using UnityEngine;

using System.Collections.Generic;

using System.Collections;

public class NonLinearMovement : MonoBehaviour

{

public float speed = 2f;

private Vector3 targetPosition;

private Vector3 initialPosition;

private float startTime;

private float journeyLength;

private bool moving = false;

private int xyz; // Change the type to int to correctly select the axis

private float amplitude;

private float frequency;

public GameObject spherePrefab;

public float sphereSpawnRate = 0.1f;

public List<GameObject> forceSpheres { get; private set; } = new List<GameObject>();

public float sphereDistance = 0.1f; // Distance between spheres along the path

public float deleteThreshold = 0.5f; // Distance threshold to delete spheres near the target

private void Start()

{

initialPosition = transform.position;

GenerateRandomTarget();

GenerateSpheresAlongPath();

}

private void Update()

{

if (moving)

{

float distanceCovered = (Time.time - startTime) \* speed;

float journeyFraction = distanceCovered / journeyLength;

Vector3 newPosition = Vector3.Lerp(initialPosition, targetPosition, journeyFraction);

/\*if (xyz == 0)

{

newPosition.x += Mathf.Sin(journeyFraction \* Mathf.PI \* 2 \* frequency) \* amplitude;

}

else if (xyz == 1)

{

newPosition.y += Mathf.Sin(journeyFraction \* Mathf.PI \* 2 \* frequency) \* amplitude;

}

else

{

newPosition.z += Mathf.Sin(journeyFraction \* Mathf.PI \* 2 \* frequency) \* amplitude;

}\*/

// Calculate an easing factor to reduce the amplitude as the object approaches the target

float easingFactor = 1f - Mathf.Clamp01(journeyFraction \* 1f); // Adjust the multiplier as needed

newPosition.x += Mathf.Sin(journeyFraction \* Mathf.PI \* 2 \* frequency) \* amplitude \* easingFactor;

newPosition.y += Mathf.Sin(journeyFraction \* Mathf.PI \* 2 \* frequency) \* amplitude \* easingFactor;

newPosition.z += Mathf.Sin(journeyFraction \* Mathf.PI \* 2 \* frequency) \* amplitude \* easingFactor;

/\*// Apply position constraints

newPosition.x = Mathf.Clamp(newPosition.x, -10f, 10f);

newPosition.y = Mathf.Clamp(newPosition.y, 0f, 10f);

newPosition.z = Mathf.Clamp(newPosition.z, -10f, 10f);\*/

transform.position = newPosition;

if (journeyFraction >= 1f)

{

moving = false;

initialPosition = targetPosition; // Set new initial position

DeleteForceSpheres();

GenerateRandomTarget();

GenerateSpheresAlongPath();

/\*foreach (var sphere in forceSpheres)

{

Destroy(sphere);

}

forceSpheres.Clear();\*/

}

}

}

private void GenerateRandomTarget()

{

Vector3 newTarget;

do

{

newTarget = initialPosition + new Vector3(

Random.Range(-10, 10),

Random.Range(0, 10),

Random.Range(-10, 10)

);

/\*if (newTarget.x > 9 || newTarget.y > 9 || newTarget.z > 9)

{

newTarget = new Vector3(0f, 2f, 0f);

}

if (newTarget.x < -9 || newTarget.y < 0 || newTarget.z < -9)

{

newTarget = new Vector3(0f, 5f, 0f);

}\*/

} while (Vector3.Distance(initialPosition, newTarget) < 10f);

targetPosition = newTarget;

startTime = Time.time;

journeyLength = Vector3.Distance(transform.position, targetPosition);

moving = true;

xyz = Random.Range(0, 3);

amplitude = Random.Range(1f, 5.0f);

frequency = Random.Range(1.0f, 2.0f);

if (Vector3.Distance(initialPosition, targetPosition) < 10f)

{

DeleteForceSpheres();

}

}

private void GenerateSpheresAlongPath()

{

float totalDistance = Vector3.Distance(initialPosition, targetPosition);

int numSpheres = Mathf.FloorToInt(totalDistance / sphereDistance);

for (int i = 0; i <= numSpheres; i++)

{

float fraction = (float)i / numSpheres;

Vector3 spherePosition = Vector3.Lerp(initialPosition, targetPosition, fraction);

float easingFactor = 1f - Mathf.Clamp01(fraction \* 1f);

spherePosition.x += Mathf.Sin(fraction \* Mathf.PI \* 2 \* frequency) \* amplitude \* easingFactor;

spherePosition.y += Mathf.Sin(fraction \* Mathf.PI \* 2 \* frequency) \* amplitude \* easingFactor;

spherePosition.z += Mathf.Sin(fraction \* Mathf.PI \* 2 \* frequency) \* amplitude \* easingFactor;

GameObject sphere = Instantiate(spherePrefab, spherePosition, Quaternion.identity);

forceSpheres.Add(sphere);

}

}

private void DeleteForceSpheres()

{

foreach (GameObject sphere in forceSpheres)

{

Destroy(sphere);

}

forceSpheres.Clear();

}

}