Tower defense 2D

Scene1: Creating enemies that follow a path.

Step 1: WaypointFollower

POI

- SerializeField
- MoveTowards
- Time.deltaTime
- Distance

Text tutorial

- Start with a new "Create empty" gameobject and rename it "Path".
- Add 4 "Create empty" gameobjects as children to the "Path" gameobject and rename them to "Waypoint (0..3)".
- Edit the positions of the Waypoints to (-7.5, 2.5, 0) (7.5, 2.5, 0) (7.5, -2.5, 0) (-7.5, -2.5, 0) in order.
- Add a new "2D Object" -> "Sprites" -> "Circle" to the Scene and rename it to "Enemy".
- Create a new C# Script called "WaypointFollower" and place it on the "Enemy" in the Scene.
- Add a new variable `waypoints`: `[SerializeField] private Transform[] waypoints;` and fill the 4 created waypoints into the array via the Unity Inspector.
- Set the Enemy position to the first waypoint position in the `void Start()` function: `transform.position = waypoints[0].position;`.
- Add a new variable `speed`: `[SerializeField] private float speed = 1;`
- Move the Enemy with the `MoveTowards` function to the second waypoint in the `void Update()` function: `transform.position = Vector3.MoveTowards(transform.position, waypoints[1].position, Time.deltaTime * speed);`.
- Add a new variable `nextWaypointIndex`: `[SerializeField] private int nextWaypointIndex = 1;`.
- And add a new variable `reachedWaypointClearance`: `[SerializeField] private float reachedWaypointClearance = 0.25f`.
- Change the hardcoded "next" `[1]` in the `Update` function to the newly created `nextWaypointIndex` variable.
- Below the `MoveTowards` call add a check for the distance between the Enemy and the next waypoint is lesser than the newly created `reachedWaypointClearance`: `if (Vector3.Distance(transform.position, waypoints[nextWaypointIndex].position) <= reachedWaypointClearance) { }`
- If this if statement is true increment the `nextWaypointIndex` by 1.
- Add a new if statement that checks if the `nextWaypointIndex` is greater or equal than the amount of waypoints: `if (nextWaypointIndex >= waypoints.Length) { }`.
- If this is true set the `nextWaypointIndex` back to `0`: `nextWaypointIndex = 0;`.
- Press play and change the speed to `10`.

By now you should have a white circle moving in a square around your screen! If not check if you did every step or ask the teacher for help.

Step 2: EnemySpawner

POI

- Prefabs
- Coroutines
- WaitForSeconds

Text tutorial

- Add a new Folder "Scipts" to the Unity assets and place the `WaypointFollower` script to this folder
- Add a new "Create empty" gameobject to the scene and rename it to "EnemySpawner"
- Create a new C# Script called "EnemySpawner" and place it on the "EnemySpawner" in the Scene.
- Add a new variable `enemyPrefab` to the EnemySpawner script: `[SerializeField] private GameObject enemyPrefab;`.
- Drag the Enemy gameobject from the scene to the Assets folder to create a Prefab of the gameobject.
- Fill the `enemyPrefab` variable in the Unity inspector with the newly created Enemy prefab.
- Delete the Enemy gameobject from the scene.
- Create a Prefabs folder and drag the prefab to this folder.
- Create a new function: `IEnumerator SpawnEnemy()` to the EnemySpawner script.
- Add a `while(true) {}` statement with a `yield return new WaitForSeconds(1);` in this block to execute the code within the statement every 1 second.
- Also `Instantiate` the `enemyPrefab` inside of the while statement.
- Start the coroutine in the `Start` function with the `StartCoroutine()` function.

By now you should have an enemy spawning every 1 second but not moving around the set waypoints. Making a prefab out of a gameobject removes any set variable from the scene. Lets fix this in the next step.

Step 3: Combining the WaypointFollower and the EnemySpawner

POI

- FindAnyObjectByType

Text tutorial

- Create a new C# Script called "Path" and add it to the "Path" gameobject in the scene.
- Move the `waypoints` variable from the WaypointFollower script to the Path script and make the variable public.
- In the WaypointFollower script add a new variable `path`: `[SerializeField] private Path path;`.
- Refactor all references to `waypoints` to `path.waypoints`.
- In the Unity inspector fill the 4 waypoints to the path variable on the Path gameobject.
- Add the Unity `Awake` function to the WaypointFollower script.
- In this function find the Path script on the Path gameobject via `FindAnyObjectByType`: `path = FindAnyObjectByType<Path>();`.

By now you should have an enemy spawning every 1 second and immediately start moving around the 4 waypoints!