BAB 2
LANDASAN KEPUSTAKAAN

2.1 Projection Mapping

Projection mapping, juga dikenal sebagai *video mapping* dan *augmented reality*, adalah teknologi proyeksi yang digunakan untuk mengubah benda, sering berbentuk tidak teratur, menjadi tampilan permukaan untuk proyeksi video. Objeknya dapat berupa lanskap industri yang kompleks, seperti bangunan, benda-benda dalam ruangan kecil ataupun panggung teater. Dengan menggunakan perangkat lunak khusus, benda dua atau tiga dimensi spasial akan dipetakan pada program virtual yang meniru lingkungan nyata dimana hal itu diproyeksikan. *Software* ini dapat berinteraksi dengan proyektor untuk mencocokkan gambar yang diinginkan ke permukaan objek (Jones, 2015).

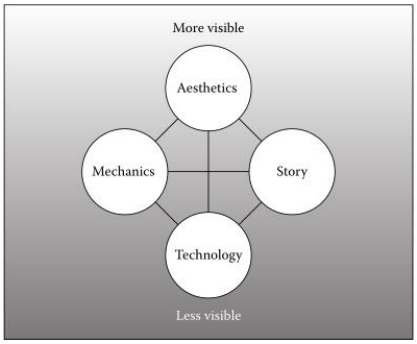
Projection mapping sering digunakan untuk iklan, konser, teater, komputasi, dekorasi dan hal lain yang bisa di-imanijasi-kan seperti game. Penggabungan konten virtual dan benda dapat dilakukan menggunakan *software* khusus maupun dengan usaha sendiri.

2.2 *Game*

Game merupakan kata di dalam Bahasa Inggris yang diartikan sebagai permainan di dalam Bahasa Indonesia. *Game* diartikan sebagai sebuah aktivitas yang setidaknya mempunyai satu pemain, mempunyai peraturan dan mempunyai syarat kemenangan. Dengan begitu *video game* berarti adalah sebuah *game* yang dimainkan di layar monitor (Rogers, 2010).

2.2.1 Elemen di dalam *Game*

Menurut Schell (2015), ada begitu banyak cara untuk membagi dan mengklasifikasikan begitu banyak *element* yang membentuk suatu *game*, dan mengelompokkannya menjadi 4 bagian penting yang berhubungan satu sama lain dan menggambarkannya seperti diagram di gambar 2.1. Schell menyebutnya sebagai *elemental tetrad*.



Gambar 2.1 *Elemental Tetrad*

Sumber: (Schell, 2015)

Empat dasar *element game* yang saling berhubungan satu sama lain tersebut adalah sebagai berikut.

1. ***Mechanic***
   Mechanic adalah prosedur dan aturan dari suatu *game*. *Mechanic* menjelaskan tujuan dari *game* tersebut, bagaimana cara *player* supaya dapat ataupun tidak dapat mencapai tujuan tersebut, dan apa yang terjadi jika *player* mencoba melakukannya. Jika *game* dibandingkan dengan media hiburan *linear* yang lain (seperti buku, *film*, dll), bisa diketahui bahwa media hiburan *linear* tersebut mempunyai elemen-elemen teknologi, cerita, *aesthetic*, tetapi media tersebut tidak mempunyai mekanik, dimana mekanik itulah yang membuat *game* itu menjadi *game.*
2. ***Story* (Cerita)**
   Cerita merupakan urutan-urutan kejadian yang terjadi di dalam *game*. Bisa berupa *linear* dan sesuai naskah*,* bisa juga bercabang dan tampil secara tiba-tiba. Ketika ada cerita yang ingin disampaikan melalui *game* tersebut, mekaniklah yang juga perlu dipilih untuk memperkuat dan memunculkan cerita tersebut.
3. ***Aestethic* (Estetika)**
   Estetika meliputi bagaimana tampak, bunyi, bau, rasa, dan perasaan dari sebuah *game.* Estetika adalah aspek yang sangat penting di dalam *game design* karena aspek inilah yang paling terasa dalam pengalaman bermain seseorang. Contohnya seperti ketika ada suatu tampilan, atau bunyi yang ingin pemain rasakan dan ingin terjun ke dalam dunia *game* tersebut.
4. ***Technology***
   Teknologi di sini tidak berarti hanya termasuk "teknologi tingkat tinggi" saja, tetapi juga termasuk *material* dan interaksi yang memungkinkan game berjalan. Teknologi yang dipilih untuk suatu *game* akan memungkinkan suatu hal melakukan sesuatu, dan juga melarang/membatasi *game* untuk melakukan hal yang lain.

2.2.2 Genre Game

Dalam beberapa tahun belakangan ini, *game* telah terbagi ke dalam banyak *genre* dan *sub-genre*. *Genre game* digunakan untuk menjelaskan *style* dan konsep dari *gameplay game* tersebut (Rogers, 2010). *Genre-genre game* secara umum menurut Rogers dalam bukunya yang berjudul "*Level Up! The Guide to Great Video Game Design*" yaitu:

1. ***Action***
   Genre ini membutuhkan koordinasi tangan dan mata untuk memainkannya. *Genre action* terbagi menjadi beberapa *sub genre* seperti *action adventure, action arcade, platformer, stealth, fighting* dan terakhir *Beat'em up/hack 'n' slash*.

2. ***Shooter***
   Genre yang berfokus pada kegiatan membidikkan senjata dan menembakkan musuh. *Genre shooter* terbagi menjadi 3 *sub genre* seperti *first person shooter*, *shoot e'm up & third person shooter*
3. ***Adventure***
   Genre yang berfokus pada pemecahan teka-teki, mengkoleksi barang, & memanajemen persediaan. *Genre adventure* terbagi menjadi beberapa *sub genre* seperti *graphical adventure*, *role-playing games* (RPG), *massive multiplayer online role - playing game* (MMORPG) dan *survival* atau *horror*.
4. ***Construction/Management***
   Genre ini menantang pemain untuk membangun dan memperluas lokasi yang ada dengan sumber daya terbatas yang ada. Bisa didasari dari sebuah cerita atau "mainan".
5. ***Life simulation***
   Permainan dengan *genre* ini hampir mirip dengan permainan ber-*genre management*, tetapi khususnya hanya berputar-putar pada suatu tempat tertentu dan memelihara hubungan dengan kehidupan buatan dalam *game*.
6. ***Music/rhythm***
   Genre ini membuat pemain fokus untuk mencocokan ritme musik atau mencapai batas skor.
7. ***Party***
   Genre yang berdasarkan bermain kompetitif yang dirancang khusus agar dapat dimainkan oleh beberapa pemain.
8. ***Puzzle***
   Genre puzzle didasarkan pada logika dan penyelesaian sebuah pola. Permainan *genre* ini bisa berlangsung sangat lama dan penuh dengan metode-metode dalam menyelesaikannya.
9. ***Sport***
   Genre permainan didasarkan kompetisi atletik, baik olahraga *extreme* maupun tradisional. *Genre sport* ini terbagi menjadi *sport management*, yaitu permainan untuk memanajamen dan mengelola langsung pemain maupun tim.
10. ***Strategy***
    Genre yang didasarkan pada berpikir dan berencana dalam menyusun sebuah strategi. *Genre strategi* terbagi menjadi 3 *sub genre* seperti *Real Time Strategi* (RTS), *Turn-based,* dan*Tower defense.*
11. ***Vehicle simulation***
    Genre ini mensimulasikan *piloting*/mengendarai kendaraan supaya membuat pemain merasakan pengalaman seakan-akan mengendarai kendaraan seperti pada aslinya. *Genre vehicle simulation* terbagi menjadi 2 *sub genre* seperti *driving* dan *flying*.

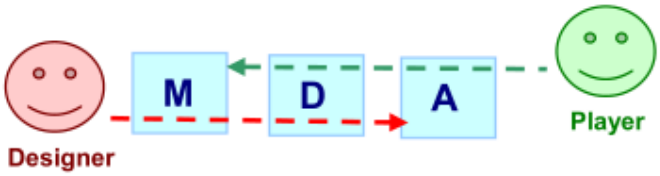
2.2.3 Game Platformer

Sebuah *game platform* (atau *platformer*) adalah *video game* yang mempunyai *gameplay* membimbing karakter untuk melompat antara *platform*, rintangan, atau keduanya untuk melanjutkan permainan. Tantangan-tantangan ini dikenal sebagai *jumping puzzle* atau *freerunning*. Pemain mengontrol lompatan untuk menghindari avatar jatuh dari *platform* atau gagal melakukan lompatan yang diperlukan. Unsur pemersatu yang paling umum dari game genre ini adalah tombol melompat. Contoh terkenal dari *game platformer* ini ialah *Super Mario Bros*. Mekanik permainan ini dimana pemain membimbing karakter untuk melompati antara *platform* dan rintangan, dalam genre lain yang biasa disebut *platforming*, sebuah kata lain yang berasal dari kata *platform* (Greenslade, 2006).

2.3 MDA (*Mechanics/Dynamics/Aesthetics*)

MDA merupakan singkatan dari *mechanic*, *dynamics* dan *aesthetics*. *Mechanics*, *Dynamics,* dan *Aesthetic* adalah tiga lapisan yang mendefinisikan sebuah *game*. Di dalam MDA *framework* hal ini memiliki arti yang sangat spesifik (Hunicke, LeBlanc, & Zubek, 2001), yaitu:

1. ***Mechanics*** menggambarkan komponen tertentu dari permainan, pada tingkat representasi data dan algoritma.
2. ***Dynamics*** menggambarkan perilaku *run-time* dari mekanik yang bekerja pada *input* dan *output* pemain satu sama lain dari waktu ke waktu.
3. ***Aesthetics*** menjelaskan respon emosional yang diinginkan bangkit pada pemain, ketika ia berinteraksi dengan sistem permainan.



Gambar 2.2 MDA dari Perspektif *Game Designer* dan Pemain

Sumber: (Hunicke, LeBlanc, & Zubek, 2001)

Desainer game hanya memiliki kontrol langsung dari *mechanics* permainan; mekanik bekerja sama untuk menghasilkan *dynamics*, yang pada gilirannya menghasilkan *aesthetics*. Mereka ingin membuat permainan mereka menyenangkan dan menarik, tetapi hanya memiliki kontrol tidak langsung kepada pengalaman pemain. Dengan demikian, desainer cenderung melihat *mechanics* dan bekerja ke arah luar.

Sebaliknya, pemain akan segera akrab dengan respons emosional mereka sendiri kepada sebuah permainan, terlepas dari apakah mereka memahami aturan-aturan yang mendasari permainan tersebut. Seseorang dapat menikmati *Wii Sports Tennis* tanpa harus mengetahui dimensi yang tepat dari lapangan virtual

yang ada. Hanya melalui beberapa jam melalui pengamatan dan deduksi, *dynamics* dan *mechanics game* secara bertahap menjadi jelas.

2.4 *Teori Pengujian*

2.4.1 *Whitebox Testing*

White box testing merupakan suatu teknik pengujian yang mengevaluasi kode dan struktur internal dari program. Teknik pengujian *whitebox testing* digunakan oleh pengembang dan penguji. Pengujian ini membantu mereka memahami kode mana yang berjalan dengan benar dan mana yang tidak. Teknik ini dapat menunjukkan jikalau ada logika yang kurang ataupun *typo* di dalam kode, yang dapat mengakibatkan beberapa konsekuensi negatif bagi program (Roy, 2016).

Ada 2 tahap sederhana dalam melakukan *whitebox testing* yaitu:

1. Mengerti fungsionalitas dari program tersebut melalui *source code*.
2. Melakukan pengujiannya.

Menurut Roy (2016), ada tiga teknik utama di dalam *White box Testing* yang dimana setiap teknik tersebut memiliki metode tersendiri, yaitu:

1. *Statement Coverage*

   Statement coverage, seperti arti dari namanya merupakan metode dengan memvalidasi setiap dan seluruh baris dari kode yang akan dieksekusi, minimal sekali.

2. *Branch Coverage*

   "Branch" di dalam bahasa pemrograman memiliki dua cabang yaitu *false* dan *true*. Jadi di dalam *branch coverage* setiap cabang yang dieksekusi akan divalidasi minimal sekali. Dan di dalam "If statement" akan ada 2 kondisi pengujian yaitu:

   - Satu dalam memvalidasi cabang "true"
   - Dan lainnya dalam memvalidasi cabang "false"

3. *Path Coverage*

   Path coverage menguji semua *path* di dalam program. Ini merupakan teknik komprehensif yang memastikan seluruh *path* di dalam program dijalankan setidaknya sekali.

2.4.2 *Playtesting*

Playtesting adalah suatu cara menguji sebuah *game* tanpa memikirkan sisi koding atau *bug/error* dari *game* tersebut tetapi untuk menilai seberapa menyenangkan dan seberapa efisiennya *game* ini (Levy & Noak, 2010). Playtesting memerlukan penguji untuk mencoba atau menguji semua kemungkinan cara bermain dan memastikan semuanya itu menyenangkan. Hal ini merupakan hal

yang penting karena kesalahan di suatu *sub-system* dalam *game* dapat mencederai seluruh kesenangan pengalaman bermain.

Menurut Felder (2015), ada 5 tahap dari *Playtesting* dimana setiap tahapnya tersebut memliki tujuan tersendiri yaitu:

1. *Concept Testing*

   Tahap awal dimana hal yang diuji adalah hal yang paling fundamental yaitu konsep utama *game*. Konsep cara bermain tersebut diuji apakah sudah betul menarik dan menyenangkan dari dasarnya. Tahap ini adalah tahap yang dapat dilakukan dengan sedikit alat bantuan.

2. *Scattershot Testing*

   Tahap selanjutnya di mana di antara mekanik-mekanik permainan yang telah diimplementasikan diuji satu per satu dan dipastikan dimana bagian mekanik yang paling menarik dan dapat menjadi suatu daya tarik utama *game*. Dan dari mekanik yang paling dianggap menarik itulah bisa diterapkan ide-ide yang ada di dalamnya ke dalam mekanik-mekanik *game* lainnya. Kekurangan dari tahap pengujian ini ialah tingginya kompleksitas yang diperlukan dalam menguji dan dibutuhkan *playtester* yang betul-betul memahami seluk-beluk *game* tersebut.

3. *Experience Testing*

   Tahap dimana saatnya memastikan opini dan perasaan dari playtester dari setiap bagian di game seperti mekanik, map, cerita, fitur baru, dll. *Form* yang dipakai di bagian ini berupa kuesioner. Hasil dari setiap bagian yang diuji dituangkan dalam beberapa bagian berupa, bagus, buruk, ataupun biasa saja.

4. *Gameplay Stress Testing*

   Di tahap ini dibutuhkan banyak *playtester* dan sebisa mungkin dapat mengecek *gameplay* secara detail karena di tahap ini akan diuji apakah ada *glitch* ataupun *bug* yang mengganggu jalannya permainan ataupun yang merusak mekanik permainan yang telah disediakan.

5. *Accessibiilty Testing*

   Di tahap akhir ini perlu diuji apakah *game* yang telah dibuat tersebut mudah diakses dan dimengerti oleh semua pemain. *Playtester* di sini perlu memberikan opininya, khususnya jikalau mereka tidak mengerti jalannya permainan ataupun *stuck* di suatu tempat di dalam *game*.