Cosas fundamentales para el desarrollo de videojuegos:

Documento para facilitar el desarrollo de videojuegos con Unity3D ,Godot Engine, Pilas engine 2 y jmonkey engine 100% en español

“Comunidad de desarrollo de videojuegos”

Aleatoriedad:

La aleatoriedad se suele usar con 2 valores, uno mínimo y uno máximo, el resultado es un número aleatorio entre esos 2 valores. La aleatoriedad no es perfecta, se suele usar el reloj de la computadora entre otras cosas para así poder tener un resultado difícil de descifrar.

-Un ejemplo práctico de aleatoriedad en el juego flappybird son las columnas.Al resposicionarse toman un valor aleatorio en el eje "Y" ya que algunas están más abajo o más arriba.

-Otro ejemplo de aleatoriedad en el juego Pong, cada vez que reinicia la pelota y la lanzamos se toma un valor aleatorio en el eje "Y" para que al tirar la pelota el jugador no adivine su ubicación.

Unity:

```
float variable = Random.Range(valorMinimo, valorMaximo);
```

Godot engine:

```
var aleatoriedad = rand_range(ValorMinimo,ValorMaximo)
var randomEntero = randi()%(ValorMaximo-ValorMinimo)+ValorMinimo
```

Godot engine c#:

```
int posicionX = (int)Godot.GD.RandRange(0,400);//creo una aleatoriedad entre 2 numeros dobles convierto a entero al final
float posicionY = (float)Godot.GD.RandRange(0,400);//randRange es double hay que convertirlo a float para meterlo en el position
```

Pilas engine 2:

(esta función puede retornar valores que se incluyen en los extremos, por ejemplo si llamamos a `pilas.azar(1, 3)` podría retornar `1`, `2`, o `3`.).

```
pilas.azar(desde, hasta)
```

Jmonkey Engine:

(aún no hay información)

Semilla en Aleatoriedad:

También existe el concepto de semilla y es algo fundamental para tener resultados diferentes.Lo que hace la semilla es generar una aleatoriedad cada vez que inicia el juego y con esto no siempre obtener el mismo resultado.Esta función se suele poner en función READY de Godot.

-En el juego flappy bird cada vez que morimos las columnas aparecen en distinta posición, se suma a la aleatoriedad una semilla que genera diferentes partidas.

Unity:

(aún no hay información).

Godot engine:

(usar `randomize()` para obtener números aleatorios en cada inicio del juego).

randomize()

Godot engine c#:

```
//creo una semilla para que al inicio siempre sea diferente
Godot.GD.Randomize();
```

Pilas engine 2:

(aún no hay información).

Jmonkey Engine:

(aún no hay información)

Etiquetas o grupos para obtener parámetros de objetos:

Las etiquetas son una forma de obtener las propiedades de muchos objetos utilizando un nombre. Esto es muy útil para usar con muchos objetos y casi todos los game engine lo utilizan.

-Por ejemplo pueden usar los grupos para determinar una serie de enemigos y mediante esa etiqueta acceder a sus propiedades y hacerlos atacar.

-También se podría usar para dar vuelta varios sprite. Si tengo un personaje con 2 sprite separados al ponerlos en la misma etiqueta puede acceder a sus propiedades y darlos vuelta juntos sin tener que acceder a cada uno.

Unity:

(nota importante en unity para acceder a todos los objetos etiquetados hay que declarar una variable de tipo arreglo.En la función ready asignar a la variable creada el método gameobject.findGameWithTag y luego para cambiar sus propiedades utilizar un bucle foreach)
https://www.youtube.com/watch?v=9nIITkT0KJs&feature=youtu.be

GameObject.FindGameObjectWithTag("Nombre del Tag");

Ejemplo:

```
public GameObject[] SpritesConEtiqueta;//tomó los sprites en un arreglo para luego darlo vuelta

void Start()
{
	SpritesConEtiqueta = GameObject.FindGameObjectsWithTag("Sprite_player");
}

void Update()
{
	foreach (GameObject i in SpritesConEtiqueta) //se usa la letra i de iteración
	{
		if (i.GetComponent<SpriteRenderer>().flipX == true)
		{
			i.GetComponent<SpriteRenderer>().flipX = false;
		}
	}
}
```

Godot engine:

(nota importante los corchetes al final de la función determinan la ubicación del nodo comenzando con el cero por el primero,osea en el caso de querer acceder a muchos objetos tendrías que hacer un bucle foreach como en unity)
https://www.youtube.com/watch?v=7QwekZj6q5I&list=PL-EPeghw5sXhVskw_YRQ83N6GZk4p9RQh&index=17

Get_tree.get_nodes_in_group("nombre del grupo")[0]

Godot engine c#:

```
(tipoDeNodo)GetTree().GetNodesInGroup("nombreDelGrupo")[0];
```

Pilas engine 2:

(aún no hay información).

Jmonkey Engine:

(aún no hay información)

Buscar hijos:

Tanto en godot como en Unity3D necesitamos acceder a los “gameObject hijos” o a los “nodos hijos” para poder acceder a sus propiedades desde el script donde está ubicado el padre y así modificar o tomar sus valores.Hay que tener en cuenta que UNITY tiene la posibilidad de agregar componentes a los Gameobject, en cambio Godot Engine tiene nodos con parámetros definidos.La principal diferencia es que en Unity se construyen los objetos con componentes y en Godot Engine ya vienen Definidos con sus propiedades ó “algo así podría decirse”.
-Un ejemplo práctico seria acceder al “nodo hijo” o “gameObject hijo” y activar o desactivar animaciones de caminar.

Unity:

transform.Find("Nombre Del Gameobject Hijo") //accede al gameobject hijo

Godot engine:

$Nombre Del Nodo //delante del nombre va un signo pesos accede al nodo hijo

Godot engine C#:

getNode<tipo de nodo>(“ubicación / nombre del nodo”)

Pilas engine 2:
(aún no hay información).

Jmonkey Engine:
(aún no hay información)

Buscar objetos o nodos hijos en una escena por nombre y usarlos como límites en un escenario
Segunda explicación.

Es importante poder acceder a los objetos de la escena o nodos mediante un nombre o una referencia

-Un ejemplo practico seria usar la posición de un personaje para evitar que se salga de la cámara.Para esto tendríamos que buscar al player y acceder a la propiedad de posición en el escenario.

-En los juegos de pelea2D o los juegos de pelea callejera como las tortugas ninjas o street of rage de sega génesis se suele ver que los personajes van hasta cierto punto y se traban,(como si habría una pared), esto se puede lograr usando la posición de una referencia en cierto punto de la pantalla para evitar que el jugador supere esa coordenada.

Unity:
https://www.youtube.com/watch?v=ecjBqZjWzi0
https://www.youtube.com/watch?v=MPH-tdsD-uI

https://www.youtube.com/watch?v=CbK3hvBJspE

```
GameObject.Find("nombreDelGameObject").transform.position; // es unVector3
```

Godot engine:

https://www.youtube.com/watch?v=0IWIngMt06E
https://www.youtube.com/watch?v=ucFjRTmhjAwhttps://www.youtube.com/watch?v=0IWIngMt06E
(En godot engine para estas cosas se suele usar grupos ya que es mucho más práctico,si el script está en el mismo nodo es posible acceder a las propiedades directamente,si queremos acceder a sus nodos hijos se puede usar la siguiente forma).

```
$nombreDelNodo.translation #nota translation es para escenas 3D es un vector3
```

Pilas engine 2:

(aún no hay información).

Jmonkey Engine:

(aún no hay información)

Detener el flujo de código por unos segundos
CORRUTINAS y Cooldown:

Cuando creamos un juego a veces necesitamos que el flujo del código se detenga por unos segundos.
-Por ejemplo en los juegos de disparo como el counter strike para que las balas no salgan una al lado de la otra hay que detener el flujo del código por 1 segundo.Esto permite configurar la cantidad de tiros por segundo y tener un tiempo de enfriamiento entre las balas

Unity:

https://docs.unity3d.com/es/current/Manual/Coroutines.html
(creamos una función con la parte del código que queremos ejecutar luego de un tiempo.En la función “Invoke” el primer parámetro es el nombre de la función como string y el segundo parámetro un número flotante que va a determinar el tiempo en segundos)

```
Invoke("nombreDeLaFuncion", 1.0f);
```

(también existe una función llamada IEnumerator que nos permite usar un yield como en Godot y detener el flujo del código por unos segundos..Se crea la función IEnumerator y dentro se coloca yield para detener el flujo de código.Para llamarla se escribe la función StartCoroutine con el Nombre de la función entre corchetes)

```
 void Start()
{
   StartCoroutine("Nombre_de_la_funcion");
}

IEnumerator Nombre_de_la_funcion()
{
   velocidad.x += 1;
```

```
    yield return new WaitForSeconds(3.0f); //espera 3 segundos antes de seguir con el flujo
    velocidad.x += 1;
}
```

Godot engine:

(esperamos 1 segundo antes de seguir, donde 1.0 es el tiempo en segundos con número flotante)

```
yield(get_tree().create_timer(1.0),"timeout")
```

Tambien existe el nodo timer con la señal timeOUT

Pilas engine 2:

(aún no hay información)

Jmonkey Engine:

(aún no hay información)

Cambiar de escenas:

Cambiar de escenas es algo fundamental en el desarrollo de videojuegos

- En los juegos de arcade como pacman se suele ver un menú principal y al tener un crédito y presionar el botón enter se cambia de la escena menú a la escena del juego

-cambiar de escena nivel 1 a nivel 2.

-cada escena tiene sus comportamientos, música, scripts y demás

Unity:

(importante cargamos la librería para poder acceder a los métodos de la clase)

```
using UnityEngine.SceneManagement;
```

(Entre comillas va el nombre de la escena que queremos cargar)

(no olvidarse de ir a “file/build setting” y arrastrar las escenas al cuadro “scenes in build” y guardar cambios)

```
SceneManager.LoadScene("nombreDeLaescena");
```

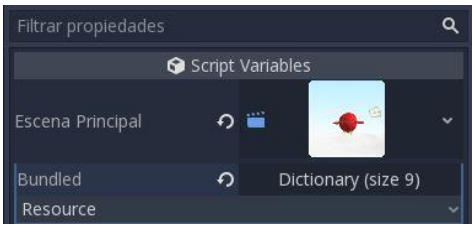
Godot engine:

(Creamos una variable de tipo export para poder cargar la escena en el editor)

```
export(PackedScene) var nombreDeLaEscena
```

Godot engine c#:

```
[Export]
public PackedScene escenaEnEditor;
```



(Cambiamos de escena usando la variable como parámetro de la función).
get_tree().change_scene_to(nombreDeLaEscena)

Pilas engine 2:

(aún no hay información)

Jmonkey Engine:

(aún no hay información) .

Sistema de waypoint:

El sistema de waypoint es un sistema que permite mover a objetos de la escena entre puntos.Para esto se eligen ubicaciones en la escena con nodosPosition(en godot) o emptys en unity y luego se mueve el objeto hacia esos puntos.
-Un ejemplo práctico sería un enemigo que da vueltas entre 3 puntos para patrullar y detectar jugadores
-Otro ejemplo práctico podría ser una carrera de autos donde el auto está controlado por la computadora y tiene que ir por ciertos puntos para no chocar contra las paredes

Unity:

https://www.youtube.com/watch?v=sTU4oXFNyGk

Godot engine:

Próximamente iki capitán va a hacer un tutorial

Pilas engine 2:
(aún no hay información).

Jmonkey Engine:
(aún no hay información)

Instanciar objetos en la escena

Instanciar es algo muy común en un videojuego, ya que se usa para agregar objetos en una escena mientras el juego se está ejecutando

-Un ejemplo muy practico seria en los juegos de naves que al disparar instancias balas.

-En el juego marvel super heroes vs street fighters los personajes instancian poderes como arrojar fuego,tirar bolas de energía, lanzar un rayo entre otras cosas.Todas son instancias de objetos que se crean al tocar una serie de botones mientras jugamos.

Unity:
https://www.youtube.com/watch?v=XErqdGOxMKA

(para instanciarlo conviene crear una lista de Gameobject, cargar el prefact en el editor y luego instanciar..En la funciona van 3 parámetros,el nombre del gameobject,la ubicación inicial como vector3 y la rotación en quaternions)

Instantiate(NombreDelGameObject, UbicacionInicial, RotacionQuaternions);

Godot engine:

https://www.youtube.com/watch?v=0yu5LYgAZVM&t=414s

Godot engine C#:

```
Spatial nuevaEscena = (Spatial)escena.Instance();//instancio la escena empaquetada mosaicos
GetNode("nodoPadre").AddChild(mosaico);//agrego el nodo a la escena

mosaico.Translation = new Vector3(0,0,0);//cambia la posicion de los mosaicos
```

Pilas engine 2:

(aún no hay información).

Jmonkey Engine:

(aún no hay información)

Destruir objetos:

Destruir objetos es algo fundamental, ya que se usa constantemente para dejar de ver cosas en un juego, se podría decir que se eliminan los nodos o los gameobjects de una escena

-Por ejemplo destruir a los enemigos

-destruir balas al salir de la pantalla.

-En los juegos tipo infinite runner o flappyBird destruir objetos consume muchos recursos, en el caso de instanciar muchos enemigos a veces es conveniente volverlos a su ubicación inicial antes que destruirlos.

Unity:

```
Destroy(nombreGameObject, tiempo en float);  //esta función tiene 2 parámetros
```

Godot engine:

(destruir nodo)

```
queue_free()
```

(toma la jerarquía y destruye al padre)

```
get_parent().queue_free()
```

Pilas engine 2:

(aún no hay información)

Jmonkey Engine:

```
rootNode.detachChild(spatial obj);  //toma al árbol y destruye al hijo según su nombre en variable spatial
```

Enemigos persiguen a personajes
“NAVIGATION PATHFINDING”

Una de las cosas fundamentales en el desarrollo de videojuegos es que los enemigos persigan y detecten a los personajes,esto puede ser algo muy tedioso, ya que el enemigo no solamente tiene que perseguir al personaje, sino que también tiene que esquivar obstáculos y de alguna forma saber por dónde ir eligiendo el

mejor camino(supuestamente).Vendría a ser como un principio de inteligencia artificial.Esto es aplicable a juegos 3D, 2D asimétricos,2D superior, desconozco si es posible aplicarlos en juegos con cámara lateral..Tanto en unity como en godot existe una herramienta llamada NAVIGATION..Lo que hace esta herramienta es crear como una selección en los planos o mallas que limitan el lugar por donde el enemigo puede ir mientras persigue al personaje,(para evitar chocar con árboles,rocas o cosas por el estilo)..Imagínese lo que sería programar esto si el game engine no traería esta herramienta.Algunos de sus parámetros son distancia máxima a la que el enemigo puede acercarse al personaje,distancia a la que el enemigo detecta al personaje entre muchas otras cosas.
.Un ejemplo sería el metal gear solid de ps1, los enemigos al estar a corta distancia del personaje lo persiguen y cuando están al lado lo golpean,ellos evitan obstáculos y buscan el mejor camino para atacar...Si el personaje se esconde o está lejos del radar de visión ya no lo van a perseguir
-En el juego MarioBros64 de Nintendo vemos como los enemigos bombas al estar cerca de Mario lo persiguen, sin embargo si Mario se aleja a un rango superior al de la vista de los enemigos ya no lo persiguen.
-En el juego pacman los fantasmas buscan el mejor camino para encontrar al jugador cada fantasma tiene diferente forma de reaccionar ante la pared y el jugador.Todos los fantasmas saben por dónde ir y no se chocan con las paredes.

Unity:
https://www.youtube.com/watch?v=sQwNE-ETtAw

Godot engine:
(ejemplo)
https://www.youtube.com/watch?v=_LwMY9CEsd0
(tutorial en ingles con pacman)
https://www.youtube.com/watch?v=2xiE27j4iiw
(tutorial en en español)https://www.youtube.com/watch?v=N5GqnGgChf0

https://www.youtube.com/watch?v=Q_5lRei-8D4&feature=push-u-sub&attr_tag=35clYQEyiOyVgyfl%3A6

Pilas engine 2:
(aún no hay información).

Jmonkey Engine:
(aún no hay información)

Tipos de cuerpos:

Cuerpos Rígidos, Cuerpos Estáticos, Cuerpos Dinámicos

Mover/Colisiones, chocar con objetos en una escena y física:

Todos los juegos manejan colisiones,si pensamos que sería esto,podríamos referirnos a la propiedad que tienen los objetos para poder chocar/colisionar/golpearse/moverse.Si un personaje y un escenario no tiene un cuerpo de colisión el personaje nunca quedaría arriba del plano o del escenario.Las colisiones están relacionadas a la física, (cosas técnicas que sin práctica no tienen sentido de existencia)….Cuando nos referimos a colisiones va estrictamente relacionado con el tipo de cuerpo.Hay kinematico,estático y dinámico

El dinamico se refiere a cuerpos que colisionan y son manejados por la gravedad y la física del game engine,(cuando nos referimos a física estamos hablando de las propiedades de los objetos para interactuar en las colisiones entre sí o algo por el estilo).Se suele usar en pelotas o cosas que no maneja el jugador

El Kinematico se refiere a un cuerpo con físicas pero que tienen que ser configuradas mediante código, podría decirse que tiene todas las posibilidades de un dinámico,pero hay que programar las funcionalidades.Se suele usar en personajes controlados por el jugador o la computadora como plataformas

El Estático es un cuerpo que no se mueve, como un piso,pero tiene la facultad de poder colisionar con los objetos.Está optimizado ya que se supone que no va a ser movido en un escenario.

Para que todo esto funcione es necesario crear dentro del personaje un nodo(en godot engine) o un Componente(en unity) que dibuje una caja de colisión para ser usado entre los 2 objetos.

-En el juego angry bird las físicas son fundamentales,los cuerpos son dinámicos y tienen colisiones que interactúan entre sí para poder derribar los edificio y ganar el juego

-En los juegos de plataforma 2D suele usarse cuerpos kinematicos para poder mover el personaje,chocar con el suelo,saltar entre otras propiedades físicas que es necesario tener el control sobre ellas.

-En el juego GTA san andreas los autos tienen colisiones y estos pueden chocar.Al chocar activan un evento donde cambia el auto normal por un auto roto,es tan rápido que no notamos cuando pasa esto, simplemente vemos el mismo auto, pero con las imperfecciones..

Unity:

https://www.youtube.com/watch?v=znzLA0OapcU

https://www.youtube.com/watch?v=-cFxFUaWRJY

Godot engine:

https://www.youtube.com/watch?v=OtddXOaK-GI
https://www.youtube.com/watch?v=wnMgtqD3NRc&list=PL-EPeghw5sXjWgz_uPhxqrf4AQM39AuxV&index=4

Pilas engine 2:
(aún no hay información).

Jmonkey Engine:
(aún no hay información)

Áreas y cuerpos para activar eventos:
Al entrar o salir de un área

Una de las cosas fundamentales en el desarrollo de videojuegos es activar eventos cuando un cuerpo entra o sale de un área.Para poder usar esta técnica suele haber un cubo o plano que representa un área donde el jugador puede entrar o salir.Estos cuerpos suelen tener funciones predefinidas que detectan cuando un cuerpo entra o sale de esa área.
Para que esto funcione el personaje también tiene que tener un área de colisión, sin embargo no choca con esa área ya que solamente se usa para que el game engine compruebe si el personaje entró o salió del área..
-Un ejemplo practico seria las monedas de mario bros.Cuando mario bros entra al área de la moneda suma un punto y la moneda desaparece.
-Otro ejemplo práctico puede ser el counter strike con las puertas...Las puertas no siempre abren,para poder abrirlas necesitamos entrar al área(tenemos que estar cerca de la puerta) para activar la función que ejecuta la animación abrir puerta

Unity:
https://www.youtube.com/watch?v=y9aEoCosbxA

Godot engine:
https://www.youtube.com/watch?v=HOZi-GY47go
https://www.youtube.com/watch?v=PkNoDsnwHLc&t=443s

Pilas engine 2:
(aún no hay información).

Jmonkey Engine:
(aún no hay información)

Destruir objetos al salir de la cámara “Técnica Culling”:

En la mayoría de los juegos las balas o poderes salen de la pantalla,por lo tanto la cámara tiene un límite y al salir de ese límite a veces es conveniente que los objetos se destruyan, ya que sino aunque no sean renderizados van a seguir estando en memoria consumiendo recursos.Para facilitar este trabajo los game engine suelen traer herramientas que verifican cuando un objeto sale de la cámara.En el caso de godot es un nodo con señales y en el caso de unity es una función que detecta cuando los objetos salen de la cámara.

-Un ejemplo práctico son los juegos de aviones como el Galaga de nintendo NES..Cuando las balas no colisionan con los enemigos salen de la pantalla,al salir de la pantalla son destruidas para no seguir consumiendo recursos.

-En el street fighters 2 cuando ryu lanza un hadouken al salir de la pantalla el objeto es destruido-

Unity:

https://www.youtube.com/watch?v=9A0h5jRjpTg

Godot engine:

https://www.youtube.com/watch?v=lo04cPHFIMw&list=PL-EPeghw5sXiCTYE7Tjv9hFqyEtCUpc3n&index=17

Pilas engine 2:

(aún no hay información).

Jmonkey Engine:

(aún no hay información)

Punto de SPAWN para los objetos en la escena:

El punto de spawn es el lugar donde los personajes,poderes enemigos van a comenzar, se suele usar una referencia como un nodo position o un gameobject para usar su posición como como referencia.
-En Mario bros todo comienza en 1 lugar específico,los hongos,los poderes,todos tienen una posición donde salir,ya que sino fuera así los poderes como el fuego se instanciarian a la escena pero no saldrán del personaje,sino que de cualquier lado
-En el juego de naves con disparos galaga las balas salen de la punta de la nave

Unity:
https://www.youtube.com/watch?v=TnPmwvhvqQg

Godot engine:
https://www.youtube.com/watch?v=DnFrHHWj15w

Pilas engine 2:
(aún no hay información).

Jmonkey Engine:
(aún no hay información)

En este artículo te enseñaremos como usar un control de cambios para tus videojuegos y también cómo usar nuestro flujo de trabajo con la herramienta git y así evitar pérdida de datos o proyectos.

GIT es un sistema que permite llevar el control de cambio del código fuente de cualquier aplicación. En nuestro caso nos permitirá llevar el control de cambios de nuestro videojuego.

-Sirve para guardar el progreso del proyecto

-Suele ser usado por grandes equipo de desarrollo para que todos juntos puedan trabajar separados en el mismo proyecto.

https://rdckgames-docs.readthedocs.io/es/latest/extra/git.html

Singleton.

Los singleton se utilizan para manejar scripts entre diferentes escenas.

Concretamente es crear una script con variables como un puntaje y que ese puntaje se pueda cargar entre las diferentes escenas.

Godot Engine:

Página web sobre Diseño de Juegos:
¿Cómo hacer un juego interesante?

Usualmente, cuando un desarrollador comienza a trabajar en la creación de un proyecto, sea un novato o no, comienza a pensar en hacer lo que otro desarrollador o estudio hizo con su proyecto. Por ejemplo, si quieres hacer un juego de plataformas inspirado en Super Mario Bros, en parte vas a hacer cada mecanica de Super Mario Bros sin siquiera entendiendo porque ellos las hicieron en su juego. Así que, si en Super Mario Bros puede romper bloques saltando, en tu juego voy a hacerlo también, de la misma forma y sin motivo, y eso lo hará más aburrido. Solo porque Nintendo lo hizo no significa que tu deberías hacer lo mismo, es como lo que decían tus padres de gente que salta de puentes. Ellos hacen mecánicas específicas para sus proyectos porque ellos están creando las mecánicas de su juego prácticamente desde 0.
https://axsajim.com/es/2019/06/08/1-diseno-de-juegos-sensato-como-hacer-un-juego-interesante/?fbclid=IwAR35O2nl0mrF_ow7_HQ_eG1A6D-DaJAPgaIO-fS-NczsoNs3h9o3NwQs28Y

Interpolación lineal:

La interpolación lineal es algo muy importante a la hora de crear videojuegos ya que nos permite cambiar 2 valores en un cierto tiempo.Por ejemplo si necesitamos rotar un objeto y no queremos que esa rotación sea instantánea,es necesario usar una interpolación lineal para que haya un tiempo entre la primer rotación y la última.Esto también puede ser muy útil en los casos donde queramos que un objeto apunte a otro teniendo en cuenta su posición,también puede ser usado en la escala entre otros.

Unity:

https://www.youtube.com/watch?v=KJ7yT7KGuMM

Godot engine:

(En Godot engine esto se puede hacer de 3 formas-Una es usando la función matemática lerp otra es usando una función especial que trae el transform llamada “linear_interpolate()” más info aquí http://docs.godotengine.org/es/latest/tutorials/math/interpolation.html
Por último y creo que es la más practica es usar un nodo tween)

```
func rotar_90_interpolado(delta):
	if Input.is_action_pressed("d"):#si presiono a
		rotacion.y = -45
		puede_rotar = true
		tiempo_rotacion = 0

	if puede_rotar:
		tiempo_rotacion += delta
		#rotation_degrees.y = lerp(self.rotation_degrees.y,rotacion.y,tiempo_rotacion)
		rotation_degrees = rotation_degrees.linear_interpolate(rotacion,tiempo_rotacion

	if tiempo_rotacion > 1:
		puede_rotar = false
```

Recomiendo usar Nodo tween en Godot para interpolación lineal.Tutorial con explicación
https://www.youtube.com/watch?v=Um1ImiKFmhE

Pilas engine 2:

(aún no hay información).

Jmonkey Engine:

(aún no hay información)

RAYCASTING rayo para detectar colisiones:

El raycasting consiste en usar un rayo que va desde un punto ”A” hasta un punto “B”.La forma de utilizarlo es lanzar el rayo y colisionar con un objeto.

Un ejemplo práctico puede ser detectar objetos 3D desde la cámara(Vista en primera persona),osea al salir el rayo del centro de la cámara colisiona con lo que tiene enfrente y de esta forma es posible levantar objetos al mover el mouse.

Otro ejemplo puede ser con juegos tipo Roguelike donde los personajes suelen moverse en casillas y para detectar paredes pueden tener un rayo que está por delante del personaje comprobando constantemente si hay un obstáculo.

Otro ejemplo puede ser una cámara de vigilancia,como las del metal gear solid,cuando el personaje pasa por el rayo que sale de esa cámara de vigilancia,el personaje es detectado o visto por la cámara que activa un evento atacar al jugador en la posición donde el rayo colisionó.

Unity ejemplo:

https://www.youtube.com/watch?v=sBUSY37P2_o

Godot Engine c# ejemplo:

https://www.youtube.com/watch?v=L6XeFMI3bfU&list=PLqsK8f12rhT8Ua04pfOrUk7tiL1XqOgpK&index=23

Matemática en videojuegos y dificultad:

Logaritmos:

Se puede usar los logaritmos para aumentar la dificultad más lentamente.

Por ejemplo en Godot Engine c#.

Video con una simple explicación de logaritmo..Es la inversa de la exponencial y solamente tenes que saber multiplicar o sacar un exponente para entenderlo.

https://www.youtube.com/watch?v=pZTuEHrnOMg

```
//Ejemplo en Código
int enemyCount = (int)Math.Log(level,2);
```

aqui tengo un logaritmo en base 2 que dependiendo el nivel dará un resultado

Ejemplo teniendo en cuenta que el resultado es entero:

log base 2 de 0 = 0...Ojo en la calculadora científica CASIO fx-82MS da error, pero en godot da 0

log base 2 de 1 = 0

log base 2 de 2 = 1

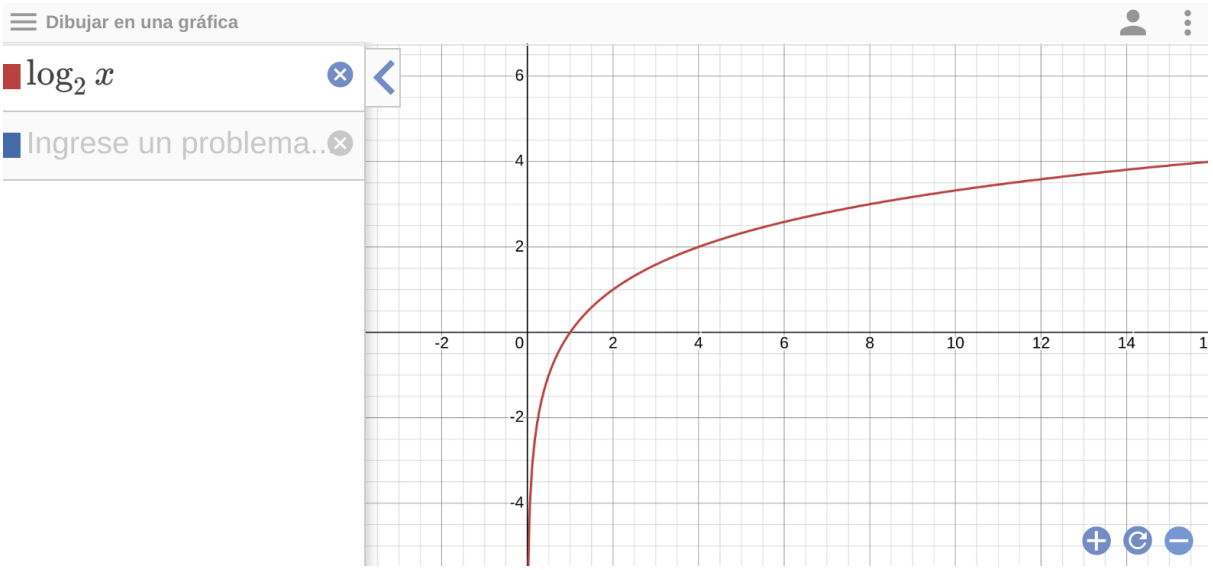
log base 2 de 3 = 1

log base 2 de 4 = 2

Esto puede ser representado en una función logarítmica donde los niveles aumentan exponencialmente. “G(x) = log x”...La consecuencia real es que la dificultad aumenta más lentamente.

Ejemplo logaritmo base 2 de x

Verificamos en la función que un logaritmo nunca es cero



Divisiones:

Se puede dividir un número para disminuir la dificultad dependiendo el nivel

Por ejemplo en Godot Engine c#.

```
//Ejemplo en Código
int enemyCount = level / 2;
```

Ejemplo teniendo en cuenta que el resultado es entero:

level / 2= si level es 1 el resultado es 0

level / 2= si level es 2 el resultado es 1

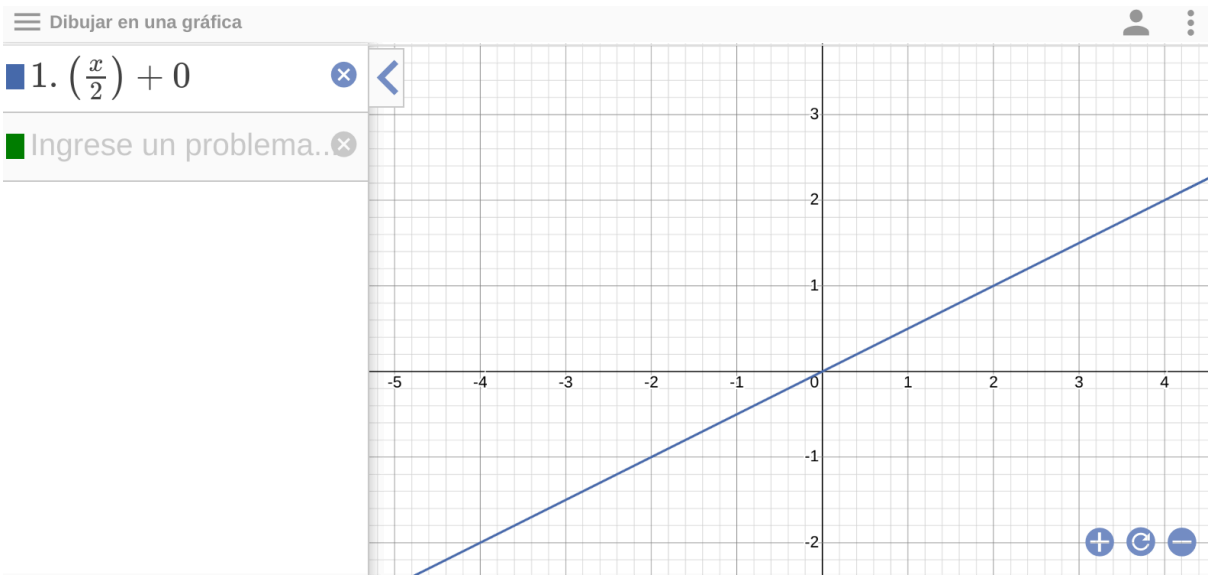
level / 2=si level es 3 el resultado es 1

level / 2= si level es 4 el resultado es 2

Esto puede ser representado en una función lineal donde los niveles aumentan linealmente. “M.X + B”.La consecuencia es que la dificultad aumenta de forma pareja y cambiando los valores podemos obtener diferentes resultados lineales

ejemplo “level / 2”

osea sería 1.(x/2)+0


Dibujar en una gráfica
1. $\left(\frac{x}{2}\right)+0$
Ingrese un problema...
3
2
1
-1
-2
-5
-4
-3
-2
-1
0
1
2
3
4