6.Matemáticas para videojuegos con Unity

***ESTA PARTE ES SOBRE TODO TEÓRICA***

1 Sistemas de coordenadas.

1.1 Coordenadas cartesianas.

También conocidas como coordenadas rectangulares son usadas para la representación gráfica de una relación matemática:

-Determinan la posición en el espacio.

Las coordenadas cartesianas:

-Están compuestas por ejes ortogonales.

-Se cruzan en el origen.

1 solo eje:

-Línea (espacio unidimensional)

2 Ejes

-Plano (dimensión horizontal y vertical) 2D.

3 Ejes

-Espacio(horizontal vertical y profundidad) 3D.

Tiene una aplicación muy amplia y constante:

-Posiciones.

-Posiciones relativas.

-Velocidades.

-Fuerza.

También hay otros tipos de coordenadas:

-Polares (plano 2D).

-Coordenadas UV(Texturas).

-Viewport(Pantalla).

1.2 Pivote,coordenadas globales y locales

Pivote:

Es el origen del sistema de referencia y sobre él se aplican las transformaciones.

1D es el punto 0.

2D es el punto (0,0).

3D es el punto (0,0,0).

Coordenadas globales:

Sistema de referencia absoluto, está asociado a “el mundo” de nuestro juego.

Todos los objetos de nuestro mundo comparten el mismo punto de origen.

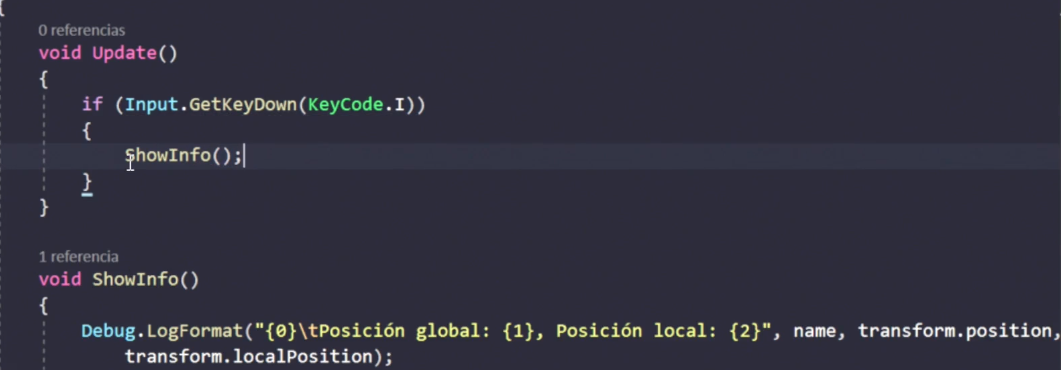
Coordenadas locales:

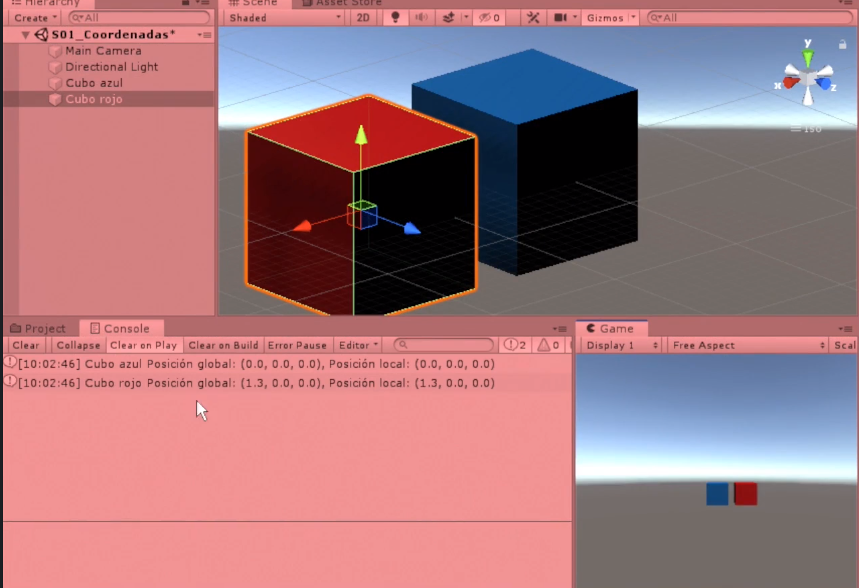
Está asociado a un objeto local, o propio.

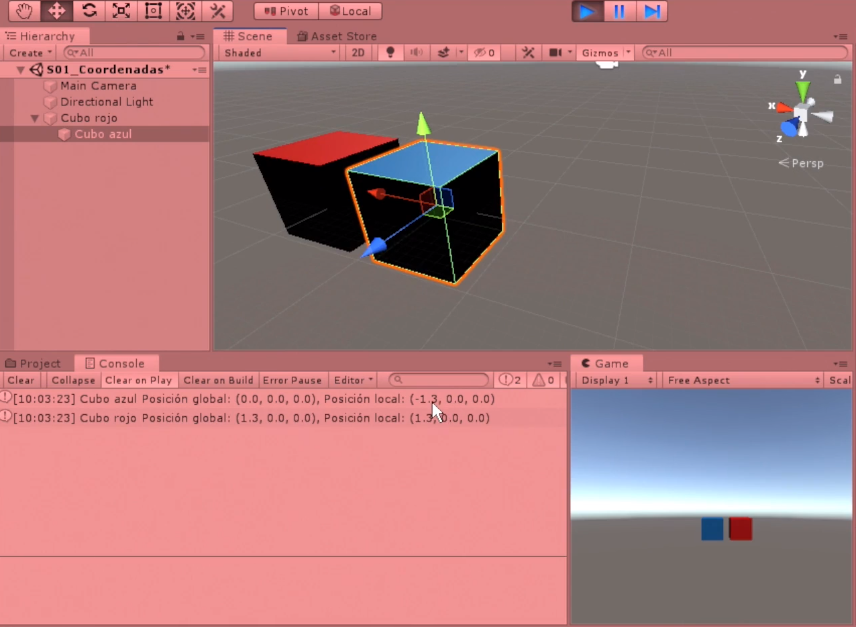
Cada objeto tiene su sistema de coordenadas locales propio.

1.3 Jerarquía de transformaciones

En Unity cada objeto tiene su propio sistema de jerarquía.







Como se puede ver en la primera imagen de los cubos, al no estar dentro de una jerarquía, el cubo azul y el cubo rojo tendrían una posición global y una local coincidentes, pero en el momento que el Cubo azul pasa a ser un hijo del cubo rojo, la Posición local pasaría a ser referente al padre y no al global, porque dejaría de ser un hijo del “mundo” por asi decirlo y ser hijo del cubo rojo.

Es decir:

-En una jerarquía las transformaciones de espacio local se van acumulando.

-Hay una jerarquía de objetos de la escena.

-Las representaciones son en coordenadas locales respecto al padre(todos los objetos son hijos del mundo).

-Haciendo referencia a lo anterior, los padres que no tienen padre están emparentados con el mundo, por lo que sus coordenadas locales y globales son las mismas.

-Los hijos heredan las transformaciones de los padres.

2 Vectores.

2.1 Puntos,vectores y representaciones.

Un punto es la localización en el espacio.

Un vector es el desplazamiento en el espacio.

En Unity están representados por:

-Vector2: Coordenadas X Y.

-Vector3: Coordenadas X Y Z

-Vector4: No es un Vector de 4 dimensiones, simplemente se usa para añadir semántica porque con la cuarta coordenada podemos saber si nos referimos a posición o vector:

-Si w==0 -> Dirección.

-Si w!=0-> Posición.

2.2 Aritmética de vectores.

Operar con vectores es fundamental en el panorama de los videojuegos:

-Gráficos.

-Física.

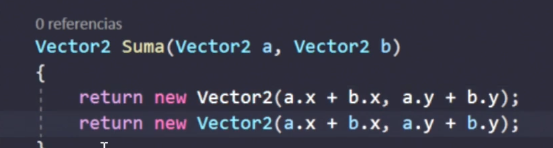
-Lógica

.Aritmética de vectores:Suma

-Se suman los componentes de los vectores.

-Geométricamente corresponde a colocar un vector en el extremo del otro.

-Operación conmutativa.

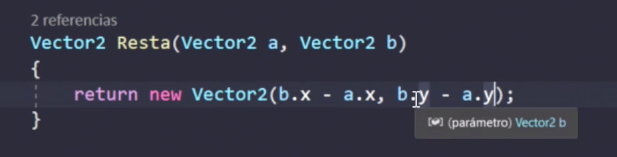


Aritmética de vectores: Resta

-Obtener distancias y direcciónes.

-Resta de cada componente de los vectores.

-No conmutativo (a-b no es lo mismo que b-a).



Aritmética de vectores: Multiplicación y División:

-Incrementa o decrementa la longitud de un vector según el escalar(valor) que nosotros le demos:



2.3 Magnitud o longitud de un vector

-Es la longitud de un vector.-

-Se realiza mediante el teorema de pitágoras

-Se usa la función SqrtMagnitude ya que está optimizado porque obtener la magnitud es una operación costosa debido al uso de la raíz cuadrada.



Normalización de un vector:

-Consiste en dividir un vector por su magnitud, para obtener un vector de distancia 1 con la misma dirección y sentido.

2.4 Producto escalar

Devuelve un valor escalar.

Indica la relación entre los vectores normalizados:

-1 Misma dirección y sentido.

-0:Perpendiculares.

- -1: Sentido opuesto.

Ejemplo:

Calcular la velocidad de un vehículo relativa a su posición

La función se llama Dot();

2.5 Producto vectorial

Devuelve un valor vectorial.

En Unity se usa la regla de la mano izquierda.

Su longitud indica el área de un polígono.

Se utiliza la función cross.

2.6 Vector normal

-Es un vector perpendicular a una superficie.

Calculable a partir de 3 puntos usando el producto escalar.

Usos:

-Iluminación.

-Mecánicas de juego:

-Billar.

-Rebotes.

-Determinar direcciónes.

3 Trigonometría.

3.1 Seno y coseno.

Seno:

Con el seno podemos saber la relación entre un cateto opuesto al ángulo y la hipotenusa del triángulo.

También podemos identificarlo como la proyección del punto sobre el eje vertical.

Coseno:

Con el coseno podemos saber la relación que hay entre el cateto contiguo al ángulo y la hipotenusa del triángulo.

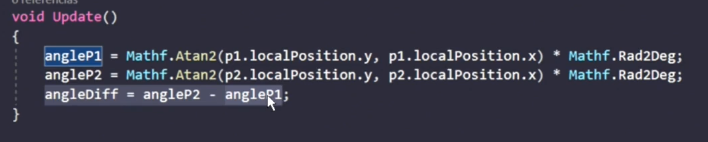
Se puede identificar el coseno como la proyección del punto correspondiente al ángulo con el eje horizontal.

3.2 Tangente y arcotangente.

La tangente es la relación que hay entre cateto opuesto y adyacente.

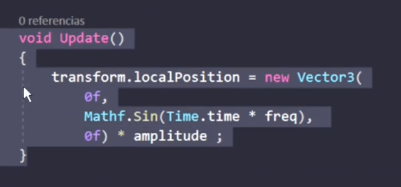
La arcotangente(la inversa de la tangente) se utiliza más en el desarrollo de videojuegos ya que permite obtener el ángulo que forma un vector con el eje horizontal.

La mayoría de las librerías matemáticas incluyen atan2 que es un método utilizado para evitar errores derivados del cambio de signo o dividir por 0 y asi poder obtener el angulo correcto.



Uso de seno y coseno en animación

Seno



Coseno



Ambos



Video todos

4 Interpolación lineal.

4.1 Definición de interpolación lineal.

Interpolar es obtener valores a partir de valores ya conocidos.

Hay varios métodos con distintos grados de precisión

-Por vecino más cercano(la menos precisa de todas).

-Lineal.

-Polinómica.

-Spline.

Algunos casos de interpolación son:

-Realizar desplazamientos suaves entre varios puntos.

-Realizar degradados de color.

-Cambios de transparencia.

-Rellenar barras de progreso.

También existe la extrapolación que se utiliza sobre todo en juegos online.

La extrapolación en vez de obtener valores entre los extremos del intervalo los obtiene de fuera.

4.2 LERP.

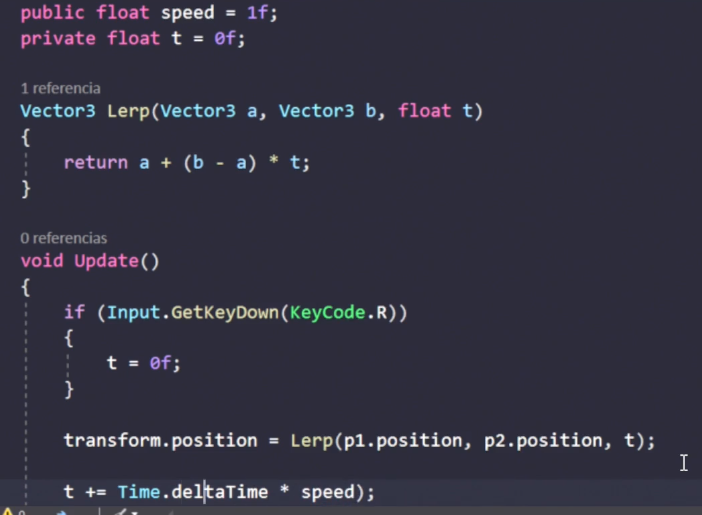
-Abreviatura de Linear Interpolate (interpolación lineal).

-Sirve para obtener valores entre dos valores extremos que se encuentran en un rango conocido usando una aproximación lineal.

C = A + (B-A)\*t => A - A\*t + B\*t => A\*(1-t) + B\*t

-A y B son los valores extremos.

-t es el porcentaje de posición que se quiere averiguar(normalizado entre 0 y 1 (0 es el 0% y 1 el 100%).

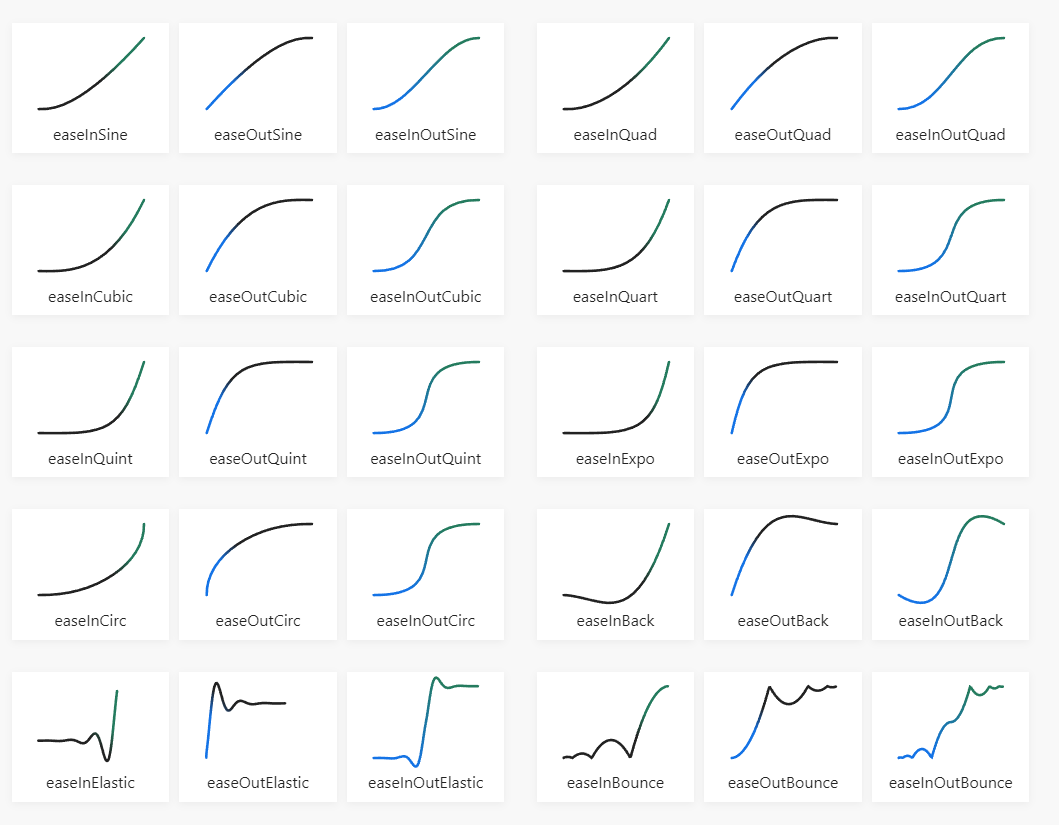
LERP:

Video LERP

4.3 Distintos tipos de interpolación.

Variando la forma en la que evoluciona t se pueden obtener distintos tipos de interpolaciones.

https://easings.net/



5 Cuaterniones.

5.1 Los cuaterniones en Unity.

-Son un instrumento matemático.

-Principalmente rotaciones.

-Extensión de números complejos.

-4 Dimensiones

-1 Real.

-3 Imaginarias.

Tienen una serie de ventajas y desventajas:

Ventajas:

-Rápidos de procesar.

-Faciles de interpolar

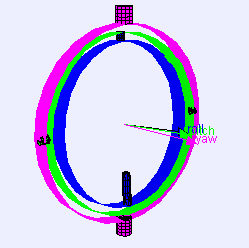
–Evitan problemas como el Gimbal Lock

Desventajas:

-Complejos de entender.

Gimbal Lock:

Es una pérdida de un grado de libertad y se produce cuando dos ejes se alinean.



5.2 Clase Quaternion

Unity ofrece la clase Quaternion para poder trabajar con cuaterniones, dentro de esta clase se puede encontrar una serie de propiedades para manipular cuaterniones.

La propiedad mas importante es la que al ser aplicada a otro cuaternion no lo modifica.

Quaternion.identity.

Se puede instanciar directamente un Quaternion usando su constructor

Quaternion q = new Quaternion().

Pero lo más común es utilizar los métodos que permiten crearlos a partir de informacion.

-Quaternion.Euler(Vector3): Crea un cuaternión a partir de los ángulos de Vector3.

-Quaternion.FromToRotation(Vector3 from, Vector3 to): Crea un cuaternión para rotar desde ángulo origen a ángulo destino.

-Quaternion.Angle(Quaternion a, Quaternion b): Devuelve el ángulo entre los cuaterniones a y b.

-Quaternion.Dot(Quaternion a, Quaternion b): Producto escalar entre los cuaterniones a y b.

-Quaternion.Lerp(Quaternion a, Quaternion b, float t): Realiza una interpolación lineal entre los cuaterniones a y b con el porcentaje t.

-Quaternion.Slerp(Quaternion a, Quaternion b, float t): Realiza una interpolación lineal esférica entre los cuaterniones a y b con el porcentaje t.

-Quaternion.LookRotation(Vector3 direction, Vector3 up): Devuelve una rotación que apunta en la dirección especificada y tiene como vector hacia arriba el especificado por el parámetro up.

-Quaternion.RotateTowards(Quaternion from, Quaternion to, float maxDelta): Devuelve un cuaternión que rota desde el cuaternión from hacia el cuaternión to a una velocidad máxima de maxDelta grados.

6 Clase Mathf.

6.1 Mathf.

Es una clase estática con métodos matemáticos.

Propiedades estáticas:

-Grados y Radianes.

-Pi.

-Epsilon.(Mínimo valor representable)

-Infinito.

Algunos de los métodos mas usados:

-Abs.

-Sin,Cos,Tan,Atan2.

-Clamp/Clamp01.(acotar números entre rangos, Clamp01 entre 0 y 1).

-Lerp/LerpAngle.

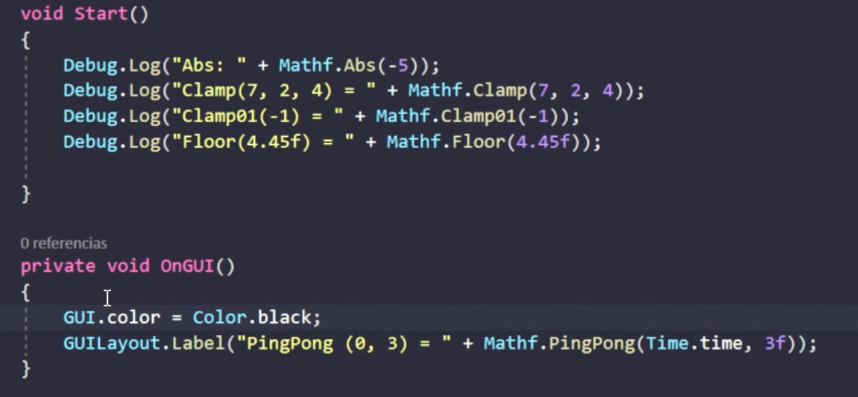
-Max/Min.

-MoveTowards/MoveTowardsAngle.(Avanzar de un valor a otro a una velocidad dada)

-PingPong.(rebota un valor entre 0 y un número dado

-Sign.

-sqrt.



VideoPingPong:

6.2 Random

No confundir con System.Random

-UnityEngine.System -> Estatico.

-System.Random -> Instanciado.

System.Random es mejor en el caso de necesitar varios generadores de números aleatorios, ya que el de Unity es compartido por todos los elementos del juego debido a ser una clase estática pero al uso son similares, aunque el random de Unity permite guardar el estado..

Se puede usar en métodos y propiedades para aleatoriedades:

-Números.

-Puntos.

-Vectores.

-Rotaciones.

Random.State-> Permite guardar estados.

\*\*\*\*\* CLASE DE MATES DE 4 DE LA ESO SOCORRO \*\*\*\*\*

\*\*\*\*\* Diría que esta parte del curso hubiese estado mejor antes del curso de Unity en sí porque ya irías con algunos conceptos más claros \*\*\*\*\*