BAB II
TINJAUAN PUSTAKAN DAN DASAR TEORI

2.1 Tinjauan Pustaka

Beberapa penelitian sebelumnya yang dilakukan terkait pembuatan *game* dengan implementasi metode *Finite State Machine* (FSM), antara lain terdapat pada tabel 2.1.

Tabel 2.1 Tabel Perbandingan Penelitian

Penulis	**Objek / Topik / Genre**	**Teknologi / Metode**	**Bahasa Pemrograman**	**Perangkat yang digunakan**
Huda, Ahmad Samsul (2016), Universitas Islam Negeri Maulana Malik Ibrahim.	*Game edukasi "Cepat Tepat"* pada smartphone	FSM (*Finite State Machine*)	Java	*Eclipse*
Miftah Fauzan, 2016 (universitas Mulawarman)	*Game* "*The Relationship*"	FSM (*Finite State Machine*)	C#	*Unity3D*
Silvia Rostianingsih, 2013, (Universitas Kristen Petra)	Simulasi Pertanian dan Peternakan	FSM (*Finite State Machine*)	*Actionscript*	*Adobe Flash Professional CS3*
Virgananta, 2013 (Politeknik Elektronika Negeri Surabaya)	Cerita "Malin Kundang"	FSM (*Finite State Machine*)	-	*Stencyl*
Dahlan, Baihaqi, Erliana, 2015 (Universitas Malikussaleh)	*Game* Edukasi untuk anak-anak	FSM (*Finite State Machine*)	RGSX (*Ruby Game Scripting System*)	*RPG Maker VX.*
Ari Kustanto, 2023, (Universitas Teknologi Digital Indonesia Yogyakarta)	*Game "*Rama dan Shinta"	FSM (*Finite State Machine*)	C#	*Unity3D*

Pada aplikasi yang akan dibuat memiliki perbedaan alur cerita yang diambil memuat unsur kebudayaan di Indonesia, untuk membuat *gameplay* yang menarik, digunakan implementasi metode *Finite State Machine* pada karakter musuh. Metode *Finite State Machine* berfokus pada tingkah laku musuh (*Non Player Character*) yang menggunakan 3 hal, yaitu *State* (Keadaan), *Event* (kejadian) dan *action* (aksi).

2.2 Dasar Teori

2.2.1 Pengertian *Game*

Dalam kamus Bahasa Indonesia "*Game*" diartikan sebagai permainan. Sebuah *game* merupakan sebuah sistem di mana pemain terlibat dalam konflik buatan, di sini pemain berinteraksi dengan sistem dan konflik dalam permainan yang merupakan rekayasa atau buatan. Dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan. *Game* biasanya diperuntukkan untuk hiburan. Tetapi terkadang juga *game* ditujukan untuk edukasi, karena sebenarnya *game* juga penting untuk mengasah perkembangan otak, meningkatkan konsentrasi, dan melatih memecahkan masalah dengan tepat dan cepat.

2.2.2 *Unity Game Engine*

Unity Game Engine adalah *software* yang digunakan untuk membuat video *Game* berbasis dua atau tiga dimensi dan dapat digunakan secara gratis, selain untuk membuat *Game*, *Unity* 3D juga dapat digunakan untuk membuat konten yang interaktif lainnya seperti, visual arsitektur dan *real-time* 3D animasi, selain sebagai

Game engine Unity 3D juga dapat digunakan sebagai sebuah editor bagi *Game* yang sudah ada.

Unity 3D dibuat dengan menggunakan bahasa program C++, tapi pengguna tidak perlu menggunakan Bahasa C++ yang sulit, karena *Unity 3D* mendukung Bahasa program lain seperti *JavaScript*, C#, dan Boo, Unity memiliki kemiripan dengan *Game engine* lainnya seperti, Blender *Game engine*, *Virtools*, *Game*studio, adapun kelebihan dari *Unity 3D*, *Unity* dapat dioperasikan pada platform Windows dan *Mac Os* dan dapat menghasilkan *Game* untuk Windows, Mac, Linux, Wii, iPad, iPhone, *google Android* dan juga browser. Untuk browser, kita memerlukan sebuah *plugin*, yaitu Unity Web *Player*, sama halnya dengan *Flash Player* pada *Browser*. *Game* Unity 3D juga mendukung dalam pembuatan *Game* untuk *console Game* Xbox 360 dan PlayStation3.

2.2.3 Bahasa Pemrograman C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET *Framework* (Fowler, 2005).

C# adalah Java versi Microsoft, sebuah bahasa multi platform yang didesain untuk bisa berjalan di berbagai mesin. C# adalah pemrograman berorientasi *Object* (OOP). C# memiliki kekuatan bahasa C++ dan probabilitas seperti Java. Fitur-fitur yang diambilnya dari bahasa C++ dan Java adalah desain berorientasi objek, seperti *garbage collection*, *reflection*, akar kelas (*root class*), dan juga penyederhanaan terhadap pewarisan jamak (*multiple inheritance*). Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang bersifat *general-purpose* (untuk tujuan jamak),

berorientasi objek, modern, dan sederhana. C# ditujukan agar cocok digunakan untuk menulis program aplikasi baik dalam *system clien-server* (*hosted system*) maupun *system embedded* (*embedded system*), mulai dari program aplikasi yang sangat besar yang menggunakan sistem operasi yang canggih hingga kepada program aplikasi yang sangat kecil. Meskipun aplikasi C# ditujukan agar bersifat 'ekonomis' dalam hal kebutuhan pemrosesan dan memori komputer, bahasa C# tidak ditujukan untuk bersaing secara langsung dengan kinerja dan ukuran program aplikasi yang dibuat dengan menggunakan bahasa pemrograman C (Komputer, 2008).

2.2.4 Android

Android adalah sistem operasi perangkat *mobile* berbasis Linux, yang menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi sendiri. Saat perilisan perdana *Android* yaitu 5 November 2017, Android bersama *Open Headset Alliance* menyatakan mendukung pengembangan open *source* pada perangkat *mobile*. Di lain pihak, Google merilis kode-kode *Android* di bawah lisensi *Apache*, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler. (Safaat H, 2012).

2.2.5 *Finite State Machine* (FMS)

Finite State Machine adalah model matematis yang digunakan untuk menjelaskan perilaku sistem yang memiliki keadaan atau *state* yang terbatas. Dalam model ini, sistem dianggap sebagai mesin yang dapat berada dalam satu dari beberapa keadaan (*state*) yang berbeda pada waktu tertentu.

FSM terdiri dari beberapa komponen utama:

a. Status (*States*):

   Status adalah kondisi atau situasi khusus di mana sistem dapat berada. Contoh dalam permainan adalah "Berjalan", "Lari", "Menyerang", "Bersembunyi", dan sebagainya. Status merupakan elemen utama yang ditentukan dalam FSM.

b. Peristiwa (*Events*):

   Peristiwa adalah pemicu yang menginisiasi perubahan dari satu status ke status lainnya. Dalam konteks permainan, peristiwa dapat berupa *input* pemain, aktivasi tombol, *timer* berakhir, dan banyak lagi.

c. Transisi (*Transitions*):

   Transisi menggambarkan bagaimana sistem pindah dari satu status ke status lain ketika peristiwa tertentu terjadi. Ini dinyatakan dalam diagram sebagai panah yang menghubungkan status-status.

d. Aksi (*Actions*):

   Setiap transisi dapat disertai dengan aksi atau perilaku yang terkait. Misalnya, ketika karakter berpindah dari status "Lari" ke status "Menyerang", aksi yang terjadi bisa berupa animasi serangan.

FSM sangat berguna dalam pembuatan *game* karena dapat menggambarkan perilaku karakter, objek, dan entitas lainnya dengan cara yang terstruktur dan mudah dipahami. Dengan mengatur FSM dengan benar, pengembang dapat merancang sistem interaksi yang kompleks, mengelola transisi yang tepat antara status-status, dan memberikan pengalaman bermain yang konsisten dan menyenangkan bagi pemain.