**PEMROGRAMAN GAME**

**JOB SHEET MINGGU KE-8**

****

Mata Kuliah : Pemrograman Game  
Dosen :

Disusun oleh

Nama :

NIM :

**PROGRAM STUDI TEKNIK INFORMATIKA JURUSAN TEKNIK ELEKTRO**

**POLITEKNIK NEGERI SEMARANG**

**2019**

1. **Tujuan Instruksional Khusus**
2. **Dasar Teori**

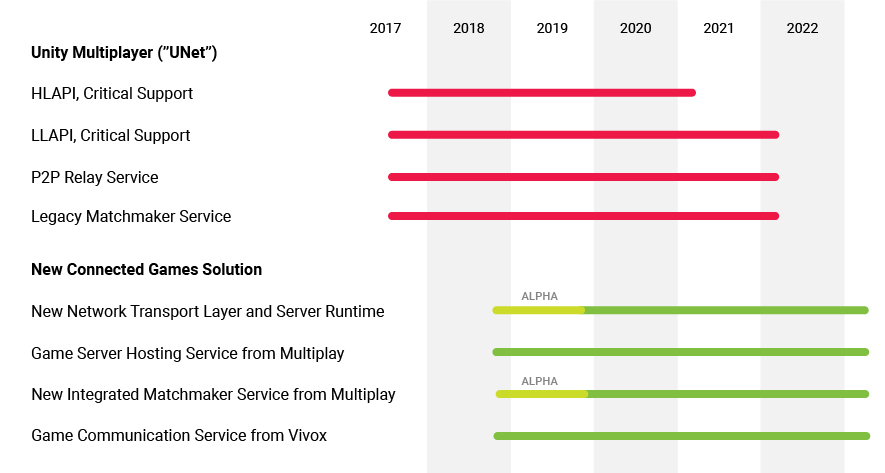
## ****Konsep Game Multiplayer****

Bermain game sendiri tentunya menyenangkan, tetapi akan lebih menyenangkan jika dapat bermain bersama teman. Di modul ini, kita akan membuat game multiplayer dari game yang sebelumnya telah kita rintis. Game Multiplayer adalah game yang dapat dimainkan dengan komputer yang berbeda tetapi saling berkomunikasi.

Di modul ini Anda akan belajar logika penerapan prinsip dan teknik multiplayer di game Pukul Bola, dari hal yang sederhana sampai akhirnya menjadi game lengkap. Pembahasan dijelaskan secara bertahap sehingga Anda dapat mengimplementasikan hal serupa pada game yang Anda miliki. Anda akan mengenal tentang penggunaan Network Manager, Network Transform, Network Identity, Network Start Position, SyncVar, ClientRpc. Semua akan dirangkum menjadi beberapa submodul untuk memperdalam implementasi masing-masing komponen.

### ****Hal-hal yang akan kita pelajari antara lain:****

* Network Manager
* UI Connection
* Menampilkan Player
* Menampilkan Bola
* Menampilkan Skor
* Menampilkan Halaman Selesai

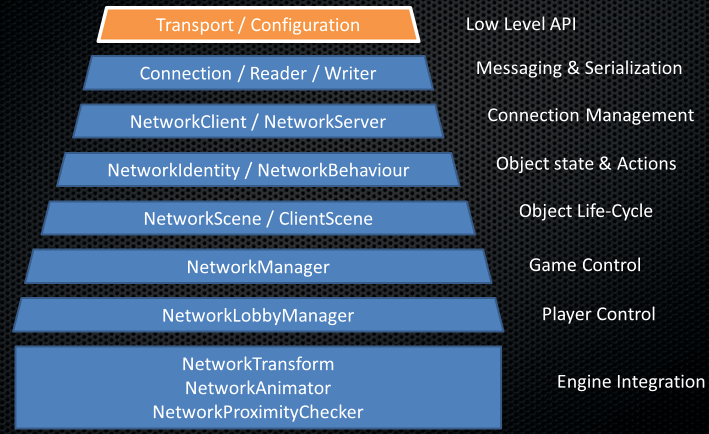


Note : Pada modul Multiplayer ini, disarankan menggunakan versi Unity 2018.4 kebawah karena Unity Networking sudah tidak bisa digunakan pada Unity versi diatas 2018.4

**Network Manager**

Pada submodul ini kita akan membahas tentang cara kerja Network Manager. Network Manager yang digunakan di Unity merupakan assosiasi dari High Level API (HLAPI), sehingga dapat memudahkan developer untuk membuat project multiplayer.

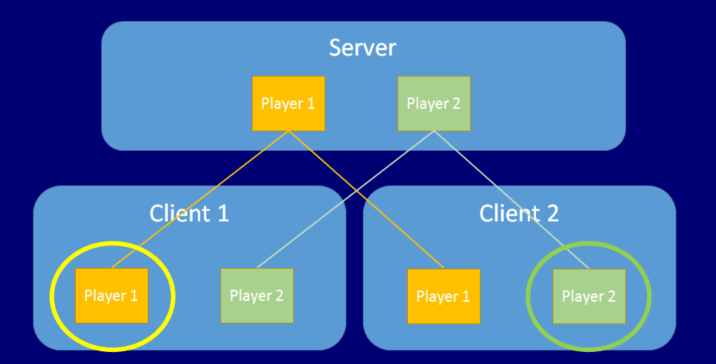
Berikut ini adalah gambaran pada **HLAPI**:



**HLAPI**adalah sebuah pengaturan baru dari networking yang sudah tertanam di Unity, dengan menggunakan namespace: UnityEngine.Networking. Pengaturan tersebut lebih fokus pada kemudahan dalam pengembangan game. Layanan yang disediakan sangat bermanfaat untuk game multiplayer seperti:

* Message handlers
* General purpose high performance serialization
* Distributed object management
* State synchronization
* Network classes: Server, Client, Connection, etc

Sebelum kita melangkah lebih jauh, kita akan membahas konsep Multiplayer dan komponen seperti apa yang akan kita gunakan. Konsep multiplayer yang kita bangun adalah seperti bagan di bawah ini.



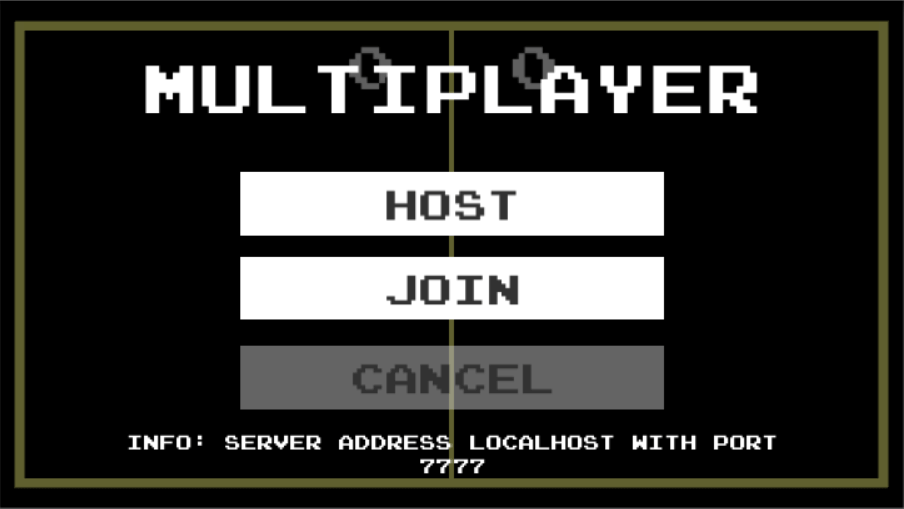
Ketika Anda membuat game yang dapat dimainkan di satu komputer, mungkin Anda tidak banyak menemukan masalah. Tetapi akan cukup menantang ketika kita membuat game yang dapat dimainkan secara bersama dengan menggunakan komputer/mesin yang berbeda. Kendala yang bisa ditemui antara lain:

* Objek lawan yang ditampilkan di **Client 1** merupakan object yang berbeda dari objek lawan di**Client 2**.
* Objek lawan di **Client 1**bergerak berdasarkan dari data yang dikirimkan oleh Objek lawan di **Client 2**.
* Pengiriman data memerlukan waktu beberapa milidetik sehingga pergerakan objek antara **Client 1**dan di**Client 2** berbeda.

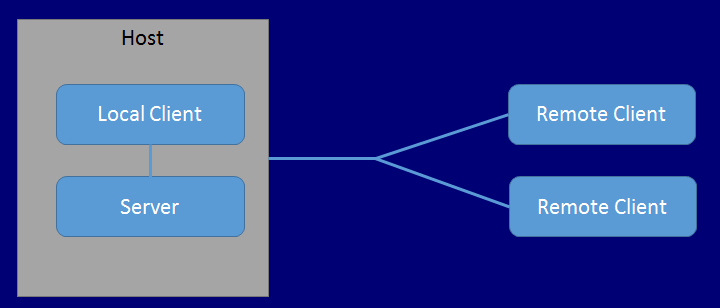
Untuk memastikan tampilan di **Client 1**dan di**Client 2** sama tanpa masalah-masalah di atas, kami telah merangkum materinya di submodul ini.

## ****UI Connection****

Tampilan UI untuk koneksi sudah dibuat. Kini kita fokus mengamankan koneksi antara **Host** dan **Join**.



Ketika user mengklik Host, maka dia akan berperan sebagai penyedia server. Sedangkan ketika user mengklik Join maka dia akan berperan sebagai Client. Namun ketika User klik Cancel, ia tidak bisa menjadi Host ataupun Client. Berikut ini adalah gambaran hubungan antara Host, Client dan Server.



Host adalah sebuah server dan sebuah client dalam proses yang sama. Host yang menggunakan sebuah client disebut Local Client, sedangkan client lain disebut Remote Client.

Beberapa method yang akan digunakan seperti:

* **ClientScene.ready:** ketika client terhubung dengan server.
* **NetworkServer.active:** ketika server dalam kondisi aktif.
* **NetworkClient.active:** ketika client dalam kondisi aktif walaupun tidak terhubung dengan server.
* **network.networkAddress**: IP Address yang digunakan oleh server.
* **network.networkPort:** port yang digunakan oleh server.
* **network.StartClient():** mengkoneksikan dengan server.
* **network.StopClient():** menghentikan koneksi dengan server.
* **network.StartHost():** membuat server sekaligus juga menjadi client.
* **network.StopHost():**  menghentikan server atau client yang terhubung dengan server.

Kode untuk mengetahui IP Address pada sebuah device.

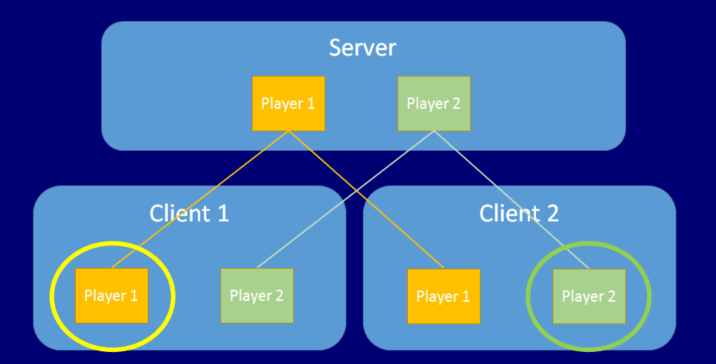
1. **string ip = Network.player.ipAddress;**

Untuk lebih detailnya, Anda dapat mengunjungi laman berikut ini:

* [UNet Using High Level API](https://docs.unity3d.com/Manual/UNetUsingHLAPI.html)
* [NetworkManagerHUD Code](https://forum.unity.com/threads/networkmanagerhud-code-network-control-and-matchmaking-example.326604/)
* [Port](https://id.wikipedia.org/wiki/Port_(Jaringan_Komputer))

## ****Menampilkan Player****

Setelah kita selesai membuat Area Pemukul, selanjutnya kita akan menampilkan Player dengan gambaran konsep sebagai berikut:

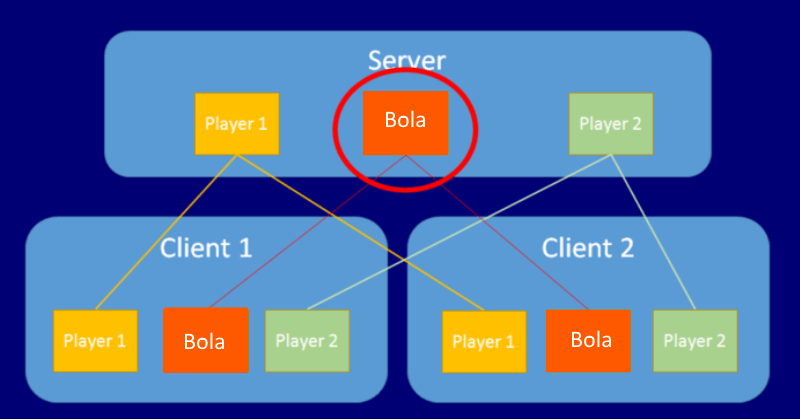


Berikut ini rangkuman permasalahan yang akan dibahas dan kata kunci yang akan kita pelajari.

|  |  |
| --- | --- |
| **Permasalahan** | **Kata Kunci** |
| Bagaimana Memunculkan Player ke Client lain? | Anda dapat memanfaatkan komponen Network Manager untuk memunculkan Player ke semua Client. Object yang dapat dimasukkan ke dalam Playerprefab tersebut adalah object Prefab yang memiliki Komponen Network Identity sebagai Local Player Authority. |
| Bagaimana bisa berjalan dari inputan masing-masing client padahal Object dan Script yang dimiliki Oleh Player1 dan Player2 adalah sama? | Anda dapat memanfaatkan scriptif (!isLocalPlayer) return; untuk membatasi player/paddle yang bukan milik kita agar tidak dipengaruhi oleh input kita. |
|
| Bagaimana memposisikan letak Player1 dan Player2 agar berbeda? | Anda dapat menggunakan komponen Network Start Position pada gameobject yang merupakan titik poin munculnya Player. Kemudian pada Player Spawn Method menggunakan Round Robin. |
| Bagaimana mengkondisikan pergerakan Player(Paddle) di Client1 dan di Client 2 dengan hasil yang sama? Padahal setiap pengiriman data memerlukan waktu sehingga apa yang dilihat player dari client lain terlihat lebih lambat. Seperti contoh object Player 2 yang dilihat di Client 1 merupakan transfer data dari Client 2 kemudian diteruskan ke Server lalu selanjutnya diteruskan ke Client 1 | Untuk menyamakan pergerakan di Client lain dapat menggunakan Komponen Network Transform. |
| Bagaimana membuat Player yang dimainkan user, lebih unik? | Anda dapat memanfaatkan function public override void OnStartLocalPlayer(). Anda dapat melakukan inisialisasi hanya pada Player yang sedang digunakan. Misal menggunakan gambar yang berbeda atau menambahkan sesuatu misal efek. |
|
|

## ****Menampilkan Bola****

Selanjutnya kita akan menampilkan sebuah bola yang sifatnya cukup berbeda jika dibandingkan dengan **Player**(Pemukul).



Berikut ini rangkuman permasalahan yang akan dibahas dan kata kunci yang akan kita pelajari.

|  |  |
| --- | --- |
| **Permasalahan** | **Kata Kunci** |
| Bagaimana memunculkan bola? | **Register Spawnable Prefabs**, Daftar segala object selain player yang dimunculkan di komponen Network Manager. |
| Bagaimana bola dapat terlihat di server dan client? | **NetworkServer.Spawn(bola)**, Kode digunakan untuk memunculkan object ke Network misal object bola |
| Bagaimana supaya bola tidak ada duplikat proses (menerima inputan dari Client1, Client2, dan Server)? | **isServer**, variable yang menghasilkan boolean yang digunakan untuk mendeteksi apa yang sedang dijalankan di client atau server |

1. **Alat dan Bahan**
2. PC / Laptop
3. **Materi**

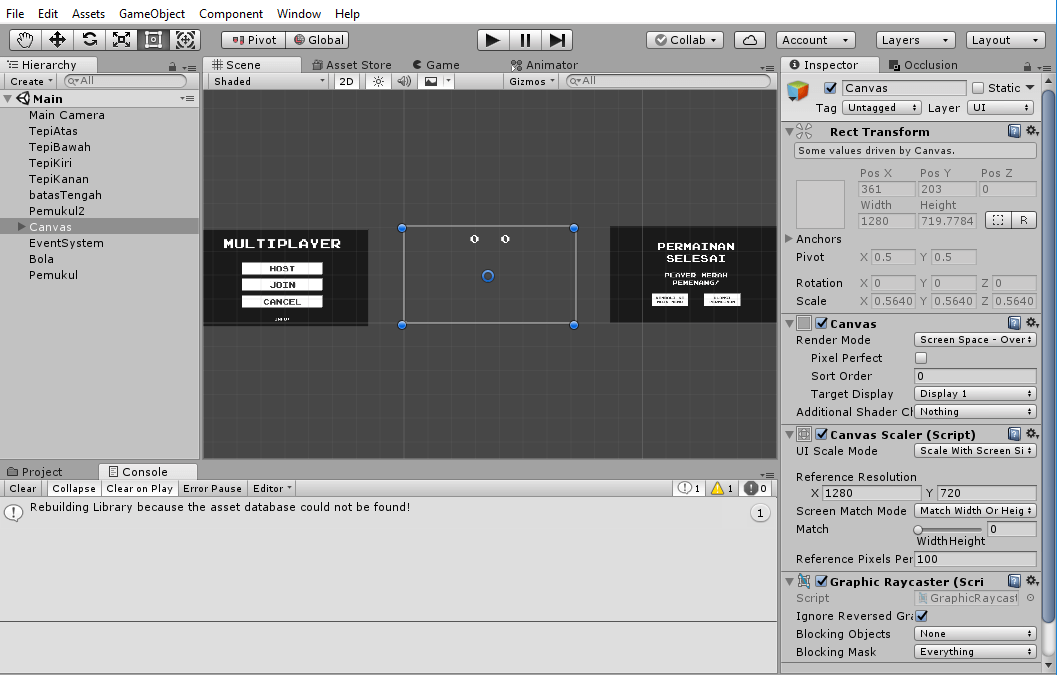
### ****Praktik: Menyiapkan Project Pukul Bola****

Kita akan mengembangkan proyek Pukul Bola yang sebelumnya dapat dimainkan berdua dengan satu komputer. Selanjutnya kita akan berusaha membuat proyek tersebut agar bisa dimainkan dengan peranti (device) berbeda.

**Beberapa perbedaan dari Project Pukul Bola sebelumnya**

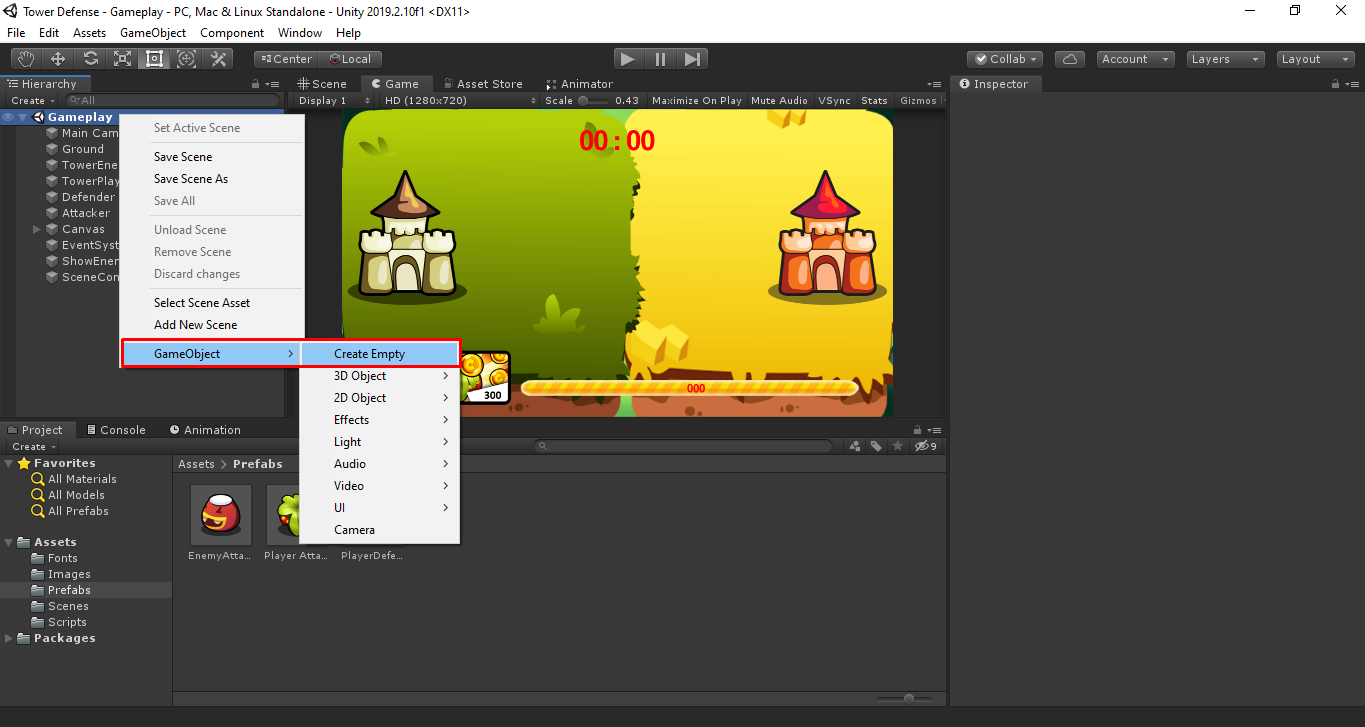
* Menghilangkan penggunaan fungsi destroy object.
* Meringkas beberapa kode..
* Menambah folder Prefabs.
* Menambahkan tampilan UI Multiplayer di scene Main.

Tampilan pada project Pukul Bola seperti berikut ini.

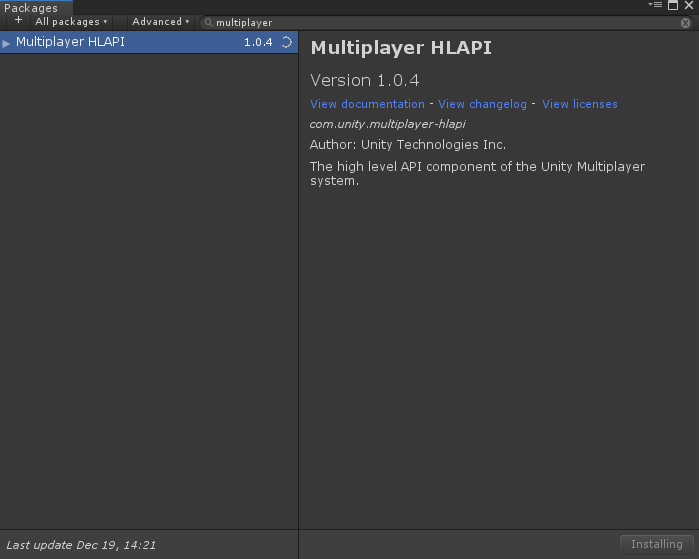


### ****Praktik: Menambahkan Network Manager****

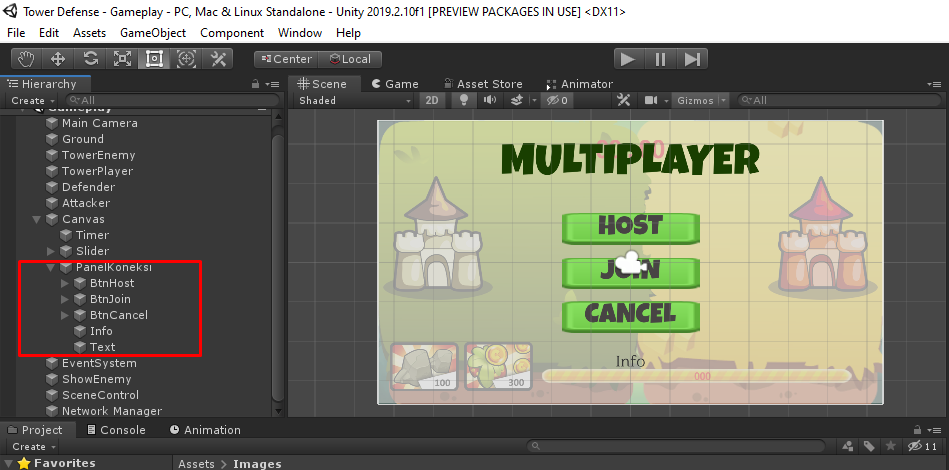
Pertama kita akan menyiapkan proyek untuk dapat dikembangkan ke multiplayer dengan menambahkan Network Manager. Selain itu kita juga menghapus obyek Paddle dan Bola.

1. Tambahkan GameObject kosong kemudian beri nama **Network Manager.** 

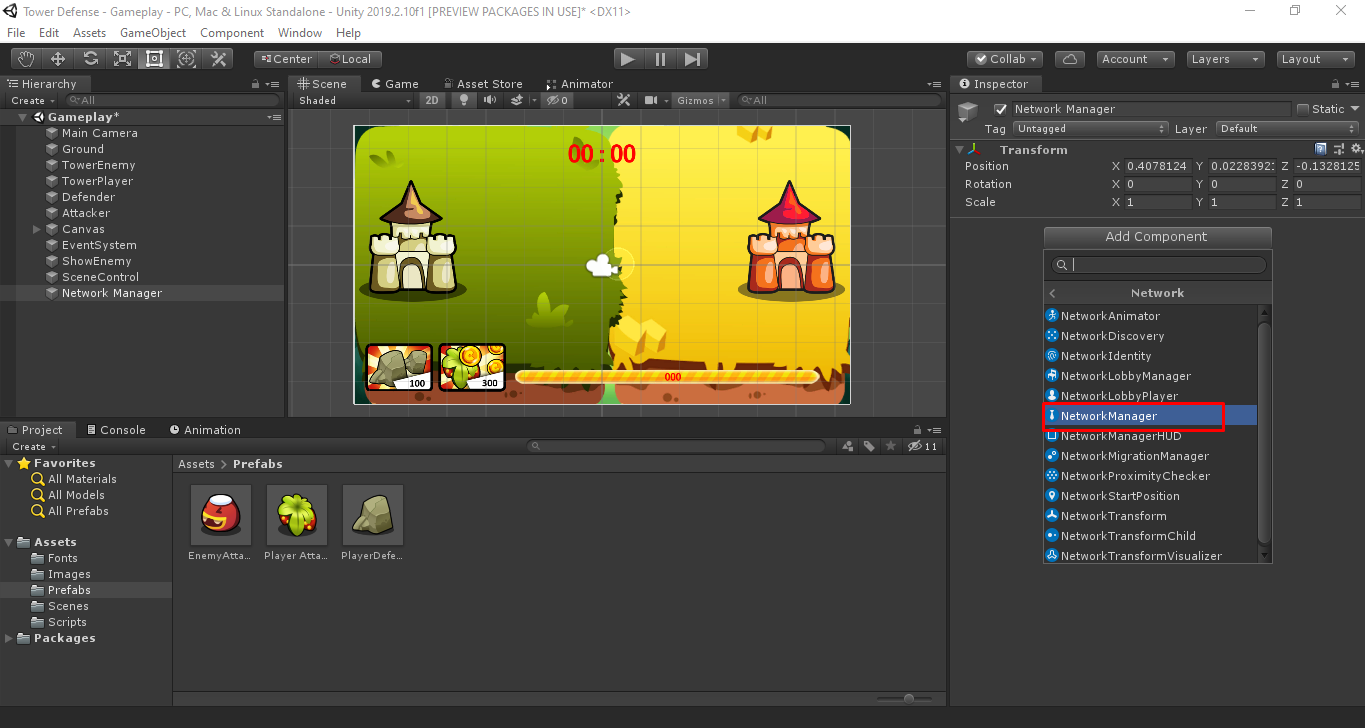
Mendapatkan Package Multiplayer



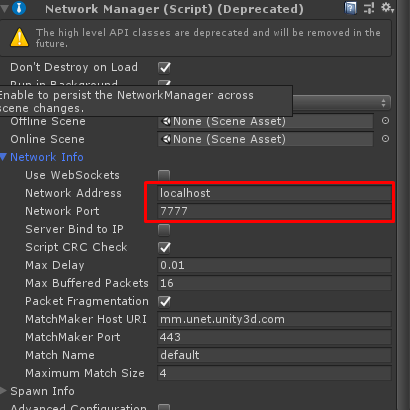
Panel yang telah dibuat :



1. Tambahkan komponen **Network Manager.**



Buka Network Info kemudian aturlah **IP** dan **Port**yang Anda gunakan sebagai server.

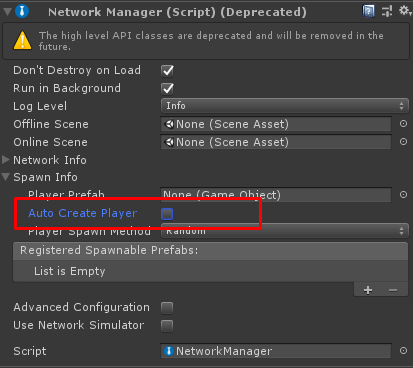


Network Address digunakan sebagai Alamat Server, sedangkan Port digunakan sebagai jalur jaringan. Pastikan Network Address dan Port tidak sedang digunakan oleh aplikasi yang lain

### ****Praktik: Menyiapkan Project****

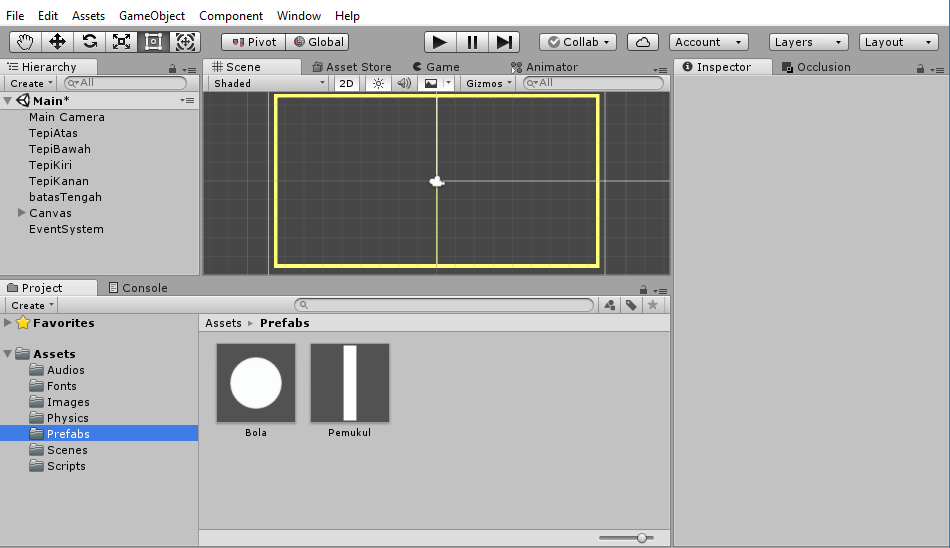
Selanjutnya kita akan menyiapkan proyek untuk submodul selanjutnya, yaitu pembahasan tentang UI. Langkah-langkahnya sebagai berikut:

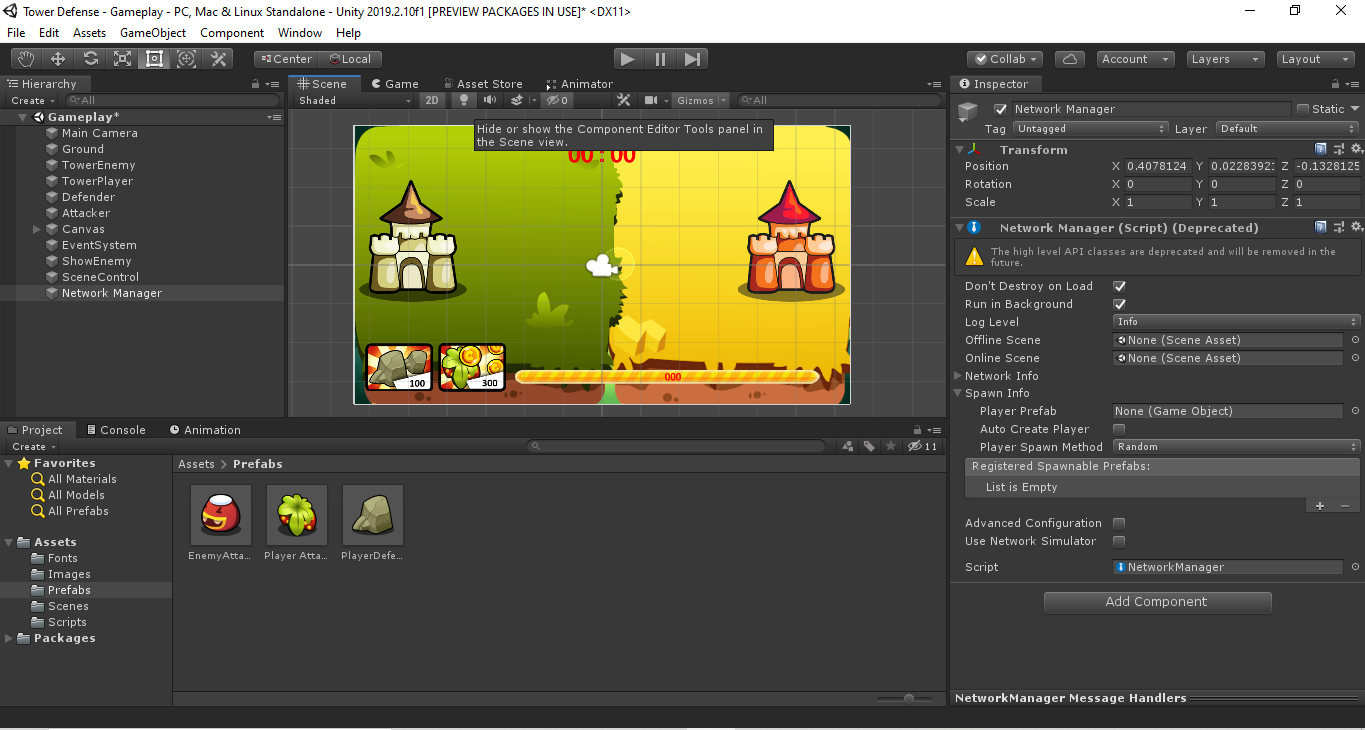
1. Buka Spawn Info,**h**ilangkan centang pada **Auto Create Player.**



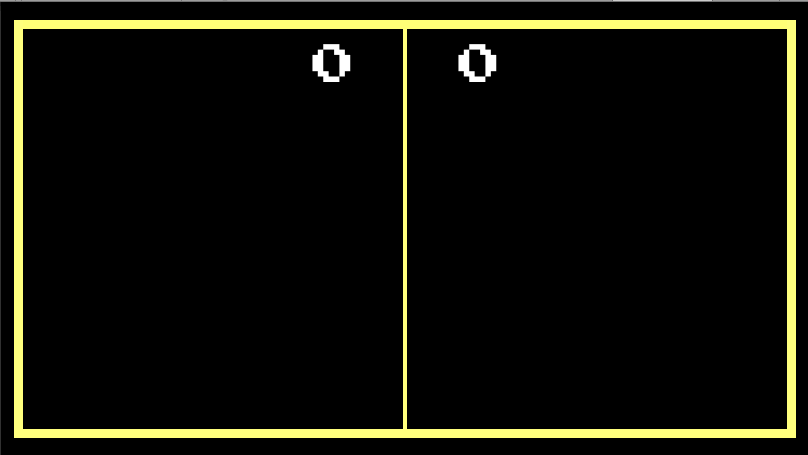
Langkah ini perlu diambil untuk menghindari eror.

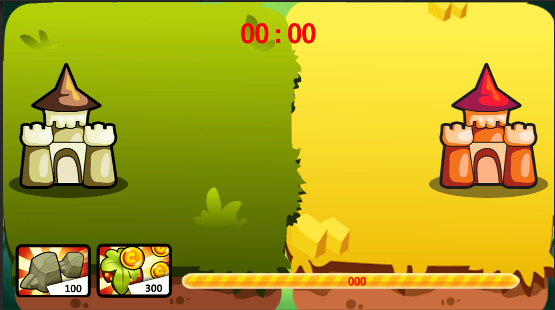
1. Hapus GameObject **Pemukul, Pemukul2**dan **Bola**di Hierarchy.  
   Semua gameobject yang dimunculkan di dalam game, baik itu gameobject sebagai pemain/pelaku utama maupun sebagai bola, harus dimunculkan melalui Network Manager. Object yang digunakan di Network Manager harus berupa Prefabs (gameobject Bola dan Pemukul sudah tersedia di folder **Prefabs**).





Seperti ini ketika dijalankan:





Tidak muncul Paddle , Bola dan Eror pada panel Game setelah kita menambahkan Network Manager. Selanjutnya, kita akan membahas tentang UI Connection untuk mengatur bagaimana tata kelola koneksi dan tampilan untuk menjadi Host atau Client di sebuah game.

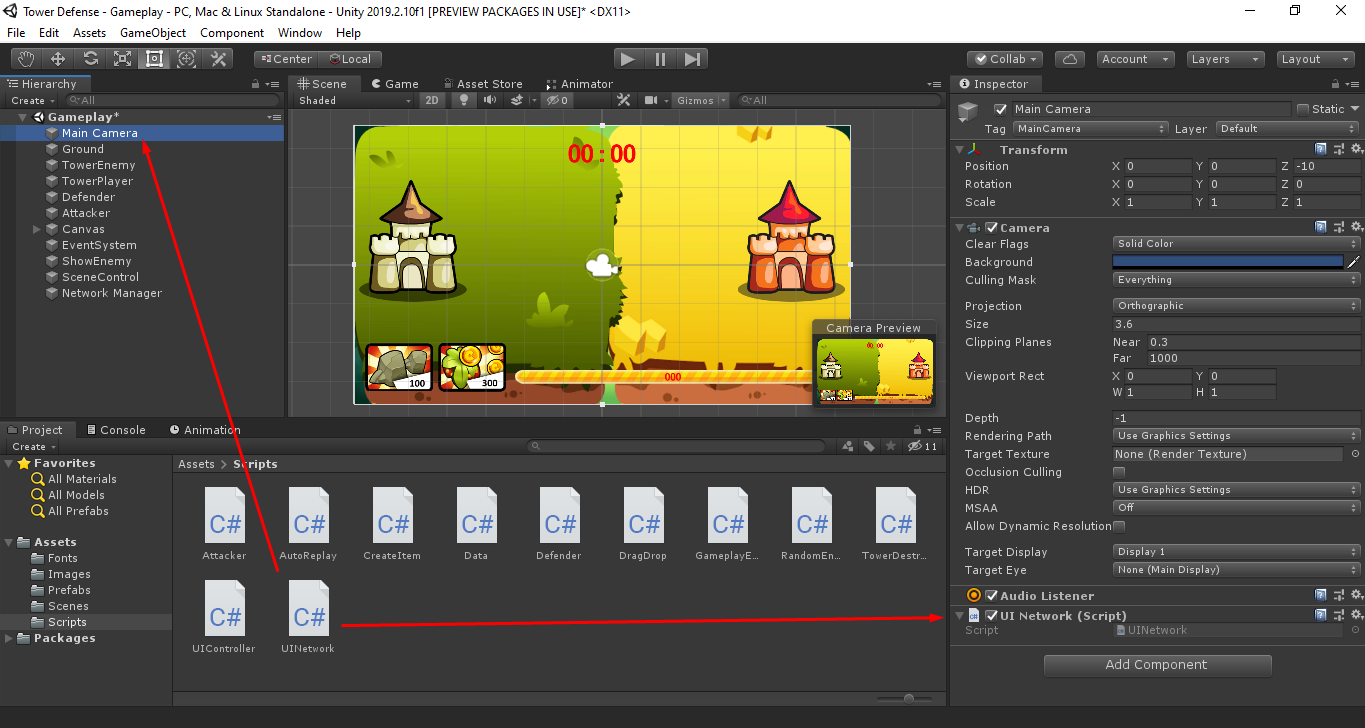
### ****Praktik: Menampilkan Halaman Koneksi****

1. Buka folder **Scripts** kemudian buat script baru dengan nama **UINetwork**. Tambahkan kode sebagai berikut.
   1. Tambahkan variabel di dalam class **UINetwork.**
      1. GameObject panelKoneksi;
   2. Tambahkan kode ini di dalam function **Start().**
      1. panelKoneksi = GameObject.Find("PanelKoneksi");
      2. panelKoneksi.transform.localPosition = Vector3.zero;

**panelKoneksi = GameObject.Find("PanelKoneksi"):**Mencari GameObject yang bernama PanelKoneksi yang terdapat di dalam Hierarchy.

**panelKoneksi.transform.localPosition = Vector3.zero:**Mengubah posisi titik tengah dari posisi parent (Canvas).

1. Simpan perubahan dengan tekan **Ctrl+S**, sehingga keseluruhan kode menjadi seperti ini.
   1. **using UnityEngine;**
   3. **public class UINetwork : MonoBehaviour**
   4. **{**
   5. **GameObject panelKoneksi;**
   7. **// Use this for initialization**
   8. **void Start()**
   9. **{**
   10. **panelKoneksi = GameObject.Find("PanelKoneksi");**
   11. **panelKoneksi.transform.localPosition = Vector3.zero;**
   12. **}**
   14. **// Update is called once per frame**
   15. **void Update()**
   16. **{**
   18. **}**
   19. **}**
2. Masukkan script **UINetwork**ke GameObject **MainCamera**dan hapus komponen **HalamanManager**di **MainCamera**. Kita sudah tidak memerlukan HalamanManager karena UI Network yang akan meng-handle semuanya.



### ****Praktik: Menghandle Button dan Text pada Halaman Koneksi****

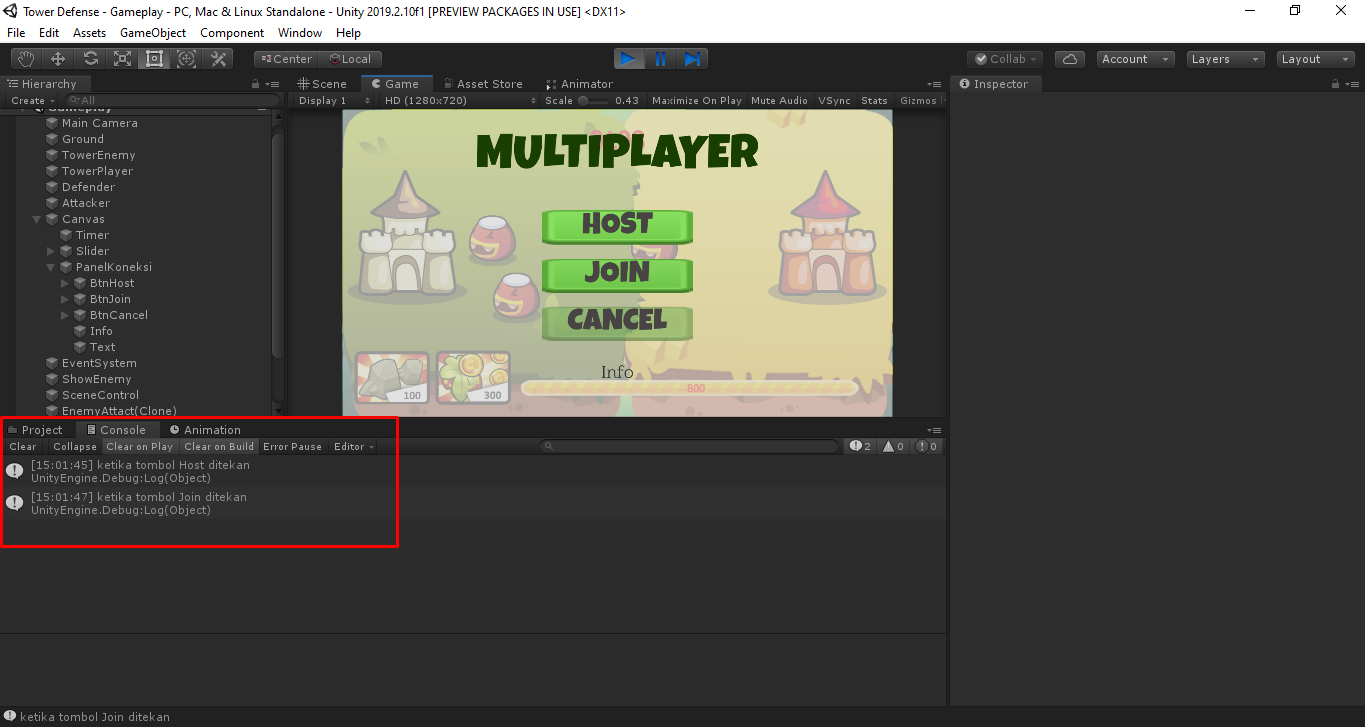
Kita akan membuat listener ketiga tombol dan text info di script UINetwork. Tujuannya, agar kita bisa dengan mudah mengelola komponen tersebut sesuai keinginan. Khusus untuk tombol Cancel, di awal kita akan buat menjadi tombol yang tidak aktif. Sedangkan untuk info, kita akan menampilkan infomasi **Network Address** dan **Port**yang digunakan oleh server. Langkah-langkahnya sebagai berikut.

1. Buka script **UINetwork**lalu tambahkan kode di bawah ini.
   1. Menambahkan **Library.**
      1. using UnityEngine.UI;
   2. Deklarasi variabel di dalam class **UINetwork.**
      1. Button btnHost;
      2. Button btnJoin;
      3. Button btnCancel;
      4. Text txInfo;
   3. Tambahkan inisialisasi pada **Text**dan **Button**di dalam function **Start().**
      1. btnHost = GameObject.Find("BtnHost").GetComponent<Button>();
      2. btnJoin = GameObject.Find("BtnJoin").GetComponent<Button>();
      3. btnCancel = GameObject.Find("BtnCancel").GetComponent<Button>();
      4. txInfo = GameObject.Find("Info").GetComponent<Text>();
      5. btnHost.onClick.AddListener(StartHostGame);
      6. btnJoin.onClick.AddListener(StartJoinGame);
      7. btnCancel.onClick.AddListener(CancelConnection);
      8. btnCancel.interactable = false;

GameObject.Find("**Nama\_GameObject**"). GetComponent<**Nama\_Komponen**>(): Mengambil komponen **Button**atau **Text**dengan melakukan pencarian gameobject yang terdapat di **Hierarchy**.  
**BtnKomponen**.onClick.AddListener(**Nama\_Function**): Merespon ketika button tersebut di klik kemudian menjalankan function yang terdapat di dalam parameter **AddListener**.

* 1. Tambahkan function.
     1. private void StartHostGame()
     2. {
     3. Debug.Log("Dijalankan ketika tombol Host ditekan");
     4. }
     5. private void StartJoinGame()
     6. {
     7. Debug.Log("Dijalankan ketika tombol Join ditekan");
     8. }
     9. private void CancelConnection()
     10. {
     11. Debug.Log("Dijalankan ketika tombol Cancel ditekan");
     12. }

1. Simpan dengan **Ctrl+S**sehingga secara keseluruhan sebagai berikut.
   1. using System.Collections;
   2. using System.Collections.Generic;
   3. using UnityEngine;
   4. using UnityEngine.UI;
   6. public class UINetwork : MonoBehaviour
   7. {
   8. GameObject panelKoneksi;
   9. Button btnHost;
   10. Button btnJoin;
   11. Button btnCancel;
   12. Text txInfo;
   14. // Use this for initialization
   15. void Start()
   16. {
   17. panelKoneksi = GameObject.Find("PanelKoneksi");
   18. panelKoneksi.transform.localPosition = Vector3.zero;
   19. btnHost = GameObject.Find("BtnHost").GetComponent<Button>();
   20. btnJoin = GameObject.Find("BtnJoin").GetComponent<Button>();
   21. btnCancel = GameObject.Find("BtnCancel").GetComponent<Button>();
   22. txInfo = GameObject.Find("Info").GetComponent<Text>();
   23. btnHost.onClick.AddListener(StartHostGame);
   24. btnJoin.onClick.AddListener(StartJoinGame);
   25. btnCancel.onClick.AddListener(CancelConnection);
   26. btnCancel.interactable = false;
   27. }
   29. // Update is called once per frame
   30. void Update()
   31. {
   33. }
   35. private void StartHostGame()
   36. {
   37. Debug.Log("ketika tombol Host ditekan");
   38. }
   40. private void StartJoinGame()
   41. {
   42. Debug.Log("ketika tombol Join ditekan");
   43. }
   45. private void CancelConnection()
   46. {
   47. Debug.Log("ketika tombol Cancel ditekan");
   48. }
2. Ketika dijalankan hasilnya sebagai berikut:

  
Pastikan masing-masing tombol berjalan dengan baik.

### ****Praktik: Menambahkan Network Controller****

Kita telah memastikan tombol yang dibuat berjalan dengan baik. Selanjutnya kita membuat sambungan dengan network manager. Ketika user klik tombol Host maka ia menjadi penyedia server. Sedangkan ketika user tekan Join maka ia pun akan menjadi Client. Sementara itu, untuk membatalkan cukup dengan dengan klik Cancel. Langkah-langkahnya sebagai berikut.

1. Buka kembali script **UINetwork**kemudian tambahkan kode di bawah ini.
   1. Tambahkan **Library**.
      1. using UnityEngine.Networking;

Berisi method yang digunakan untuk mengelola Network.

* 1. Deklarasi **variabel**di dalam class **UINetwork.**
     1. NetworkManager network;
  2. Tambahkan **inisialisasi.**
     1. network = GameObject.Find("Network Manager").GetComponent<NetworkManager>();
     2. txInfo.text = "Info: Server Address " + network.networkAddress + " with port " + network.networkPort;

Inisialisasi variabel network dengan mengambil komponen NetworkManager yang terdapat di GameObject Network Manager. Kemudian di baris berikutnya, tampilkan IP dan Port yang digunakan oleh server.

* 1. Tambahkan kode di dalam function**StartHostGame().**
     1. if (!NetworkServer.active)
     2. {
     3. network.StartHost();
     4. }
     5. if (NetworkServer.active) txInfo.text = "Info: Menunggu Player lain (Jika Server Aktif)";
  2. Tambahkan kode di dalam function **StartJoinGame().**
     1. if (!NetworkClient.active)
     2. {
     3. network.StartClient();
     4. network.client.RegisterHandler(MsgType.Disconnect, ConnectionError);
     5. }
     6. if (NetworkClient.active) txInfo.text = "Info: Mencoba mengkoneksikan dengan Server";

Pastikan setiap melakukan koneksi, cek terlebih dahulu apakah client sudah terhubung dengan server atau tidak. Jika server putus secara tiba-tiba, gunakan **network.client.RegisterHandler(MsgType.Disconnect, ConnectionError)**. Jika kondisi tersebut terpenuhi, maka function **ConnectionError()**akan berjalan.

* 1. Tambahkan function **ConnectionError** untuk meng-handle terjadinya gangguan koneksi.
     1. private void ConnectionError(NetworkMessage netMsg)
     2. {
     3. network.StopClient();
     4. txInfo.text = "Info: Koneksi terputus dari Server";
     5. }

Dalam kondisi ini, client terputus dari server meski masih terus menghubungi. Untuk menghentikan, gunakan **network.StopClient().**User mendapat informasi tentang kondisi yang terjadi.

* 1. Tambahkan kode di dalam function **CancelConnection.**
     1. network.StopHost();
     2. btnHost.interactable = true;
     3. btnJoin.interactable = true;
     4. btnCancel.interactable = false;
     5. txInfo.text = "Info: Server Address " + network.networkAddress + " with port " + network.networkPort;
  2. Tambahkan kode ini di dalam function **Update.**
     1. if (NetworkClient.active || NetworkServer.active)
     2. {
     3. btnHost.interactable = false;
     4. btnJoin.interactable = false;
     5. btnCancel.interactable = true;
     6. }
     7. else
     8. {
     9. btnHost.interactable = true;
     10. btnJoin.interactable = true;
     11. btnCancel.interactable = false;
     12. }

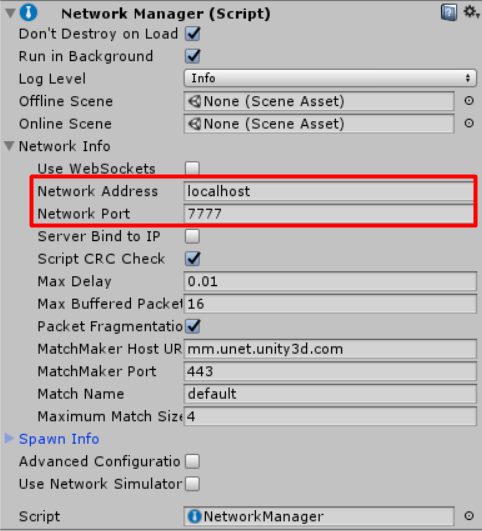
Jika Server telah siap atau client sudah terhubung dengan server, disable kedua tombol tersebut dan aktifkan tombol cancel.

1. Simpan dengan **Ctrl+S** sehingga seluruh kode sebagai berikut.
   1. using System.Collections;
   2. using System.Collections.Generic;
   3. using UnityEngine;
   4. using UnityEngine.UI;
   5. using UnityEngine.Networking;
   7. public class UINetwork : MonoBehaviour
   8. {
   9. GameObject panelKoneksi;
   10. Button btnHost;
   11. Button btnJoin;
   12. Button btnCancel;
   13. Text txInfo;
   14. NetworkManager network;
   16. // Use this for initialization
   17. void Start()
   18. {
   19. panelKoneksi = GameObject.Find("PanelKoneksi");
   20. panelKoneksi.transform.localPosition = Vector3.zero;
   21. btnHost = GameObject.Find("BtnHost").GetComponent<Button>();
   22. btnJoin = GameObject.Find("BtnJoin").GetComponent<Button>();
   23. btnCancel = GameObject.Find("BtnCancel").GetComponent<Button>();
   24. txInfo = GameObject.Find("Info").GetComponent<Text>();
   25. btnHost.onClick.AddListener(StartHostGame);
   26. btnJoin.onClick.AddListener(StartJoinGame);
   27. btnCancel.onClick.AddListener(CancelConnection);
   28. btnCancel.interactable = false;
   29. network = GameObject.Find("Network Manager").GetComponent<NetworkManager>();
   30. txInfo.text = "Info: Server Address " + network.networkAddress + " with port " + network.networkPort;
   31. }
   33. // Update is called once per frame
   34. void Update()
   35. {
   36. if (NetworkClient.active || NetworkServer.active)
   37. {
   38. btnHost.interactable = false;
   39. btnJoin.interactable = false;
   40. btnCancel.interactable = true;
   41. }
   42. else
   43. {
   44. btnHost.interactable = true;
   45. btnJoin.interactable = true;
   46. btnCancel.interactable = false;
   47. }
   48. }
   50. private void StartHostGame()
   51. {
   52. Debug.Log("ketika tombol Host ditekan");
   53. if (!NetworkServer.active)
   54. {
   55. network.StartHost();
   56. }
   57. if (NetworkServer.active) txInfo.text = "Info: Menunggu Player lain (Jika Server Aktif)";
   58. }
   60. private void StartJoinGame()
   61. {
   62. Debug.Log("ketika tombol Join ditekan");
   63. if (!NetworkClient.active)
   64. {
   65. network.StartClient();
   66. network.client.RegisterHandler(MsgType.Disconnect, ConnectionError);
   67. }
   68. if (NetworkClient.active) txInfo.text = "Info: Mencoba mengkoneksikan dengan Server";
   69. }
   71. private void CancelConnection()
   72. {
   73. Debug.Log("ketika tombol Cancel ditekan");
   74. network.StopHost();
   75. btnHost.interactable = true;
   76. btnJoin.interactable = true;
   77. btnCancel.interactable = false;
   78. txInfo.text = "Info: Server Address " + network.networkAddress + " with port " + network.networkPort;
   79. }
   81. private void ConnectionError(NetworkMessage netMsg)
   82. {
   83. network.StopClient();
   84. txInfo.text = "Info: Koneksi terputus dari Server";
   85. }
   86. }

### ****Praktik: Pengujian Multiplayer****

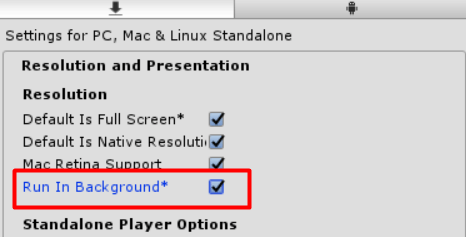
Kita akan menguji bagaimana menjalankan multiplayer yang telah kita buat dengan cara sebagai berikut.

1. Sebelum build, pastikan **IP**dan **Port**sudah sesuai dengan alamat Server Anda. Untuk mengentahui IP Address sebuah device, Anda dapat menggunakan kode sebagai berikut: **string ip = Network.player.ipAddress;** Kemudian Anda dapat mengisikan pada **Network Address**.



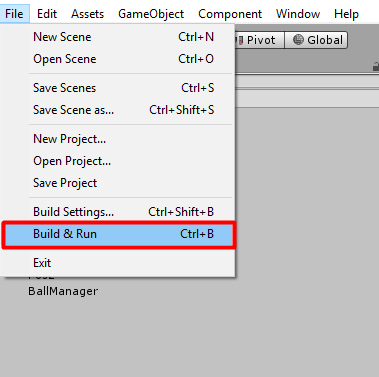
Alamat IP yang digunakan Localhost adalah **127.0.0.1**. Pastikan Port tidak sedangkan digunakan oleh aplikasi lain. Untuk mengetahui daftar port yang digunakan, Anda dapat membuka **Command Prompt (CTRL+R lalu ketik CMD)**kemudian ketik **netstat**lalu tekan **Enter**.

1. Supaya Aplikasi dapat berjalan di background maka centang **Run In Background** di **Player Settings** (**Menu Edit**> **Project Settings** > **Player**).

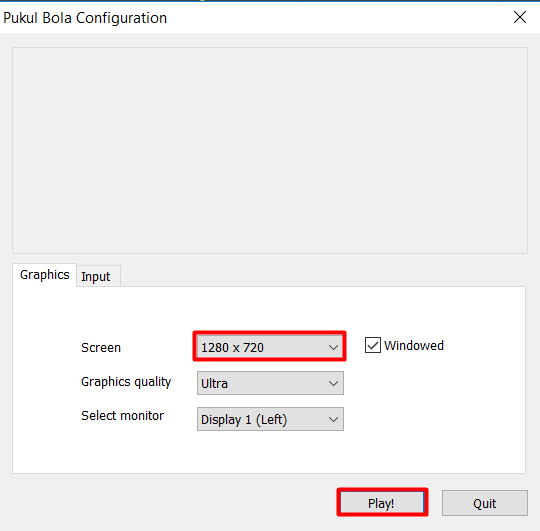


Aktifkan “**Run In Background**” seperti tertulis di atas. Tujuannya, agar ketika jendela game tidak dalam kondisi ter-select, aplikasi tetapi jalan. Dengan demikian, respon multiplayer akan lebih cepat ketika kita melakukan pengujian game.

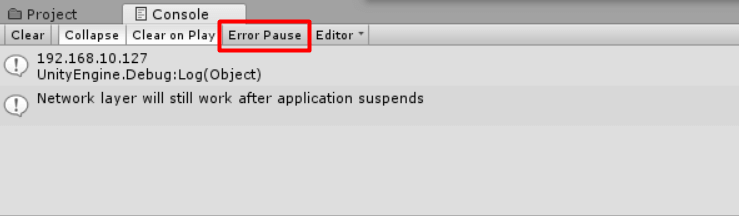
1. Build dan jalankan game dengan cara klik menu **File > Build & Run**.



1. Kemudian muncul jendela **Configuration**. Aturlah resolusi layar sesuai dengan resolusi game yang Anda buat. Kemudian pilih **Play.**



1. Lalu Game Anda berjalan secara stand-alone. Anda dapat masuk ke halaman Network.Kembali ke Unity, pada panel **Console**. Pastikan tombol **Error Pause** dalam posisi tidak aktif.



Ketika kita mencoba mengamankan koneksi namun kemudian terputus dari client, maka akan muncul kesalahan timeout yang menghentikan game Anda. Kesalahan seperti itu dapat Anda abaikan.

### ****Praktik: Masuk ke Halaman Permainan****

Kita telah memastikan script untuk mengelola network manager sudah berjalan dengan baik. Selanjutnya kita akan memberi akses network ke halaman permainan ketika dua-duanya (Stand-alone dengan Editor) saling terhubung.

* Membuat kondisi Host masuk ke permainan ketika ada Client yang terhubung.
* Membuat kondisi Client masuk ke permainan ketika sudah terhubung dengan server.
* Tekan Escape untuk keluar dari permainan dan kembali ke halaman Network.
* Kembali ke halaman Network ketika Client terputus dari Server.

Langkah-langkahnya sebagai berikut.

1. Buka script **UINetwork**, kemudian tambahkan kode sebagai berikut.
   * Tambahkan **Library.**
     1. using UnityEngine.SceneManagement;
   * Tambahkan **variabel.**
     1. int status = 0;

Digunakan untuk memberi kondisi apakah masih di halaman network atau ke halaman permainan.

* + Tambahkan **function.**
    1. public void MulaiPermainan ()
    2. {
    3. panelKoneksi.transform.localPosition = new Vector3 (-1500, 0, 0);
    4. }
    5. public void KembaliKeMain ()
    6. {
    7. network.StopHost ();
    8. SceneManager.LoadScene ("Main");
    9. }
    10. public void KembaliKeMenu ()
    11. {
    12. network.StopHost ();
    13. SceneManager.LoadScene ("Menu");
    14. }

Function **MulaiPermainan()**, mengubah posisi panel koneksi ke lebih jauh supaya tidak terlihat.  
Function **KembaliKeMain()** sebelum load kembali ke scene **Main (Halaman Network)**, pastikan koneksi tidak terhubung dengan server.

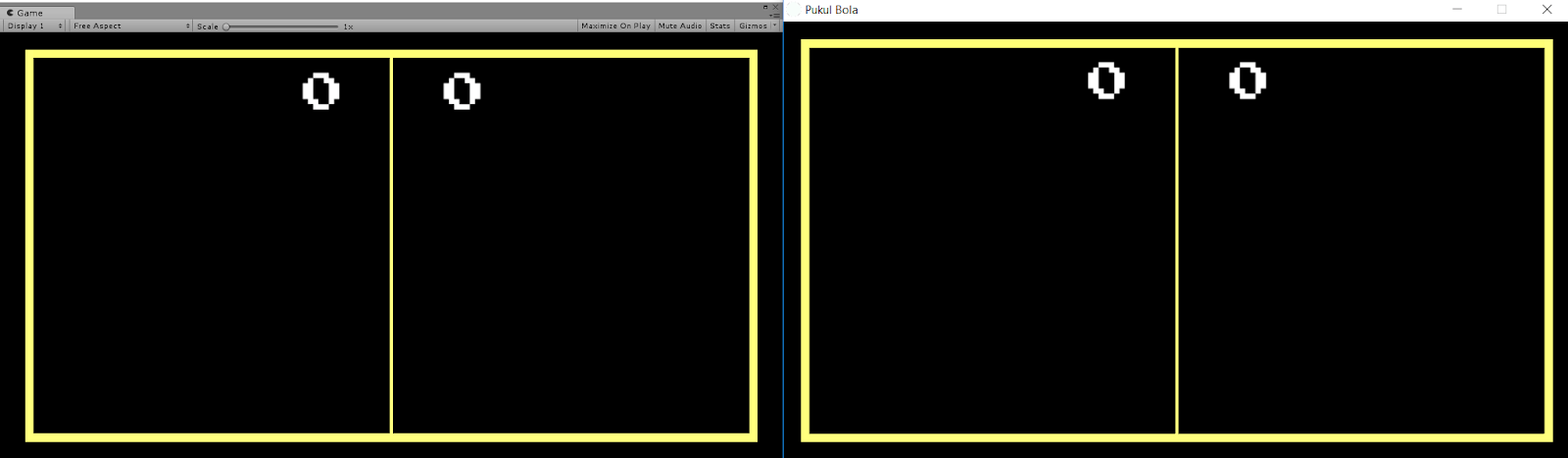
* + Tambahkan di function **Update.**
    1. if (NetworkServer.connections.Count == 2 && status == 0)
    2. {
    3. status = 1;
    4. MulaiPermainan();
    5. }
    6. if (ClientScene.ready && !NetworkServer.active && status == 0)
    7. {
    8. status = 1;
    9. MulaiPermainan();
    10. }
    11. if (Input.GetKeyUp(KeyCode.Escape))
    12. {
    13. KembaliKeMenu();
    14. }

**if (NetworkServer.connections.Count == 2 && status == 0),**kondisi ini hanya dijalankan oleh Server atau Host. Karena kita menggunakan Host maka juga dihitung sebagai client. Ketika ada client baru maka terdapat total 2 connection. Setelah kondisi terpenuhi maka masuk ke permainan.  
**if (ClientScene.ready && !NetworkServer.active && status == 0),**kondisi ini hanya dijalankan oleh Client. Ketika Client terhubung dengan server maka masuk ke permainan.  
**if (Input.GetKeyUp(KeyCode.Escape))**, ketika menekan tombol escape pada keyboad maka client kembali ke halaman menu.

* + Ubah kode di dalam function **ConnectionError**.
    1. //network.StopClient();
    2. //txInfo.text = "Info: Koneksi terputus dari Server";
    3. KembaliKeMain();

Saat menjalankan function **KembaliKeMain()** yang terdapat load ulang scene main. Maka data-data sebelumnya tidak diperlukan lagi kecuali Anda menyimpan nilai tersebut di variable static.

1. Simpan dengan **Ctrl+S**, **sehingga seluruh kode menjadi seperti ini:**
   1. using System.Collections;
   2. using System.Collections.Generic;
   3. using UnityEngine;
   4. using UnityEngine.UI;
   5. using UnityEngine.Networking;
   6. using UnityEngine.SceneManagement;
   8. public class UINetwork : MonoBehaviour
   9. {
   10. GameObject panelKoneksi;
   11. Button btnHost;
   12. Button btnJoin;
   13. Button btnCancel;
   14. Text txInfo;
   15. NetworkManager network;
   16. int status = 0;
   18. // Use this for initialization
   19. void Start()
   20. {
   21. panelKoneksi = GameObject.Find("PanelKoneksi");
   22. panelKoneksi.transform.localPosition = Vector3.zero;
   23. btnHost = GameObject.Find("BtnHost").GetComponent<Button>();
   24. btnJoin = GameObject.Find("BtnJoin").GetComponent<Button>();
   25. btnCancel = GameObject.Find("BtnCancel").GetComponent<Button>();
   26. txInfo = GameObject.Find("Info").GetComponent<Text>();
   27. btnHost.onClick.AddListener(StartHostGame);
   28. btnJoin.onClick.AddListener(StartJoinGame);
   29. btnCancel.onClick.AddListener(CancelConnection);
   30. btnCancel.interactable = false;
   31. network = GameObject.Find("Network Manager").GetComponent<NetworkManager>();
   32. txInfo.text = "Info: Server Address " + network.networkAddress + " with port " + network.networkPort;
   33. string ip = Network.player.ipAddress;
   34. Debug.Log(ip);
   35. }
   37. // Update is called once per frame
   38. void Update()
   39. {
   40. if (NetworkClient.active || NetworkServer.active)
   41. {
   42. btnHost.interactable = false;
   43. btnJoin.interactable = false;
   44. btnCancel.interactable = true;
   45. }
   46. else
   47. {
   48. btnHost.interactable = true;
   49. btnJoin.interactable = true;
   50. btnCancel.interactable = false;
   51. }
   52. if (NetworkServer.connections.Count == 2 && status == 0)
   53. {
   54. status = 1;
   55. MulaiPermainan();
   56. }
   57. if (ClientScene.ready && !NetworkServer.active && status == 0)
   58. {
   59. status = 1;
   60. MulaiPermainan();
   61. }
   62. if (Input.GetKeyUp(KeyCode.Escape))
   63. {
   64. KembaliKeMenu();
   65. }
   66. }
   68. private void StartHostGame()
   69. {
   70. Debug.Log("ketika tombol Host ditekan");
   71. if (!NetworkServer.active)
   72. {
   73. network.StartHost();
   74. }
   75. if (NetworkServer.active)
   76. txInfo.text = "Info: Menunggu Player lain (Jika Server Aktif)";
   77. }
   79. private void StartJoinGame()
   80. {
   81. Debug.Log("ketika tombol Join ditekan");
   82. if (!NetworkClient.active)
   83. {
   84. network.StartClient();
   85. network.client.RegisterHandler(MsgType.Disconnect, ConnectionError);
   86. }
   87. if (NetworkClient.active)
   88. txInfo.text = "Info: Mencoba mengkoneksikan dengan Server";
   89. }
   91. private void CancelConnection()
   92. {
   93. Debug.Log("ketika tombol Cancel ditekan");
   94. if (NetworkServer.active)
   95. network.StopHost();
   96. if (NetworkClient.active)
   97. network.StopClient();
   98. btnHost.interactable = true;
   99. btnJoin.interactable = true;
   100. btnCancel.interactable = false;
   101. txInfo.text = "Info: Server Address " + network.networkAddress + " with port " + network.networkPort;
   102. }
   104. private void ConnectionError(NetworkMessage netMsg)
   105. {
   106. //network.StopClient();
   107. //txInfo.text = "Info: Koneksi terputus dari Server";
   108. KembaliKeMain();
   109. }
   111. public void MulaiPermainan()
   112. {
   113. panelKoneksi.transform.localPosition = new Vector3(-1500, 0, 0);
   114. }
   116. public void KembaliKeMain()
   117. {
   118. network.StopHost();
   119. SceneManager.LoadScene("Main");
   120. }
   122. public void KembaliKeMenu()
   123. {
   124. network.StopHost();
   125. SceneManager.LoadScene("Menu");
   126. }
   127. }
2. Kalau ingin mencoba lagi, tutup lebih dahulu aplikasi yang sedang berjalan secara stand-alone, kemudian ulangi langkah praktik di atas. Jika sama-sama terkoneksi, maka akan masuk ke halaman permainan. Karena kita belum menambahkan objek player maka hanya muncul **Area**dan **Text Score**seperti di bawah ini.

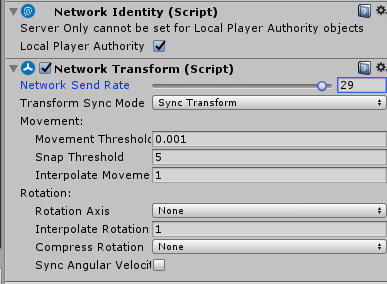


Sekarang Anda sudah bisa mengatur UI Connection yang dapat masuk ke dalam halaman permainan. Selanjutnya, kita akan mengisi halaman permainan tersebut dengan objek player yaitu **GameObject Paddle**.

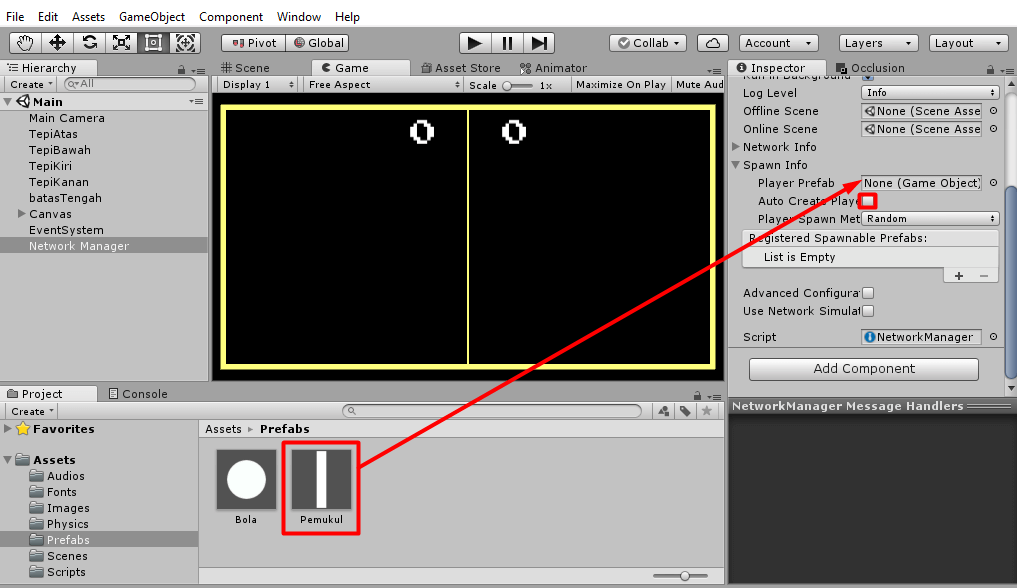
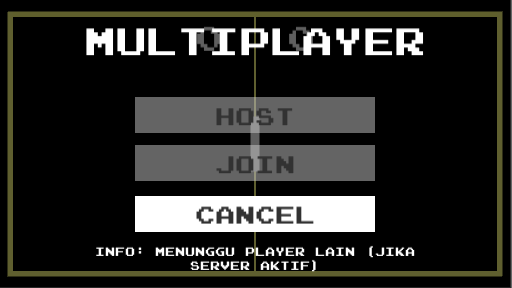
### ****Praktik: Menambahkan Komponen pada Pemukul****

Sebelum kita memasukkan prefab Pemukul ke Network Manager, kita harus menambahkan beberapa komponen yang diperlukan oleh Pemukul. Langkah-langkahnya sebagai berikut:

1. Klik file prefab **Pemukul**yang terdapat di folder **Prefabs**.
2. Tambahkan komponen **Network Identity**(**Add Component** > **Network**>**Network Identity**), lalu centang **Local Player Authority.**
3. Tambahkan komponen**Network Transfrom**(**Add Component** > **Network**>**Network Identity**) lalu atur Nilai Network Send Rate menjadi **29** dan **Rotation. .Axis** menjadi **None**. Sehingga, tampilan kedua komponen sebagai berikut:

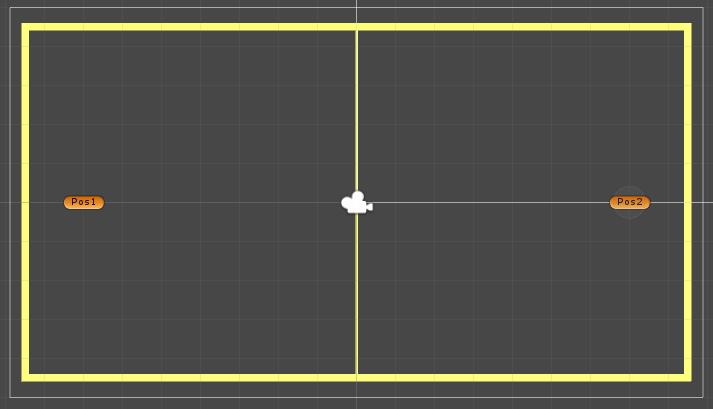


Network Sent Rate adalah berapa kali data paddle melakukan penyesuaian dengan server selama 1 detik. Kita mengisi nilai 29 yang artinya selama 1 detik maksimal data paddle melakukan penyesuaian hingga sebanyak 29 kali.

1. Kemudian tambahkan prefab **Pemukul**ke field **Player Prefab** dan jangan lupa untuk centang**Auto Create Player.**  
   ****
2. Silakan Anda jalankan kemudian aktifkan Host. Paddle berada di posisi tengah-tengah.

### ****Praktik: Mengatur Posisi Pemukul****

Jika kita tidak mengatur posisi maka posisi paddle berada di titik 0. Langkah-langkahnya sebagai berikut:

1. Buat gameobject kosong dan beri nama **Pos1**dan tambahkan **Icon Label** (supaya terlihat di panel Scene).
2. Tambahkan Komponen **Network Start Position .**
3. Pada **Pos1**, atur posisi X:**-7**, Y: 0, Z: 0. Sedangkan untuk rotasi**0** dan untuk scale**1**.
4. Duplikasi Pos1 kemudian ubah nama menjadi**Pos2.`**
5. Pada **Pos2**, atur posisi menjadi X:**7**, Y: 0, Z: 0. Sedangkan untuk rotasi**0** dan untuk scale **1.**
6. Klik gameobject Network Manager. Pada Insepector atur **Player Spawn Method** yang terdapat di**Spawn Info**menjadi **Round Robin**.

### ****Praktik: Mengatur Gerakan Pemukul (Player)****

Ketika Anda gerakkan dengan keyboard maka kedua-duanya bergerak secara bersamaan. Untuk mengatasi hal demikian, Anda perlu melakukan beberapa hal sebagai berikut:

1. Buka Script **PaddleController.cs**, Kemudian tambahkan code sebagai berikut:
   1. Menambahkan Library.
      1. **using UnityEngine.Networking;**

Digunakan untuk menyediakan method-method yang berhubungan dengan networking.

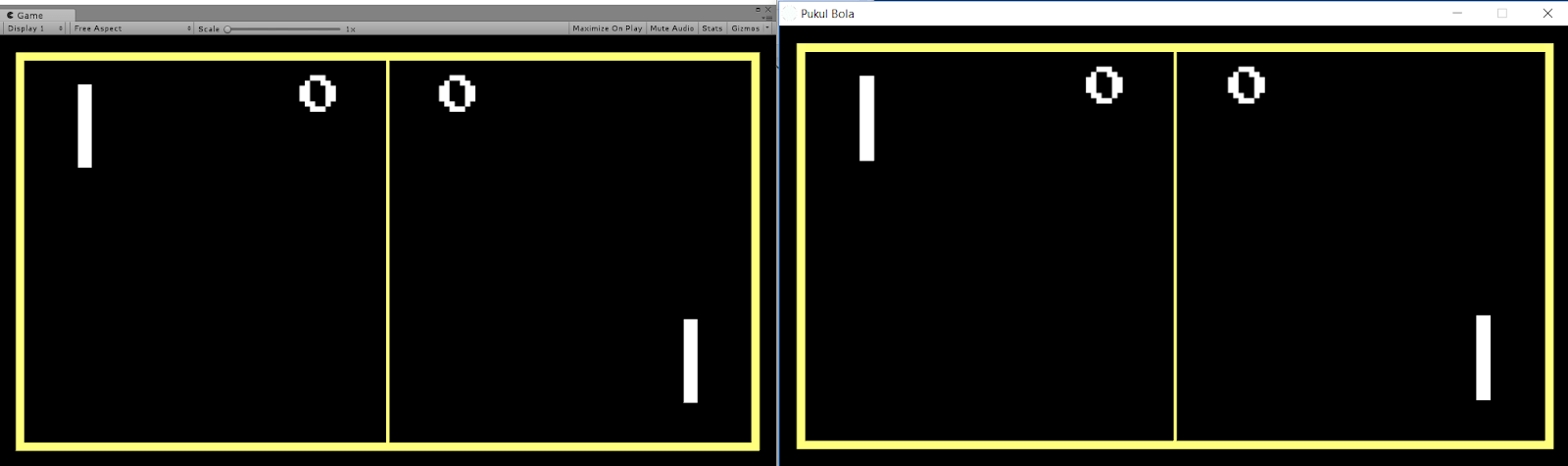
* 1. Ubah kelas turunan dari **MonoBehavior** menjadi **NetworkBehaviour.**

NetworkBehavior merupakan turunan dari MonoBehaviour. Tetapi memiliki perbedaan dengan penambahan-penambahan method untuk networking.

* 1. Tambahkan kode ini di dalam function **Update.**
     1. **if (!isLocalPlayer) return;**

Digunakan untuk membatasi supaya tidak menggerakkan 2 pemukul sekaligus. Sehingga jika Player1 menjadi pemain utama maka dia hanya menggerakkan player tersebut.

1. Simpan dengan **Ctrl+S**sehingga seluruh kode sebagai berikut:
   1. using UnityEngine;
   2. using UnityEngine.Networking;
   3. public class PaddleController : NetworkBehaviour
   4. {
   5. public float batasAtas;
   6. public float batasBawah;
   7. public float kecepatan;
   8. public string axis;
   9. // Use this for initialization
   10. void Start()
   11. {
   12. }
   13. // Update is called once per frame
   14. void Update()
   15. {
   16. if (!isLocalPlayer)
   17. return;
   18. float gerak = GetInputPC();
   19. float nextPos = transform.position.y + gerak;
   20. if (nextPos > batasAtas)
   21. {
   22. gerak = 0;
   23. }
   24. if (nextPos < batasBawah)
   25. {
   26. gerak = 0;
   27. }
   28. transform.Translate(0, gerak, 0);
   29. }
   30. float GetInputPC()
   31. {
   32. return Input.GetAxis(axis) \* kecepatan \* Time.deltaTime;
   33. }
   34. }
2. Kemudian kalankan sehingga hasilnya sebagai berikut:



### ****Praktik: Mengatur Identitas Player****

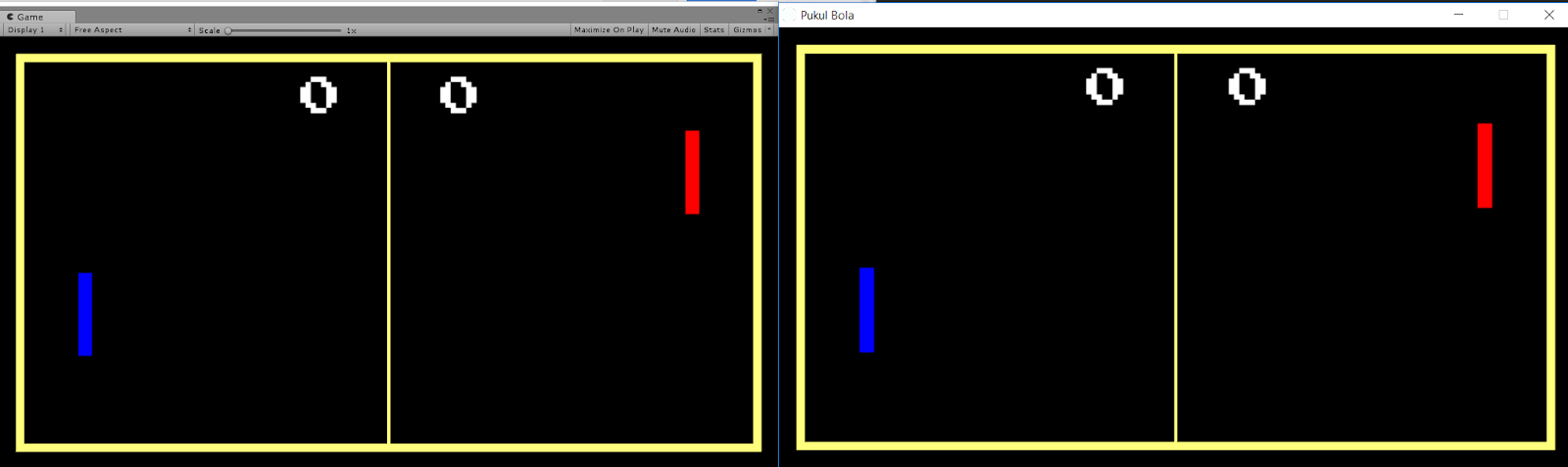
Anda dapat membedakan pemukul dengan menggunakan warna yang berbeda. Langkah-langkahnya sebagai berikut:

1. Buka Script**PaddleController.cs** kemudian tambahkan function berikut.
   1. private void Awake()
   2. {
   3. if (transform.position.x > 0) transform.GetComponent<SpriteRenderer>().color = Color.red;
   4. else transform.GetComponent<SpriteRenderer>().color = Color.blue;

**Awake(),**adalah sebuah function yang dijalankan sebelum start dan update.

**Baris 3-4**,  Jika posisi object tersebut sebelah kanan maka akan berwarna merah dan jika posisi object tersebut sebelah kiri maka akan berwarna biru.

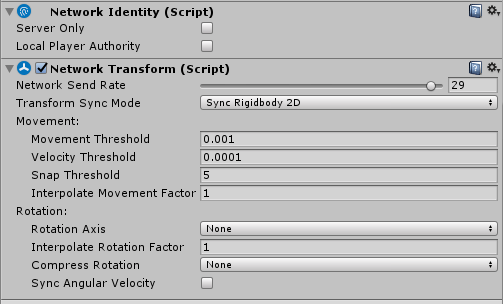
1. Simpan dengan **Ctrl+S,** sehingga seluruh kode sebagai berikut:
   1. using System.Collections;
   2. using System.Collections.Generic;
   3. using UnityEngine;
   4. using UnityEngine.Networking;
   5. public class PaddleController : NetworkBehaviour
   6. {
   7. public float batasAtas;
   8. public float batasBawah;
   9. public float kecepatan;
   10. public string axis;
   11. private void Awake()
   12. {
   13. if (transform.position.x > 0) transform.GetComponent<SpriteRenderer>().color = Color.red;
   14. else transform.GetComponent<SpriteRenderer>().color = Color.blue;
   15. }
   16. // Use this for initialization
   17. void Start()
   18. {
   20. }
   22. // Update is called once per frame
   23. void Update()
   24. {
   25. if (!isLocalPlayer) return;
   26. float gerak = GetInputPC();
   27. float nextPos = transform.position.y + gerak;
   28. if (nextPos > batasAtas)
   29. {
   30. gerak = 0;
   31. }
   32. if (nextPos < batasBawah)
   33. {
   34. gerak = 0;
   35. }
   36. transform.Translate(0, gerak, 0);
   37. }
   38. float GetInputPC()
   39. {
   40. return Input.GetAxis(axis) \* kecepatan \* Time.deltaTime;
   41. }
   42. }
2. Seperti ini hingga hasilnya ketika dijalankan.



Tips: Jika ada ingin menambahkan tanda untuk player mana yang dapat Anda gerakan, Anda dapat memanfaatkan function  public override void OnStartLocalPlayer().

### ****Praktik: Menampilkan Bola****

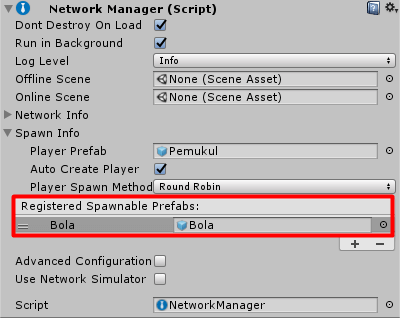
Sebelum menambahkan ke Register Spawn Prefabs, tambahkan beberapa komponen terlebih dahulu.

1. Klik berkas prefab **Bola**yang terdapat di folder **Prefabs**.
2. Tambahkan komponen **Network Identity**(**Add Component** > **Network** > **Network Identity**), tanpa ada yang dicentang. Karena gameobject bola akan diproses di server dan juga akan dimunculkan di client.
3. Tambahkan komponen**Network Transfrom**(**Add Component** > **Network**> **Network Identity**) lalu atur Nilai Network Send Rate menjadi **29, Transform Sync Mode**menjadi**Sync Rigidbody 2D** dan **Rotation Axis** menjadi **None.**
4. Sehingga tampilan kedua komponen menjadi seperti ini: Catatan: KDi game FPS objek peluru bergerak lurus dan memliki kecepatan berdasarkan lontaran (kecepatan yang sama) sehingga peluru tidak memerlukan nilai Rate yang lebih tinggi. Cukup diberi nilai 0. Hal ini juga dimaksudkan agar game terlihat lebih ringan dan tidak patah-patah.

### ****Praktik: Mendaftarkan Bola Prefabs ke Network Manager****

Setiap objek yang akan dibuat, harus didaftarkan terlebih dahulu di NetworkManager. Langkah-langkahnya sebagai berikut:

1. Klik GameObject **Network Manager** di **Hierarchy.**
2. Buka **Inspector.**
3. Tambahkan **Prefab Bola**ke **Registered Spawnable Prefabs.**



### ****Praktik: Membuat GameObject untuk Menampilkan Bola****

1. Buat GameObject kosong dengan klik menu **GameObject**>**Create Empty.**
2. Ganti Nama**GameObject** dengan **Bola Manager.**
3. Tambahkan Component > Klik **Add Component**> **Network**> **Network Identity.**Kemudian centang **Server Only**.

IMG_258

Kita perlu men-generate cukup 1 kali bola, tidak lebih. Oleh karena itu, kita menggunakan server yang hanya ada satu, untuk melakukan perintah ini. Jangan menggunakan client, karena jumlahnya ada lebih dari satu, sementara jumlah bola yang di-generate pun akan mengikuti jumlah client.

### ****Praktik: Membuat Kode untuk Menampilkan Bola****

Buka Folder Scripts, kemudian buat script C# baru dengan nama**BolaManager.cs**

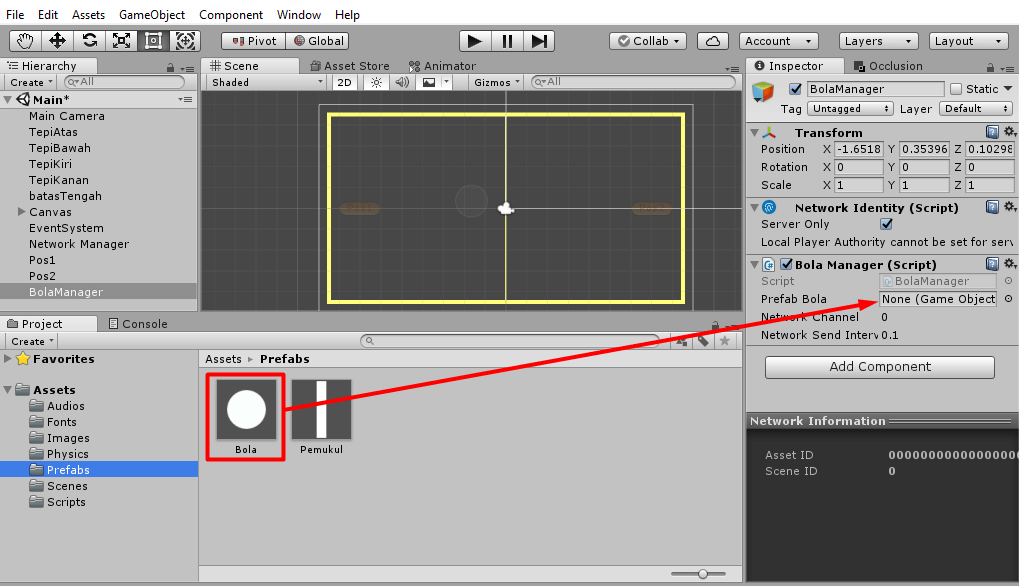
1. Buka Script **BolaManager.cs** dengan klik 2 kali.
2. Tambahkan kode di bawah ini:
   1. Menambahkan Library **using UnityEngine.Networking.**
   2. Mengganti **MonoBehaviour**menjadi **NetworkBehaviour.**
   3. Menambahkan variabel sebagai berikut:
      1. public GameObject prefabBola;
      2. bool bolaMuncul = false;
      3. GameObject bola;

4. Kemudian masukkan kode di bawah ini ke dalam function Update().

* + 1. if (!isServer || bolaMuncul)
    2. return;
    3. if (NetworkServer.connections.Count == 2)
    4. {
    5. bola = (GameObject)Instantiate(prefabBola);
    6. NetworkServer.Spawn(bola);
    7. bolaMuncul = true;
    8. }

Variabel **isServer** digunakan untuk membatasi kode yang dijalankan hanya di Server. Sedangkan variable**bolaMuncul** digunakan untuk membatasi supaya bola muncul sekali saja ketika menerima perintah.  
**if (NetworkServer.connections.Count == 2),** ketika terdapat dua player maka permainan dimulai  
**bola = (GameObject)Instantiate (prefabBola);** dan **NetworkServer. Spawn (bola);** digunakan untuk memunculkan bola ke seluruh client dan server

1. Simpan perubahan kode dengan**Ctrl+S**sehingga seluruh kode menjadi seperti berikut:
   1. using UnityEngine;
   2. using UnityEngine.Networking;
   3. public class BolaManager : NetworkBehaviour
   4. {
   5. public GameObject prefabBola;
   6. bool bolaMuncul = false;
   7. GameObject bola;
   8. // Use this for initialization
   9. void Start()
   10. {
   11. }
   12. // Update is called once per frame
   13. void Update()
   14. {
   15. if (!isServer || bolaMuncul)
   16. return;
   17. if (NetworkServer.connections.Count == 2)
   18. {
   19. bola = (GameObject)Instantiate(prefabBola);
   20. NetworkServer.Spawn(bola);
   21. bolaMuncul = true;
   22. }
   23. }
   24. }
2. Masukkan script **BolaManager.cs** ke gameobject **BolaManager**di Hierarchy
3. Kemudian **Inspector**pada **BolaManager**, masukkan prefab **Bola**ke field **Prefab Bola**.



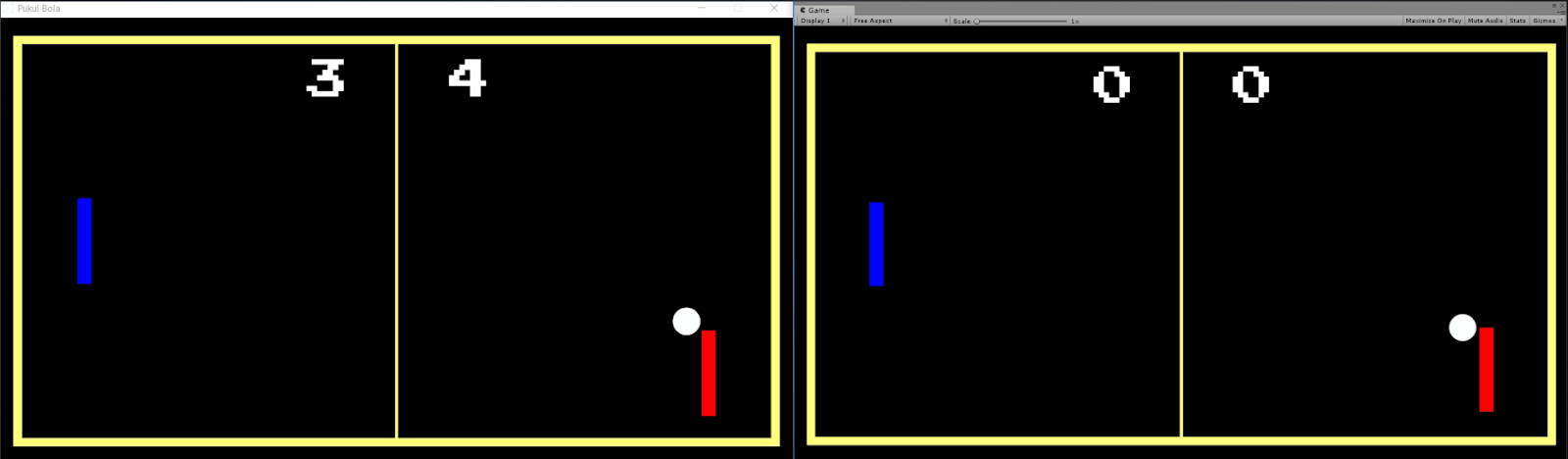
### ****Praktik: Mengatur Gerakan Bola****

Sekarang ini Bola mendapat respon dari 3 inputan yaitu Server, Client1 dan Client2. Supaya tidak ada proses ganda maka kita perlu membatasi dengan 1 inputan saja. Pergerakkan bola hanya diproses di server karena paling dekat dengan Client dari pada dari client ke client. Langkah-langkahnya sebagai berikut:

1. Buka Script **BallController.cs**kemudian tambahkan kode.
   1. Tambahkan Library.
      1. **using UnityEngine.Networking;**
   2. Ubah kelas turunan dari **MonoBehavior**menjadi **NetworkBehaviour.**
   3. Tambahkan paling atas di function **OnCollisionEnter2D.**
      1. **if (!isServer) return;**

## ****Menampilkan Skor****

 Masalah sebelumnya tentang nilai skor, hanya muncul di server/host saja, namun tidak muncul di Client. Hal ini terjadi karena kode hanya dijalankan di server.



Anda dapat menyamakan nilai variabel dengan menggunakan **SyncVar**. Walaupun proses perhitungan atau proses yang lain pada bola dilakukan oleh server, Anda masih dapat memanfaatkan fungsi **SyncVar**supaya seluruh nilai variabel, sama. Modul ini akan membahas cara penerapannya.

### ****Praktik: Mengubah Code BallController.cs****

1. Buka kode BallController.cs kemudian hapus beberapa kode berikut.
   1. Hapus kode**TampilkanScore();** di dalam **OnCollisionEnter2D().**
   2. Hapus juga function.
      1. void TampilkanScore()
      2. {
      3. Debug.Log("Score P1: " + scoreP1 + " Score P2: " + scoreP2);
      4. scoreUIP1.text = scoreP1 + "";
      5. scoreUIP2.text = scoreP2 + "";
      6. }
2. Kemudian ubah atau tambahkan kode berikut:
   1. Ubah variabel **scoreP1**dan **scoreP2.**
      1. [SyncVar (hook = "OnChangeScore1")]
      2. public int scoreP1;
      3. [SyncVar (hook = "OnChangeScore2")]
      4. public int scoreP2;

**[SyncVar(hook = "OnChangeScore1")],**Digunakan untuk menyamakan nilai pada variabel di bawahnya. Kemudian ketika berganti, nilai tersebut akan dikirim ke function OnChangeScore1  sehingga perubahan terlihat cepat. Anda juga bisa melakukan persamaan dengan hanya menambahkan [SyncVar].

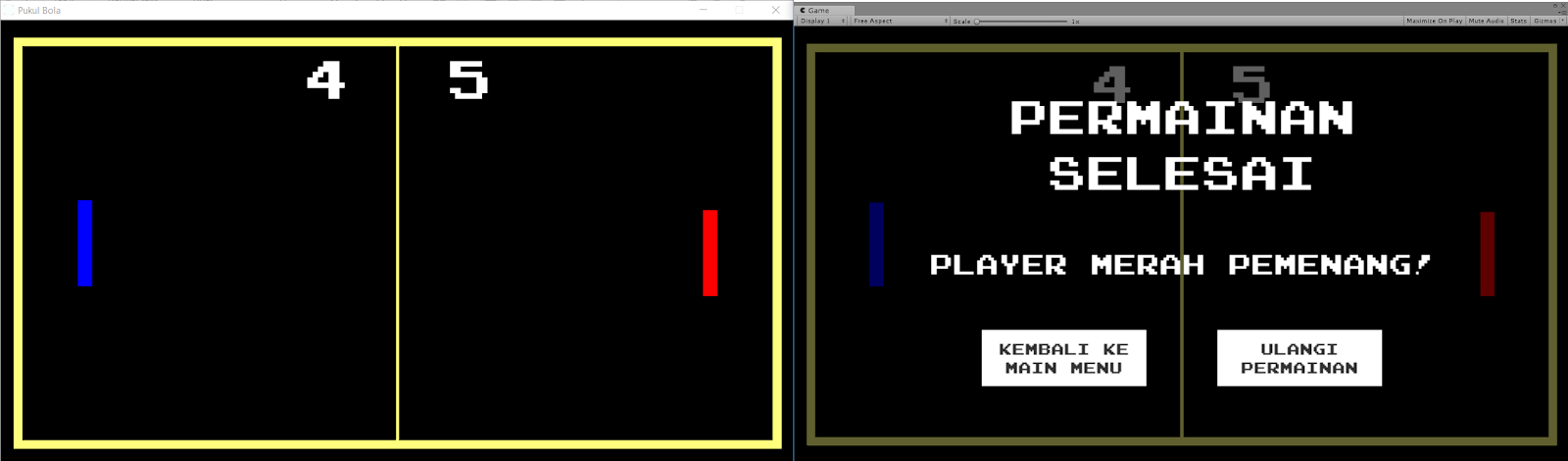
* 1. Tambahkan function berikut di dalam class**BallController.cs**
     1. void OnChangeScore1(int score)
     2. {
     3. if (scoreUIP1 != null)
     4. scoreUIP1.GetComponent<Text>().text = "" + score;
     5. }
     6. void OnChangeScore2(int score)
     7. {
     8. if (scoreUIP2 != null)
     9. scoreUIP2.GetComponent<Text>().text = "" + score;
     10. }

Nilai yang dikirim akan langsung ditampilkan di halaman permainan.

1. Simpan perubahan dengan **Ctrl+S**sehingga seluruh kode sebagai berikut:
   1. using System.Collections;
   2. using System.Collections.Generic;
   3. using UnityEngine;
   4. using UnityEngine.UI;
   5. using UnityEngine.Networking;
   6. public class BallController : NetworkBehaviour
   7. {
   8. public int force;
   9. Rigidbody2D rigid;
   10. [SyncVar(hook = "OnChangeScore1")]
   11. public int scoreP1;
   12. [SyncVar(hook = "OnChangeScore2")]
   13. public int scoreP2;
   14. Text scoreUIP1;
   15. Text scoreUIP2;
   16. GameObject panelSelesai;
   17. Text txPemenang;
   18. AudioSource audio;
   19. public AudioClip hitSound;
   20. // Use this for initialization
   21. void Start()
   22. {
   23. rigid = GetComponent<Rigidbody2D>();
   24. Vector2 arah = new Vector2(2, 0).normalized;
   25. rigid.AddForce(arah \* force);
   26. scoreP1 = 0;
   27. scoreP2 = 0;
   28. scoreUIP1 = GameObject.Find("Score1").GetComponent<Text>();
   29. scoreUIP2 = GameObject.Find("Score2").GetComponent<Text>();
   30. panelSelesai = GameObject.Find("PanelSelesai");
   31. audio = GetComponent<AudioSource>();
   32. }
   33. // Update is called once per frame
   34. void Update()
   35. {
   36. }
   37. private void OnCollisionEnter2D(Collision2D coll)
   38. {
   39. if (!isServer)
   40. return;
   41. if (coll.gameObject.name.Contains("Pemukul"))
   42. {
   43. float sudut = (transform.position.y - coll.transform.position.y) \* 5f;
   44. Vector2 arah = new Vector2(rigid.velocity.x, sudut).normalized;
   45. rigid.velocity = new Vector2(0, 0);
   46. rigid.AddForce(arah \* force \* 2);
   47. }
   48. else if (coll.gameObject.name == "TepiKanan")
   49. {
   50. scoreP1 += 1;
   51. if (scoreP1 == 5)
   52. {
   53. TampilanSelesai("Biru");
   54. return;
   55. }
   56. ResetBall();
   57. Vector2 arah = new Vector2(2, 0).normalized;
   58. rigid.AddForce(arah \* force);
   59. }
   60. else if (coll.gameObject.name == "TepiKiri")
   61. {
   62. scoreP2 += 1;
   63. if (scoreP2 == 5)
   64. {
   65. TampilanSelesai("Merah");
   66. return;
   67. }
   68. ResetBall();
   69. Vector2 arah = new Vector2(-2, 0).normalized;
   70. rigid.AddForce(arah \* force);
   71. }
   72. audio.PlayOneShot(hitSound);
   73. }
   74. void TampilanSelesai(string warna)
   75. {
   76. panelSelesai.transform.localPosition = Vector3.zero;
   77. txPemenang = GameObject.Find("Pemenang").GetComponent<Text>();
   78. txPemenang.text = "Player " + warna + " Pemenang!";
   79. gameObject.SetActive(false);
   80. }
   81. void ResetBall()
   82. {
   83. transform.localPosition = new Vector2(0, 0);
   84. rigid.velocity = new Vector2(0, 0);
   85. }
   86. void OnChangeScore1(int score)
   87. {
   88. if (scoreUIP1 != null)
   89. scoreUIP1.GetComponent<Text>().text = "" + score;
   90. }
   91. void OnChangeScore2(int score)
   92. {
   93. if (scoreUIP2 != null)
   94. scoreUIP2.GetComponent<Text>().text = "" + score;
   95. }
   96. }

## ****Menampilkan Halaman Selesai****

Di modul sebelumnya, halaman selesai hanya muncul pada server/host, tidak di Client, seperti terlihat di bawah ini:



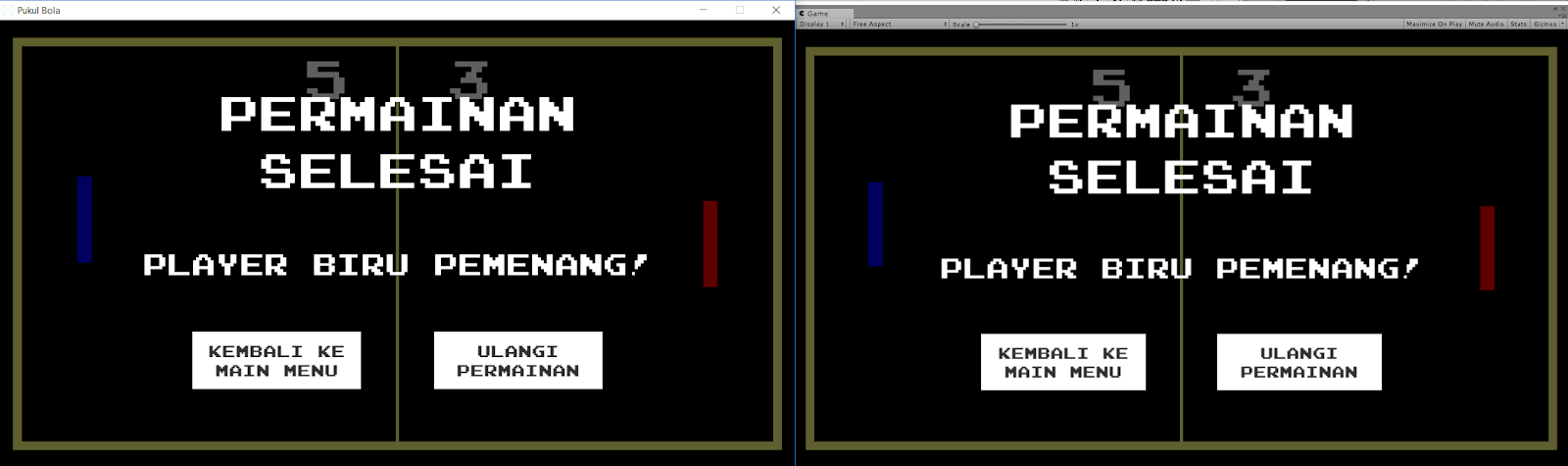
Masalah tersebut terjadi karena kode hanya dijalankan di server, sehingga halaman selesai yang ditampilkan hanya di Host. Pada submodul ini kita akan membahas tentang **[ClientRpc]**.

Kapan kita menggunakan **[ClientRpc]?** --> Ketika sesuatu dijalankan di server tetapi hasilnya diimplementasikan ke semua Client.

Submodul ini akan membahas implementasinya.

### ****Praktik: Menampilkan Halaman Selesai****

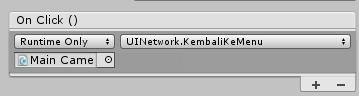
Supaya function untuk menampilkan halaman Selesai dapat diimplementasi di Client, lakukan langkah-langkah berikut:

1. Buka Script **BallController.cs** kemudian ubah scriptnya sebagai berikut:
   1. Ubah nama function void**TampilanSelesai(string warna)** menjadi**RpcTampilanSelesai (string warna)**. Setiap attribute [ClientRpc] digunakan, maka setiap nama function harus diawali dengan kata Rpc. Jangan lupa untuk mengganti juga setiap penggunaan function tersebut
   2. Tambahkan attribute**[ClientRpc]** di atas function RpcTampilanSelesai (string warna).
   3. Simpan Perubahan dengan menekan Tombol **Ctrl+S.**
2. Sehingga ketika dijalankan, Halaman Selesai akan muncul secara bersamaan. Jika game Anda berakhir seperti di atas maka Anda telah sukses menampilkan halaman selesai.

### ****Praktik: Mengaktifkan tombol ke Menu dan Ulangi Permainan****

Kedua tombol dalam posisi belum aktif. Untuk mengaktifkan, ikuti langkah-langkah berikut:

1. Klik gameobject **BtnMenu**. Kemudian pada list **On Click()**, tambahkan Main Camera dan masukkan function **UINetwor** > **KembaliKeMenu.**



1. Klik gameobject **BtnMain**. Kemudian pada list **On Click()**, tambahkan Main Camera dan masukkan function **UINetwork**> **KembaliKeMain.**



### ****Praktik: Respon ketika Client Terputus****

Ketika Host kehilangan Client maka permainan yang terdapat di Host terus berlanjut sampai permainan selesai. Jika ingin menyudahi permainan ketika Client terputus, Anda dapat melakukan beberapa langkah sebagai berikut:

1. Klik prefab Pemukul di Folder **Prefabs**. Pada Tag di **Inspector**, Tambahkan **Tag Player**.
2. Buka Script **BallController.cs** lalu tambahkan kode sebagai berikut:
   1. Tambahkan **Function**di dalam class **BallController.cs.**
      1. void ClientDisconnect ()
      2. {
      3. GameObject.Find ("Main Camera").SendMessage ("KembaliKeMain");
      4. }

Cari GameObject yang memiliki nama MainCamera yang terdapat di Hierarchy. Kemudian kirim atau panggil seluruh komponen script yang memiliki Method KembaliKeMain. Function ini tidak memerlukan Rpc karena function ini hanya diperuntukkan sebagai server.

* 1. Tambahkan kode di dalam function **OnCollisionEnter2D().**
     1. if (GameObject.FindGameObjectsWithTag ("Player").Length == 1)
     2. {
     3. ClientDisconnect ();
     4. }

Cari gameobject di hierarchy yang memiliki nama Tag Player. Jika jumlah gameobject tersebut hanya satu (hanya milik Host) maka permainan selesai dengan menjalankan function ClientDisconnect().

* 1. Simpan perubahan dengan Tekan Ctrl+S.

1. Sehingga seluruh kode seperti berikut ini:
   1. using System.Collections;
   2. using System.Collections.Generic;
   3. using UnityEngine;
   4. using UnityEngine.UI;
   5. using UnityEngine.Networking;
   6. public class BallController : NetworkBehaviour
   7. {
   8. public int force;
   9. Rigidbody2D rigid;
   10. [SyncVar(hook = "OnChangeScore1")]
   11. public int scoreP1;
   12. [SyncVar(hook = "OnChangeScore2")]
   13. public int scoreP2;
   14. Text scoreUIP1;
   15. Text scoreUIP2;
   16. GameObject panelSelesai;
   17. Text txPemenang;
   18. AudioSource audio;
   19. public AudioClip hitSound;
   20. // Use this for initialization
   21. void Start()
   22. {
   23. rigid = GetComponent<Rigidbody2D>();
   24. Vector2 arah = new Vector2(2, 0).normalized;
   25. rigid.AddForce(arah \* force);
   26. scoreP1 = 0;
   27. scoreP2 = 0;
   28. scoreUIP1 = GameObject.Find("Score1").GetComponent<Text>();
   29. scoreUIP2 = GameObject.Find("Score2").GetComponent<Text>();
   30. panelSelesai = GameObject.Find("PanelSelesai");
   31. audio = GetComponent<AudioSource>();
   32. }
   33. // Update is called once per frame
   34. void Update()
   35. {
   36. }
   37. void OnChangeScore1(int score)
   38. {
   39. if (scoreUIP1 != null)
   40. scoreUIP1.GetComponent<Text>().text = "" + score;
   41. }
   42. void OnChangeScore2(int score)
   43. {
   44. if (scoreUIP2 != null)
   45. scoreUIP2.GetComponent<Text>().text = "" + score;
   46. }
   47. private void OnCollisionEnter2D(Collision2D coll)
   48. {
   49. if (!isServer)
   50. return;
   51. audio.PlayOneShot(hitSound);
   52. if (coll.gameObject.name.Contains("Pemukul"))
   53. {
   54. float sudut = (transform.position.y - coll.transform.position.y) \* 5f;
   55. Vector2 arah = new Vector2(rigid.velocity.x, sudut).normalized;
   56. rigid.velocity = new Vector2(0, 0);
   57. rigid.AddForce(arah \* force \* 2);
   58. }
   59. else if (coll.gameObject.name == "TepiKanan")
   60. {
   61. scoreP1 += 1;
   62. if (scoreP1 == 5)
   63. {
   64. RpcTampilanSelesai("Biru");
   65. return;
   66. }
   67. ResetBall();
   68. Vector2 arah = new Vector2(2, 0).normalized;
   69. rigid.AddForce(arah \* force);
   70. }
   71. else if (coll.gameObject.name == "TepiKiri")
   72. {
   73. scoreP2 += 1;
   74. if (scoreP2 == 5)
   75. {
   76. RpcTampilanSelesai("Merah");
   77. return;
   78. }
   79. ResetBall();
   80. Vector2 arah = new Vector2(-2, 0).normalized;
   81. rigid.AddForce(arah \* force);
   82. }
   83. **if (GameObject.FindGameObjectsWithTag("Player").Length == 1)**
   84. **{**
   85. **ClientDisconnect();**
   86. **}**
   87. }
   88. [ClientRpc]
   89. void RpcTampilanSelesai(string warna)
   90. {
   91. panelSelesai.transform.localPosition = Vector3.zero;
   92. txPemenang = GameObject.Find("Pemenang").GetComponent<Text>();
   93. txPemenang.text = "Player " + warna + " Pemenang!";
   94. gameObject.SetActive(false);
   95. }
   96. void ResetBall()
   97. {
   98. transform.localPosition = new Vector2(0, 0);
   99. rigid.velocity = new Vector2(0, 0);
   100. }
   101. **void ClientDisconnect()**
   102. **{**
   103. **GameObject.Find("Main Camera").SendMessage("KembaliKeMain");**
   104. **}**
   105. }

Jika dijalankan dan kemudian salah satu Client terputus,, akan muncul halaman Network kita.

