NavMesh

GUILD

TUTOR: C. Freddy Sacaca López

Gmail: [freddyjefemaster@gmail.com](mailto:freddyjefemaster@gmail.com)

(Se recomienda ver las diapositivas)

1. **NAVMESH**

Es un navegador del plano o terreno en el que se desplazará nuestro personaje y nuestros enemigos. Es y hace un mapeo del plano en el que se desplazará nuestro personaje, traza rutas dentro del mapa. Es muy útil ya que este trazara hasta rutas cortas en nuestro plano y nuestro jugador o enemigo ya sabrá por cual ruta tomas en el juego.

Para que los objetos a no colisionar funcionen entonces estos deben permanecer estáticos. Se los configura en la vista inspector de nuestro Game Object en la parte superior.

Nos permite transitar en nuestro mapa

* + 1. **Propiedades**

El primer lugar como creamos nuestro NavMesh??? Ok es sencillo vamos a la pestaña de Unity y le damos click al menú Window y seleccionamos Navegation y ya nos aparecerá la ventana de navegación.

* Bake

En esta pestaña aplácese las propiedades de terreno como ser la altura en la que se restringe para cruzar nuestro objeto, el radio de acercamiento de nuestro objeto, desde que inclinación no se puede pasar los objetos

Una ves que ya hayamos configurado estas propiedades a nuestro modo bueno entonces nos ponemos a “hornear” el terreno apretando el botón de Bake que se encuentra en la propiedad de Bake en la parte Inferior derecha de la pestaña Navegation

OBS.- Evitar que se “hornee” las luces ya que esto hará a nuestro equipo tarde más en “hornear” las rutas y los polígonos de nuestro plano

Los métodos y necesarios y útiles en esta practica será :

# [NavMeshAgent](https://docs.unity3d.com/es/current/ScriptReference/NavMeshAgent.html).SetDestination

public bool **SetDestination**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**target**);

**Descripción**

Establece o actualiza el destino desencadenando así el cálculo de una nueva ruta.

Tenga en cuenta que la ruta de acceso puede no estar disponible hasta después de unos pocos fotogramas más tarde. Mientras se está calculando el path, pathPending será true. Si un camino válido está disponible entonces el agente reanudará el movimiento.

using UnityEngine;

using System.Collections;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html) {

private [NavMeshAgent](https://docs.unity3d.com/es/current/ScriptReference/NavMeshAgent.html) agent;

void Start() {

agent = GetComponent<NavMeshAgent>();

}

void Update() {

[RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html) hit;

if ([Input.GetMouseButtonDown](https://docs.unity3d.com/es/current/ScriptReference/Input.GetMouseButtonDown.html)(0)) {

[Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html) ray = Camera.main.ScreenPointToRay([Input.mousePosition](https://docs.unity3d.com/es/current/ScriptReference/Input-mousePosition.html));

if ([Physics.Raycast](https://docs.unity3d.com/es/current/ScriptReference/Physics.Raycast.html)(ray, out hit))

agent.SetDestination(hit.point);

}

}

}

# [Input](https://docs.unity3d.com/es/current/ScriptReference/Input.html).GetMouseButtonDown

public static bool **GetMouseButtonDown**(int **button**);

**Descripción**

Devuelve true durante el fotograma en que el usuario presionó el botón del ratón.

Es necesario llamar a esta función desde la función de actualización, ya que el estado se restablece cada trama. No devolverá true hasta que el usuario haya soltado el botón del ratón y vuelva a presionarlo. los valores del botón son 0 para el botón izquierdo, 1 para el botón derecho, 2 para el botón central.Returns true during the frame the user pressed the given mouse button.

using UnityEngine;

using System.Collections;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html) {

void Update() {

if ([Input.GetMouseButtonDown](https://docs.unity3d.com/es/current/ScriptReference/Input.GetMouseButtonDown.html)(0))

[Debug.Log](https://docs.unity3d.com/es/current/ScriptReference/Debug.Log.html)("Pressed left click.");

if ([Input.GetMouseButtonDown](https://docs.unity3d.com/es/current/ScriptReference/Input.GetMouseButtonDown.html)(1))

[Debug.Log](https://docs.unity3d.com/es/current/ScriptReference/Debug.Log.html)("Pressed right click.");

if ([Input.GetMouseButtonDown](https://docs.unity3d.com/es/current/ScriptReference/Input.GetMouseButtonDown.html)(2))

[Debug.Log](https://docs.unity3d.com/es/current/ScriptReference/Debug.Log.html)("Pressed middle click.");

}

}

# [Camera](https://docs.unity3d.com/es/current/ScriptReference/Camera.html).ScreenPointToRay

public [Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html)**ScreenPointToRay**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**position**);

## Parámetros

|  |
| --- |
|  |

## Descripción

Devuelve un rayo que va desde la cámara a través de un punto de pantalla.

El rayo resultante está en el espacio mundial, comenzando en el plano cercano de la cámara y pasando por las coordenadas de los píxeles de la posición (x, y) en la pantalla (la posición .z es ignorada).

El espacio de la pantalla se define en píxeles. La parte inferior izquierda de la pantalla es (0,0); la parte superior derecha es (pixelWidth, pixelHeight).

using UnityEngine;

using System.Collections;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html) {

[Camera](https://docs.unity3d.com/es/current/ScriptReference/Camera.html) camera;

void Start() {

camera = GetComponent<Camera>();

}

void Update() {

[Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html) ray = camera.ScreenPointToRay(new [Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)(200, 200, 0));

[Debug.DrawRay](https://docs.unity3d.com/es/current/ScriptReference/Debug.DrawRay.html)(ray.origin, ray.direction \* 10, [Color.yellow](https://docs.unity3d.com/es/current/ScriptReference/Color-yellow.html));

}

}

# [Physics](https://docs.unity3d.com/es/current/ScriptReference/Physics.html).Raycast

public static bool **Raycast**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**origin**, [Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**direction**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**origin**, [Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**direction**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**origin**, [Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**direction**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**origin**, [Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**direction**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

## Parámetros

|  |  |
| --- | --- |
| **origin** | The starting point of the ray in world coordinates. |
| **direction** | La dirección del rayo. |
| **maxDistance** | The max distance the ray should check for collisions. |
| **layerMask** | A [Layer mask](https://docs.unity3d.com/es/current/Manual/Layers.html) that is used to selectively ignore Colliders when casting a ray. |
| **queryTriggerInteraction** | Specifies whether this query should hit Triggers. |

## Valor de retorno

**bool** Es cierto si el rayo intersecta con un Collider, de lo contrario falso.

## Descripción

Emite un rayo, desde el punto de origen, en dirección dirección, de longitud maxDistance, contra todos los colisionadores de la escena.

Usted puede proporcionar opcionalmente un LayerMask, para filtrar cualquier Colliders que no estén interesados en generar colisiones con.

Especificar queryTriggerInteraction le permite controlar si los colisionadores Trigger generan un resultado o si deben utilizar la configuración global Physics.queriesHitTriggers.

Este ejemplo crea un Raycast simple, proyectándose hacia delante desde la posición de la posición actual del objeto, extendiéndose para 10 unidades.

using UnityEngine;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html)

{

void FixedUpdate()

{

[Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html) fwd = transform.TransformDirection([Vector3.forward](https://docs.unity3d.com/es/current/ScriptReference/Vector3-forward.html));  
  
 if ([Physics.Raycast](https://docs.unity3d.com/es/current/ScriptReference/Physics.Raycast.html)(transform.position, fwd, 10))

print("There is something in front of the object!");

}

}

**Notas: Raycasts no detectará los colisionadores para los cuales el origen Raycast está dentro del Collider.**

**Si mueve Colliders desde secuencias de comandos o por animación, debe permitir que se ejecute al menos un FixedUpdate para que la biblioteca de física pueda actualizarse antes de que un Raycast toque el Collider en su nueva posición.**

public static bool **Raycast**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**origin**, [Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**direction**, out [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)**hitInfo**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**origin**, [Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**direction**, out [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)**hitInfo**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**origin**, [Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**direction**, out [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)**hitInfo**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**origin**, [Vector3](https://docs.unity3d.com/es/current/ScriptReference/Vector3.html)**direction**, out [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)**hitInfo**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

## Parámetros

|  |  |
| --- | --- |
| **origin** | The starting point of the ray in world coordinates. |
| **direction** | La dirección del rayo. |
| **hitInfo** | If true is returned, hitInfo will contain more information about where the collider was hit (See Also: [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)). |
| **maxDistance** | The max distance the ray should check for collisions. |
| **layerMask** | A [Layer mask](https://docs.unity3d.com/es/current/Manual/Layers.html) that is used to selectively ignore colliders when casting a ray. |
| **queryTriggerInteraction** | Specifies whether this query should hit Triggers. |

## Valor de retorno

**bool** True when the ray intersects any collider, otherwise false.

## Descripción

Echa un rayo contra todos los colisionadores en la escena y devuelve información detallada sobre lo que fue golpeado.

Este ejemplo informa la distancia entre el objeto actual y el Collider:

using UnityEngine;  
  
public class RaycastExample : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html){

public Missile missile;  
  
 void FixedUpdate(){

[RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html) hit;  
  
 if ([Physics.Raycast](https://docs.unity3d.com/es/current/ScriptReference/Physics.Raycast.html)(transform.position, -Vector3.up, out hit))

print("Found an object - distance: " + hit.distance);

}

}

This example re-introduces the maxDistance parameter to limit how far ahead to cast the Ray:

using UnityEngine;  
  
public class RaycastExample : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html)

{

public Missile missile;  
  
 void FixedUpdate()

{

[RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html) hit;  
  
 if ([Physics.Raycast](https://docs.unity3d.com/es/current/ScriptReference/Physics.Raycast.html)(transform.position, -Vector3.up, out hit, 100.0f))

print("Found an object - distance: " + hit.distance);

}

}

public static bool **Raycast**([Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html)**ray**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html)**ray**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html)**ray**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html)**ray**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

## Parámetros

|  |  |
| --- | --- |
| **ray** | The starting point and direction of the ray. |
| **maxDistance** | The max distance the ray should check for collisions. |
| **layerMask** | A [Layer mask](https://docs.unity3d.com/es/current/Manual/Layers.html) that is used to selectively ignore colliders when casting a ray. |
| **queryTriggerInteraction** | Specifies whether this query should hit Triggers. |

## Valor de retorno

**bool** True when the ray intersects any collider, otherwise false.

## Descripción

Igual que arriba usando ray.origin y ray.direction en lugar de origen y dirección.

using UnityEngine;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html)

{

void Update()

{

[Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html) ray = Camera.main.ScreenPointToRay([Input.mousePosition](https://docs.unity3d.com/es/current/ScriptReference/Input-mousePosition.html));

if ([Physics.Raycast](https://docs.unity3d.com/es/current/ScriptReference/Physics.Raycast.html)(ray, 100))

print("Hit something!");

}

}

public static bool **Raycast**([Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html)**ray**, out [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)**hitInfo**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html)**ray**, out [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)**hitInfo**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html)**ray**, out [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)**hitInfo**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

public static bool **Raycast**([Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html)**ray**, out [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)**hitInfo**, float **maxDistance** = Mathf.Infinity, int **layerMask** = DefaultRaycastLayers, [QueryTriggerInteraction](https://docs.unity3d.com/es/current/ScriptReference/QueryTriggerInteraction.html)**queryTriggerInteraction** = QueryTriggerInteraction.UseGlobal);

## Parámetros

|  |  |
| --- | --- |
| **ray** | The starting point and direction of the ray. |
| **hitInfo** | If true is returned, hitInfo will contain more information about where the collider was hit (See Also: [RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html)). |
| **maxDistance** | The max distance the ray should check for collisions. |
| **layerMask** | A [Layer mask](https://docs.unity3d.com/es/current/Manual/Layers.html) that is used to selectively ignore colliders when casting a ray. |
| **queryTriggerInteraction** | Specifies whether this query should hit Triggers. |

## Valor de retorno

**bool** True when the ray intersects any collider, otherwise false.

## Descripción

Igual que arriba usando ray.origin y ray.direction en lugar de origen y dirección.

Este ejemplo dibuja una línea a lo largo de la longitud del rayo cuando se detecta una colisión:

using UnityEngine;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html)

{

void Update()

{

[Ray](https://docs.unity3d.com/es/current/ScriptReference/Ray.html) ray = Camera.main.ScreenPointToRay([Input.mousePosition](https://docs.unity3d.com/es/current/ScriptReference/Input-mousePosition.html));

[RaycastHit](https://docs.unity3d.com/es/current/ScriptReference/RaycastHit.html) hit;  
  
 if ([Physics.Raycast](https://docs.unity3d.com/es/current/ScriptReference/Physics.Raycast.html)(ray, out hit, 100))

[Debug.DrawLine](https://docs.unity3d.com/es/current/ScriptReference/Debug.DrawLine.html)(ray.origin, hit.point);

}

}

**Ejemplos**

Los Ejemplos ya se les están proporcionando ahora mismo no todos pero lo esencial…

* 1. **Practica:**
* Crea un objeto que resuelva un laberinto y que tenga una pared traspasable…