per frame
    void Update ()
    {

        float t = Mathf.PingPong(Time.time, duration) / duration
;
        if (Input.GetKeyDown (KeyCode.Alpha1))
        {
            Debug.Log (t);
        }
        luz_point.color = Color.Lerp (mi_color1, mi_color2, t);
```

```
        }
```

28. Draw Lines y Gizmos

```
void OnDrawGizmosSelected()
    {
        if(objetivo)
        {
            Gizmos.color = Color.blue;
            Gizmos.DrawLine (transform.position, objetivo.position
);

            Gizmos.color = Color.white;
            Gizmos.DrawWireSphere (transform.position, 1.5f);

        }


    }
```

Funciones en Unity

1. Lerp

Vector3.Lerp (transform.position, target_position, move_speed*Time.deltaTime);

Lerp interpola los valores de inicio a fin de manera lenta y continua.

Usado en una cámara no va tan pegada al objetivo, va lentamente.

2. Velocity en RigidBody

- rb.velocity = new Vector2(axis.x*velocidad_movimiento,rb.velocity.y);

Modifica la velocidad del personaje y lo hace caminar sin detención en una dirección

3. Clamp

Para anclar el valor de una variable como de la cámara usamos clamp

Mecánica BOUNDS

```
cam = GetComponent<Camera> ();
alto_vision_mitad = cam.orthographicSize;
ancho_vision_mitad = alto_vision_mitad * Screen.width/Screen.height;
float clampedX = Mathf.Clamp (transform.position.x, min_bound.x + ancho_vision_mitad,
max_bound.x -  ancho_vision_mitad);
float clampedY = Mathf.Clamp (transform.position.y, min_bound.y + alto_vision_mitad,
max_bound.y -  alto_vision_mitad);
transform.position = new Vector3 (clampedX, clampedY, transform.position.z);
```

clamp no retorna un valor menor al mínimo o mayor al máximo

4. Translate

translate -> le avisa al motor que queremos cambiar las posiciones x,y,z de transform

```
transform.Translate (new Vector3
(Input.GetAxisRaw("Horizontal")*velocidad_movimiento*Time.deltaTime,0,0));
```

5. Slerp para hacer que un enemigo persiga con la mirada

```
transform_nueva.rotation = Quaternion.Slerp(transform_nueva.rotation,
Quaternion.LookRotation(-target.position + transform_nueva.position),
rotationSpeed*Time.deltaTime);
```

```
Vector3 direction = transform.TransformDirection (Vector3.forward);
Quaternion rotation = Quaternion.FromToRotation (Vector3.up,shoot.normal);
Mathf.SmoothDamp(machine_gun.transform.localPosition.x,posxy_next.x,ref velxy.x,0.04f)
Mathf.Lerp();
```

```
Vector3.Angle(transform.forward, targetDir);
transform.RotateAround (Vector3.zero, Vector3.up, velAngular * Tim
e.deltaTime);
```

Anclar la vision de la cámara

En funciones privadas no es necesario agregar private. En los datos tampoco, pero en los datos es recomendable para facilitar la lectura