MODUL 16 – Posisi, Gerakan, dan Navigasi Untuk GameObject Character Bagian 2

A. TUJUAN

- Memilih destinasi – mencari point spawn terdekat atau titik spawn acak
- Memilih destinasi – melakukan respawn pada checkpoint

B. PETUNJUK

1. Awali setiap kegiatan praktikum dengan berdoa
2. Baca dan pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik
3. Kerjakan tugas-tugas praktikum dengan baik, sabar dan jujur
4. Tanyakan kepada dosen apabila ada hal-hal yang kurang jelas

C. ALOKASI WAKTU: 4 jam pelajaran

D. DASAR TEORI

- Banyak GameObject dalam Movement games yang bias dikendalikan oleh pemain, oleh simulasi fisik lingkungan, atau oleh logic NPC (Non-Player Character). Sebagi contoh, objek yang mengikuti jalur tertentu, atau bergerak maju, atau terbang menjauh dari posisi tertentu dari karakter Unity menyediakan beberapa controller, untuk karakter first dan third-person, juga untuk kendaraan seperti mobil dan pesawat. Movement dari GameObject dapat dikendalikan juga oleh state machines dari Unity Mecanim animation system. Unity menyediakan banyak Class dan Component termasuk Class Vector3 dan rigid physic body untuk memodelkan pergerakan realistic, gaya, dan tumbukan pada games. Modul ini akan menggunakan fitur game engine tersebut untuk mengimplementasikan NPC dan pergerakan karakter musuh.
- Untuk game 3D (pada beberapa bagian dari game 2D), class dasar dari object adalah class Vector3 – object yang menyimpan dan memanipulasi nilai (x, y, z) yang merepresentasikan posisi pada ruang 3D. Jika digambarkan garis imaginer dari titik awal (0, 0, 0) ke titik tertentu pada ruang, maka arah dan panjang dari panah ini (vector) dapat merepresentasikan kecepatan atau gaya (atau disebut dengan nilai magnitude pada arah tertentu).
- Jika diabaikan semua komponen controller karakter, collider, dan system physics dalam Unity, kita dapat menulis kode yang melakukan teleport object secara langsung pada lokasi tertentu (x, y, z) pada scene. Dan kadangkala hal ini juga dilakukan, contoh jika ingin memunculkan object secara tiba-tiba di titik tertentu. Akan tetapi pada banyak kasus, object dipindahkan secara realistic, maka harus diterapkan gaya (force) pada object, atau mengubah kecepatan pergerakannya. Atau jika misalkan object memiliki komponen Controller Character, maka dapat menambahkan pesan Move( ). Dengan adanya Unity NavMeshAgents (dan class lain dari Navigation Meshes), kita sekarang dapat men-set tujuan dari object dengan NavMeshAgent, kemudian logic pathfinding yang tersedia akan melakukan pergerakan object NPC menuju lokasi tujuan (x, y, z) tertentu.

- Selain menentukan teknik apa yang digunakan untuk menggerakkan object, game yang dibuat juga harus menentukan bagaimana menentukan lokasi tujuan, atau arah dan magnitude dari perubahan pergerakan.
- Konsep inti lain pada pergerakan dan pembuatan NPC antara lain:
  a. Titik Spawn : Lokasi tertentu dalam scene dimana object dibuat atau dipindahkan.
  b. Waypoints : lokasi sekuens untuk mendefinisikan path untuk NPC atau character Player untuk diikuti.
  c. Checkpoints : lokasi yang sekali dilalui, akan mengubah atau mengaktifkan suatu kejadian dalam game (contoh, menambah extra time, atau jika pemain mati, akan hidup kembali di titik cekpoint terakhir)

E. LATIHAN PRAKTIKUM

1. Memilih Destinasi – Mencari titik spawn terdekat

Banyak game yang menggunakan titik spawn. Titik spawn dapat berupa random atau titik terdekat dari object tertentu (missal character player). Praktikum ini menggunakan scene dari praktikum 16-2. Copy project tersebut ke project baru 16-3. Berikut adalah tahapan untuk menentukan titik spawn acak:

1. Buat Object Sphere berukuran (1,1,1) pada posisi (2,2,2) dan berikan material m_red.
2. Pada Assets, buat Prefab baru dan beri nama Prefab-ball, drag sphere dari hierarchy kedalam prefab tersebut kemudian hapuskan sphere dari hierarchy.
3. Buat object capsule baru dan beri nama Capsule-spawnPoint pada posisi (3, 0.5, 3), beri tag Respawn (tag default yang telah disediakan Unity). Untuk percobaan, saat ini Respawn point dibuat visible. Nantinya akan diinvisible dengan hilangkan centang Mesh Renderer untuk GameObject Respawn, agar tidak terlihat oleh player.
4. Buat duplikat dari Capsule-spawnPoint dan pindahkan kebeberapa lokasi yang berbeda.
5. Tambahkan script SpawnBall pada gameObject Cube-player.

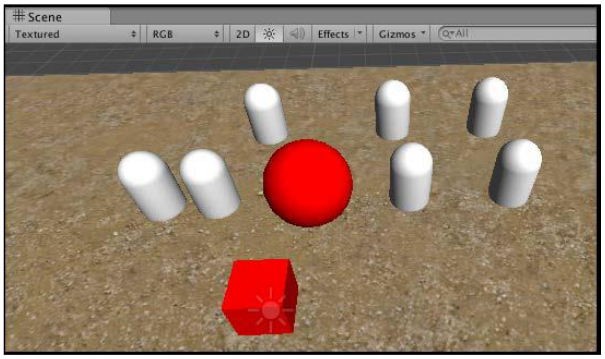
```
using UnityEngine;
using System.Collections;
public class SpawnBall : MonoBehaviour
{
    public GameObject prefabBall;
    private SpawnPointManager spawnPointManager;
    private float destroyAfterDelay = 1;
    private float testFireKeyDelay = 0;
    void Start()
    {
        spawnPointManager = GetComponent<SpawnPointManager>();
        StartCoroutine("CheckFireKeyAfterShortDelay");
    }
    IEnumerator CheckFireKeyAfterShortDelay()
    {
        while (true)
        {
            yield return new WaitForSeconds(testFireKeyDelay);
            // having waited, now we check every frame
            testFireKeyDelay = 0;
            CheckFireKey();
        }
    }
    private void CheckFireKey()
    {
        if (Input.GetButton("Fire1"))
        {
            CreateSphere();
            // wait half-second before alling next spawn
            testFireKeyDelay = 0.5f;
        }
    }
    private void CreateSphere()
    {
        GameObject spawnPoint = spawnPointManager.RandomSpawnPoint();
        GameObject newBall = (GameObject)Instantiate(prefabBall,
        spawnPoint.transform.position, Quaternion.identity);
        Destroy(newBall, destroyAfterDelay);
    }
}
```

6. Tambahkan script SpawnPointManager pada GameObject Cube-player:

```
using UnityEngine;
using System.Collections;
public class SpawnPointManager : MonoBehaviour
{
    private GameObject[] spawnPoints;
    void Start()
    {
        spawnPoints = GameObject.FindGameObjectsWithTag("Respawn");
    }
    public GameObject RandomSpawnPoint()
    {
        int r = Random.Range(0, spawnPoints.Length);
        return spawnPoints[r];
    }
}
```

7. Pilih Cube-player dari panel Hierarchy. Pada SpawnBall component, drag Prefab-ball dari Assets ke variable public Prefab Ball dai SpawnBall. Hilangkan centang Mesh Renderer untuk tiap capsule.

8. Saat game dijalankan dan klik tombol mouse, maka sphere akan muncul tiba-tiba secara acak di titik spawn capsule dan hilang setelah setengah detik.



Selain secara acak, spawnpoint juga dapat dilakukan dengan memilih titik terdekat dengan player. Berikut adalah langkah-langkahnya:

1. Tambahkan method berikut pada script SpawnPointManager

```
public GameObject NearestSpawnpoint(Vector3 source)
    {
        GameObject nearestSpawnPoint = spawnPoints[0];
        Vector3 spawnPointPos = spawnPoints[0].transform.position;
        float shortestDistance = Vector3.Distance(source,
        spawnPointPos);
        for (int i = 1; i < spawnPoints.Length; i++)
        {
            spawnPointPos = spawnPoints[i].transform.position;
            float newDist = Vector3.Distance(source, spawnPointPos);
            if (newDist < shortestDistance)
            {
                shortestDistance = newDist;
                nearestSpawnPoint = spawnPoints[i];
            }
        }
        return nearestSpawnPoint;
        }
```

2. Ubah baris pertama dari script SpawnBall sehingga variable spawnPoint diset dengan memanggil method NearestSpawnPoint();

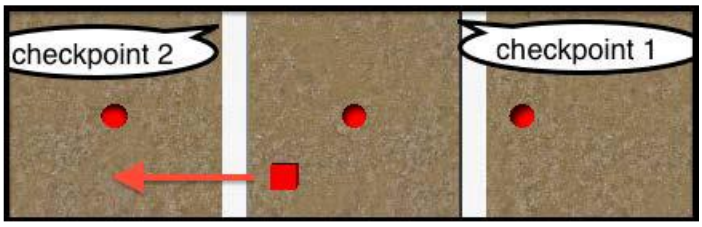
```
private void CreateSphere()
    {
        GameObject spawnPoint = spawnPointManager.
        NearestSpawnpoint(transform.position);
        GameObject newBall = (GameObject)Instantiate(prefabBall,
        spawnPoint.transform.position, Quaternion.identity);
        Destroy(newBall, lifeDuration);
    }
```

2. Memilih Destinasi – respawn pada checkpoint terakhir yang dilewati

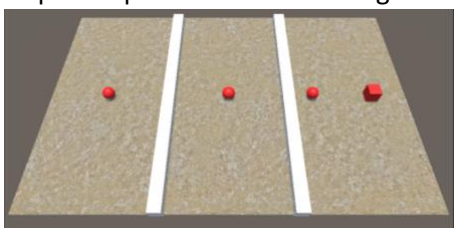
Checkpoint merepresentasikan jarak tertentu yang telah dilalui dalam sebuah game (atau bias juga track) dimana user sukses melaluinya. Berhasil melalui checkpoint biasanya akan mendapatkan bonus, seperti extra time, point, peluru, dan lain sebagainya. Jika pemain memiliki beberapa lives, maka player yang mati akan direspawn pada titik checkpoint terakhir yang baru saja dilaluinya. Praktikum ini akan menunjukkan pendekatan sederhana dari checkpoint, dimana

saat character pemain melalui checkpoint, maka saat mati akan direspwan pada checkpoint terakhir yang telah dilaluinya.

Praktikum ini akan menggunakan project yang telah dibuat di praktikum 16.2. Copy project tersebut pada praktikum 16.4 berikut



1. Pindahkan GameObject Cube-player pada posisi (12, 0.5, 0).
2. Pilih Cube-player pada panel Inspector dan tambahkan komponen Character Controller dengan meng-klik Add Component | Physics | Character Controller (untuk meng-enable pesan Collision OnTriggerEnter).
3. Buat Cube dengan nama Cube-checkpoint-1 pada (5,0,0) dengan skala (1,1,20).
4. Pilih Cube-checkpoint-1 pada hierarchy, beri centang Is Trigger pada komponen Box Collider di panel Inspector.
5. Berikan Tag CheckPoint pada Cube-checkpoint-1.
6. Duplikat Cube-checkpoint-1 dan beri nama Cube-checkpoint-2, letakkan pada posisi (-5,0,0)
7. Buat Sphere baru dan beri nama Sphere-Death pada (7,0.5,0). Berikan material m_red pada sphere untuk membuatnya berwarna merah.
8. Pilih Sphere-Death dari Hierarchy, centang property Is Trigger pada komponen Sphere Collider di panel Inspector.
9. Buat tag Death, dan pilih tag ini untuk Sphere-Death.
10. Duplikat Sphere-Death dan tempatkan pada (0, 0.5, 0).
11. Duplikat Sphere-Death sekali lagi dan tempatkan pada (-10, 0.5, 0).



12. Tambahkan script CheckPoints pada GameObject Cube-player:

```
using UnityEngine;
using System.Collections;
public class CheckPoints : MonoBehaviour
{
    private Vector3 respawnPosition;
    void Start()
    {
        respawnPosition = transform.position;
    }
    void OnTriggerEnter(Collider hit)
    {
        if (hit.CompareTag("CheckPoint"))
        {
            respawnPosition = transform.position;
        }
        if (hit.CompareTag("Death"))
        {
            transform.position = respawnPosition;
        }
    }
    }
```

13. Jalankan scene. Jika kubus mengenai sphere merah sebelum melalui checkpoint, maka akan kembali ke titik awa. Jika telah melalui checkpoint dan sphere merah ditabrak, maka kubus akan direspawn pada checkpoint terakhir yang dilalui.

A. TUGAS PRAKTIKUM

1. Tambahkan objek sphere hitam sebagai objek yang paling berbahaya (jika terkena maka harus kembali ke titik awal)!
2. Tambahkan aturan jika kubus telah berada di checkpoint akhir maka warna kubus berubah menjadi hijau!

--- SELAMAT BELAJAR ---