Writing & Presentation Assignment Week 10

GDD & GCD

Berikut link GDD Kelompok 4 :

https://docs.google.com/document/d/1FevNBjvFe6ITbThIosKcocWBH68jvHpvQhyG2-9X8bU/edit?usp=sharing

Hari ini saya belajar mempelajar Game Desain Computer dan terbagi menjadi breakout room untuk mengerjakan GDD & GDC dan mempresentasikannya.

A. Game Concept Document (GCD)

Game Concept Document adalah **untuk menjelaskan tentang konsep game yang akan dibuat secara ringkas (biasanya terdiri dari 1 – 2 halaman)**. GCD juga menjelaskan tentangesensi game, seperti plot, setting, dan mekanik game dasar. Game konsep harus menjawab pertanyaan dasar, seperti :

1. Apa yang player lakukan di game?
2. Apa obstacle yang dilakukan player?
3. Apa tujuan yang player capai?

Semua game harus punya 3 pertanyaan dasar tersebut. Tapi jika ingin lebih detail lagi, harus mencakup :

1. Introduction
2. Deskripsi game
3. Kunci fitur apa saja yang ada di game
4. Genre
5. Platform membangun game
6. Konsep gambar (tema)

B. Game Design Document

Game Design Document adalah **sumber utama pembuatan game (*single source of truth* = dasar) agar pengembangan jelas arah tanpa perlu lagi bertanya dengan Game Designer dan tidak melenceng apa yang sudah dicanangkan**. Biasanya sudah ditentukan oleh Game Manager di studio.

GDD ini sebagai tempat Game Designer untuk menyimpan ide game design (dalam bentuk gambar/diagram/tulisan, untuk media tidak ada aturan tertentu, biasanya lebih baik gambar daripada tulisan untuk lebih jelas flowchartnya) dan development (programmer & artist) untuk menerima instruksi apa yang mau mereka buat. Intinya GDD sebagai **tools komunikasi antar tim**.

1. Tujuan lain pembuatan GDD ini mendetail, seperti :
    - Visi game (menyatukan beberapa ide dari beberapa kepala, agar tidak bentrokharus direncanakan oleh beberapa orang)
    - Menjelaskan konten (biasanya alur story, monetisasi, event, tema hari besar)
    - Merencenakan implementasi (missal, 3 bulan pertama sampai 10 level, mainmenu & inventory bisa bulan depan)

Fungsi GDD ada 2 :

- Untuk tim internal (game development). Semua tim harus membaca GDD ini agar mengerti.
- Untuk tim eksternal (investor & publisher). Di dalamnya harus mencakup :
  1) Analisis market, keunggulan produk dari kompetitor
  2) Spesifikasi teknikal mencakup
     a. **Game ini akan dibuat di platform mana? Apakah PS 5? MacOS? Android?**Hal ini berguna untuk menentukan kapasitas asset agar tidak memakan memori berlebihan).
     b. Perjanjian projek, seperti struktur folder dan penamaan file. Untuk asset penamaan file, apakah nama karakternya? Ditampilkan di level mana? Biasanya developer indie berantakan. Untuk developer pro, nama parent & child folder, perbedaan folder texture dan material untuk karakter tercantum dalam GDD.
     c. Analisis biaya, ditentukan oleh produser, **tidak wajib ada**
     d. Pemasukan projek. ditentukan oleh produser, **tidak wajib ada**

2. Tujuan GDD : Reduce / remove
   - Miskomunikasi, jangan ada salah paham dan beda pemikiran
   - Kecanggungan tentang pertanyaan “apa yang harus saya lakukan”
   - Menghilangkan kebingungan/ketidakjelasan

GDD ini lebih kearah teknis (bukan story dengan Bahasa novel). Di GDD ini membuat formula alur cerita. Lebih ke arah kelugasan, tidak ada ambiguitas, dan ringkas.

Manfaat GDD dengan pedoman :

- Dokumen utama yang pada dasarnya mencatat setiap aspek proyek
- Hapus hypebring bisa dilakukan
- Kejelasan dan kepastian

Beberapa kesalahan dalam membuat GDD :

- Struktur kebahasaan buruk dan penggunaan Bahasa salah
- Terlalu besar *scoop* (besaran beban kerja) biasanya dilakukan oleh pemula
- Terlalu umum dan kurang mendetail
- Off base / tidak dapat diterapkan dalam konteks proyek / perusahaan (tidak berguna /tidak perlu)

Media pembuatan GDD : Notion. Ms.Word, Figma, PPT

Kalau di gdocs harus menambah daftar isi Ketika ada isian baru. Jadi alangkah lebih baik pakai notion. Semakin berkembang dunia virtual semakin menembus Batasan yang ada di dunia riil, buktinyatidak membutuhkan dokumen berlembar – lembar, bisa juga membuatnya lewat wiki.

UNITY TEST FRAMEWORK

Hari ini membahas tentang testing di unity. Software testing adalah proses evaluasi dan verifikasi software apakah behaviour sudah sesuai atau belum, bahwa software harus mengeluarkan output yang sesuai (periksa kode, play mode apakah masih ada bug atau animasijalan?).

Kalau pakai testing ini, developer prevent bug dan sudah tes diawal sebelum di develop. Ke depannya tidak akan mengulang testing di bagian tertentu (khusus game besar dan kolaborasi).Hal ini dilakukan agar tidak muncul bug, mengurangi biaya produksi game, dan mengurangi bentrok jobdesk antar kelompok (kolaborasi tim).

Kalau di game, ada hal-hal yang tidak bisa diotomasi dengan bot (harus manual dicoba denganplayer manusia) makanya studio game besar punya ruangan khusus puluhan computer dan player untuk testing game terus menerus cari bug.

Ada 2 cara testing :

End To End	Bikin script / bot yang melakukan semua hal dari awal – akhir, dari buka menu – exit, seolah – olah ada yang klik tombol berapa detik nyala, dan diotomasi dengan script (bot).
Unit Testing	Tes 1 komponen/individu unit bagian dari software yang dibuat. Biasanya yang di tes adalah **methodnya**. Bisa juga nilai setelah dilakukan sesuatu proses baru bisa lihat nilainya dan biasanya hanya tes 1 method saja.

Unit Testing disini sudah dikasih frameworknya di unity. Ini semua programmer harus tau karena perusahaan besar dilakukan khusus 1 orang programmer. Programmer ini kerjanya ngetes kerjaannya yang dilakukan koleganya.

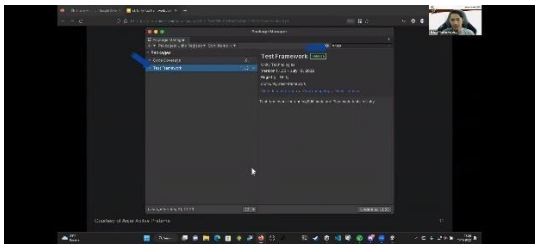
Cara untuk Unit Testing di unity :

Pakai **Unity Testing Framework** (**UTF**) package, dites saat **play mode** dan **edit** (plugin/script editor) dengan target platform iOS, Android, Windows. Fitur yang tersedia :

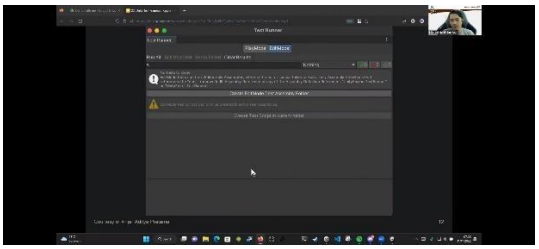
a. Unit Testing
b. Automatic Testing

Cara installnya :

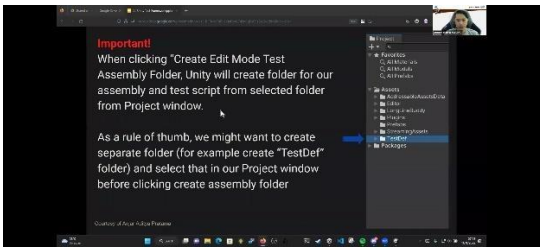
a) Install **UTF** di **package manager (windows windows => package manager)**
b) Windows => General => Test Runner
c) Membuat **assembly definitions** dipisah untuk game dan pengembangan game (testing)
d) Membuat **test case** dalam bentuk fungsi yang akan di run
e) Membuat **assembly definitions references** (bedanya yang test harus referenced dengangame dan sebaliknya. Tapi Ketika di build game nya, yang test tidak akan ikut ke build(terpisah)).



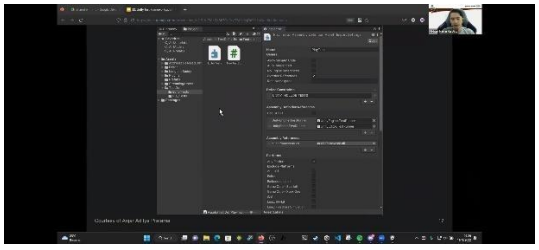
Setelah install ada window khusus, seperti gambar dibawah ini dan buat folder khusus,
assembly folder sama buat scriptnya.



Test Runner API untuk **unit test** sebetulnya dari semua script Cuma khusus folder (**assembly folder**) itu aja. Memungkinkan developer membuat daftar untuk memisahkan edit mode, playmode, atau keduanya, atau tidak semuanya.



Folder nya itu biasanya dinamakan bebas oleh developer tapi langsung di tes oleh unity tapi sebaiknya gunakan **TES** seperti gambar, agar tidak bingung.



Jangan lupa bagi lagi menjadi 2 folder untuk **play test & editor test** secara manual karena unity membebaskan tapi lebih baik ikuti gambar. Tampilannya mirip Cuma beda penamannya.

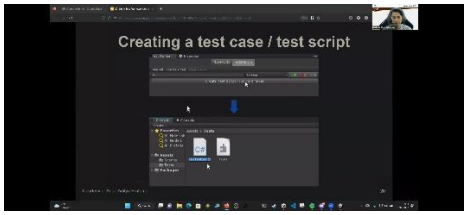
Lihat pada gambar

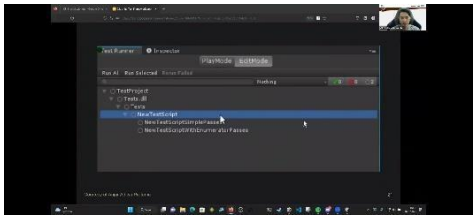
- Sebelah kiri (**berbentuk puzzle warna biru**) = **Assembly Definitions**. Ketika di klik akan seperti di gambar.
- Sebelah kanan (**berbentuk pagar warna hijau** = **Script untuk testing**

ISTILAH PENTING (DALAM GAMBAR)	
Define Constraint	Biasanya digunakan untuk mengkompilasi atau mereferensikan program, saat ini memakai **UNITY_INCLUDE_TEST**
Assembly Definitions References	Dalam 1 file bisa mereferensikan kelas di file lain sesuai apa yang ada di dalam **Assembly Definitions**. Jika tidak terkait, maka tidak akan tahu kalau ada script **Assembly Definitions** di flder lain.
Name & General	Untuk memberi nama file dan setting **Assembly Definitions**, seperti Allow 'unsafe' code, Auto Referenced, dsb
Platform	Developer bisa memilih platform untuk testing dimana. Untuk play test semua dicentang saja.

Assembly adalah koleksi / daftar / tipe data / kelas yang udah dibuat dikumpulkan dalam satufungsionalitas bahwa mereka bisa bekerja sama (referenced). Dia lebih kearah menyimpan alamat file dan mirip seperti folder karena menyimpan **testscript.cs** developer mendefinisikandan akan di lihat oleh compiler.

Intinya, **Assembly** adalah **folder tempat menyimpan testscript**.





Bisa dilihat kalau udah dibuat akan ada testscript dan didalamnya ada fungsi tes yang dibuat. Contoh Script :

```
using System.Collections;
using System.Collections.Generic;
using NUnit.Framework;
using UnityEngine;
using UnityEngine.TestTools;

public class NewTestScript
{
    // A Test behaves as an ordinary method
    [Test]
    public void NewTestScriptSimplePasses()
    {
        string defaultName = "player";
        string playerName = "foo";

        // Use the Assert class to test conditions
        Assert.AreEqual(defaultName, playerName);
    }

    // A UnityTest behaves like a coroutine in Play Mode. In Edit Mode you can use
    // `yield return null;` to skip a frame.
    [UnityTest]
    public IEnumerator NewTestScriptWithEnumeratorPasses()
    {
        // Use the Assert class to test conditions.
        // Use yield to skip a frame.
        yield return null;
    }
```

Di dalamnya ada 2 fungsi yang mau di test, yaitu **public void NewTestScriptSimplePasses()** dan **public void NewTestScriptWithEnumeratorPasses()**. Ini bisa di run dan akan terceklis di listnya.