IT Journal Research and Development (ITJRD)
Vol.4, No.1, Agustus 2019, E-ISSN : 2528-4053 | P-ISSN : 2528-4061
DOI : 10.25299/itjrd.2019.vol4(1).3517

Pembuatan Plugin Tile-Based Game Pada Unity3D

Salhazan Nasution[1], Arbi Haza Nasution[2], Arif Lukman Hakim[3]
Teknik Informatika, Universitas Riau[1,3]
Teknik Informatika, Universitas Islam Riau[2]
salhazan@lecturer.unri.ac.id[1], arbi@eng.uir.ac.id[2], arif.lukman@student.unri.ac.id[3]

Article Info

History:

Dikirim 28 Juli 2019
Direvisi 05 Agustus 2019
Diterima 08 Agustus 2019

Kata Kunci:

Unity
Plugin
Tile based game
Level editor

Abstrak

Saat ini video games sudah menjadi hal umum dalam kehidupan masyarakat dunia. Sejalan dengan itu, proses pengembangan sebuah game menjadi lebih baik dengan kemunculan *game engine*. Salah satu dari sekian banyak *game engine* yang paling sering digunakan adalah *Unity*. *Unity* memberikan berbagai macam fitur, salah satunya adalah kemampuan untuk menggunakan *plugin*. Hanya saja, *Unity* sendiri belum memiliki plugin untuk pengembangan game berbasis tile. Tanpa dukungan *plugin*, pengembangan *tile-based game* akan memakan waktu sangat lama, karena setiap tile harus diatur ulang masing-masing posisinya pada koordinat x, y, dan z dengan sangat presisi setiap kali tile baru dibuat. Solusi dari masalah tersebut adalah dengan membuat *GUI (Graphical User Interface)* pada *editor Unity*, dengan melakukan ekstensi kelas *Editor* milik *Unity*. Dengan melakukan ekstensi kelas tersebut, sebuah sistem menu baru dapat dibuat khusus untuk melakukan level editing pada *tile-based game*. Dengan menggunakan plugin ini, pengembangan *tile-based game* dapat menjadi lebih efektif dan efisien, baik dari segi sumber daya, waktu, dan kemudahan pengerjaan.

Koresponden:

Salhazan Nasution
Program Studi Teknik Informatika, Fakultas Teknik
Universitas Riau,
Jl. HR Subrantas km 12,5 Pekanbaru, Riau
Email: salhazan@lecturer.unri.ac.id

1. PENDAHULUAN

Video games sudah menjadi semakin terkenal setelah kemunculan pertamanya pada tahun 1950-an. *Video games* selalu menjadi komponen dari budaya populer, dan menjadi pusat dari industri hiburan *multi-miliar* dolar. Saat ini karena adanya *internet*, ketersediaan *software* pengembangan *game* telah meningkat seiring dengan kemudahan distribusi *game-ready device* dan bahkan distribusi *game* itu sendiri, mendorong kemajuan pada industri pengembangan *game* [1].

Software yang digunakan pada proses pengembangan *game*, atau biasa disebut *game engine* adalah sebuah alat yang dirancang untuk mengurangi biaya, kompleksitas, dan waktu yang dibutuhkan untuk memasarkan sebuah *game* pada sebuah proses pengembangan *game*. *Software* ini memberikan lapisan abstraksi diatas fungsionalitas utama dalam membuat sebuah *video game*. Lapisan abstraksi ini dikemas dengan beberapa peralatan yang dirancang untuk berfungsi sebagai komponen fungsional yang dapat dimodifikasi atau ditambahkan dengan penambahan komponen dari pihak ketiga [2].

Game engine sendiri juga sudah semakin terkenal. *Game engine* yang paling terkenal dengan jumlah pengembangnya adalah *Unity*, dengan jumlah pengembang lebih dari 5,5 juta orang terdaftar pada tahun 2016, kemudian diikuti dengan *Unreal Engine* yang telah mencapai 4 juta pengembang pada tahun 2017 [3].

Unity adalah sebuah game engine yang memberikan keuntungan yang besar dibandingkan game engine lainnya yang terdaftar di pasaran saat ini. *Unity* memberikan kapabilitas *drag-and-drop* pada alur kerja visualnya serta mendukung pemrograman pada bahasa C#, yang mana bahasa tersebut sangat terkenal. *Unity* sudah mendukung pengembangan grafis 3D dan 2D, juga menyediakan seperangkat peralatan untuk dua jenis grafis ini yang selalu berkembang, menjadi semakin mudah digunakan pada setiap pembaharuan. *Unity* juga dibuat khusus untuk mendukung pengembang menggunakan plugin dari *software* pihak ketiga. *Unity* juga menyediakan toko aset (*Asset Store*) sendiri yang menyediakan berbagai *plugin* yang diperlukan untuk pengembang *game*, dari pengembang, oleh pengembang dan untuk pengembang [2].

Plugin sendiri terdiri dari beberapa hal, diantaranya adalah aset seperti model 3D, sprite 2D, texture, material, efek suara, music, script, efek partikel, dan masih banyak lagi. Semuanya tersedia pada *Unity Asset Store*, mulai dari yang gratis hingga yang berbayar.

Disamping itu, sistem berbasis *tile* (*tile-based system*) untuk pembuatan *game* telah digunakan secara luas, sehingga menjadi salah satu dari standar pendekatan yang digunakan untuk membuat *game* pada sebagian banyak teknologi *game design*. Sistem ini tidak hanya untuk *game* 2D, melainkan game 3D juga dapat menggunakan sistem yang sama. *Tile-based system* ini terkenal karena memberikan solusi untuk masalah-masalah yang berbeda yang mana begitu kompleks untuk diselesaikan dengan cara yang lain [4].

Unity tidak memberikan dukungan secara langsung untuk membuat game menggunakan *tile based system*, sementara *plugin* yang terkait dengan fungsionalitas yang memadai dibatasi oleh pembayaran, dan *plugin* yang *gratis* kurang akan fungsionalitas yang baik, jadi pengembang memiliki kesulitan dalam hal akses kepada *tile-based system plugin* yang baik. Studi ini dilakukan untuk membuat *tile-based system* yang lebih kuat dalam hal fungsionalitas serta dapat diakses oleh pengguna *Unity* terutama dalam bidang edukasi dan pembelajaran.

Pada penelitian yang dilakukan oleh Ross Tredinnick, Brady Boettcher, Simon Smith, Sam Solovy, dan Kevin Ponto dengan judul *Uni-CAVE: A Unity3D Plugin for Non-head Mounted VR Display Systems*, *Unity3D* telah menjadi *game engine* 3D yang populer dan tersedia secara gratis untuk desain dan konstruksi lingkungan *virtual*. Namun, *Unity3D* belum bisa mengimplementasi tampilan *Virtual Reality* (VR) berbasis proyeksi imersif secara cuma-cuma. Plugin Uni-CAVE memungkinkan teknik stereo yang berbeda, seperti OpenGL quad buffered *stereo*, *passive stereo* dan *side-by-side stereo*. Tujuan dari karya ini adalah memberikan solusi gratis yang bisa diadaptasi dengan sistem proyeksi VR imersif apapun [5].

Pada penelitian yang dilakukan oleh Ivan Carmosino, Francesco Belotti, Riccardo Berta, Alessandro De Gloria, dan Nicola Secco dengan judul *A Game Engine Plug-in for Efficient Development of Investigation Mechanics in Serious Games*, *Serious Game* (SG) edukatif semakin sering digunakan dalam pendidikan, pelatihan, dan *domain* seperti olahraga dan perawatan kesehatan. Terlepas dari meningkatnya minat terhadap SG, penyebarannya terhalang oleh beberapa kendala, termasuk sulitnya mengkombinasikan efektifitas edukasional dengan hiburan, tingginya sumber daya yang dibutuhkan untuk proses pengembangan, dan melibatkan berbagai disiplin ilmu yang berbeda selain desain *game* dan pengembangan perangkat lunak. Pada penelitian ini, diusulkan *framework* pengembangan untuk mendukung investigasi dalam OWSG. *Framework* ini menyediakan *description template* yang memungkinkan pengembang untuk mendefinisikan *item* apa saja yang harus diinvestigasi dalam arah yang terstruktur secara langsung. Deskriptor diproses saat *runtime* oleh *plug-in game engine*, yang bertanggung jawab mengatur lingkungan *virtual* dan mengatur *gameplay* yang sesuai. Tujuannya adalah untuk membuat pengembangan OWSG menjadi lebih mudah juga untuk pakar *knowledge domain* yang belum memiliki pengalaman programming, dan memisahkan konten dari kode, yang mana memberikan lingkungan kerja modular untuk produksi, manajemen, dan perawatan dari *game* tersebut [6].

Pada penelitian yang dilakukan oleh Nicolo Balzarotti dan Gabriel Baud-Bovy dengan judul *HPGE: An Haptic Plugin for Game Engines*, *Serious Games* adalah *game* yang bertujuan untuk

mengintregasikan tujuan pendidikan dengan mekanik *game evidence-based* untuk mendukung pembelajaran dan generalisasi pembelajaran tersebut. Motivasi untuk menggunakan *haptic feedback* dalam *game* adalah manfaat pembelajaran yang diperoleh dari pengalaman sensorik dan motorik, serta eksplorasi aktif dari pemain. Selain itu, berbagai aplikasi VR yang memiliki perangkat *force-feedback* telah menunjukkan hasil yang menjanjikan. Tujuan dari penelitian ini adalah untuk mengimplementasikan *plug-in* yang memungkinkan integrasi perangkat *haptic* menggunakan *Unity3D*, karena *Unity3D* tidak mendukung integrasi perangkat *haptic* secara langsung [7].

Pada penelitian yang dilakukan oleh Carlo Luongo dan Paolo Leoncini dengan judul *An Unreal Engine* 4 *Plugin to Develop CVE Applications Leveraging Participant's Full Body Tracking Data*, ketersediaan game engine andalan yang gratis dengan kode open-source memberikan fitur yang powerful, serta lingkungan pengembangan yang produktif, bahkan untuk pengembangan aplikasi VR sekalipun. Laboratorium-laboratorium penelitian di bidang VR ini telah melakukan migrasi menggunakan *game engine*, yang merupakan lingkungan pengembangan baru yang jauh lebih produktif, kustomisabel dan ekstensibel, *interface* yang bersifat *easy-to-use*, kaya akan fitur, dan hampir tidak memerlukan biaya (terkecuali proyek yang memiliki keuntungan besar), dengan tambahan penyebaran *multi-platform* termasuk *mobile device*. Di sisi lain, CVE (*Collaborative Virtual Environment*) sendiri, dengan sebutan lainnya "VR *multiplayer* yang terhubung via jaringan" memungkinkan dua orang atau lebih untuk bergabung dalam lingkungan kerja *virtual* untuk melakukan kerja sama suatu pekerjaan yang tidak mungkin dilakukan oleh satu orang. Salah satu contohnya adalah simulasi dari pekerjaan nyata. Fitur khas yang paling menonjol pada CVE adalah kemampuan untuk membawa gerak tubuh manusia secara penuh dari dunia nyata ke dalam VR. Tujuan dari penelitian ini adalah untuk mengimplementasikan *plugin* yang dapat mengatur *full-body tracking* pada lingkungan jaringan *multiplayer* CVE berbasis *Unreal Engine* 4 [8].

Pada penelitian yang dilakukan oleh Marc O. Rudel, Johannes Ganser, Rene Weller, dan Gabriel Zachmann dengan judul *UnrealHaptics: A Plugin-System for High Fidelity Haptic Rendering in the Unreal Engine*, perangkat-perangkat VR sudah semakin terjangkau, seperti *Oculus Rift* dan HTC *Vive*. Game engine modern seperti Unity dan Unreal juga telah menyederhanakan pengembangan untuk game VR secara dramatis. Namun, perangkat VR ini hanya terbatas pada dua jenis indera manusia, yaitu penglihatan dan pendengaran. Hal ini mencegah sejumlah besar orang yang tidak dapat menikmati konten pada VR yaitu orang-orang tunanetra. Untuk mengatasi hal ini, terdapat perangkat berupa *haptic device*, dimana *haptic* ini mensimulasikan sentuhan. Tujuan dari penelitian ini adalah membuat *plug-in* UnrealEngine yang mendukung penggunaan perangkat *haptic* bernama *UnrealHaptics* [9].

2. STUDI LITERATUR

Tile sudah sangat terkenal dalam konteks pengembangan *game* 2D, sebagai elemen *terrain* yang *reusable*, yang memungkinkan konstruksi *terrain* yang lebih besar, yang mana karena ukuran dan kendala manajemen, tidak bisa ditangani melalui *bitmap* secara normal. Teknik ini sangat populer di masa lalu, namun teknik ini kembali diminati dalam konteks perangkat *hand-held*. Selain itu, ada tren yang sedang meningkat kearah kebangkitan dari *game* 2D, dengan penekanan pada *terrain* (lahan) yang lebih terpopulasi, aksi yang lebih cerdas, efek yang lebih kuat, serta desain karakter menggunakan instrument pemodelan 3D dengan penekanan pada pembuatan 2D. *Tile* adalah *bitmap* kecil, biasanya berukuran 16x16 atau 32x32 *pixel*, yang digunakan sebagai elemen penyusun *game terrain*. Teknik ini masih digunakan sampai sekarang, dan teknik ini menawarkan banyak keutungan untuk pengembang *game*. Selain memperkecil ukuran *game*, pemrosesan *terrain* juga lebih cepat, karena hanya sejumlah kecil informasi yang perlu dijabarkan. Keuntungan lain dari sistem *tile* ini adalah jika pengembang ingin membangun *terrain* yang lebih besar lagi, pengembang hanya perlu menambahkan *tile*, tanpa merancang seluruh *bitmap terrain* dari awal. *Game* yang menggunakan *tile* sebagai dasar komponen grafiknya disebut *tile-based game* [10].

Gambar 1. *Tile based terrain* pada 2D *game* (kiri) dan 3D *game* (kanan).

Sebuah *game engine* mewakili seluruh dasar dari sebuah *game*, menyediakan fungsionalitas untuk melakukan *rendering* grafik yang dioptimalkan dan efisien, file akses sistem, input pemain melalui perangkat seperti *keyboard* dan *mouse*, pemutar suara, konektivitas jaringan, serta penyimpanan dan pemuatan status permainan [11].

Game engine adalah kerangka kerja untuk pengembangan *game* yang memiliki beberapa bagian inti, yaitu *audio engine*, *rendering engine*, dan *physics engine*. *Audio engine* memiliki peran penting pada sebuah *game*. Jika karakter pemain sedang bertarung dengan musuh, dan mekanisme pertarungan tersebut menggunakan pedang, ketika pemain melakukan kontak dengan pedang musuh perlu menghasilkan efek suara. Pekerjaan itu dilakukan oleh *audio engine*. Untuk membuat suasana dari game lebih menarik, *audio engine* juga dapat memutar musik latar dan efek suara. *Rendering engine* membantu menentukan apa yang ditampilkan sebagai *output* kepada pengguna. *Output* tersebut merupakan suguhan *visual* untuk pemain saat sedang memainkan *game*. *Rendering* membantu menghidupkan konten grafis game dengan cara yang diinginkan oleh pengembang. *Physics engine* membantu mensimulasikan fisika pada game [12].

Unity (umumnya dikenal dengan *Unity3D*) adalah *game engine* dan *Integrated Development Environment* (IDE) untuk membuat media interaktif, biasanya *video game*. *Chief Executive Officer* (CEO) *Unity*, David Helgason mengatakan bahwa "*Unity* adalah toolset yang digunakan untuk membuat *game*, dan *Unity* merupakan teknologi yang mengeksekusi grafis, *audio*, fisika, interaksi, dan *networking*." *Unity* terkenal dengan kemampuan *prototyping* yang cepat dan target *publishing platform* yang berjumlah besar [13].

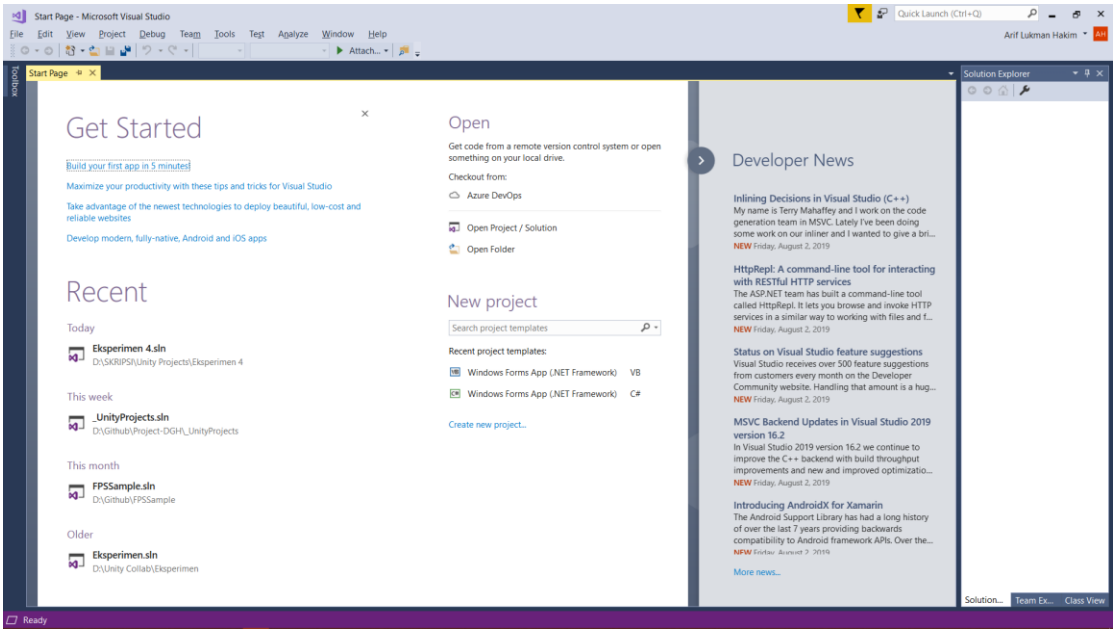
Gambar 2. *Interface Unity game engine*.

Plugin adalah unit ekstensi biner untuk aplikasi yang mana arsitekturnya memungkinkan fungsionalitas untuk diperkenalkan kepada *end-users* setelah instalasi aplikasi. *Plugin* adalah entitas perangkat lunak yang berhubungan erat dengan komponen. *Component-based development* (pengembangan berbasis komponen) biasanya tidak mempertimbangkan bahwa komponen-

komponen tersebut dapat ditambahkan ke dalam aplikasi setelah proses instalasi aplikasi. Komponen biasanya digunakan untuk memfasilitasi pembangunan aplikasi itu sendiri [14].

Plugin adalah bagian dari kode yang memodifikasi perilaku *runtime*. Bahasa dapat dianggap sebagai sebuah framework yang menyediakan seperangkat bagian ekstensi yang mana plugin dapat mengimplementasikan penambahan fungsionalitas terhadap bahasa tersebut. Menurut aspek ekstensi bahasa pemrograman, ada tiga kategori dari bagian ekstensi: (1) berhubungan dengan algoritma, (2) berhubungan dengan pembuatan aturan baru dan manajemen dari rule execution cycle, dan (3) terkait dengan sintaks bahasa [15].

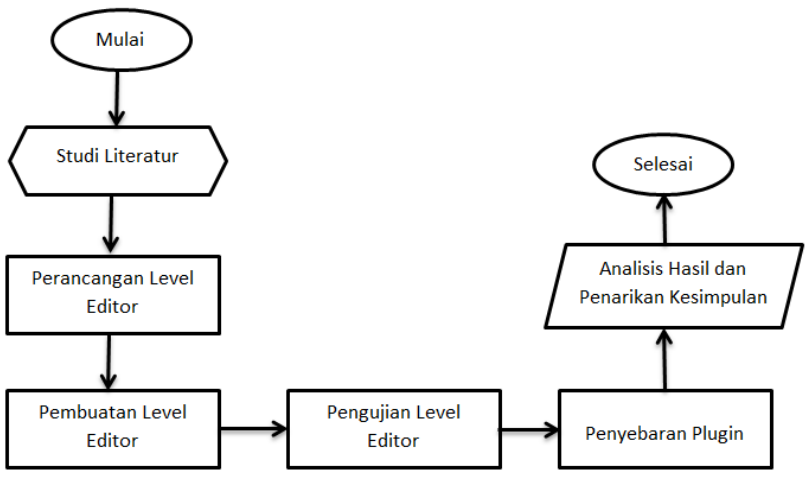
Microsoft Visual Studio 2017 adalah sebuah *integrated development environment* (IDE) dari *Microsoft*. *Visual Studio* digunakan untuk mengembangkan *program* komputer seperti *website*, *web app*, *web service*, dan *mobile app*. Versi terakhir dari *Visual Studio* 2017 ini memiliki fitur penuh untuk pengembangan aplikasi *Android*, *iOS*, *Windows*, *web*, dan *cloud* [16].



Gambar 3. *Interface Microsoft Visual Studio.*

3. METODE PENELITIAN

Gambar 4 menggambarkan metodologi yang digunakan pada penelitian ini.



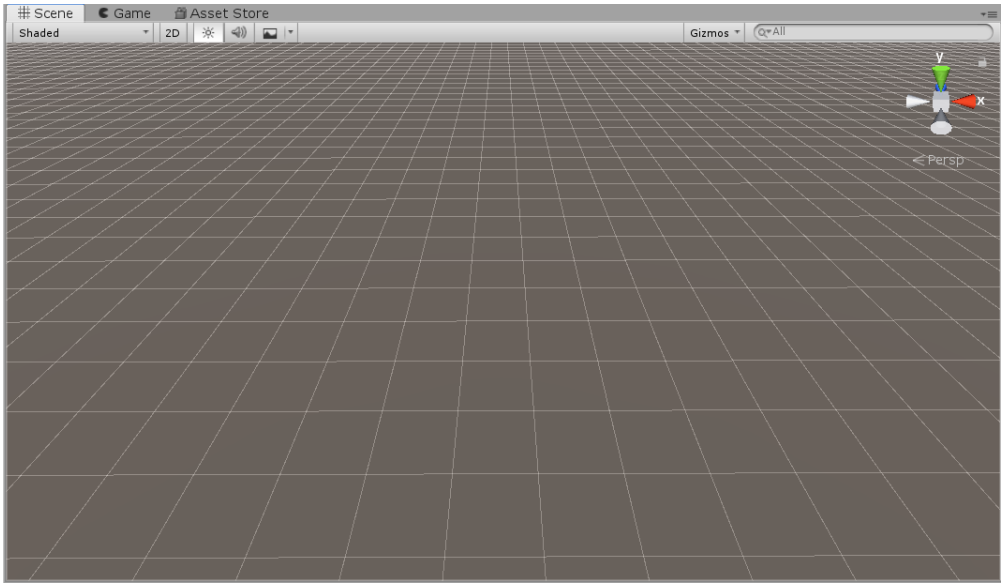
Gambar 4. Metodologi Penelitian.

3.1 Studi Literatur

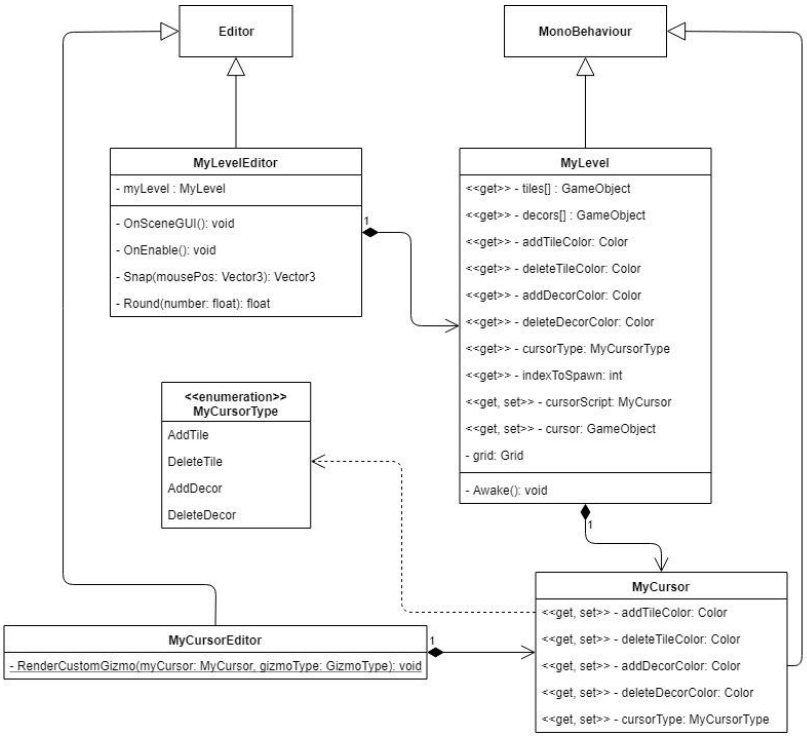
Studi literatur dilakukan untuk mengetahui dasar teori yang dibutuhkan untuk melakukan penelitian ini. Literatur yang dikaji pada penelitian ini berhubungan dengan *tile-based game*, *game engine*, *Unity3D*, dan *plug-in*.

3.2 Perancangan Level Editor

Setelah alat dan bahan dipersiapkan, langkah selanjutnya yang akan dilakukan adalah merancang *level editor*. *Level editor* akan diimplementasikan menggunakan sistem *grid*, yang mana *grid* ini berupa sistem koordinat, dimana *grid* ini yang akan menjadi patokan posisi masing-masing *tile* nantinya.



Gambar 5. Sistem *Grid* Pada *Unity*.



Gambar 6. Diagram Kelas *Level Editor*.

Bagian utama dari sistem ini adalah script *MyLevelEditor*. Dalam *script MyLevelEditor* terdapat *method OnSceneGUI* yang berguna untuk menentukan posisi kursor 3D pada *grid*, dan mengatur apa saja yang akan terjadi ketika mouse diklik, digerakkan, klik dan tarik, serta ketika mouse meninggalkan *scene view*. Dalam *script* tersebut juga terdapat *method Snap* yang berfungsi untuk menempelkan kursor 3D tepat pada posisi kotak pada *grid*. *Method Round* berfungsi membantu pembulatan nilai pada *method Snap* untuk memperoleh posisi kotak yang tepat.

Kemudian terdapat *script MyLevel* yang berfungsi menampung seluruh *property* yang nantinya akan dapat diubah dan dikonfigurasi sesuai dengan keinginan pengembang *game*. *Property-property* ini nantinya akan digunakan pada *script MyLevelEditor*. Di dalam *script MyLevel* hanya terdapat satu *method* saja yaitu *Awake*. *Method Awake* biasanya dieksekusi pada saat mode *play* dilakukan di dalam *game view*. Namun untuk kasus *level editor* ini, *Awake* akan dieksekusi dalam *mode edit*, karena ada beberapa variabel yang harus diatur oleh *script* saat objek *level editor* ini dimasukkan ke dalam *hierarchy* nantinya.

Script MyLevel membutuhkan *script* lain yang bernama *MyCursor*. *Game object* yang memiliki *script MyCursor* inilah yang nantinya akan berfungsi sebagai kursor 3D pada *scene view*. *Script MyCursor* juga berfungsi menampung konfigurasi warna kursor 3D yang dilakukan pada *script MyLevel*.

Setelah itu, terdapat *script MyCursorEditor* yang berfungsi mengubah warna kursor 3D sesuai konfigurasi yang telah disimpan pada *script* MyCursor. Warna kursor 3D juga akan berubah sesuai dengan jenis operasi yang terdapat pada enumerasi *MyCursorType*. Konfigurasi warna kursor 3D berdasarkan jenis operasi ini juga dapat dilakukan pada *script MyLevel*, juga akan disimpan pada *script MyCursor*.

3.3 Pembuatan Level Editor

Rancangan *level editor* yang sudah dibuat akan diimplementasikan pada *Unity*. *Level editor* dibuat dengan cara melakukan ekstensi script *Unity editor*, kemudian menambahkan elemen GUI pada *scene view Unity*, sehingga pengembang dapat meletakkan, memindahkan, serta menghapus objek yang dibutuhkan di *level* tersebut.

3.4 Pengujian Level Editor

Level editor yang sudah diimplementasikan tadi akan diuji, apakah masing-masing fungsi berjalan dengan baik dan sesuai dengan apa yang diinginkan. Jika masih ada kekurangan dan *bug*, maka dapat dilakukan perbaikan sehingga fungsi-fungsi tersebut dapat berjalan sebagaimana mestinya.

3.5 Penyebaran Plugin

Setelah pengujian dan perbaikan dilakukan, *plug-in* akan disebarkan kepada pengembang *game*. Pengembang dapat mencoba seluruh fitur yang telah diimplementasikan dalam *plug-in* ini menggunakan aset yang telah mereka buat. *Plug-in* ini nantinya akan disertakan dengan sebuah panduan dan sebuah kuesioner. Panduan akan berguna sebagai informasi kepada pengembang game bagaimana cara menggunakan *plug-in* ini, sementara kuesioner digunakan untuk memperoleh *feedback* dari pengembang *game* tentang bagaimana kinerja dari *plug-in* ini.

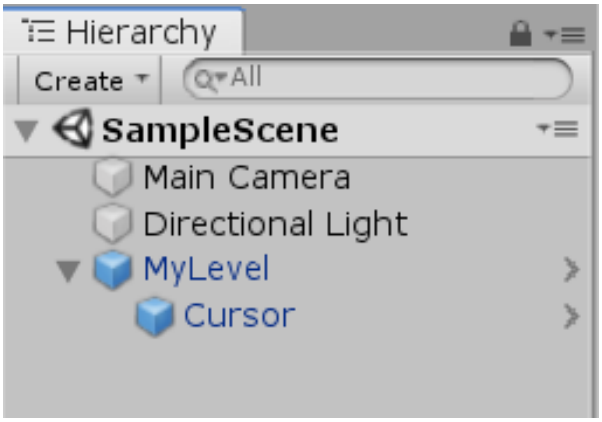
3.6 Analisis Hasil dan Penarikan Kesimpulan

Hasil dari penelitian ini adalah kuesioner yang diperoleh dari para pengembang *game* yang telah mengembangkan *plug-in* ini. Dari seluruh kuesioner yang didapatkan, dapat dilakukan analisis dan penarikan kesimpulan, apakah *plug*-in ini efektif dalam mempercepat kinerja pembuatan *tile-based* game.

4. HASIL DAN PEMBAHASAN

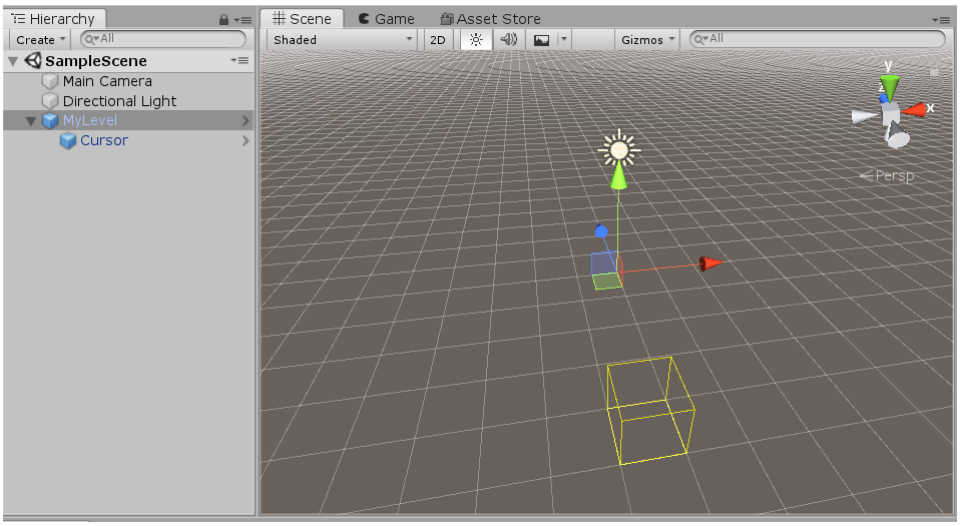
4.1 *Graphical User Interface* (GUI)

Level editor yang diimplementasikan menggunakan sebuah *game object* pada Unity. *Game object* ini diletakkan pada *hierarchy*, dan akan muncul di dalam *scene view*. Untuk melakukan pembuatan *level*, pengguna harus mengklik *game object level editor* pada *hierarchy* terlebih dahulu. Kemudian *level editor* dapat digunakan di dalam *scene view*.



Gambar 7. Objek *level editor* bernama "*MyLevel*" pada *hierarchy*.

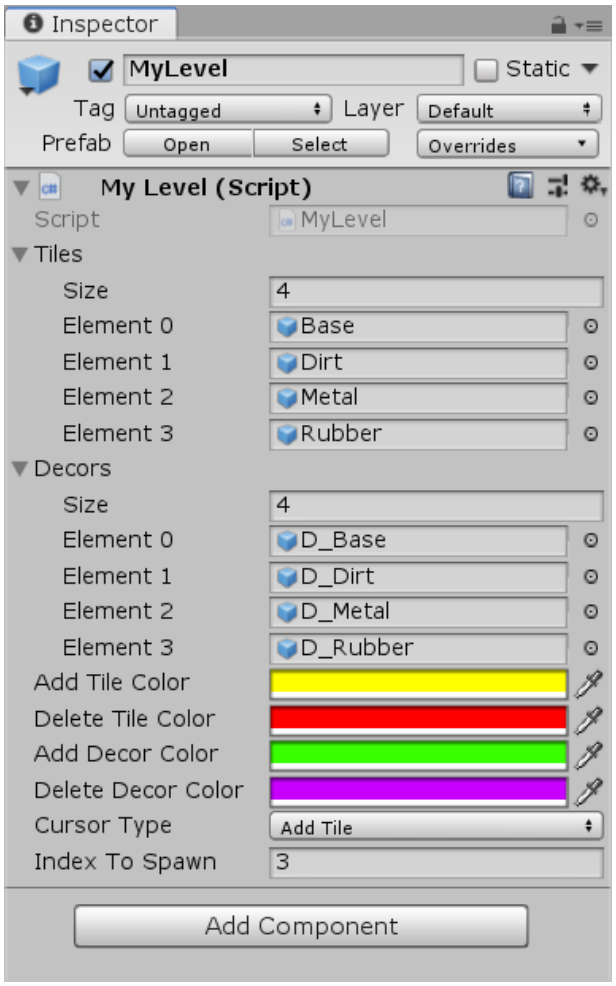
Game object ini diberi nama *MyLevel*. *MyLevel* memiliki *child game object* yang bernama *Cursor*. *Cursor* berfungsi sebagai kursor 3D yang akan muncul di dalam *scene view* untuk membantu proses pembuatan *level*. Selain kursor 3D, akan tampil juga *grid* yang akan membantu pengembang *game* untuk melihat posisi mana saja yang dapat diletakkan *tile* ke dalam *scene view*. Posisi *tile* tidak akan melenceng dari posisi kotak yang terdapat pada *grid.*



Gambar 8. Tampilan *level* editor pada *scene view* saat *game object MyLevel* dipilih.

Kursor 3D dapat digerakkan dengan cara menggerakkan *mouse* di dalam *scene view*. Kursor 3D inilah yang akan bekerja untuk memilih *tile* mana yang ingin ditambah dan dihapus pada *grid*.

Kemudian, saat *MyLevel* dipilih di dalam *hierarchy*, akan muncul sebuah komponen dari *game object MyLevel* pada *inspector*, yaitu *script* dari *object MyLevel*. Di dalam komponen ini, terdapat *property-property* yang dapat dikonfigurasikan oleh pengembang *game*.



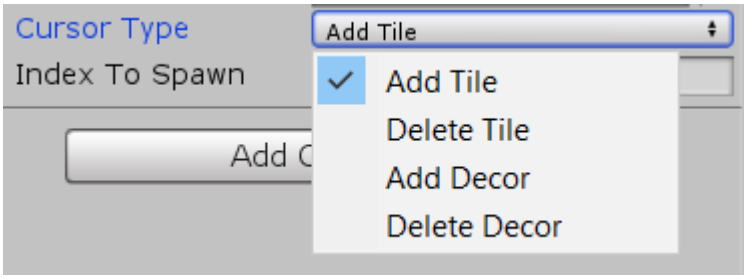
Gambar 9. *Game Object MyLevel* Pada *Inspector*

Tabel 1. *Property* Dari Komponen *Script MyLevel.*

No.	***Property***	**Fungsi**
1.	*Tiles*	Berisi jumlah dan jenis objek *tile* apa saja yang dapat ditambahkan ke dalam *scene view*. Jumlah objek *tile* dan jenis objek *tile* dapat diubah sesuai kehendak pengembang *game*.
2.	*Decors*	Berisi jumlah dan jenis objek *decor* apa saja yang dapat ditambahkan ke dalam *scene view*. Objek *decor* berfungsi sebagai dekorasi yang nantinya akan diletakkan diatas *tile*. Jumlah objek *decor* dan jenis objek *decor* dapat diubah sesuai kehendak pengembang *game*.
3.	*Add Tile Color*	Berisi warna dari kursor 3D untuk operasi menambah *tile*. Warna dapat diubah oleh pengembang *game* sesuai keinginan dan warna dari

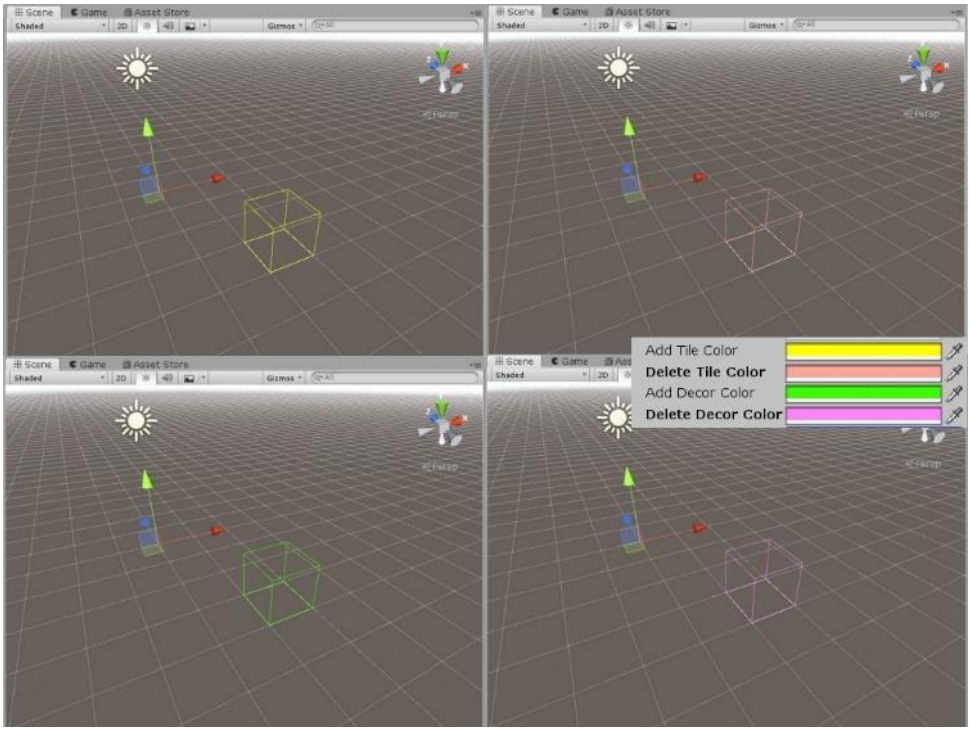
		kursor 3D akan menyesuaikan dengan *property* ini.
4.	*Delete Tile Color*	Berisi warna dari kursor 3D untuk operasi menghapus *tile*. Warna dapat diubah oleh pengembang *game* sesuai keinginan dan warna dari kursor 3D akan menyesuaikan dengan *property* ini.
5.	*Add Decor Color*	Berisi warna dari kursor 3D untuk operasi menambah *decor*. Warna dapat diubah oleh pengembang *game* sesuai keinginan dan warna dari kursor 3D akan menyesuaikan dengan *property* ini.
6.	*Delete Decor Color*	Berisi warna dari kursor 3D untuk operasi menghapus *decor*. Warna dapat diubah oleh pengembang *game* sesuai keinginan dan warna dari kursor 3D akan menyesuaikan dengan *property* ini.
7.	*Cursor Type*	Berisi operasi apa saja yang dapat dilakukan oleh pengembang game di dalam *level editor*. Warna kursor 3D akan berubah sesuai dengan jenis operasi yang dipilih dan sesuai dengan konfigurasi warna pada empat *property* diatas. Jenis operasi yang tersedia adalah menambah dan menghapus *tile*, serta menambah dan menghapus *decor*.
8.	*Index To Spawn*	Memilih *index* dari objek *tile* atau *decor* yang akan diletakkan dari *scene view*. Jenis objek yang akan ditambahkan ke *scene view* akan berubah sesuai dengan jenis operasi .Jika memilih operasi *decor* maka yang akan diletakkan adalah *decor*, dan sebaliknya.

Seluruh *property* diatas dapat dimodifikasi sesuai dengan keinginan pengembang *game*. Untuk menambah atau menghapus objek, baik itu *tile* maupun *decor*, pengembang game dapat memilih tipe operasi pada *property CursorType*.



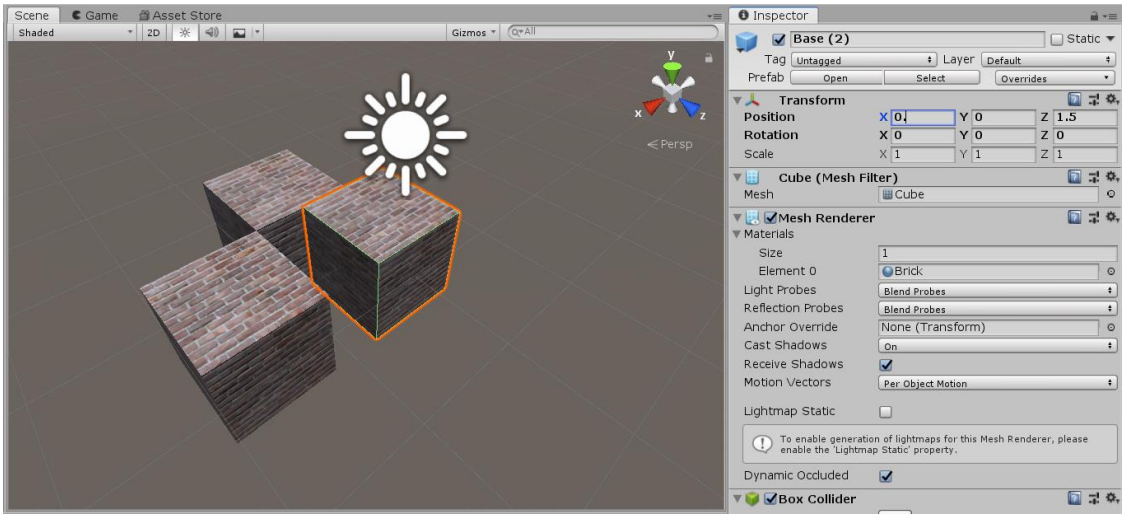
Gambar 10. Tampilan *property CursorType* setelah diklik pada *inspector*.

Kemudian warna kursor 3D pada *scene view* akan berubah sesuai dengan konfigurasi warna masing-masing jenis operasi pada *property game object* MyLevel.



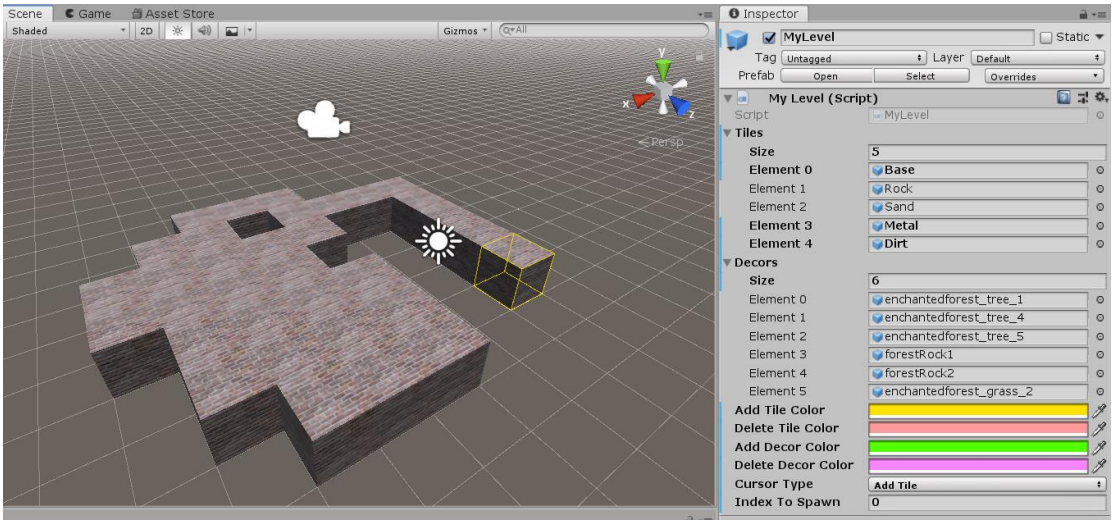
Gambar 11. Warna kursor 3D sesuai jenis operasi, A) *Add* Tile, B) *Delete Tile*, C) *Add Decor*, dan D) *Delete Decor*.

Setelah memilih jenis operasi, pengembang dapat menambah/menghapus objek, baik itu *tile* maupun *decor*, dengan cara mengklik kursor 3D pada posisi yang diinginkan di dalam *scene view*. Tanpa menggunakan *plugin* ini, pengembang *game* akan membutuhkan waktu yang lama untuk membuat sebuah *level* pada sistem *tile based game*, karena untuk membuat sistem *tile based game*, pengembang harus mengatur koordinat masing-masing *tile* setiap kali *tile* baru ditambahkan ke dalam *level*. Koordinat juga harus diatur dengan tepat agar masing-masing *tile* tidak bertubrukan dan tidak bergeser sedikit ke salah satu arah, sehingga posisi *tile* menjadi rapi sehingga tidak merusak estetika dunia *game* yang akan dibuat. Berikut adalah contoh implementasi sistem *tile based game* tanpa menggunakan *plugin* seperti pada gambar 12.



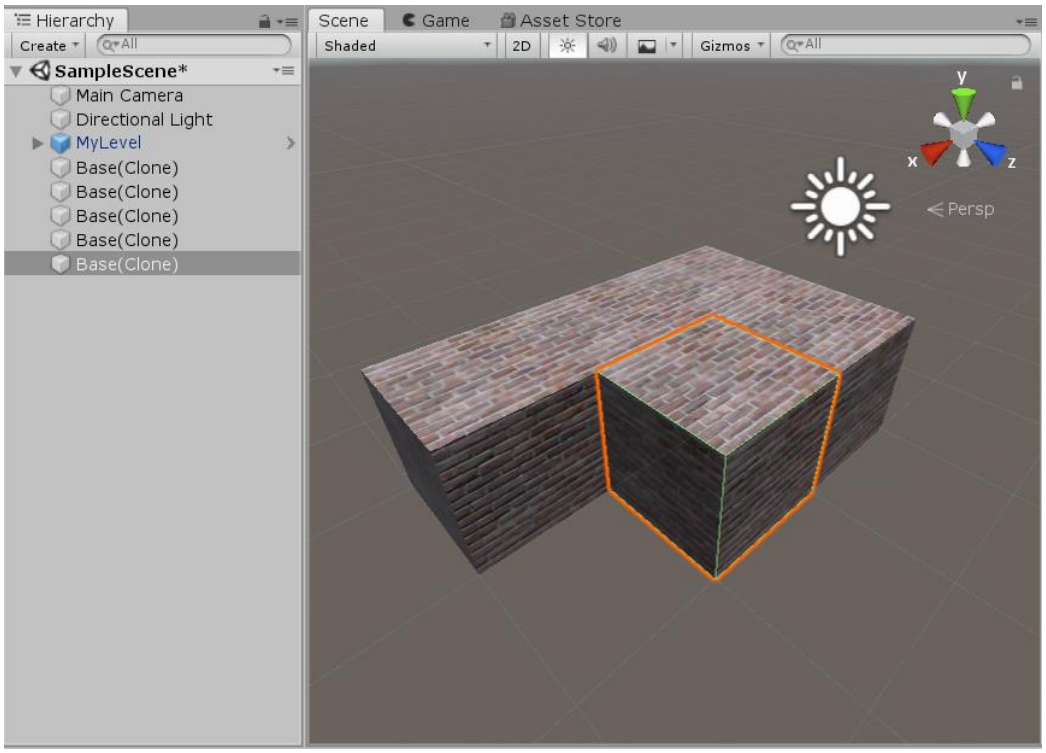
Gambar 12. Pengaturan Koordinat Pada Setiap *Tile* Yang Baru Ditambahkan Ke Dalam *Scene View* Tanpa Menggunakan *Plugin*.

Dengan menggunakan *plugin*, pengembang tidak perlu menghabiskan waktu yang lama untuk membuat sebuah *level*, karena *plugin tile based game* ini memberikan fitur *point and click* untuk melakukan operasi *tile* pada *scene view* seperti pada gambar 13.



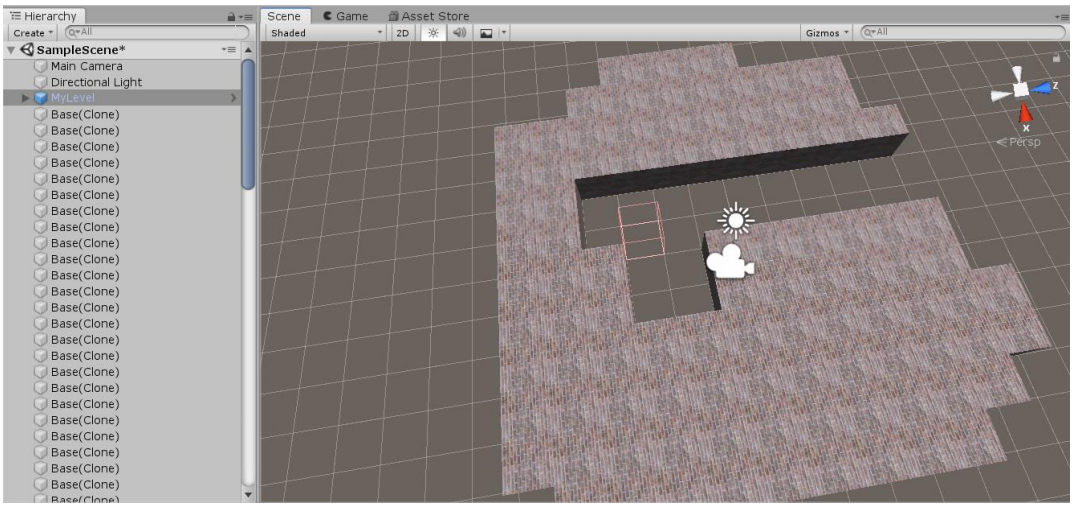
Gambar 13. Fitur *Point And Click* Pada Penambahan Setiap *Tile Baru* Ke Dalam *Scene View*.

Proses yang sama juga berlaku untuk menghapus *tile* yang telah diletakkan. Tanpa penggunaan *plugin*, penghapusan *tile* dilakukan dengan cara menghapus satu-satu objek pada *hierarchy* dengan cara memilihnya dari *scene* view, kemudian menekan tombol *Delete* seperti terlihat pada gambar 14.



Gambar 14. Menghapus *Tile* Pada *Scene View* Tanpa Menggunakan *Plugin*.

Sementara jika menggunakan *plugin*, pengembang *game* hanya perlu memilih jenis operasi menghapus, kemudian klik atau *drag* pada *tile* yang akan dihapus seperti terlihat pada gambar 15.



Gambar 15. Menghapus *Tile* Pada *Scene View* Dengan Menggunakan *Plugin*.

4.2 Hasil Uji Kemudahan *Plugin*

Kemudahan penggunaan plug-in ini telah diuji oleh responden berjumlah lima orang, dan data pengujian kemudahan penggunaan plug-in ini didapat dari kuesioner berskala likert yang diisi oleh masing-masing responden. Kuesioner yang memiliki pertanyaan yang berskala likert dijawab berdasarkan pemilihan dari beberapa jawaban yang telah didefinisikan, dimana jawaban tersebut mewakili penilaian dan pendapat responden tentang kriteria tertentu seperti kepuasan, kebaikan, persetujuan, dan lain sebagainya [17].

Tingkat kemudahan penggunaan *plug-in* dari jawaban yang diberikan oleh responden dihitung dengan rumus:

$$\text{Tingkat Kemudahan (\%)} = \frac{\text{Skor Rata-Rata}}{\text{Skor Ideal}} \text{ x } 100\% \quad (1)$$

Skor rata-rata dihitung dengan rumus:

$$\text{Skor Rata-Rata} = \frac{\text{Total Skor}}{\text{Jumlah Pertanyaan}} \quad (2)$$

Berikut adalah hasil kuesioner uji kemudahan *plug-in* sesuai dengan perhitungan skala likert pada tabel 2.

Tabel 2. Hasil Kuesioner Uji Kemudahan *Plug-in.*

No.	Pertanyaan	Total Jawaban					Total Skor	%
		STS	TS	N	S	SS		
1.	Pengendalian kursor 3D pada *scene view* sangat mudah.			1	3	1	20	80
2.	Pemilihan jenis operasi pada objek *tile* dan dekorasi sangat mudah.			2	2	1	19	76
3.	Menambah objek *tile* dan dekorasi pada *scene view* sangat mudah.			1	1	3	22	88
4.	Menghapus objek tile dan dekorasi pada scene view sangat mudah.			1	1	3	22	88
5.	Pemilihan warna untuk setiap jenis kursor 3D sangat mudah.			1	2	2	21	84

6.	Penambahan jenis objek *tile* dan dekorasi ke dalam *inspector* sangat mudah.			2	2	1	19	76
7.	Pemilihan jenis objek tile dan dekorasi yang ingin ditambahkan ke dalam scene view sangat mudah.		3	1	1		13	52
Total							**136**	
Rata-Rata							**19,4**	**77,7**

Tabel 3. Bobot Nilai

No.	Kategori	Nilai
1.	Sangat Setuju (SS)	5
2.	Setuju (S)	4
3.	Netral (N)	3
4.	Tidak Setuju (TS)	2
5.	Sangat Tidak Setuju (STS)	1

Tabel 4. Presentase Nilai

No.	Keterangan	Presentase
1.	Sangat Tidak Memudahkan	0% - 19.99%
2.	Tidak Memudahkan	20% - 39.99%
3.	Cukup Memudahkan	40% - 59.99%
4.	Memudahkan	60% - 79.99%
5.	Sangat Memudahkan	80% - 100%

Hasil pengujian kemudahan *plug-in* menggunakan kuesioner telah mendapatkan hasil seperti terlihat pada tabel 2. Pada pertanyaan pertama, dapat dilihat bahwa ada satu orang memilih pilihan netral yang bernilai tiga poin, tiga orang memilih pilihan setuju yang bernilai empat poin, dan satu orang lagi memilih pilihan sangat setuju yang bernilai lima poin. Kemudian jumlah pilihan dari responden tersebut akan dikalikan dengan skor pilihannya dan didapatkan hasil total skor untuk pertanyaan pertama berjumlah 20 poin. Total skor ini nanti akan digunakan untuk menghitung nilai persentase kemudahan plug-in untuk pertanyaan pertama dengan cara membagi total skor dengan jumlah responden sebanyak lima dan dikalikan dengan 100%. Didapatkanlah persentase kemudahan pada pertanyaan pertama sebesar 80%. Sesuai dengan skala likert yang telah dirancang, maka didapatkanlah 80% ini sebagai hasil yang termasuk pada kategori sangat mudah. Perhitungan untuk pertanyaan kedua hingga pertanyaan ketujuh menggunakan cara yang sama. Kemudian nilai total skor dan persentase kemudahan pada masing-masing pertanyaan akan dirata-ratakan, dan didapatkanlah hasil akhir persentase kemudahan *plug-in* sebesar 77,7%. Merujuk kepada skala likert, hasil akhir persentase kemudahan *plug-in* ini termasuk pada kategori memudahkan.

5. KESIMPULAN

Plugin ini dibuat untuk mempermudah pengembangan *tile-based game* pada *Unity game engine*. Tanpa menggunakan *plugin*, pengembang game akan membutuhkan waktu yang lama untuk membuat sebuah *level* pada sistem *tile based game*, karena pengembang harus mengatur koordinat masing-masing *tile* setiap kali *tile* baru ditambahkan ke dalam *level*. Dengan menggunakan *plugin* ini, pengembang tidak perlu menghabiskan waktu yang lama untuk membuat sebuah *level*, karena *plugin* ini memiliki fitur *point and click* serta *drag and drop* untuk melakukan operasi *tile* pada *scene view*. *Plugin* ini telah diuji oleh lima pengembang game dengan pengalaman pada bidang pembuatan game. Hasil pengujian menunjukkan dari plugin ini dikategorikan sebagai "memudahkan" dengan persentase kemudahan rata-rata sama dengan 77.7%.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh pengembang game yang telah terlibat pada uji kemudahan plugin untuk dukungan dan masukan yang jujur selama proses penelitian ini berlangsung.

DAFTAR PUSTAKA

[1] R. Bergonse, "Fifty Years on, What Exactly is a Videogame? An Essentialistic Definitional Approach," *Comput. Games J.*, vol. 6, no. 4, pp. 239–255, 2017.
[2] J. Halpern, *Developing 2D games with Unity : independent game programming with C#*. 2018.
[3] A. Salomão, F. Andaló, and M. Luiz Horn Vieira, "How Popular Game Engine Is Helping Improving Academic Research: The DesignLab Case," *Adv. Hum. Factors Wearable Technol. Game Des.*, vol. 795, pp. 416–424, 2019.
[4] R. Van Der Spuy, *Game Design with Flash*. 2010.
[5] R. Tredinnick, B. Boettcher, S. Smith, S. Solovy, and K. Ponto, "Uni-CAVE: A Unity3D plugin for non-head mounted VR display systems," *IEEE Virtual Real.*, pp. 393–394, 2017.
[6] I. Carmosino, F. Bellotti, R. Berta, A. De Gloria, and N. Secco, "A game engine plug-in for efficient development of investigation mechanics in serious games," *Entertain. Comput.*, vol. 19, pp. 1–11, 2017.
[7] N. Balzarotti and G. Baud-bovy, "HPGE: An Haptic Plugin for Game Engines," *Games Learn. Alliance*, vol. 10653, pp. 330–339, 2017.
[8] C. Luongo and P. Leoncini, "An UE4 Plugin to Develop CVE Applications Leveraging Participant's Full Body Tracking Data," *Augment. Reality, Virtual Reality, Comput. Graph.*, pp. 610–622, 2018.
[9] M. O. Rudel, G. Johannes, R. Weller, and G. Zachmann, "UnrealHaptics: A Plugin-System for High Fidelity Haptic Rendering in the Unreal Engine," *IEEE Comput. Graph. Appl.*, vol. 38, no. 2, pp. 28–30, 2018.
[10] E. Karouzaki, A. Savidis, A. Katzourakis, and C. Stephanidis, "Tile Dreamer: Game Tiles Made Easy," *Univers. Access Hum. Comput. Interact. Coping with Divers.*, vol. 4554, pp. 382–391, 2007.
[11] J. Freiknecht, C. Geiger, D. Drochtert, W. Effelsberg, and R. Dörner, "Game Engines Jonas," *Serious Games*, pp. 127–161, 2016.
[12] A. Nandy and D. Chanda, *Beginning Platino Game Engine*. 2016.
[13] J. Haas, "A History of the Unity Game Engine - An Interactive Qualifying Project," no. March, p. 44, 2014.
[14] H. Cervantes and S. C. Villalobos, "Using a Lightweight Workflow Engine in a Plugin-Based Product Line Architecture," *Compon. Based Softw. Eng.*, vol. 4068, pp. 198–205, 2006.
[15] J. S. Cuadrado and J. G. Molina, "A Plugin-Based Language to Experiment with Model Transformation Jes´us," *Model Driven Eng. Lang. Syst.*, vol. 4199, pp. 336–350, 2006.
[16] V. M. S. Durano, *Understanding Game Application Development*. 2018.
[17] M. A. Lubiano, A. Salas, S. De, R. De Sáa, M. Montenegro, and M. Á. Gil, "Soft Methods for Data Science," vol. 456, pp. 329–337, 2017.