Match Three

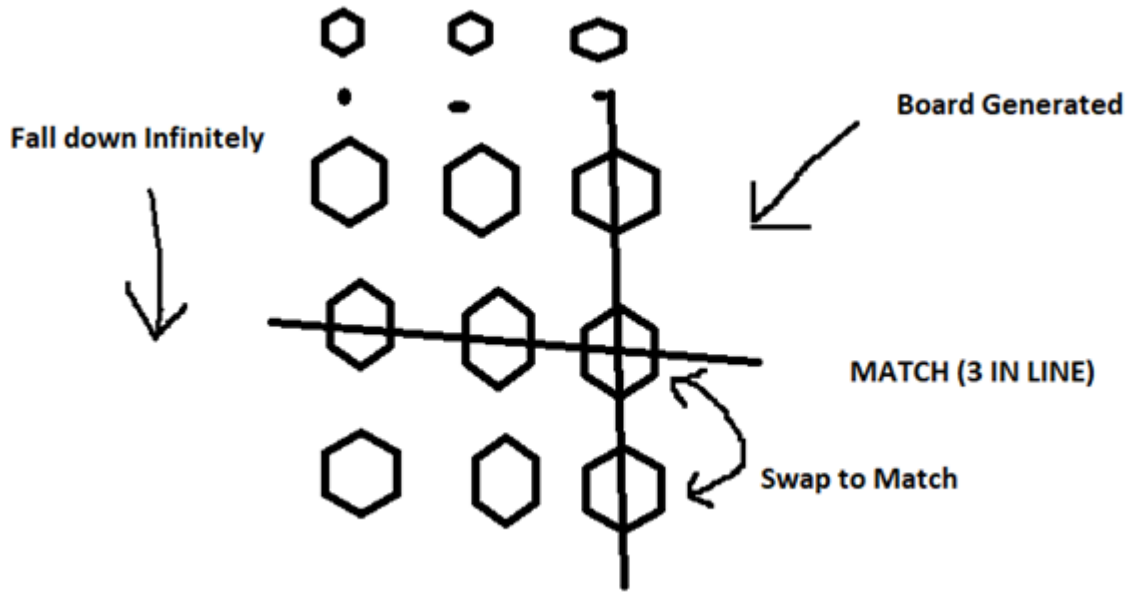
Site: DILo Game Academy
Course: Game Programming Studi Independen
Book: Match Three
Printed by: 408 Dewa Sinar Surya
Date: Monday, 4 October 2021, 7:56 PM

Table of contents

1. Pendahuluan

Pada materi kali ini kita akan membahas cara membuat game Match Three, yaitu sebuah game casual seperti candy crush, dimana kita akan menghancurkan icon-icon yang muncul dengan cara membuat minimal 3 barisan icon yang sama. Game ini lumayan banyak di pasaran dan tahukah kalian walaupun game ini terlihat cukup simpel, akan tetapi memiliki sistem yang lumayan kompleks.

Untuk gambaran awal, seperti inilah game yang akan kita buat



Dari gambar di atas, kita bisa lihat tujuan game ini adalah dengan melakukan match dengan menempatkan item yang sama pada 3 barisan, kita melakukannya dengan cara menukar posisi item. Untuk setiap match yang dilakukan item akan dihapus dan item baru akan turun dari atas terus menerus. Untuk board nya sendiri juga secara otomatis digenerate. Dari informasi ini, kita bisa bagi fitur-fiturnya ke dalam 2 bagian, yaitu sistem untuk board dan sistem untuk item atau tilenya sendiri. Selain itu kita juga akan menambahkan sistem skoring dan timer agar game nya lebih menarik. Secara garis besar inilah step-step yang akan dilakukan

- Membuat Board Generator
- Membuat sistem Swap Tile
- Membuat sistem Board
- Membuat sistem skoring dan timer

Materi ini terdiri atas 6 bagian. Berikut adalah outline bagian-bagiannya:

Bagian 1: Setting Up

1. New Project
2. Import Asset
3. Set Up Asset
4. Set Up Scene

Bagian 2: Set Up Generator

1. Creating Tile Prefab
2. Board as a Singleton
3. Generate Board
   - Using tile ID
4. Randomize Tiles
   - No Match at Start

Bagian 3: Swapping Tiles

1. Selecting Tiles
2. Moving Tiles
   - Adding Swap tiles to board
   - Using Coroutine for Scripted Animation (Tween)
   - Swapping back Tiles
3. Getting to know our Neighbour
   - Using Raycast
   - Limit Swapping Tiles
4. Checking Match
   - Getting Matched Sprite around Tiles
   - Adding Match check to board for all Tiles
   - Add checking before swapping back tiles

Bagian 4: Board as the Manager

1. Adding Process to board
2. Creating match process
3. Creating after match process
    - Creating drop process
    - Creating destroy and fill process
    - Creating reposition process (actual animation)

Bagian 5: Score and Timer

1. Create Score Handler
2. Create Timer Handler
3. Calculating score
4. Create UI mockup
5. Adding Game Over from Timer

Bagian 6: Adding Audio

1. Adding select and deselect sound
2. Adding match sound
3. Adding combo match sound
4. Adding not match sound

2. Download Asset

Untuk membuat game match three berdasarkan materi ini, terlebih dulu download asset berikut di link berikut:

Hexagon Tiles:

- Tiles\tileAutumn_full.png
- Tiles\tileDirt_full.png
- Tiles\tileGrass_full.png
- Tiles\tileLava_full.png
- Tiles\tileMagic_full.png
- Tiles\tileRock_full.png
- Tiles\tileSand_full.png
- Tiles\tileStone_full.png
- Tiles\tileWater_full.png

Interface Audio:

- confirmation_001.ogg
- confirmation_002.ogg
- glass_005.ogg
- error_006.ogg

LINK DOWNLOAD:

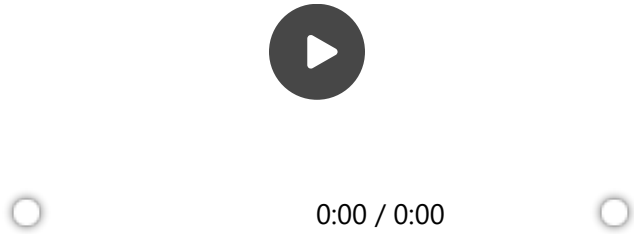
https://drive.google.com/file/d/1sbEwfDoqKV4Ce2eQBRQdMIPWZ67OnRvF/view?usp=sharing

2.1. Tutorial Versi Video

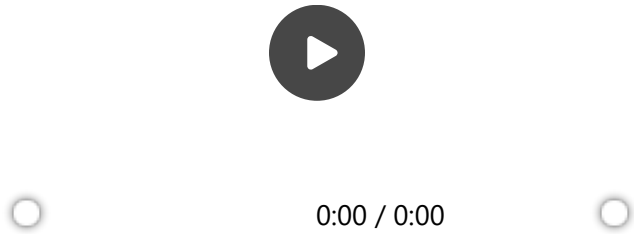
Selain versi tertulis, kamu juga bisa mengikuti materi chapter ini lewat versi video. Terdapat sedikit perbedaan langkah dari apa-apa yang ada pada versi tertulis tetapi script akhirnya sama.

Videonya bisa ditonton di link berikut

Bagian 1 dan Bagian 2 - Setting Up & Setup Generator



Bagian 3 - Swapping Tiles



Bagian 4 - Board as the Manager

0:00 / 0:00

Bagian 5 - Timer and Score

0:00 / 0:00

Bagian 6 - Audio

0:00 / 0:00

3. Bagian 1 - Setting Up

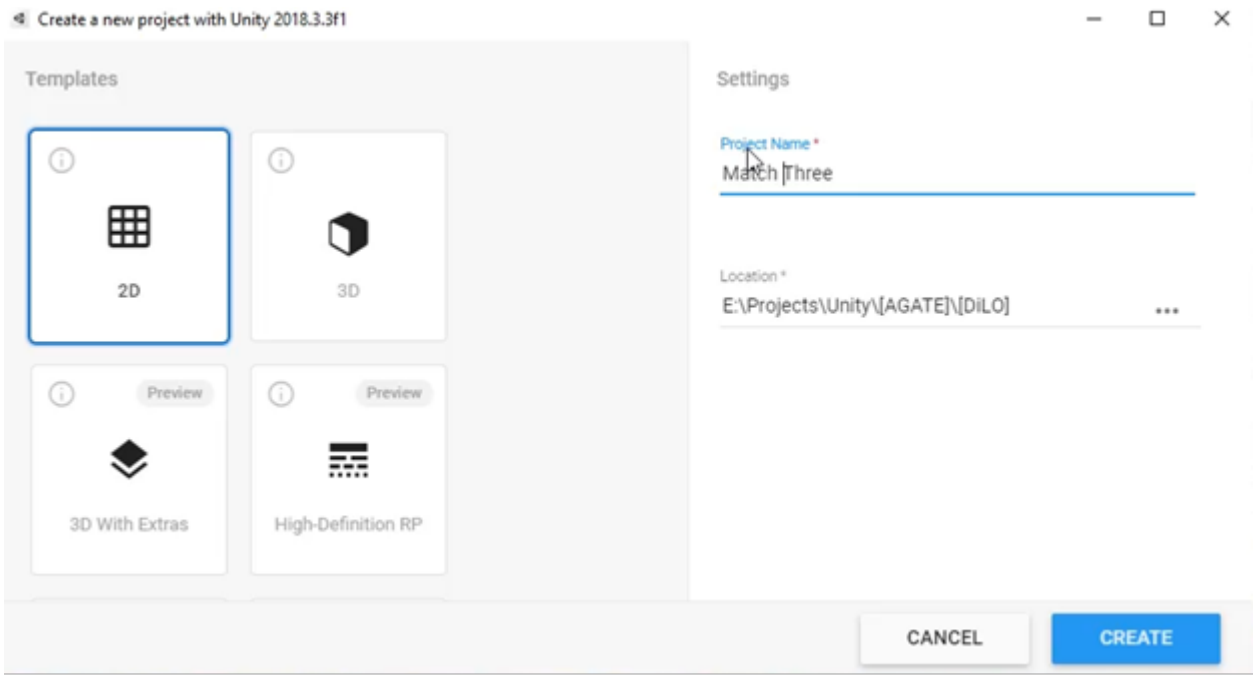
Inilah hal-hal yang akan kamu pelajari pada bagian 1:

Bagian 1: Setting Up

1. New Project
2. Import Asset
3. Set Up Asset
4. Set Up Scene

New Project

Pertama, kita buat project baru dulu, di pilihan project nya kita bisa pilih 2D project agar unity mengatur project kita supaya langsung bisa dimainkan secara 2D.



Import Asset

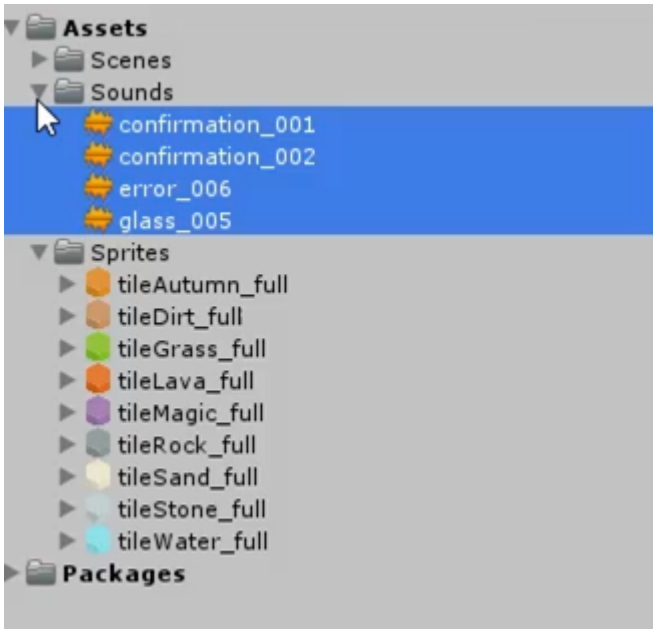
Pastikan bahwa kalian sudah mendownload asset untuk project ini. file nya dalam bentuk zip jadi kalian perlu untuk mengekstrak nya terlebih

dahulu, kemudian masukan file nya ke dalam project unity kita dengan cara drag n drop

Set Up Asset

Setelah kita masukan asset-nya, kita rapikan foldernya agar tidak berantakan.

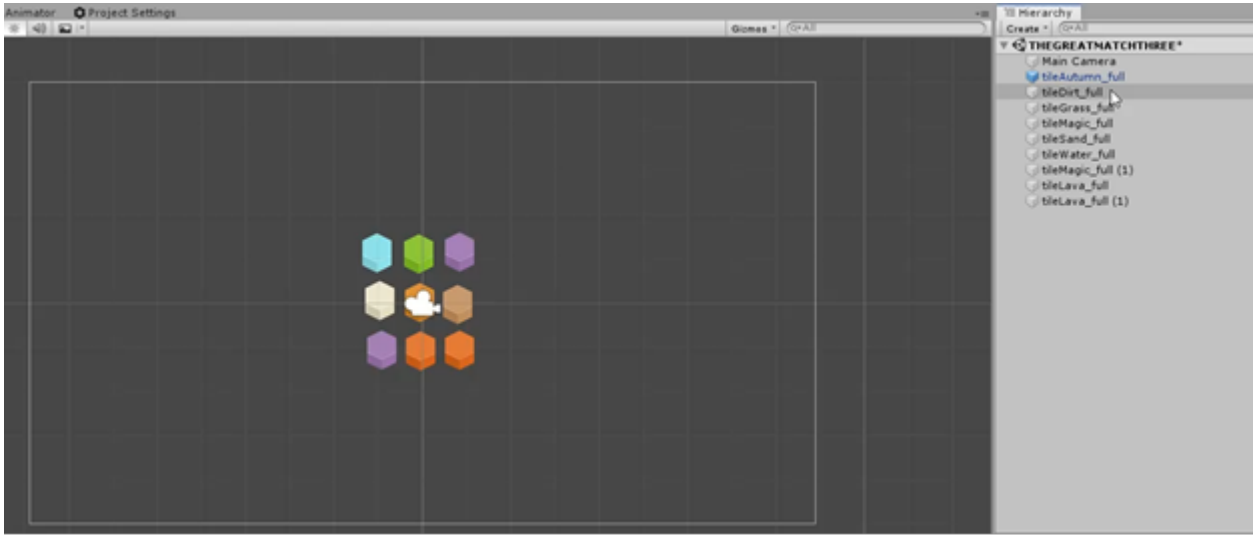
- Masukan asset gambar (png) ke dalam folder Sprites
- Masukan asset suara (ogg) ke dalam folder Sounds



Hal ini penting dilakukan agar kita tidak sulit untuk mencari asset dalam project kita, kalian bisa atur folder sesuai keinginan kalian juga kok, yang penting jangan sampai berantakan file nya karena akan menyulitkan kalian sendiri nanti saat project game kalian semakin membesar

Set Up Scene

Sekarang, agar kita bisa melihat seperti apa game yang akan kita buat, kita bisa langsung memasukan sprite-sprite yang sudah kita potong tadi ke dalam scene, masukan tile-tile yang kalian inginkan, buat seperti board match three, lalu bayangkan apa yang akan terjadi saat game nya berjalan



Step ini sebenarnya memang terlihat tidak terlalu penting bagi proses pembuatan game, akan tetapi ini akan membantu kita untuk membayangkan gameplay dari game yang akan kita buat.

4. Bagian 2 - Setup Generator

Pada bagian 2 ini berikut adalah hal yang akan kamu pelajari:

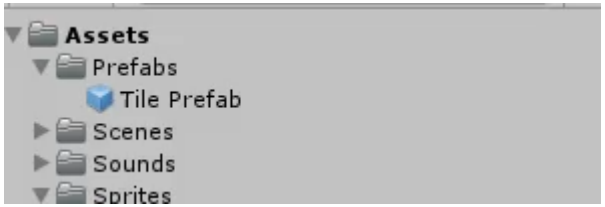
Bagian 2: Set Up Generator

1. Creating Tile Prefab
2. Board as a Singleton
3. Generate Board
    - Using tile ID
4. Randomize Tiles
    - No Match at Start

4.1. Creating Tile Prefab

Creating Tile Prefab

Pada game match three biasanya susunan tile yang kita pasang tadi dibuat secara otomatis oleh generator. Sebelum kita membuat generatornya, kita harus terlebih dahulu membuat object yang akan di generate sebagai prefab. Disini kita buat buat salah satu object tile yang ada pada scene kita menjadi prefab dengan cara mendrag object nya dari scene hierarchy ke project. Setelah prefabnya dibuat, jangan lupa untuk menghapus semua object tile dari scene karena nanti tile akan disusun menggunakan script. Untuk memastikan bahwa kamu sudah membuat prefab yang benar, pastikan bahwa object dalam prefab hanyalah 1 tile yang sudah diberi komponen box collider 2D.



4.2. Board as a Singleton

Board as a Singleton

Pastikan ketika kamu mencapai bagian ini kamu sudah menghapus semua tile yang ada pada scene kita, lalu kita langsung buat game object baru bernama board di scene kita. Lalu kita buat script baru bernama BoardManager, dan kita pasangkan pada object board tadi. Object board inilah yang nantinya akan secara otomatis menggenerate tile. Object board ini berjenis empty.

Kita akan membuat BoardManager sebagai singleton. Fungsi dari singleton sendiri adalah supaya object kita bisa di akses oleh script apapun tanpa harus membuat reference terlebih dahulu. Kita buka script nya, lalu kita tambahkan script yang akan membuatnya menjadi singleton di line paling atas sebelum fungsi start

```
#region Singleton

private static BoardManager _instance = null;

public static BoardManager Instance
{
   get
   {
      if (_instance == null)
      {
         _instance = FindObjectOfType<BoardManager>();

         if (_instance == null)
         {
            Debug.LogError("Fatal Error: BoardManager not Found");
         }
      }

      return _instance;
   }
}

#endregion
```

4.3. Generate Board

Generate Board

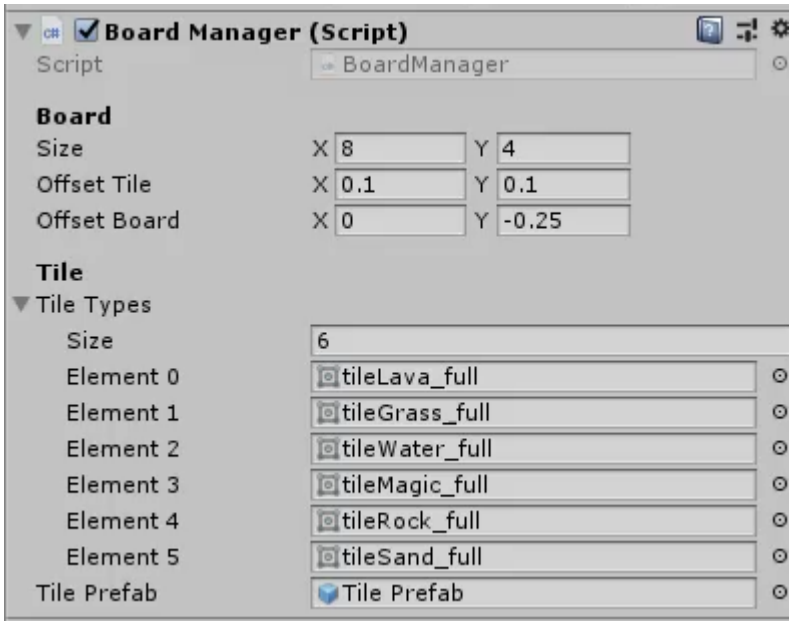
Sebelum kita membuat fungsi untuk menggenerate tile, kita buat dulu TileController yang berfungsi untuk mengontrol tile-tile yang digenerate oleh board manager ini. Kita buat script baru bernama TileController, lalu kita buka Tile Prefab yang sudah kita buat tadi, dan kita pasangkan TileController kita di Tile Prefab.

Sekarang kita langsung membuat fungsi untuk menggenerate tile. Fungsinya cukup sederhana, hanya saja ada beberapa tambahan untuk mengatur posisi agar berada di tengah layar serta dapat di kostumisasi. Pada dasarnya kita hanya membuat double array, lalu kita loop double array tersebut sesuai jumlah baris dan kolom yang kita inginkan. Pada proses loop nya kita Instantiate prefab kita, lalu posisikan di tempat sesuai indeks nya. Caranya, masukkan code berikut pada script BoardManager:

```
[Header("Board")]
public Vector2Int size;
public Vector2 offsetTile;
public Vector2 offsetBoard;

[Header("Tile")]
public List<Sprite> tileTypes = new List<Sprite>();
public GameObject tilePrefab;


private Vector2 startPosition;

private Vector2 endPosition;

private TileController[,] tiles;


private void Start()
{
   Vector2 tileSize = tilePrefab.GetComponent<SpriteRenderer>().size;
   CreateBoard(tileSize);
}

private void CreateBoard(Vector2 tileSize)
{
   tiles = new TileController[size.x, size.y];

   Vector2 totalSize = (tileSize + offsetTile) * (size - Vector2.one);

   startPosition = (Vector2)transform.position - (totalSize / 2) + offsetBoard;
   endPosition = startPosition + totalSize;

   for (int x = 0; x < size.x; x++)
   {
      for (int y = 0; y < size.y; y++)
      {
         TileController newTile = Instantiate(tilePrefab, new Vector2(startPosition.x + ((tileSize.x
+ offsetTile.x) * x), startPosition.y + ((tileSize.y + offsetTile.y) * y)),
tilePrefab.transform.rotation, transform).GetComponent<TileController>();
         tiles[x, y] = newTile;
```

```
        }
    }
}
```

Setelah itu, cek inspector untuk game object board manager lalu ubah komponen scriptnya menjadi seperti berikut:



4.4. Randomize Tile

Randomize Tile

Tentu tidak menarik jika tile yang di generate oleh board kita sama semua. Maka dari itu, kita harus membuat fitur untuk mengacak tile yang di spawn. Sebelum mengacak tile, kita perlu menambahkan fungsi untuk mengubah tile pada TileController. Untuk kasus ini kita akan memakai ID untuk membedakan tipe2 antar tile dengan ID sebagai angka pada list Tile Types. Implementasinya seperti berikut

```
public int id;

private BoardManager board;
private SpriteRenderer render;

private void Awake()
{
   board = BoardManager.Instance;
   render = GetComponent<SpriteRenderer>();
}

public void ChangeId(int id, int x, int y)
{
   render.sprite = board.tileTypes[id];
   this.id = id;

   name = "TILE_" + id + " (" + x + ", " + y + ")";
}
```

Sekarang kita bisa saja tinggal mengacaknya tipe tile dengan langsung mengacak id yang masuk antara 0 sampai jumlah tipe tile yang ada seperti ini

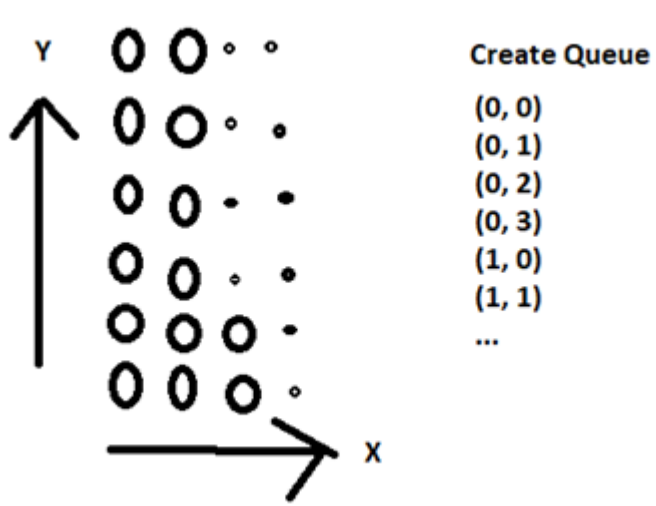
Masukkan code berikut pada script BoardManager, pada fungsi CreateBoard():

```
newTile.ChangeId(Random.Range(0, tileTypes.Count), x, y);
```

Akan tetapi, jika kita mengacak seperti itu, ada kemungkinan untuk tile yang muncul sudah berjajar tiga tipe yang sama sehingga player langsung

dapat skor. Hal ini akan membuat game tidak menarik dan membingungkan player. Untuk mengatasi hal ini, kita harus membuat list id apa saja yang bisa diambil sebelum melakukan proses change id. Agar kita lebih tercerahkan, mari kita teliti terlebih dahulu proses generate tile yang telah kita buat.

Proses generate tile yang kita lakukan pada script kita tidak lah instant, akan tetapi bertahap di generate satu per satu, pada looping x kita bergerak dari kiri ke kanan, dan pada looping y kita bergerak dari bawah ke atas. Jika diilustrasikan mungkin seperti ini

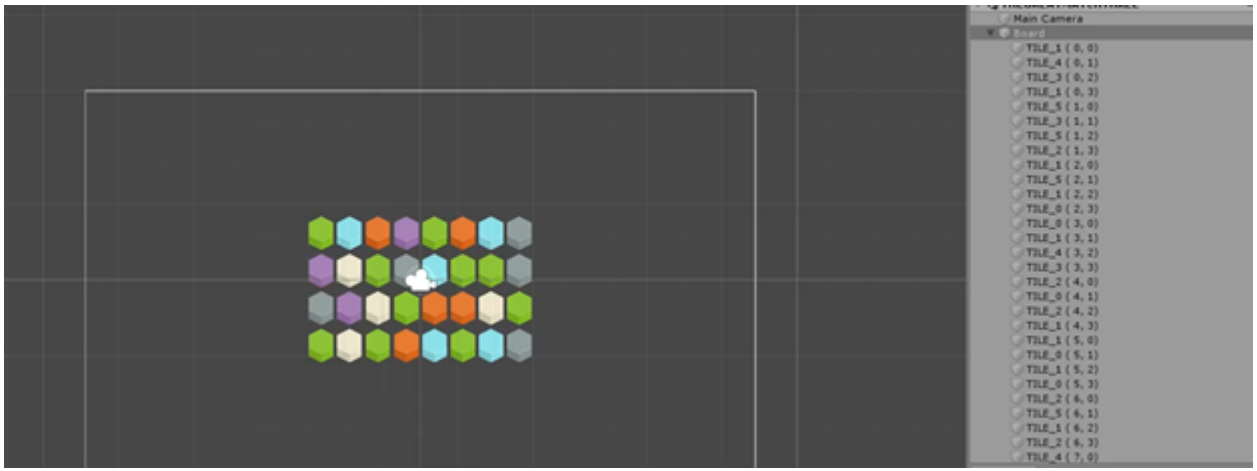


Dari hasil analisis tersebut, kita tahu bahwa untuk setiap tile baru yang di generate, dia tidak memiliki tetangga di atas dan juga di kanan. Maka dari itu untuk pengecekan match nya, kita hanya perlu mengecek untuk bawah dan kiri. Jika membuat barisan 3 tipe, maka tipe tersebut kita buang dari list. Fungsi ini kemudian kita gunakan untuk pengambilan ID baru.

```
private void CreateBoard(Vector2 tileSize)
{
   tiles = new TileController[size.x, size.y];

   Vector2 totalSize = (tileSize + offsetTile) * (size - Vector2.one);

   startPosition = (Vector2)transform.position - (totalSize / 2) + offsetBoard;
   endPosition = startPosition + totalSize;

   for (int x = 0; x < size.x; x++)
   {
      for (int y = 0; y < size.y; y++)
      {
         TileController newTile = Instantiate(tilePrefab, new Vector2(startPosition.x + ((tileSize.x
+ offsetTile.x) * x), startPosition.y + ((tileSize.y + offsetTile.y) * y)),
tilePrefab.transform.rotation, transform).GetComponent<TileController>();
         tiles[x, y] = newTile;

         // get no tile id
         List<int> possibleId = GetStartingPossibleIdList(x, y);
         int newId = possibleId[Random.Range(0, possibleId.Count)];

         newTile.ChangeId(newId, x, y);
      }
   }
}

private List<int> GetStartingPossibleIdList(int x, int y)
{
   List<int> possibleId = new List<int>();

   for (int i = 0; i < tileTypes.Count; i++)
   {
      possibleId.Add(i);
   }

   if (x > 1 && tiles[x - 1, y].id == tiles[x - 2, y].id)
   {
```

```
            possibleId.Remove(tiles[x - 1, y].id);
        }

        if (y > 1 && tiles[x, y - 1].id == tiles[x, y - 2].id)
        {
            possibleId.Remove(tiles[x, y - 1].id);
        }

        return possibleId;
    }
```

Sekarang kalian coba mainkan game nya, tile-tile akan di generate secara acak tanpa ada match seperti ini



5. Bagian 3 - Swapping Tiles (Pengantar)

Pada bagian 3 inilah yang akan kalian pelajari:

Bagian 3: Swapping Tiles

1. Selecting Tiles
2. Moving Tiles
    - Adding Swap tiles to board
    - Using Coroutine for Scripted Animation (Tween)
    - Swapping back Tiles
3. Getting to know our Neighbour
    - Using Raycast
    - Limit Swapping Tiles
4. Checking Match
    - Getting Matched Sprite around Tiles
    - Adding Match check to board for all Tiles
    - Add checking before swapping back tiles

5.1. Selecting Tile

Selecting Tile

Sekarang kita akan menambahkan kontrol pada tile-tile kita agar bisa dipilih, untuk melakukan hal ini kita tinggal menambahkan status isSelected pada tiap tile nya, lalu pada OnMouseDown kita aktifkan isSelected tersebut. Kita juga perlu mematikan status isSelected tile sebelumnya saat kita memilih tile baru, disini kita akan memakai static field yang menyimpan tile sebelumnya. Masukkan code berikut pada script TileController:

```
private static readonly Color selectedColor = new Color(0.5f, 0.5f, 0.5f);
private static readonly Color normalColor = Color.white;

private static TileController previousSelected = null;

private bool isSelected = false;

private void OnMouseDown()
{
   // Non Selectable conditions
   if (render.sprite == null)
   {
      return;
   }

   // Already selected this tile?
   if (isSelected)
   {
      Deselect();
   }
   else
   {
      // if nothing selected yet
      if (previousSelected == null)
      {
         Select();
      }

      else
      {
         previousSelected.Deselect();
         Select();
      }
   }
}

#region Select & Deselect
```

```
private void Select()
{
    isSelected = true;
    render.color = selectedColor;
    previousSelected = this;
}

private void Deselect()
{
    isSelected = false;
    render.color = normalColor;
    previousSelected = null;
}

#endregion
```

5.2. Moving Tile

Moving Tile

Game kita tentu tidak akan seru jika hanya bisa mengganti selected status saja. Kita perlu menukar posisi Tile kita saat dua tile dipilih. Untuk melakukan hal ini kita akan membuat fungsi untuk memindahkan posisi tile. Agar fungsi ini bisa berjalan dengan lancar kita perlu memasukkan sejumlah code pada script TileController dan BoardManager.

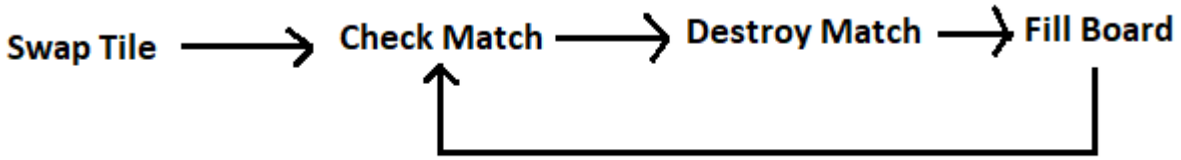
Untuk memindahkan posisi tile kita memerlukan animasi. Akan tetapi, karena posisi awal dan posisi akhir perpindahan selalu berubah2, kita tidak bisa memakai sistem animasi pada unity. Untuk kasus ini kita bisa menggunakan Tween, yaitu teknik untuk menganimasikan gerakan pada object dengan menggunakan Coroutine. Beginilah code gerakan dengan menggunakan tween. Masukkan code berikut pada script TileController:

```
private static readonly float moveDuration = 0.5f;


public IEnumerator MoveTilePosition(Vector2 targetPosition, System.Action onCompleted)
{
   Vector2 startPosition = transform.position;
   float time = 0.0f;

   // run animation on next frame for safety reason
   yield return new WaitForEndOfFrame();

   while (time < moveDuration)
   {
      transform.position = Vector2.Lerp(startPosition, targetPosition, time / moveDuration);
      time += Time.deltaTime;

      yield return new WaitForEndOfFrame();
   }

   transform.position = targetPosition;

   onCompleted?.Invoke();
}
```

Secara garis besar, flow Coroutine ini adalah:

- Dijalankan pada frame berikutnya (supaya kalkulasi tidak terganggu)
- Pada tiap frame nya hingga durasi tercapai
- Pindahkan posisi sesuai garis linier (Lerp) antara startPosition dengan TargetPosition dengan
- Tambahkan waktu dengan waktu proses frame tersebut

Pada proses Coroutine ini juga ditambahkan event onCompleted agar kita bisa menjalankan proses lanjutan setelah coroutinenya selesai. Penambahan event ini akan dipakai secara intensif pada jenis game sekuensial seperti ini dimana proses akan secara otomatis berjalan. Contoh kasus pada game ini:



Kita akan menukar posisi tile yang kita pilih dengan posisi tile yang kita pilih sebelumnya. Hal yang perlu kita lakukan hanyalah menjalankan Coroutine ini untuk kedua tile tersebut dengan posisi tile yang ditukar. Agar lebih terstruktur, kita akan menambahkan fungsi untuk menukar posisi tile pada board, karena board ini lah yang bertugas untuk mengatur segala kejadian yang akan terjadi padanya. Code berikut dimasukkan ke dalam script BoardManager:

```
public bool IsAnimating
{
   get
   {
      return IsSwapping;
   }
}

public bool IsSwapping { get; set; }

#region Swapping

public IEnumerator SwapTilePosition(TileController a, TileController b, System.Action
onCompleted)
{
   IsSwapping = true;

   Vector2Int indexA = GetTileIndex(a);
   Vector2Int indexB = GetTileIndex(b);

   tiles[indexA.x, indexA.y] = b;
   tiles[indexB.x, indexB.y] = a;

   a.ChangeId(a.id, indexB.x, indexB.y);
   b.ChangeId(b.id, indexA.x, indexA.y);

   bool isRoutineACompleted = false;
   bool isRoutineBCompleted = false;

   StartCoroutine(a.MoveTilePosition(GetIndexPosition(indexB), () => { isRoutineACompleted
= true; }));
   StartCoroutine(b.MoveTilePosition(GetIndexPosition(indexA), () => { isRoutineBCompleted
= true; }));

   yield return new WaitUntil(() => { return isRoutineACompleted && isRoutineBCompleted;
});

   onCompleted?.Invoke();
```

```
    IsSwapping = false;
}

#endregion

public Vector2Int GetTileIndex(TileController tile)
{
    for (int x = 0; x < size.x; x++)
    {
        for (int y = 0; y < size.y; y++)
        {
            if (tile == tiles[x, y]) return new Vector2Int(x, y);
        }
    }

    return new Vector2Int(-1, -1);
}

public Vector2 GetIndexPosition(Vector2Int index)
{
    Vector2 tileSize = tilePrefab.GetComponent<SpriteRenderer>().size;
    return new Vector2(startPosition.x + ((tileSize.x + offsetTile.x) * index.x), startPosition.y +
((tileSize.y + offsetTile.y) * index.y));
}
```

Pada fungsi ini kita menjalankan Corotine MoveTilePosition pada kedua tile, serta memperbarui susunan tiles pada board, serta mengganti nama tile agar sesuai dengan susunan tiles pada board dengan menggunakan ChangeId. Fungsi ini juga berupa coroutine yang akan menunggu kedua corotine yang dijalankan tadi selesai kemudian menjalankan onCompleted nya.

Selain itu kita juga menambahkan dua fungsi baru yaitu GetTileIndex dan GetIndexPosition untuk membantu fungsi swap tadi. GetTileIndex akan mengambil indeks dari tile yang dicari pada double array tiles. GetIndexPosition akan mengambil posisi tile di indeks yang dicari pada ruang 2D game kita.

Pada fungsi ini kita juga menambahkan status IsSwapping dan IsAnimating pada board agar kita bisa mengunci tile untuk tidak menerima input pada

saat IsAnimating. IsSwapping sendiri adalah salah satu kasus IsAnimating. Kita bisa mengunci input dengan cara menambahkan pengecekan IsAnimating pada OnMouseDown yang ada pada TileController.

Kita juga tambahkan fungsi untuk menjalankan coroutine SwapTilePosition ini pada TileController yang dijalankan saat ada dua tile yang dipilih. Fungsi tersebut akan dijalankan dua kali agar tile kembali ke posisi semula setelah diswap, seperti berikut ini

```
private void OnMouseDown()
{
    // Non Selectable conditions
    if (render.sprite == null || board.IsAnimating)
    {
        return;
    }

    // Already selected this tile?
    if (isSelected)
    {
        Deselect();
    }
    else
    {
        // if nothing selected yet
        if (previousSelected == null)
        {
            Select();
        }

        else
        {
            TileController otherTile = previousSelected;
            // swap tile
            SwapTile(otherTile, () => {
                SwapTile(otherTile);
            });

            // run if cant swap (disabled for now)
            //previousSelected.Deselect();
            //Select();
        }
    }
}

public void SwapTile(TileController otherTile, System.Action onCompleted = null)
{
```

```
        StartCoroutine(board.SwapTilePosition(this, otherTile, onCompleted));
    }
```

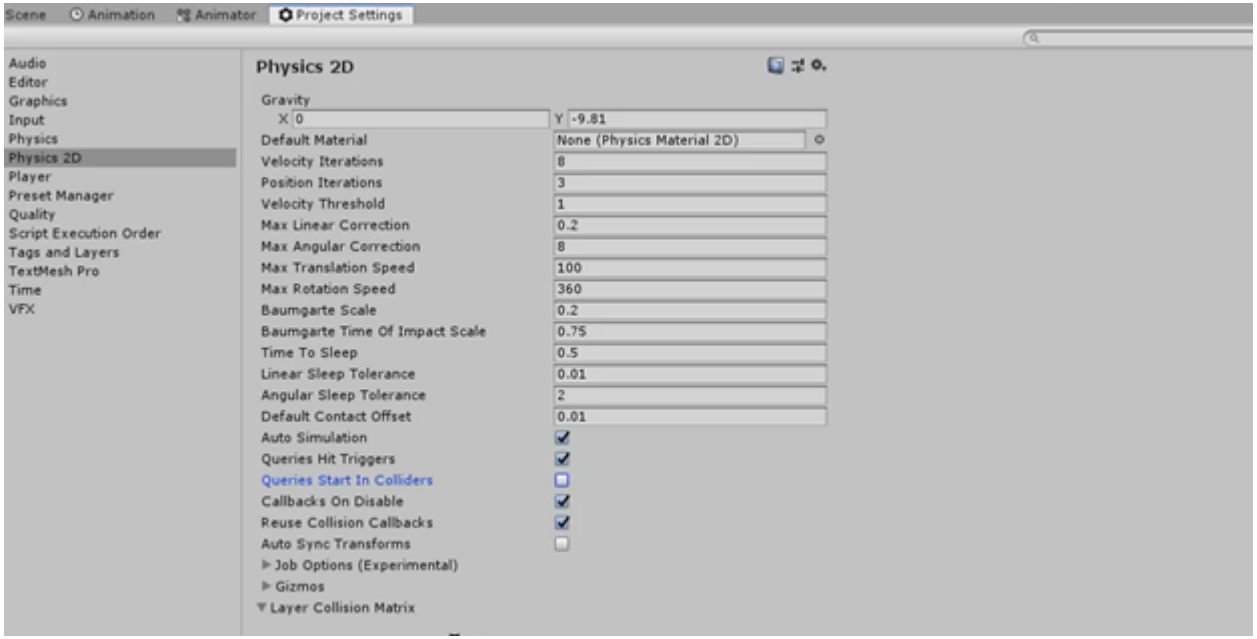
5.3. Getting to Know Our Neighbour

Getting to know our Neighbour

Proses swap tile pada game match three hanya bisa dilakukan pada tile di sekitarnya saja, baik horizontal maupun vertikal. Untuk itu, kita akan membuat fungsi pengecekan antara kedua tile sebelum dilakukan swap. Untuk proses pengecekannya kita akan memakai Raycast untuk 4 arah, yaitu atas, bawah, kiri, dan kanan. Code ini dimasukkan ke dalam script TileController:

```
private static readonly Vector2[] adjacentDirection = new Vector2[] { Vector2.up, Vector2.down,
Vector2.left, Vector2.right };

#region Adjacent

private TileController GetAdjacent(Vector2 castDir)
{
   RaycastHit2D hit = Physics2D.Raycast(transform.position, castDir, render.size.x);

   if (hit)
   {
      return hit.collider.GetComponent<TileController>();
   }

   return null;
}

public List<TileController> GetAllAdjacentTiles()
{
   List<TileController> adjacentTiles = new List<TileController>();

   for (int i = 0; i < adjacentDirection.Length; i++)
   {
      adjacentTiles.Add(GetAdjacent(adjacentDirection[i]));
   }

   return adjacentTiles;
}

#endregion
```

Dikarenakan kita menggunakan Raycast dari dalam collider tile, kita perlu matikan opsi Queries Start In Colliders pada project settings agar raycast tidak mengambil hit value dari tile tempat dia dimunculkan. Jika ini tidak dilakukan Raycast tadi akan selalu hit karena mengambil dirinya sendiri. Untuk mengakses project setting, klik edit > project setting:



Sekarang kita tambahkan proses pengecekan neighbour ini pada fungsi swap seperti ini

```
private void OnMouseDown()
{
    // Non Selectable conditions
    if (render.sprite == null || board.IsAnimating)
    {
        return;
    }

    // Already selected this tile?
    if (isSelected)
    {
        Deselect();
    }
    else
    {
        // if nothing selected yet
        if (previousSelected == null)
        {
            Select();
        }

        else
        {
            // is this an adjacent tile?
            if (GetAllAdjacentTiles().Contains(previousSelected))
            {
                TileController otherTile = previousSelected;
                previousSelected.Deselect();

                // swap tile
                SwapTile(otherTile, () => {
                        SwapTile(otherTile);
                });
            }
            // if not adjacent then change selected
            else
            {
                previousSelected.Deselect();
```

```
                Select();
            }
        }
    }
}
```

5.4. Check Match

Check Match

Sekarang kita akan membuat fungsi untuk mengecek apakah terdapat match atau tidak. Untuk mengecek match atau tidak kita akan membuat fungsi menggunakan Raycast, mirip seperti fungsi sebelumnya. Akan tetapi fungsi ini akan secara berulang mengambil tile selama tile tersebut memiliki ID yang sama.

Fungsi yang akan dibuat dibagi menjadi tiga fungsi, yang pertama fungsi akan mengambil semua tile dengan ID yang sama pada satu arah. Fungsi kedua akan menggabungkan hasil dari fungsi satu arah menjadi satu baris (atas dan bawah menjadi vertikal). Kemudian Fungsi ke tiga akan menggabungkan fungsi satu baris dan mereturn hasil dari baris vertikal dan baris horizontal. Code berikut dimasukkan ke dalam script TileController:

```
public bool IsDestroyed { get; private set; }


#region Check Match

private List<TileController> GetMatch(Vector2 castDir)
{
   List<TileController> matchingTiles = new List<TileController>();
   RaycastHit2D hit = Physics2D.Raycast(transform.position, castDir, render.size.x);

   while (hit)
   {
      TileController otherTile = hit.collider.GetComponent<TileController>();
      if (otherTile.id != id || otherTile.IsDestroyed)
      {
         break;
      }

      matchingTiles.Add(otherTile);
      hit = Physics2D.Raycast(otherTile.transform.position, castDir, render.size.x);
   }

   return matchingTiles;
}

private List<TileController> GetOneLineMatch(Vector2[] paths)
{
   List<TileController> matchingTiles = new List<TileController>();

   for (int i = 0; i < paths.Length; i++)
   {
      matchingTiles.AddRange(GetMatch(paths[i]));
   }

   // only match when more than 2 (3 with itself) in one line
   if (matchingTiles.Count >= 2)
   {
      return matchingTiles;
```

```
    }

    return null;
}

public List<TileController> GetAllMatches()
{
    if (IsDestroyed)
    {
        return null;
    }

    List<TileController> matchingTiles = new List<TileController>();

    // get matches for horizontal and vertical
    List<TileController> horizontalMatchingTiles = GetOneLineMatch(new Vector2[2] {
Vector2.up, Vector2.down });
    List<TileController> verticalMatchingTiles = GetOneLineMatch(new Vector2[2] {
Vector2.left, Vector2.right });

    if (horizontalMatchingTiles != null)
    {
        matchingTiles.AddRange(horizontalMatchingTiles);
    }

    if (verticalMatchingTiles != null)
    {
        matchingTiles.AddRange(verticalMatchingTiles);
    }

    // add itself to matched tiles if match found
    if (matchingTiles != null && matchingTiles.Count >= 2)
    {
        matchingTiles.Add(this);
    }

    return matchingTiles;
}
```

#endregion

Kita akan mengecek match seluruh tile pada board untuk menghindari kesalahan match dan untuk mempermudah proses matching sendiri. Oleh karena itu kita akan menambahkan fungsi check match ini pada script BoardManager.

```
public List<TileController> GetAllMatches()
{
    List<TileController> matchingTiles = new List<TileController>();

    for (int x = 0; x < size.x; x++)
    {
        for (int y = 0; y < size.y; y++)
        {
            List<TileController> tileMatched = tiles[x, y].GetAllMatches();

            // just go to next tile if no match
            if (tileMatched == null || tileMatched.Count == 0)
            {
                continue;
            }

            foreach (TileController item in tileMatched)
            {
                // add only the one that is not added yet
                if (!matchingTiles.Contains(item))
                {
                    matchingTiles.Add(item);
                }
            }
        }
    }

    return matchingTiles;
}
```

Sekarang kita tinggal tambahkan pengecekan Match pada proses swap kita tadi. Masukkan code berikut pada script TileController:

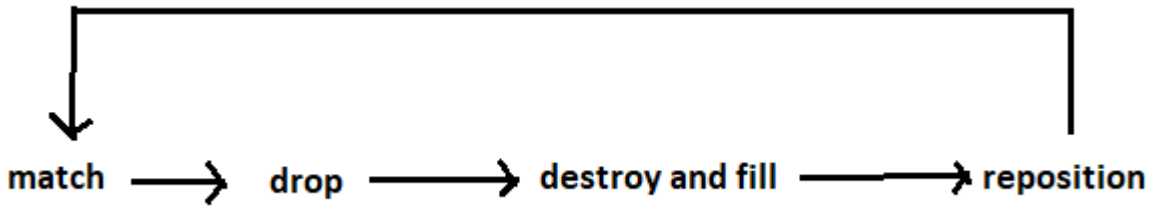
```
private void OnMouseDown()
{
    // Non Selectable conditions
    if (render.sprite == null || board.IsAnimating)
    {
        return;
    }

    // Already selected this tile?
    if (isSelected)
    {
        Deselect();
    }
    else
    {
        // if nothing selected yet
        if (previousSelected == null)
        {
            Select();
        }

        else
        {
            // is this an adjacent tile?
            if (GetAllAdjacentTiles().Contains(previousSelected))
            {
                TileController otherTile = previousSelected;
                previousSelected.Deselect();

                // swap tile
                SwapTile(otherTile, () => {
                    if (board.GetAllMatches().Count > 0)
                    {
                        Debug.Log("MATCH FOUND")
                    }
                    else
                    {
                        SwapTile(otherTile);
```

```
                }
            });
        }
        // if not adjacent then change selected
        else
        {
            previousSelected.Deselect();
            Select();
        }
    }
  }
}
```

Sekarang kita bisa coba play game nya dan kita bisa swap tile dengan tetangganya. Saat match ditemukan akan muncul debug “MATCH FOUND”.

6. Bagian 4 - Board as the Manager

Sekarang kita akan membuat proses yang akan dijalankan board pada saat debug “MATCH FOUND” terjadi. Proses yang akan dilakukan board adalah proses yang lumayan kompleks pada saat match terjadi. Proses ini kita akan bagi menjadi 4 proses seperti grafik ini



Pada bagian ini kamu akan mempelajari:

Bagian 4: Board as the Manager

1. Adding Process to board
2. Creating match process
3. Creating after match process
    - Creating drop process
    - Creating destroy and fill process
    - Creating reposition process (actual animation)

6.1. Creating Match Process

Creating Match Process

Pada proses match kita akan melakukan matching sekaligus mendestroy tile yang sudah match. Sebelum kita membuat prosesnya, kita buat dulu Coroutine untung mendestroy tile pada Tile Controller seperti ini.

Sebelum itu, pada fungsi Start() pada script TileController masukkan code berikut:

IsProcessing = false;

IsSwapping = false;

IsDestroyed = false;

Berulah masukkan code berikut:

```
    private static readonly float moveDuration = 0.5f;
    private static readonly float destroyBigDuration = 0.1f;
    private static readonly float destroySmallDuration = 0.4f;

    private static readonly Vector2 sizeBig = Vector2.one * 1.2f;
    private static readonly Vector2 sizeSmall = Vector2.zero;
    private static readonly Vector2 sizeNormal = Vector2.one;

    public bool IsDestroyed { get; private set; }

    public IEnumerator SetDestroyed(System.Action onCompleted)
    {
        IsDestroyed = true;
        id = -1;
        name = "TILE_NULL";

        Vector2 startSize = transform.localScale;
        float time = 0.0f;

        while (time < destroyBigDuration)
        {
            transform.localScale = Vector2.Lerp(startSize, sizeBig, time / destroyBigDuration);
            time += Time.deltaTime;

            yield return new WaitForEndOfFrame();
        }

        transform.localScale = sizeBig;

        startSize = transform.localScale;
        time = 0.0f;

        while (time < destroySmallDuration)
        {
            transform.localScale = Vector2.Lerp(startSize, sizeSmall, time / destroySmallDuration);
            time += Time.deltaTime;

            yield return new WaitForEndOfFrame();
```

```
    }

    transform.localScale = sizeSmall;

    render.sprite = null;

    onCompleted?.Invoke();
}
```

Kemudian kita jalankan coroutine ini pada semua tile yang ada pada list matched tile. Kita jalankan proses pengambilan list matched tile pada Board Manager. Kita lakukan prosesnya dengan cara yang mirip swap, yaitu menunggu sampai semua proses selesai, akan tetapi ini dilakukan untuk setiap tiles yang dihancurkan. Untuk melakukan hal ini kita tambahkan fungsi yang akan mengecek list dari status boolean. Kita juga perlu mendefiniskan jika match tidak ditemukan maka harus keluar dari proses sequence nya.

```
public bool IsAnimating
{
   get
   {
      return IsProcessing || IsSwapping;
   }
}

public bool IsProcessing { get; set; }

public void Process()
{
   IsProcessing = true;
   ProcessMatches();
}

#region Match

private void ProcessMatches()
{
   List<TileController> matchingTiles = GetAllMatches();

   // stop locking if no match found
   if (matchingTiles == null || matchingTiles.Count == 0)
   {
      IsProcessing = false;
      return;
   }
}

private IEnumerator ClearMatches(List<TileController> matchingTiles, System.Action
onCompleted)
{
   List<bool> isCompleted = new List<bool>();

   for (int i = 0; i < matchingTiles.Count; i++)
   {
      isCompleted.Add(false);
```

```
    }

    for (int i = 0; i < matchingTiles.Count; i++)
    {
        int index = i;
        StartCoroutine(matchingTiles[i].SetDestroyed(() => { isCompleted[index] = true; }));
    }

    yield return new WaitUntil(() => { return IsAllTrue(isCompleted); });

    onCompleted?.Invoke();
}

#endregion

public bool IsAllTrue(List<bool> list)
{
    foreach (bool status in list)
    {
        if (!status) return false;
    }

    return true;
}
```

Lalu, pada script Tile Controller, pada fungsi OnMouseDown() cari Debug.Log("MATCH FOUND") lalu masukkan code berikut di bawahnya:

```
board.Process();
```

6.2. Creating Drop Process

Creating Drop Process

Sekarang kita buat proses selanjutnya, yaitu proses drop. Proses ini hanya akan mengupdate double list yang menyimpan susunan tile pada board. Kita tidak bisa langsung memproses jatuhnya tile karena animasi drop dan animasi fill akan kita jalankan secara bersamaan pada proses reposition.

Untuk mencari tile yang akan jatuh, kita akan mencari tile yang sudah didestroy. Kemudian semua tile yang ada di atasnya akan kita jatuhkan sebanyak 1 tile. Jika ada tile yang di destroy maka proses drop diabaikan karena nantinya tile yang sudah didestroy akan di generate ulang pada proses selanjutnya.

Sebelum itu, pada fungsi ProcessMatches() pada #region Match, masukkan code berikut:

```
StartCoroutine(ClearMatches(matchingTiles, ProcessDrop));
```

Barulah masukkan code berikut:

```
#region Drop

private void ProcessDrop()
{
   Dictionary<TileController, int> droppingTiles = GetAllDrop();
}

private Dictionary<TileController, int> GetAllDrop()
{
   Dictionary<TileController, int> droppingTiles = new Dictionary<TileController, int>();

   for (int x = 0; x < size.x; x++)
   {
      for (int y = 0; y < size.y; y++)
      {
         if (tiles[x, y].IsDestroyed)
         {
            // process for all tile on top of destroyed tile
            for (int i = y + 1; i < size.y; i++)
            {
               if (tiles[x, i].IsDestroyed)
               {
                  continue;
               }

               // if this tile already on drop list, increase its drop range
               if (droppingTiles.ContainsKey(tiles[x, i]))
               {
                  droppingTiles[tiles[x, i]]++;
               }
               // if not on drop list, add it with drop range one
               else
               {
                  droppingTiles.Add(tiles[x, i], 1);
               }
            }
         }
      }
```

```
    }

    return droppingTiles;
}

private IEnumerator DropTiles(Dictionary<TileController, int> droppingTiles, System.Action
onCompleted)
{
    foreach (KeyValuePair<TileController, int> pair in droppingTiles)
    {
        Vector2Int tileIndex = GetTileIndex(pair.Key);

        TileController temp = pair.Key;
        tiles[tileIndex.x, tileIndex.y] = tiles[tileIndex.x, tileIndex.y - pair.Value];
        tiles[tileIndex.x, tileIndex.y - pair.Value] = temp;

        temp.ChangeId(temp.id, tileIndex.x, tileIndex.y - pair.Value);
    }

    yield return null;

    onCompleted?.Invoke();
}

#endregion
```

Flow prosesnya mirip seperti proses match sebelumnya yaitu membuat list tile yang akan di proses, kemudian menjalankan proses coroutine dengan list tile yang tadi. Flow ini juga akan di implemen pada proses yang lain agar lebih mudah untuk dibaca dan diimplementasikan.

6.3. Creating Destroy and Fill Process

Creating Destroy And Fill Process

Proses selanjutnya adalah menghancurkan tile yang sudah di mark sebagai is destroyed dan menggantinya menjadi tile baru yang akan turun dari atas. Karena kita akan membuat tile lama menjadi tile baru, kita harus tambahkan fungsi untuk menggenerate tile baru secara random pada Tile Controller.

```
public void GenerateRandomTile(int x, int y)
{
   transform.localScale = sizeNormal;
   IsDestroyed = false;

   ChangeId(Random.Range(0, board.tileTypes.Count), x, y);
}
```

Kemudian kita akan menjalankan proses GenerateRandomTile ini untuk setiap tile yang sudah di mark is destroyed. Kemudian kita pindahkan posisinya ke atas board. Code berikut dimasukkan ke dalam script BoardManager:

Sebelum itu, pada fungsi ProcessDrop() pada #region Drop, masukkan code berikut:

```
StartCoroutine(DropTiles(droppingTiles, ProcessDestroyAndFill));
```

Barulah masukkan code berikut:

```
#region Destroy & Fill

private void ProcessDestroyAndFill()
{
   List<TileController> destroyedTiles = GetAllDestroyed();


}

private List<TileController> GetAllDestroyed()
{
   List<TileController> destroyedTiles = new List<TileController>();

   for (int x = 0; x < size.x; x++)
   {
      for (int y = 0; y < size.y; y++)
      {
         if (tiles[x, y].IsDestroyed)
         {
            destroyedTiles.Add(tiles[x, y]);
         }
      }
   }

   return destroyedTiles;
}

private IEnumerator DestroyAndFillTiles(List<TileController> destroyedTiles, System.Action
onCompleted)
{
   List<int> highestIndex = new List<int>();

   for (int i = 0; i < size.x; i++)
   {
      highestIndex.Add(size.y - 1);
   }

   float spawnHeight = endPosition.y + tilePrefab.GetComponent<SpriteRenderer>().size.y +
offsetTile.y;
```

```
    foreach (TileController tile in destroyedTiles)
    {
        Vector2Int tileIndex = GetTileIndex(tile);
        Vector2Int targetIndex = new Vector2Int(tileIndex.x, highestIndex[tileIndex.x]);
        highestIndex[tileIndex.x]--;

        tile.transform.position = new Vector2(tile.transform.position.x, spawnHeight);
        tile.GenerateRandomTile(targetIndex.x, targetIndex.y);
    }

    yield return null;

    onCompleted?.Invoke();
}

#endregion
```

6.4. Creating Reposition Process

Creating Reposition Process

Sekarang pada proses Reposition kita akan menjalankan memindahkan tile-tile pada posisi seharusnya yang ada pada double array tile. Pada proses ini kita hanya akan mengambil posisi berdasarkan index dari tile tersebut dan menempatkan tile nya pada posisi itu. Pada proses ini juga kita tidak perlu mencari tile karena semua tile akan di proses secara bersamaan.

Sebelum itu, pada fungsi ProcessDestroyAndFill() pada #region Destroy & Fill, masukkan code berikut:

StartCoroutine(DestroyAndFillTiles(destroyedTiles, ProcessReposition));

Barulah masukkan code berikut:

```
#region Reposition

private void ProcessReposition()
{
   StartCoroutine(RepositionTiles(ProcessMatches));
}

private IEnumerator RepositionTiles(System.Action onCompleted)
{
   List<bool> isCompleted = new List<bool>();

   int i = 0;
   for (int x = 0; x < size.x; x++)
   {
      for (int y = 0; y < size.y; y++)
      {
         Vector2 targetPosition = GetIndexPosition(new Vector2Int(x, y));

         // skip if already on position
         if ((Vector2)tiles[x, y].transform.position == targetPosition)
         {
            continue;
         }

         isCompleted.Add(false);

         int index = i;
         StartCoroutine(tiles[x, y].MoveTilePosition(targetPosition, () => { isCompleted[index] =
true; }));

         i++;
      }
   }

   yield return new WaitUntil(() => { return IsAllTrue(isCompleted); });

   onCompleted?.Invoke();
}
```

#endregion

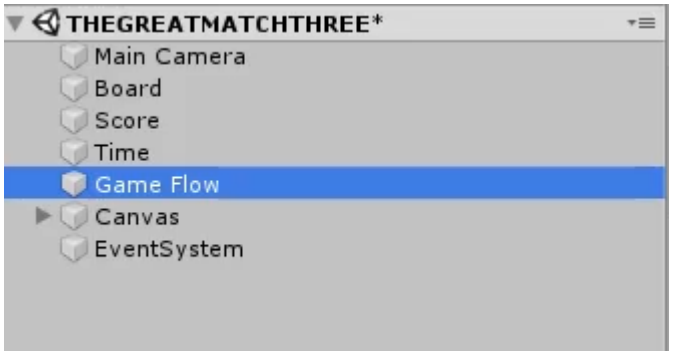
Seperti kalian lihat setelah proses reposition, board akan kembali melakukan proses match untuk mengecek tile yang match kembali karena mungkin pada susunan yang baru ada yang match baru juga yang muncul. Ini biasa dinamakan combo pada game match three.

Dan sudah selesai sistem game match three ini. Kita bisa play untuk melihat proses sistem yang sudah kita buat. Pada chapter berikutnya kita akan tambahkan timer dan skor agar sistem kita ini menjadi sebuah game.

7. Bagian 5 - Timer and Score

Sebuah game perlu adanya sistem yang membuat permainan menjadi menarik. Pada match three biasanya ada sistem skor dan timer dengan tujuan game nya adalah menambahkan skor sebanyak banyaknya sampai waktu habis. Untuk itu kita akan membuat beberapa Manager lain yang kita buat menjadi singleton supaya dapat dengan mudah diakses oleh script lainnya.

Kita buat 3 object baru pada scene game kita, yaitu Score untuk Score Manager, Time untuk Time Manager, dan Game Flow untuk Game Flow Manager.



Pada bagian ini kamu akan mempelajari:

Bagian 5: Score and Timer

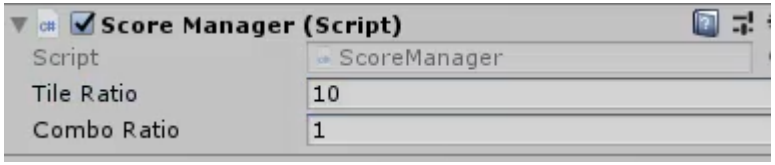
1. Create Score Handler
2. Create Timer Handler
3. Calculating score
4. Create UI mockup
5. Adding Game Over from Timer

7.1. Create Score Manager

Create Score Manager

Pertama kita buat script baru Score Manager, lalu kita kita pasangkan pada object Score. Score Manager ini anak mengatur scoring kita dan menyimpan highscore and current score. Disini kita akan membuat 3 fungsi yang bisa dipakai, yaitu fungsi reset, fungsi menambahkan score, dan fungsi untuk menjadikan current score sebagai highscore.

Kita juga bisa mengatur rasio skor tiap tile dan juga tiap combonya setelah kita menuliskan scriptnya dan memasukkan scriptnya ke dalam game objectnya.



```
private static int highScore;

#region Singleton

private static ScoreManager _instance = null;

public static ScoreManager Instance
{
   get
   {
      if (_instance == null)
      {
         _instance = FindObjectOfType<ScoreManager>();

         if (_instance == null)
         {
            Debug.LogError("Fatal Error: ScoreManager not Found");
         }
      }

      return _instance;
   }
}

#endregion

public int tileRatio;
public int comboRatio;

public int HighScore { get { return highScore; } }
public int CurrentScore { get { return currentScore; } }

private int currentScore;

private void Start()
{
   ResetCurrentScore();
}
```

```
public void ResetCurrentScore()
{
   currentScore = 0;
}

public void IncrementCurrentScore(int tileCount, int comboCount)
{
   currentScore += (tileCount * tileRatio) * (comboCount * comboRatio);
}

public void SetHighScore()
{
   highScore = currentScore;
}
```

Lalu, agar combonya berlaku, pada script BoardManager masukkan code berikut:

pada global class:

```
   private int combo;
```

pada fungsi Process():

```
      combo = 0;
```

pada fungsi ProcessMatches():

```
      combo++;
      ScoreManager.Instance.IncrementCurrentScore(matchingTiles.Count, combo);
```

7.2. Create Game Flow Manager

Create Game Flow Manager

Kita buat script baru Game Flow Manager yang nantinya akan mengatur flow game kita. Lalu kita pasangkan pada object Game Flow. Pada kasus ini Game Flow Manager hanya akan mengatur proses Game Over saja. Kita akan menambahkan marker game over pada manager ini agar dapat memberitahu ke script lain kalau game nya sudah selesai. Pada proses Game Over juga kita akan menjalankan set highscore dari score manager

```
#region Singleton

private static GameFlowManager _instance = null;

public static GameFlowManager Instance
{
   get
   {
      if (_instance == null)
      {
         _instance = FindObjectOfType<GameFlowManager>();

         if (_instance == null)
         {
            Debug.LogError("Fatal Error: GameFlowManager not Found");
         }
      }

      return _instance;
   }
}

#endregion

public bool IsGameOver { get { return isGameOver; } }

private bool isGameOver = false;

private void Start()
{
   isGameOver = false;
}

public void GameOver()
{
   isGameOver = true;
   ScoreManager.Instance.SetHighScore();
}
```

Kita juga akan mendisable input saat game over dengan cara memodifikasi fungsi OnMouseDown pada TileController

Sebelum itu, pada script TileController masukkan code berikut di global classnya:

```
private GameFlowManager game;
```

Lalu, pada fungsi Awake() pada script TileController masukkan code berikut:

```
 game = GameFlowManager.Instance;
```

Barulah pada script TileController juga kita masukkan code berikut:

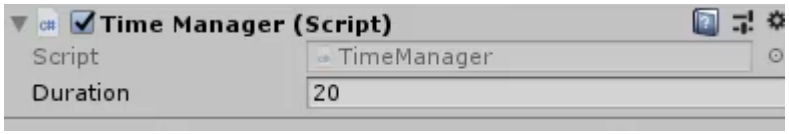
```
private void OnMouseDown()
{
    // Non Selectable conditions
    if (render.sprite == null || board.IsAnimating || game.IsGameOver)
    {
        return;
    }

    // Already selected this tile?
    if (isSelected)
    {
        Deselect();
    }
    else
    {
        // if nothing selected yet
        if (previousSelected == null)
        {
            Select();
        }

        else
        {
            // is this an adjacent tile?
            if (GetAllAdjacentTiles().Contains(previousSelected))
            {
                TileController otherTile = previousSelected;
                previousSelected.Deselect();

                // swap tile
                SwapTile(otherTile, () => {
                    if (board.GetAllMatches().Count > 0)
                    {
                        board.Process();
                    }
                    else
                    {
                        SwapTile(otherTile);
```

```
                    }
                });
            }
            // if not adjacent then change selected
            else
            {
                previousSelected.Deselect();
                Select();
            }
        }
    }
}
```

7.3. Create Time Manager

Create Time Manager

Terakhir, kita buat script baru Time Manager, kemudian kita pasangkan pada object Time. Time Manager ini bertugas untuk melakukan countdown time selama durasi yang kita inginkan. Pada saat waktu habis, Time Manager akan memanggil fungsi GameOver pada Game Flow Manager.



```
#region Singleton

private static TimeManager _instance = null;

public static TimeManager Instance
{
   get
   {
      if (_instance == null)
      {
         _instance = FindObjectOfType<TimeManager>();

         if (_instance == null)
         {
            Debug.LogError("Fatal Error: TimeManager not Found");
         }
      }

      return _instance;
   }
}

#endregion

public int duration;

private float time;

private void Start()
{
   time = 0;
}

private void Update()
{
   if (GameFlowManager.Instance.IsGameOver)
   {
      return;
```

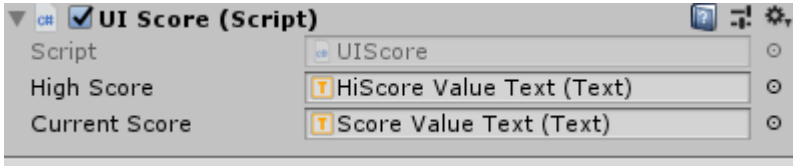
```
        }

        if (time > duration)
        {
            GameFlowManager.Instance.GameOver();
            return;
        }

        time += Time.deltaTime;
    }

    public float GetRemainingTime()
    {
        return duration - time;
    }
```

7.4. Create UI

Create UI

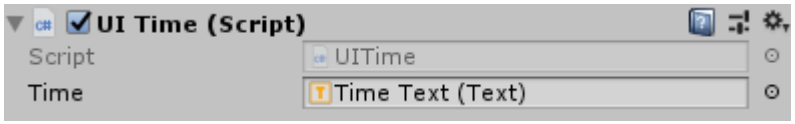
Sekarang kita perlu menambahkan UI untuk masing manager tadi seperti ini

Kita pasang script UI untuk masing panel nya. Untuk panel Score kita buat script UIScore dan kita isikan fungsi Update untuk mengganti text value nya untuk score dan highscore nya dari Score Manager



```
public Text highScore;
public Text currentScore;

private void Update()
{
   highScore.text = ScoreManager.Instance.HighScore.ToString();
   currentScore.text = ScoreManager.Instance.CurrentScore.ToString();
}

public void Show()
{
   gameObject.SetActive(true);
}

public void Hide()
{
   gameObject.SetActive(false);
}
```

Kemudian kita buat script UITime untuk panel Time. Kita isikan fungsi Update untuk mengubah text value menjadi waktu countdown. Waktu countdown ini juga kita format menjadi menit dan detik supaya lebih bagus.



```
public Text time;

private void Update()
{
   time.text = GetTimeString(TimeManager.Instance.GetRemainingTime() + 1);
}

public void Show()
{
   gameObject.SetActive(true);
}

public void Hide()
{
   gameObject.SetActive(false);
}

private string GetTimeString(float timeRemaining)
{
   int minute = Mathf.FloorToInt(timeRemaining / 60);
   int second = Mathf.FloorToInt(timeRemaining % 60);

   return string.Format("{0} : {1}", minute.ToString(), second.ToString());
}
```

Terakhir, kita tambahkan script baru UIGameOver yang akan dimunculkan saat game over. Pada fungsi Update kita hanya akan menerima tap untuk merestart game dengan cara me load ulang scene.

```
private void Update()
{
   if (Input.GetMouseButtonDown(0))
   {

UnityEngine.SceneManagement.SceneManager.LoadScene(UnityEngine.SceneManagement.SceneManager.GetActiveScene().buildIndex);
   }
}

public void Show()
{
   gameObject.SetActive(true);
}

public void Hide()
{
   gameObject.SetActive(false);
}
```

Kita hanya perlu memunculkan UIGameOver ini saat gameover terjadi. Jadi, kita matikan object Game Over ini lalu kita tambahkan pemanggilan fungsi Show nya pada fungsi GameOver di Game Flow Manager

```
[Header("UI")]
public UIGameOver GameOverUI;

public void GameOver()
{
   isGameOver = true;
   ScoreManager.Instance.SetHighScore();
   GameOverUI.Show();
}
```

Dan selesai, sekarang game kita sudah menjadi game match three yang keren dengan menambahkan timer dan skor. Kalian bisa ajak teman kalian untuk duel. Yang mendapat skor paling tinggi menang.

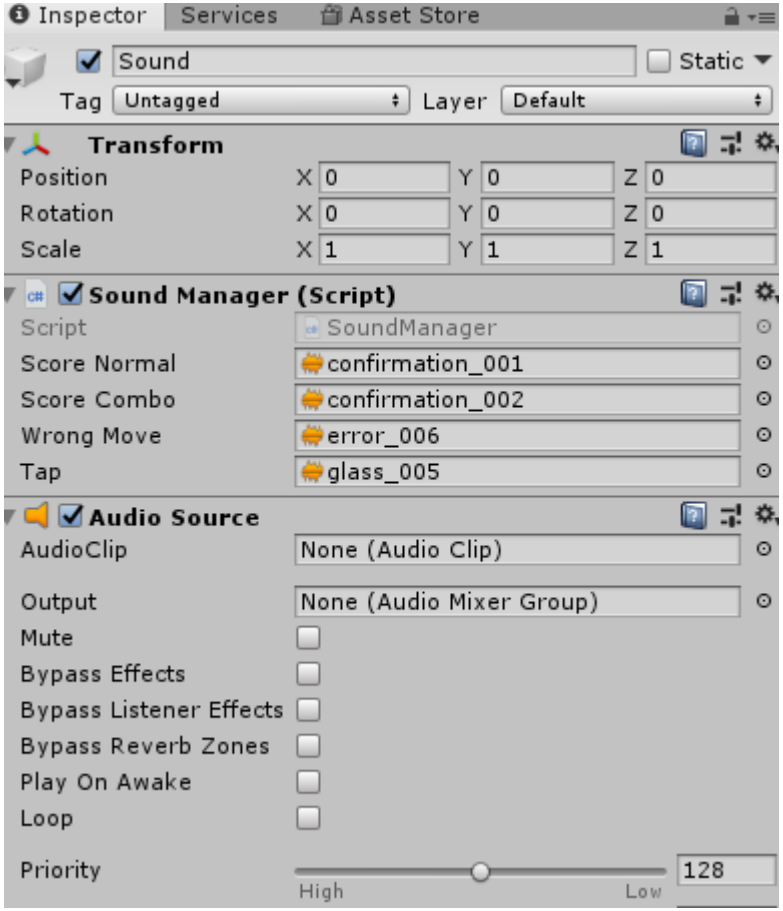
8. Bagian 6 - Audio is the Magic

Agar game kita menjadi lebih keren, kita bisa menambahkan audio yang sudah kita import di awal tadi. Hal pertama yang kita lakukan adalah dengan membuat game object berjenis audiosource bernama Sound dan membuat script SoundManager. Lalu kita menjadikannya singleton seperti manager yang lain. Kita buat field untuk setiap audio clip yang kita import tadi beserta fungsi untuk menjalankan audionya. Berikut adalah code untuk dimasukkan pada script SoundManager:

```
#region Singleton

private static SoundManager _instance = null;

public static SoundManager Instance
{
   get
   {
      if (_instance == null)
      {
         _instance = FindObjectOfType<SoundManager>();

         if (_instance == null)
         {
            Debug.LogError("Fatal Error: SoundManager not Found");
         }
      }

      return _instance;
   }
}

#endregion

public AudioClip scoreNormal;
public AudioClip scoreCombo;

public AudioClip wrongMove;

public AudioClip tap;

private AudioSource player;

private void Start()
{
   player = GetComponent<AudioSource>();
}
```

```
public void PlayScore(bool isCombo)
{
    if (isCombo)
    {
        player.PlayOneShot(scoreCombo);
    }
    else
    {
        player.PlayOneShot(scoreNormal);
    }
}

public void PlayWrong()
{
    player.PlayOneShot(wrongMove);
}

public void PlayTap()
{
    player.PlayOneShot(tap);
}
```

Lalu kita pasangkan pada object Sound dan kita pasangkan juga komponen AudioSource. Kita disable Play on Awake agar tidak otomatis play audio di awal game dimulai.



Sekarang kita tinggal tambahkan pemanggilan fungsi untuk menjalankan audio sesuai dengan kebutuhan kita, pertama kita tambahkan PlayScore saat score bertambah, yaitu fungsi IncrementCurrentScore pada Score Manager.

```
public void IncrementCurrentScore(int tileCount, int comboCount)
{
   currentScore += (tileCount * tileRatio) * (comboCount * comboRatio);

   SoundManager.Instance.PlayScore(comboCount > 1);
}
```

Lalu kita tambahkan PlayWrong setelah swap tile jika tidak menemukan tile yang match dan PlayTap saat kita memilih sebuah tile. Kita bisa memasangnya pada fungsi OnMouseDown pada TileController.

```
private void OnMouseDown()
{
    // Non Selectable conditions
    if (render.sprite == null || board.IsAnimating || game.IsGameOver)
    {
        return;
    }

    SoundManager.Instance.PlayTap();

    // Already selected this tile?
    if (isSelected)
    {
        Deselect();
    }
    else
    {
        // if nothing selected yet
        if (previousSelected == null)
        {
            Select();
        }

        else
        {
            // is this an adjacent tiles?
            if (GetAllAdjacentTiles().Contains(previousSelected))
            {
                TileController otherTile = previousSelected;
                previousSelected.Deselect();

                // swap tile
                SwapTile(otherTile, () => {
                    if (board.GetAllMatches().Count > 0)
                    {
                        board.Process();
                    }
                    else
```

```
                {
                    SoundManager.Instance.PlayWrong();
                    SwapTile(otherTile);
                }
            });
        }
        // if not adjacent then change selected
        else
        {
            previousSelected.Deselect();
            Select();
        }
    }
  }
}
```

Sekarang game kita sudah memiliki suara yang membuat game kita semakin menarik.