INSTRUCTOR

Amplía el esquema: Flowgorithm+Turtle → C# → Unity (manual)
y proporciona un ejemplo con este esquema que solicite mover un cubo del punto A al punto B con obstáculos. Con salida Turtule a de flowgorithm, la traducción a C# y el anexo de código para que funcione en el editor de Unity

Esquema completo: Flowgorithm + Turtle → C# → Unity (manual)

Ejemplo: mover un cubo del punto **A(0,0)** al punto **B(8,8)** evitando un **obstáculo circular** centrado en **(4,4) radio 1,5.**

Salida:

1. **Turtle Window** de Flowgorithm (2D)
2. **Traducción automática** → C#
3. **Código Unity** que **mueve el cubo 3D** siguiendo **la misma lógica** y **dibuja la ruta** con **LineRenderer.**

---

1. Flowgorithm + Turtle (ventana gráfica 2D)

Pasos:

- Activa **Tools → Turtle Window** dentro de Flowgorithm.
- Usa bloques **Turtle: PenDown, Forward, Turn, PenColor.**

复制

```
Programa MoverCuboTurtleObstaculo

    // Config
    real xIni = 0        // A
    real zIni = 0
    real xFin = 8        // B
    real zFin = 8
    real xObs = 4        // centro obstáculo
    real zObs = 4
    real radioObs = 1.5
    real paso = 0.5
    cadena direccion = "norte"   // norte = +Z

    Turtle.PenDown()        // comenzar a dibujar
    Turtle.PenColor("green")

    Mostrar "INICIO  ->  x:" xIni "  z:" zIni

    Mientras ( (xIni < xFin) o (zIni < zFin) ) hacer
        // 1. ¿Dentro del obstáculo?
        Si ( (xIni - xObs)*(xIni - xObs) + (zIni - zObs)*(zIni - zObs) ) <=
radioObs*radioObs entonces
            Mostrar "OBSTÁCULO  ->  Giro este"
            Turtle.PenColor("red")
            Turtle.Turn(90)          // gira 90° (este)
            direccion = "este"
            Turtle.Forward(paso)
            xIni = xIni + paso
        Sino
            // 2. Seguir dirección actual
            Si direccion = "norte" entonces
                Turtle.Forward(paso)
                zIni = zIni + paso
            Sino
                Turtle.Turn(90)      // ya está mirando este
                Turtle.Forward(paso)
                xIni = xIni + paso
            Fin Si
        Fin Si

        Mostrar "PASO  ->  x:" xIni "  z:" zIni
    Fin Mientras

    Turtle.PenColor("blue")
    Mostrar "LLEGADA  ->  x:" xIni "  z:" zIni
Fin Programa
```

Ventana Turtle → verás **línea verde**, **giro rojo** en obstáculo, **llegada azul.**

2. Exportar C# automático

File → Export → C#
Fragmento clave (reducido):

csharp 复制

```
// ---------------  C# generado por Flowgorithm  ----------------
static void Main()
{
    double xIni = 0, zIni = 0, xFin = 8, zFin = 8, xObs = 4, zObs = 4,
radioObs = 1.5, paso = 0.5;
    string direccion = "norte";
    Turtle.PenDown();   Turtle.PenColor("green");
    Console.WriteLine($"INICIO  ->  x:{xIni}  z:{zIni}");
    while ((xIni < xFin) || (zIni < zFin))
    {
        if (((xIni - xObs) * (xIni - xObs) + (zIni - zObs) * (zIni - zObs)) <=
radioObs * radioObs)
        {   // obstáculo
            Console.WriteLine("OBSTÁCULO  ->  Giro este");
            Turtle.PenColor("red");     Turtle.Turn(90);   direccion = "este";
            Turtle.Forward(paso);       xIni += paso;
        }
        else
        {
            if (direccion == "norte")   { Turtle.Forward(paso); zIni += paso;
}
            else                        { Turtle.Turn(90);
Turtle.Forward(paso); xIni += paso; }
        }
        Console.WriteLine($"PASO  ->  x:{xIni}  z:{zIni}");
    }
    Turtle.PenColor("blue");
    Console.WriteLine($"LLEGADA  ->  x:{xIni}  z:{zIni}");
}
```

3. Adaptación manual → Unity 3D + LineRenderer

Idea:

- **Reemplazamos** `Turtle.Forward()` por **Translate + LineRenderer.**
- **Conservamos** el **bucle y condiciones tal cual** (copia-pega).
- **Gizmos** para ver **obstáculo y puntos** en **Scene.**

Código listo:

csharp 复制

```csharp
using UnityEngine;
using System.Collections;
using System.Collections.Generic;   // List

public class PathAvoidTurtleUnity : MonoBehaviour
{
    [Header("Destinos")]
    public Vector3 puntoA = new Vector3(0, 0, 0);
    public Vector3 puntoB = new Vector3(8, 0, 8);

    [Header("Obstáculo")]
    public Vector3 centroObs = new Vector3(4, 0, 4);
    public float radioObs = 1.5f;

    [Header("Config")]
    public float paso = 0.5f;
    public float tiempoEntrePasos = 0.2f;  // para ver movimiento

    private Vector3 posActual;
    private string direccion = "norte";   // norte = +Z
    private List<Vector3> ruta;           // para LineRenderer

    // Referencias Unity
    private LineRenderer lr;
```

```
void Start()
{
    // inicializar línea
    lr = gameObject.AddComponent<LineRenderer>();
    lr.material = new Material(Shader.Find("Sprites/Default"));
    lr.startWidth = lr.endWidth = 0.1f;
    lr.startColor = Color.green; lr.endColor = Color.blue;
    lr.positionCount = 0;
    ruta = new List<Vector3>();

    // posición inicial
    transform.position = puntoA;
    posActual = puntoA;
    ruta.Add(posActual);
    lr.positionCount = ruta.Count;
    lr.SetPositions(ruta.ToArray());

    Debug.Log($"INICIO  ->  x:{posActual.x}  z:{posActual.z}");
    StartCoroutine(MoverHaciaB());
}
```

```
    // --------------  GIZMOS PARA VER EN SCENE  --------------
    void OnDrawGizmos()
    {
        // obstáculo
        Gizmos.color = Color.red;
        Gizmos.DrawWireSphere(centroObs, radioObs);
        // puntos clave
        Gizmos.color = Color.green;
        Gizmos.DrawSphere(puntoA, 0.2f);
        Gizmos.color = Color.blue;
        Gizmos.DrawSphere(puntoB, 0.2f);
    }
}
```

4. Instrucciones rápidas en Unity

5. **Crear 3D Project** (≥ 2020 LTS).

6. **GameObject → 3D Object → Cube** → posición (0,0,0).

7. **Arrastra el script** `PathAvoidTurtleUnity.cs` al Cube.

8. **Play:**

   - **Consola** muestra **los mismos strings** que **Flowgorithm.**
   - **Scene/Game → cubo se mueve y línea verde-roja-azul dibuja la ruta 3D.**

5. Qué logramos

- **Misma lógica** que **Flowgorithm+Turtle** (copia-pega).
- **Salida visual inmediata** (LineRenderer + movimiento).
- **Sin reconstruir nodos** de Unity VS → **salto directo** de **diagrama 2D** a **mundo 3D profesional**.