Pengembangan 2D Platformer Game Engine menggunakan Data-driven Programming

Wisnu Raditya F¹, Kemas Rahmat S.W., S.T., M.Eng.², Bayu Munajat, S.T., M.T.³

^{1,2,3}Fakultas Informatika Universitas Telkom Bandung

¹wisnu.ferdian@outlook.com, ²bagindokemas@telkomuniversity.ac.id, ³bayumunajat@outlook.com

Abstrak

Game platformer merupakan salah satu jenis game dimana pemain harus melaui platform dan menghindari rintangan untuk menyelesaikan tantangan yang diberikan. Game platformer memiliki karasteristik yang serupa: platforms, obstacles, movement aids, collectible items, dan triggers. Dari kesamaan karakteristik tersebut, untuk mempermudah dalam pengembangan game platformer, digunakan game engine yang memanfaatkan data-driven programming. Data-driven programming memisahkan antara logic dan code pada game, sehingga perubahan game dapat mudah dilakukan hanya dengan mengubah data. Tujuan dari penelitian ini adalam mencari desain data-driven programming yang dapat memenuhi kebutuhan 2-Dimensi (2D) platformer untuk diimplementasikan pada game engine. Game engine selanjutnya diuji dan dianalisis untuk mengetahui seberapa besar pengaruh data-driven programming dalam proses pengembangan game platformer.

Kata kunci: Data-driven programming, game engine, game platformer.

Abstract

Platformer game is one of game's genre that make player have to pass platforms and have to avoid obstacles to finish the game. There are common characteristic in most platformer games such as platforms, obstacles, movement aids, collectible items, and triggers. By that same game characteristic, game engine that implement data-driven programming can be used to simplify game development process. Data-driven programming separates data and logic that has been used by the game which every change of its data will affect the game runtime. In this research, the objective is to find the suitable design on data-driven programming that can satisfy 2-Dimension (2D) game platformer's requirement then implement it as game engine. Furthermore, the game engine is also tested and analyzed to know how far it could assist in game development.

Keyword: Data-driven programming, game engine, platformer game.

1. Pendahuluan

Platformer merupakan genre game dimana karakter pemain akan melewati platform yang disediakan untuk mencapai objective dari game tersebut. Beberapa game yang memiliki genre platformer adalah Super Mario Bros, Contra, Rayman, Megaman, dan Prince of Persia. Judul tersebut merupakan beberapa dari sekian banyak judul game dengan genre platformer untuk target game console, dan masih banyak pula juduljudul platformer untuk PC game, web-based game, bahkan mobile game. Dari setiap judul game tersebut, terdapat kesamaan pada tiap komponen yang membangun game tersebut, antara lain: platforms,

obstacles, movement aids, collectible items, dan Triggers[6].

Pada game studio, proses produksi game biasanya ditangani oleh 3 peran: Game Designer, Artist, dan Engineer[8]. Game designer bertugas untuk membuat Game Design Document (GDD) yang yang berisi story line, level design, karakter yang ada pada game, objective, hingga experience yang akan didapat oleh pemain. Artist bertugas pada pembuatan game assets termasuk audio dan visual pada game. Sedangkan engineer bertanggung jawab untuk merealisasikan rancangan game dengan menggabungkan ide pada

GDD dan game asset hingga menjadi sebuah game yang interaktif.

Dengan adanya kesamaan karakteristik gameplay pada game platformer dan keterbatasan sumber daya yang dimiliki oleh suatu perusahaan, tim membutuhkan sebuah pendekatan khusus agar proses produksi, terutama pada game platformer, dapat berjalan sesuai target dengan memanfaatkan sumber daya yang ada pada proses produksi sebelumnya. Selain itu diperlukan pula kondisi dimana tiap anggota tim dapat fokus pada bidang keahlian masing-masing[5]. Game designer dapat melakukan balancing dan desain level – Artist dapat mengatur animasi atau mengubah asset – Engineer dapat fokus pada fitur game, sehingga proses produksi game dapat dikerjakan secara concurrent.

Penggunaan *game engine* merupakan salah satu solusi untuk memudahkan dalam pengembangan game pada. *Game engine* mampu menangani permasalahan untuk kondisi yang spesifik. Namun game engine yang ada memiliki keterbatasan[8]. *Game engine* hanya dapat digunakan oleh *engineer* karena untuk melakukan perubahan pada game diperlukan proses *scripting*.

Pendekatan lain adalah dengan menggunakan *Data driven programming* yang meisahkan antara *data* dan *program* yang dibutuhkan oleh game[13]. *Data driven programming* dapat mengakomodasi peran selain *engineer* tanpa harus berinteraksi langsung untuk mengubah game dengan hanya mengubah data. Berdasarkan kondisi tersebut akan diketahui:

- Bagaimana desain game engine yang sesuai untuk pengembangan game platformer?
- Bagaimana mengimplementasikan Data driven programming pada game engine?

2. Landasan Teori

2.1 Game Engine

Game dapat berbentuk sederhana hingga kompleks namun pada dasarnya setiap game memilki fungsionalitas yang sama. Setiap game mampu mendeteksi input dari user sehingga terjadi interaksi antara user dengan game, serta game mampu menampilkan gambar dan animasi pada layar. Selanjutnya proses ini berulang hingga game dinyatakan selesai dengan memberikan reward menang atau kalah pada user[2].

Game engine merupakan framework yang memfasilitasi beberapa jenis pekerjaan ang dibutuhkan dalam pembuatan game seperti menampilkan gambar pada layar, pembuatan menu dan text, penanganan prioritas untuk menampikan penggunaan ataupun physcic mensimulasikan gravitasi dan *collision* pada game dan sebagainya[2].

Ketika sebuah game memiliki hard coded logic, game rule, atau menggunakan library tertentu untuk merender game objek yang spesifik, sifat game akan sangat sulit atau bahkan ada kemungkinan tidak digunakan ulang pada saat pembuatan game baru. Dengan demikian diperlukan game engine untuk sebuah software yang bersifat extensible dan dapat digunakan sebagai template untuk membuat banyak game tanpa harus melakukan perubahan secara general.

2.2 Platformer Game

Platformer merupakan istilah yang digunakan untuk game dengan ciri third-person character-based action games dengan mekanik utama dari game tersebut adalah karakter utama yang melompat dari satu platform ke platform selanjutnya. Beberapa contoh game platformer 2D adalah Space Panic, Donkey Kong, Pitfall!, dan Super Mario Brothers, Sonic The hedgehog, Megaman[8]. Game Platformer merupakan video game yang memiliki karakteristik player yang dapat melompat-lompat dari satu platform ke platform lain melewati obstacle atau penghalang dan berakhir pada tujuan tertentu. User diberikan kemampuan untuk mengontrol lompatan dan akan ada kemungkinan untuk jatuh dari platform.

Dalam game *platformer*, terdapat 5 elemen umum[6] yang sering digunakan:

- Platform: merupakan terrain yang dilewati oleh karakter player baik itu berjalan, melompat, ataupun berlari diatasnya. Platform dapat berupa permukaan yang datar, jalur berbentuk melingkar atau bagian atas dari objek lain.
- Obstacle: merupakkan semua objek yang mampu memberikan efek buruk pada karakter player.
- Movement aids: merupakan semua objek yang membantu karakter player melalui medan level yang diberikan, bisa berupa trampoline, tangga, portal dsb.

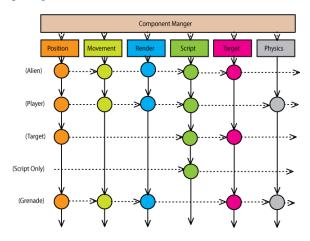
- Collectible item: merupakan semua objek yang memberikan reward jika berinteraksi dengan karakter player, bisa berupa koin, power-up, ataupun poin reward.
- Triggers: merupakan semua objek pada level yang dapat mengubah state dari level. Sebagai contoh adalah objek tombol yang jika ditekan akan menggeser salah satu bagian dari level.

2.3 Component Based Software Engineering

Game object merepresentasikan entitas yang muncul pada game world dan dapat berupa AI, mobil, pohon, bangunan, senjata dan lain-lain. Biasanya game object memiliki atribut transform yang menentukan posisi dan orientasi dari game objek, unique identifier yang dapat membedakan objek satu dengan objek lainya, dan method yang berfungsi untuk mengubah aktifitas objek[14].

Berbeda dari sistem hierarchical game object, pendekatan menggunakan component-based memisahkan fungsionalitas menjadi individual component yang bersifat independen. Dengan cara ini, tiap objek dapat memiliki fungsionalitas yang dibutuhkan dan fungsionalitas yang belum ada dapat dibuat dengan membuat komponen baru tanpa harus mengubah hirarki dari class tersebut.

Gambar 2.1 menunjukkan skema *component-based* pada pengembangan game dimana tiap entitas game terdiri dari beberapa komponen. Tiap kolom dari diagram memperlihatkan tabel kemiripan komponen, dan tiap baris memperlihatkan contoh objek yang ada pada game.



Gambar 2.1 Component Based Game Object

Component based game object system adalah penggunaan kumpulan komponen pada entitas dalam game, dimana tiap komponen menyediakan fungsionalitas dan data yang spesifik. Game object component system dapat menjadi solusi untuk menyelesaikan masalah pada banyaknya jumlah class yang harus dibuat dan kesulitan dalam menentukan peletakan fungsi baru dengan menggantikan fungsionalitas menjadi game component.

2.3.1 Pixelizer

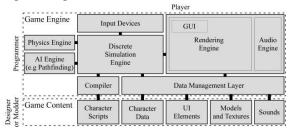
Pixelizer merupakan framework ditulis yang based menggunakan metode component untuk memudahkan pengembangan game. Dengan menggunakan konsep tersebut, pixelizer menggunakan entitas sebagai representasi game object dan penggunaan komponen pada entitas tersebut. komponen yag digunakan oleh entitas dapat berbentuk apapun mulai dari game logic hingga proses rendering. Pixelizer sendiri memiliki tujuan untuk membuat framework yag flexible, reusable, dan extendable. Proses rendering Pixelizer saat ini menggunakan proses blitting. Blitting merupakan cara cepat untuk menampilkan ratusan object tanpa harus penurunan performansi[12]. Dengan dokumentasi yang lengkap dan dukungan komunitas yang baik, Pixelizer dapat menjadi salah satu framework yang dapat digunakan dalam membuat game.

2.3 Data Driven Programming

Data driven programming memiliki pendekatan yang berbeda dibandingkan dengan prosedural programming ataupun object oriented programming. Procedural programming berfokus pada pemanggilan prosedur sebagai elemen utama, sedangkan object oriented programming berfokus pada objek. Kedua paradigma pemrograman tersebut berfokus pada pembuatan prosedur dan fungsi serta pengelompokan dan enkapsulasi kode. Data driven programming atau menggeser paradigma pemrogaraman dari sebuah objek menjadi data objek tersebut, mulai dari tipe data, bagaimana data tersebut disimpan dalam memori, serta bagaimana data tersebut akan dibaca dan diproses dalam game[11].

Pemanfaatan data driven programming memiliki sifat: mempermudah dalam penambahan ataupun pengurangan properti atau atribut dari sebuah objek, memudahkan dalam pembuatan entitas baru, behaviour dari sebuah entitas bersifat portable dan reusable, data dari sebuah entitas dapat diletakkan pada file yang berbeda, perubahan dapat dilakukan tanpa melakukan compile ulang karena program membaca data eksternal ataupun perubahan data dapat dilakukan pada saat aplikasi sedang berjalan[11]. Dengan menggunakan *Data driven programming* diharapkan akan memudahkan pengembangan aplikasi dengan melakukan perubahan pada program program seminimal mungkin[3].

Sedangkan Kostas[1] mengatakan karakteristik dari desain system game berbasis *data driven* adalah: Game entity merupakan kontainer bagi data ataupun behavior, pada dasarnya tugas dari program adalah melakukan proses rendering dan transformasi game object, Designer dapat menambahkan, mengurangi, dan mengubah game object tanpa harus menyentuh program, dan yang terakhir *statistic* permainan juga dapat disimpan kedalam data.



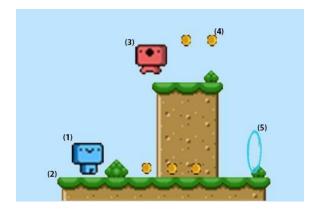
Gambar 2.2 Desain Data-driven game engine

Gambar 2.6 menunjukkan arsitektur data driven design pada game engine yang memisahkan antara logic game dengan game content[17]. Game content merupakan semua data dari game object yang akan digunakan oleh game, dapat berupa Chracter Script, Character Data, UI Elements, Models and Textures, serta Sound. Semua data tersebut dihubungkan dengan compiler dan Data Management Layer untuk selanjutnya dapat diproses oleh game engine hingga didapat game yang sesuai dengan dataset.

3. Desain dan Implementasi

3.1 Rancangan Game Platformer

Berdasarkan kesamaan karakteristik dari komponenkomponen yang ada pada game platformer, didapat desain game platformer yang dapat ditangani oleh game engine adalah sebagai berikut,



Gambar 3.1 Desain game 2D platformer

Keterangan:

- (1) Representasi dari karakter player dan dapat digerakan dengan menggunakan keyboard.
- (2) Representasidari terrain dengan memanfaatkan desain level berbasis tile.
- (3) Representasi dari musuh, yang berusaha untuk menghalangi karakter player dalam permainan.
- (4) Representasi dari koin atau objek yang dikumpulkan oleh player.
- (5) Representasi dari tujuan akhir suatu level.

3.1.1 Tilemap

Tilemap merupakan kumpulan gambar dengan cara memetakan gambar tileset pada koordinat X dan Y. Pada tilemap, terdapat berbagai macam informasi yang dapat kita simpan seperti tipe terrain, bagian yang dapat dilewati ataupun tidak dapat dilewati, bagian trigger, dsb. Informasi tersebut dapat disimpan kedalam berbagai format seperti CSV ataupun XML.



Gambar 3.2 Contoh Tileset dengan bentuk segi empat

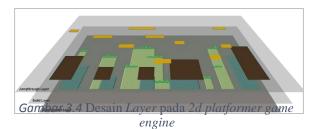
Sedangkan *tileset* sendiri merupakan beberapa gambar dengan ukuran seragam yang disatukan kedalam

sebuah file gambar. Bentuk *tileset* dapat berupa persegi, hexagonal, lingkaran tergantung dari desain level yang dibutuhkan.

<solid exportMode="Bitstring"> </solid>

Gambar 3.3 Pemetaan tileset menjadi tilemap menggunakan bitstring

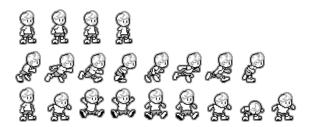
Pada studi kasus game 2d platformer, tilemap dimanfaatkan untuk mendesain level dengan tileset berbentuk persegi. Untuk memudahkan visualisasi, digunakan *OgmoEditor* sbagai tools mengonversi data visual kedalam format xml. Desain level terdiri dari 3 buah layer: background, dan solid. jump thorugh, Layer background menyimpan data visualisi dari tileset terhadap koordinat pada level, *layer jump through* menyimpan posisi dimana game object dapat melompati platform dari bawah, sedangkan layer solid menyimpan posisi dimana game object tidak dapat melalui area ini.



3.1.2 Spritesheet

Spritesheet merupakan media animasi yang disimpan kedalam sebuah gambar dua dimensi. Spritesheet dimanfaatkan untuk menggabungkan beberapa gambar animasi karakter sehingga dapat menghemat memori saat gambar tersebut dipanggil[10]. Gambar

pada spritesheet dapat dipanggil dengan mengakses posisi frame gambar.



Gambar 3.5 Contoh spritesheet

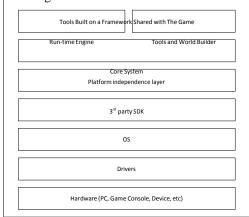
3.1.3 Background music dan Sound Effect

Salah satu yang aspek yang dapat membuat game lebih menarik adalah penggunaan sound: background music (BGM) dan sound effect (SFX). BGM pada game dimainkan secara berulang (looping), sedangkan SFX biasa dimainkan ketika terdapat event antara karakter player dengan environment pada game dan tidak berulang.

3.2 Arsitektur Data-driven Game Engine

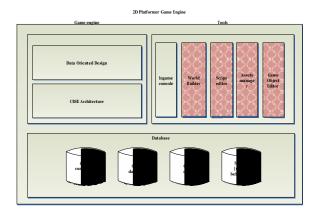
Game engine dibagi menjadi dua bagian, runtime dan non-runtime. Runtime merupakan kondisi dimana game berjalan, semua kebutuhan seperti asset dan game logic, dan interaksi antara user dengan game diproses dan ditampilkan ke monitor pada fase ini. Sedangkan non-runtime adalah kondisi dimana user dapat memaniulasi game ketika game tidak sedang berjalan.

Desain dari Runtime Game Engine Architecture adalah sebagai berikut:



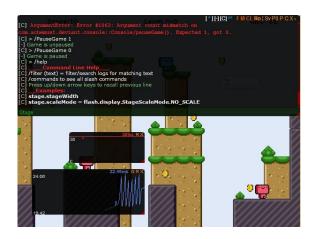
Gambar 3.6 Desain arsitektur untuk data-driven game engine

Desain runtime architecture yang digunakan untuk 2D platformer game engine sendiri memisahkan antara run-time engine dan tools. Karena menggunakan pendekatan data-driven, tools dan world builder digunakan untuk mengubah data ketika game tidak dijalankan. Text editor digunakan untuk mengubah data-data game berdasarkan format yang telah ditentukan. Sedangkan untuk memudahkan pada kondisi runtime, digunakan tools command-line console untuk dapat memanipulasi game world. Untuk tetap menjaga konfigurasi data-data yang digunakan dalam game platformer, manipulasi yang dilakukan pada command-line tools tidak akan merubah kondisi data awal.



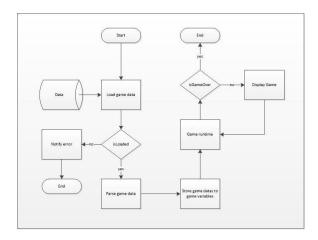
Gambar 3.7 Desain sistem pada data-driven game engine

Untuk game engine pada kasus 2d platformer, tools yang digunakan hanya in-game console. Sedangakan world builder, asset manager, game object editor ditangani dengan mengubah file xml menggunakan text editor.



Gambar 3.8 Tools pada runtime data-driven game engine

Dengan memanfaatkan data-driven programming, data yang dibutuhkan dalam game dipisah kedalam database berbentuk xml. Dengan begitu, user dapat mengubah data tanpa harus melakukan kompilasi. Untuk melihat perubahan game berdasarkan data, user hanya perlu membuka kembali executable file kemudian program game akan membaca data dan selanjutnya akan diproses untuk ditampilkan ke layar. Adapun alur dan proses dari game engine tersebut adalah sebagai berikut:



Gambar 3.9 Flow chart sistem data-driven game engine

Sistem akan membaca data yang akan digunakan dalam permainan, format file yang digunakan berupa file xmldan memiliki atribut-atribut mempengaruhi aktifitas pada entitas-entitas yang akan digunakan pada game tersebut. Data yang akan digunakan dapat berupa data transformasi, data animasi, data collision, serta data object behavior. Data yang dimuat dapat berupa data xml, data text, dan data image. Jika proses pemuatan data berhasil, data berupa behavior script dan data lainnya akan diterjemahkan kedalam bahasa sistem. Data-data yang telah melaui proses loading dan parsing, disimpan kedalam variable-variable pada game, selanjutnya data-data tersebut digunakan dalam runtime game dan dimunculkan ke layar komputer hingga permainan berhenti.

3.4 Desain Pengujian

Implementasi dari desain game engine dengan menggunakan data-driven programming selanjutnya diuji kepada 20 responden yang sebelumnya memiliki pengalaman dalam bidang pengembangan game. Pengujian dilakukan melalui dua bagian:

3.4.1 Opinion based

Opinion based bertujuan menguji desain dan implementasi data-driven game engine dengan cara memberikan kusioner kepada responden. Dari kuisioner akan didapat data mengenai tingkat kemudahan penggunaan data-driven game engine, kesesuaian hasil akhir, dan pengaruh data-driven game engine pada SDLC. Adapun skenario pengujian pada bagian opinion based adalah sebagai berikut:

- Kuisioner dibagi menjadi 3 bagian. Bagian pertama mengenai latar belakang responden pada bidang pengembangan game. Data ini nanti akan
 - digunakan untuk mengelompokkan responden untuk dapat dibandingkan tiap kelompoknya.
- 2. Responden selanjutnya akan melakukan pengujian pada point 3.4.2.
- 3. Selanjutnya reponden akan mengisi kuisioner bagian kedua mengenai experience setelah memanfaatkan *data-driven game engine* pada pengembangan game *platformer*. Bagian ini bertujuan untuk mendapatkan nilai *usability* dari game engine.
- 4. Pada bagian ketiga kuisisioner, responden akan memberikan pendapat mengenai implementasi data-driven programming pada game engine. Dari bagian ini akan didapat pengaruh data-driven programming pada fase SDLC dan apakah data-driven programming dapat membantu proses pengembangan game.

3.4.2 Factual based

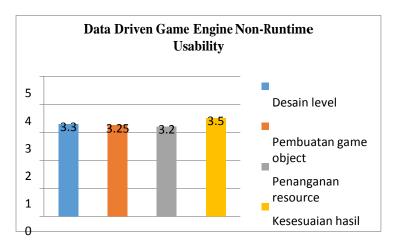
Factual based bertujuan untuk mencatat sejauh mana responden dapat memanfaatkan data-driven game engine untuk mengembangan game platformer. Responden akan diberikan checklist komponen game platformer dan manual penggunaan game engine. Dalam waktu 60 menit, semua komponen yang dapat terpenuhi oleh responden dicatat untuk kemudian diolah. Hasil data tersebut akan menentukan seberapa besar pengaruh data-driven game engine dalam pengembangan game. Adapun scenario pengujian yang

akan dilakukan adalah sebagai berikut:

1. Dalam waktu 30 menit, responden akan dijelaskan mengenai game *platformer*, komponen-komponen yang ada pada game *platformer*, dan penggunaan *game engine* dalam mengembangkan game *2D platformer*.

- 2. Dalam waktu 60 menit, responden akan menggunakan desain dari *game engine* untuk membuat satu level game *platformer*.
- Selama 60 menit tersebut akan dicatat berapa banyak kompoen penyusun game platformer yang dapat dibuat oleh responden.

4. Pengujian dan Analisis



Nilai *usability* dalam pemanfaatan data driven game engine dinilai berdasarkan skala satu sampai lima. Pada bagian bagian *non-runtime* dari game engine didapat total rata-rata dari tiap responen didapat skor sebesar 3.3125. Skor tersebut menginidikasi bahwa implementasi dari *data-diven game engine* berada diantara tingkat cukup mudah dan mudah. Data driven game engine memanfaatkan *XML* sebagai media penyimpanan data, sehingga untuk merubah data diperlukan pengetahuan dasar mengenai *XML*. Namun beberapa responden mengatakan kesulitan dalam memanipulasi data melalui file *XML*. Untuk mempermudah responden menyarankan adanya *user interface* atau *visual editor* untuk membantu interaksi bagi user yang memiliki pengetahuan rendah

Data Driven Game Engine Runtime Usability

5
4 3.45 3.65 3.75 3.75

Game log
Slash command
Memory monitor
Framerate monitor

Error Handling

0

Dari data survey menunjukkan tingkat kemudahan dalam pemakaian fitur game engine pada saat *runtime* didapat skor rata-rata sebesar 3.54. Nilai tersebut menunjukkan bahwa game engine berada diantara tingkatan cukup mudah dan mudah. Dari data responden, desain dari data-driven game engine belum memiliki fitur debugging untuk mengetahui data atau bagian mana pada game object ketika hasilnya tidak sesuai dengan yang diharapkan.

Dari 20 responden, 35% pernah menggunakan game

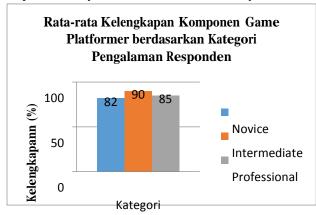
engine lain yang memanfaatkan data-driven programming sebelum pengujian dilakukan, dan 65 % sisanya belum pernah menggunakan data-driven game engine.

Game merupakan aplikasi yang tidak lepas dari proses SDLC. Dari data survey, didapat pemanfaatan data-driven game engine sangat berpengaruh pada fase *implementasi* dan *testing*.

Berdasarkan data survey, level editor pada data-

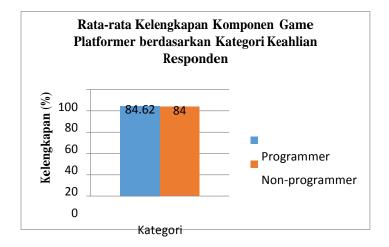
driven game engine merupakan fitur yang dirasa penting diikuti dengan game object editor, in-game console, dan scripting pada pengembangan game platformer.

Setelah dilakukan pengujian, ketertarikan responden untuk menggunakan data-driven game engine pada pengembangan game berikutnya didapat 90% responden menyatakan tertarik dan 10% sisanya tidak.

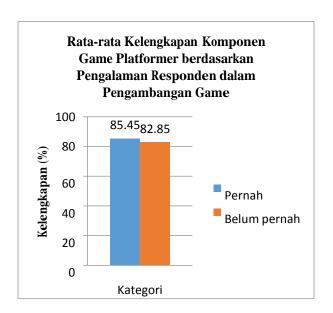


Berdasarkan pengalaman responden dalam bidang game development, data dibagi menjadi 3 kelompok: novice, intermediate, dan professional. Dari grafik diatas, didapat data terbaik dimiliki oleh responden dengan kategori Intermediate memiliki rata-rata sebesar 90% dan standar deviasi sebesar 11.54, diikuti dengan Professional sebesar 85% dengan standar deviasi 19.14, dan yang terakhir kategori Novice dengan rata-rata 82% dan standar deviasi sebesar

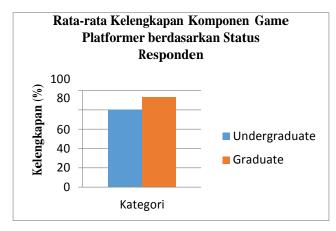
19.88. Berdasarkah data tersebut, semakin banyak pengalaman responden tidak menjamin bahwa pemanfaatan data-driven programming pada pengembangan game platformer menghasilkan nilai rata-rata yang semakin besar.



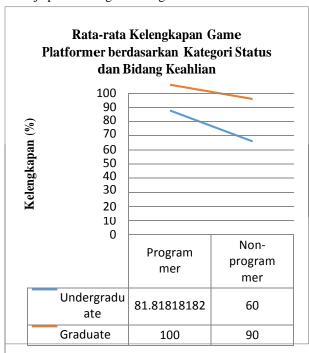
Berdasarkan kategori keahlian responden, didapat data rata-rata kelengkapan komponen platofmer sebesar 84.61% untuk programmer dengan standar deviasi 18.53, dan untuk non-programmer didapat rata-rata sebesar 84% dengan standar deviasi 16.7. Perbadaan rata-rata kelengkapan antara programmer dan non-programmer diketahui sebesar 0.61%. Dengan demikian pengembangan game dengan memanfaatkan data-driven programming tidak spesifik hanya dapat digunakan oleh programmer namun juga dapat digunakan oleh non-programmer.



Responden yang pernah membuat game platformer memiliki rata-rata kelengkapan sebesar 85.45% dengan standar deviasi 15.72, sedangkan responden yang belum pernah membaut game platformer sebelunya memiliki rata-rata sebesar 82.85% dengan standar deviasi 21.38. Rata-rata kelengkapan antara responden yang pernah dan belum pernah membuat game platformer memiliki perbedaan sebesar 2.6%.



Diketahui data rata-rata kelengkapan komponen game platformer yang telah dibuat oleh undergraduate sebesar 80% dan untuk kategori graduate sebesar 93.33%. Didapat perbadaan rata-rata sebesar 13%. Dari data ini menunjukkan bahwa pemanfaatan data-driven programming lebih mudah dimanfaatkan pada kelompok dengan status graduate atau yang telah bekerja pada bidang indsutri game.



Untuk kelompok undergraduate, antara programmer dan non-programmer didapat perbedaan rata-rata kelengkapan sebesar 21.81%. Pada kelompok graduate didapat perbedaan sebesar 10%. Sedangkan antara kelompok undergraduate dan graduate terdapat perbedaan sebesar 18.19% untuk kategori programmer dan 30% untuk kategori non-programmer. Dari hasil dapat kita lihat pemanfaatan data driven game engine lebih berpengaruh pada kelompok dengan bidang keahlian programmer dan telah bekerja pada industry game.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari desain dan

implementasi data-driven programming pada game engine adalah:

- 1. Arsitektur game engine menggunakan desain "Tools built on a framework shared with the game" dimana game engine merupakan bagian dari game yang diberikan kepada player. Dengan mengimplementasikan data-driven programming pada arsitektur tersebut, didapat sebuah game engine yang memisahkan data dan logic yang kemudian dapat digunakan untuk memudahkan dalam pengembangan game khususnya game 2d platformer.
- 2. Pada kondisi non-runtime nilai usability yang didapat sebesar 66.25%. Bagian non-runtime data-driven dapat diterima oleh responden dengan tingkat penggunaan diantara cukup mudah dan mudah untuk digunakan. Meskipun pembuatan game object dilakukan dengan hanya mengubah data pada file XML, responden memerlukan adaptasi untuk dapat menggunakannya.
- Pada kondisi runtime data-driven game engine, beberapa tools ditambahkan untuk memudahkan pengembangan game. Dari pemanfaatan tersebut, didapat nilai usability driven game engine sebesar 70.8%.
- 4. Dari pengujian yang dilakukan oleh 20 responden selama 60 menit didapat total kelengkapan komponen game platformer yang dapat dicapai memiliki rata-rata sebesar 83%. Hasil terbaik penggunaan data driven game engine oleh kelompok programmer dengan status graduate sebesar 100% dan kelompok non-programmer dengan status gradiate sebesar 90%.

5.2 Saran

- Saran-saran mengenai data-driven game engine untuk pengembangan selanjutnya adalah:
- Data driven programing memungkinkan developer membuat atau menambahkan fitur game dengan hanya mengubah data. Untuk dapat lebih memudahkan developer, ada baiknya digunakan GUI dimana data dapat data game object dapat otomatis diciptakan tanpa harus menulis secaara manual satu per satu data yang dibutuhkan.
- Pada penelitian selanjutnya dapat dilakukan penujian menggunakan metric yang sesuai untuk mengetahui tingkat reusability, maintainability, dsb.

Daftar Pustaka

- [1] Anagnostou, K. (2011, April 19). Data Driven Game Development. Retrieved November 22, 2013, from SlideShare: http://www.slideshare.net/KostasAnagnostou/da ta-driven-game-development
- [2] Bensmiley. (2012, April 5). What is Game Engine? Retrieved 11 21, 2013, from Deluge.co: http://www.deluge.co/?q=what-is-a-game-engine
- [3] Bilas, S. (2010, February 2). *A Data-Driven Game Object System*. Retrieved March 24, 2014, from New Fun Blog Scott Bilas: http://scottbilas.com/games/dungeon-siege/
- [4] Chady, M. (2009). Theory and Practice of Game Object Component Architecture. *Game Developer Converence Canada*. Canada.
- [5] Clutton, R. (2005). Data Driven Game Engines: Advanced Game Design Theory & Technology.
- [6] Gillian Smith, M. C. (2008). A Framework for Analysis of 2D Platformer Levels. *Proceeding* Sandbox '08 Proceedings of the 2008 ACM SIGGRAPH symposium on Video games, 75-80.
- [7] Gillian, S., Cha, M., & Whitehead, J. (n.d.). A Framework for Analysys of 2D Platformer Levels.
- [8] Gregory, J. (2009). *Game Engine Architecture*. Wellesley, Massachusetts: A K Peters, Ltd.
- [9] He Jifeng, X. L. (2005, August). Component-Based Software Engineering.
- [10] Löw, A. (n.d.). What is a Sprite Sheet? Retrieved December 30, 2013, from Code'n'Web:

- http://www.codeandweb.com/what-is-a-sprite-sheet/
- [11] Noel. (2009, December). Retrieved from Data Oriented Design (Or Why You Might be Shooting Yourself in the Foot With OOP): http://gamesfromwithin.com/data-oriented-design
- [12] Peitz, J. (2012, March). Retrieved from Pixelizer: http://johanpeitz.com/pixelizer/
- [13] Rabin, S. (2000). The Magic of Data Driven Design. *Game Programming Gems*.
- [14] Stoy, C. (2006). Game Object Component System. *Game Programming Gems* 6, 6, 393-403.
- [15] Sultanoðlu, Ü. K. (1998, October). Retrieved from Object Oriented Metrics: http://yunus.hun.edu.tr/~sencer/oom.html
- [16] West, M. (2007, January). Retrieved from Evolve Your Hierarchy: Refactoring Game Entities with Component: http://cowboyprogramming.com/2007/01/05/ev olve-your-heirachy/
- [17] White, W. (2011, February 02). Lectures 12 Data Driven Design. Retrieved March 24, 2014, from The Game Design Initiative at Cornell University:

 http://www.cis.cornell.edu/courses/cis3000/201
 1sp/top/lectures.php