BAB II

LANDASAN TEORI

2.1 Konsep Dasar Program

2.1.1 Pengertian *Game*

Pengertian game dalam beberapa sumber di antaranya sebagai berikut:

1. Menurut Salen dan Zimmerman bahwa "*Game* adalah sistem di mana pemain terlibat dalam konflik buatan yang didefinisikan oleh peraturan yang menghasilkan hasil yang dapat dihitung" (Wijaya & Utomo, 2017).
2. Menurut Clark C. Abt. bahwa "Sebuah *game* adalah aktivitas di antara dua atau lebih pengambil keputusan independen yang berusaha mencapai tujuan mereka dalam konteks yang membatasi. Definisi yang lebih konvensional akan mengatakan bahwa permainan adalah konteks dengan aturan di antara lawan yang mencoba untuk memenangkan tujuan" (Wijaya & Utomo, 2017).
3. Menurut Johan Huizinga bahwa "*Game* adalah aktivitas bebas berdiri dengan sangat sadar di luar kehidupan biasa sebagai hal yang tidak terlalu serius tapi pada saat bersamaan menyerap pemain dengan intens dan menyeluruh" (Wijaya & Utomo, 2017).
4. Menurut Greg Costikyan bahwa "*Game* adalah bentuk seni di mana para peserta yang disebut pemain membuat keputusan untuk mengelola sumber daya melalui token permainan dalam mengejar sebuah tujuan" (Wijaya & Utomo, 2017).

Perancangan *game* yang baik haruslah memenuhi kriteria dari *game* itu sendiri. Berikut ini adalah beberapa kriteria dari sebuah *game* (Wijaya & Utomo, 2017), yaitu:

1. Nilai Keseluruhan *(Overall Value)*

   Nilai keseluruhan dari suatu *game* terpusat pada desain dan panjang durasi *game*. Aplikasi *game* ini dibangun dengan desain yang menarik dan interaktif.

2. Dapat Digunakan *(Usability)*

   Mudah digunakan dan diakses adalah poin penting bagi pembuat *game* ataupun pemain. Aplikasi *game* ini dirancang dengan *interface* (tampilan) yang *user friendly*, sehingga *user* dengan mudah dapat mengakses dan memainkan.

3. Keakuratan *(Accuracy)*

   Keakuratan diartikan sebagaimana kesuksesan model atau gambaran sebuah *game* dapat dituangkan ke dalam percobaan atau perancangannya. Perancangan aplikasi ini harus sesuai dengan model *game* pada tahap perencanaan.

4. Kesesuaian *(Appropriateness)*

   Kesesuaian dapat diartikan bagaimana isi dan desain *game* dapat diadaptasikan terhadap keperluan *user* dengan baik. Aplikasi *game* ini menyediakan menu dan fitur yang diperlukan *user* untuk membantu pemahaman *user* dalam menggunakan aplikasi.

2.1.2 Jenis-Jenis *Game*

Di dalam dunia *game* hingga saat ini terdapat berbagai macam jenis *game* dari berbagai macam *platform* yang digunakan seperti *PC games*, *console games*, *handheld games*, hingga *mobile games*. Dengan berbagai macam jenis genre dalam *game* yang dapat dikategorikan hingga genre yang dapat dikatakan terbilang baru jenisnya dalam *mobile games*, yakni genre *idle games*. Menurut Anggra terdapat beberapa kategori *game* yang sering ditemui (Zulkifli, Jundro, & Rangan, 2017), antara lain:

1. *Arcade Game*

   Game jenis ini dapat dikatakan jenis *game* klasik. Salah satu ciri yang biasanya ditemui untuk jenis *game* ini pada umumnya memiliki tampilan 2D (Dua Dimensi) dan karakter-karakter di dalamnya bergerak ke samping diikuti dengan gerakan *background*. Contoh *game* jenis ini adalah Metal Slug, Castlevania, Super Mario, dan sebagainya.

Sumber: today.line.me & apkpure.com

Gambar II.1 Contoh *Arcade Game*

2. *Racing Game*

Jenis *game* ini berisi *game* bergenre balapan. Jenis *game* ini memacu adrenalin pemain untuk menjadi yang tercepat di arena balapan, mulai dari kendaraan seperti mobil, sepeda motor, *skateboard* hingga mainan tamiya. Contoh dari *game* jenis ini antara lain MotoGP, Need for Speed, Gran Turismo, dan lain-lain.

Sumber: m.emuparadise.me & youtube.com

Gambar II.2 Contoh *Racing Game*

3. *Fighting Game*

Game jenis ini berisi *game* pertarungan antar *player* ataupun melawan *computer*. Di dalam *game* jenis ini, pemain dapat memilih karakter yang telah disediakan yang memiliki kemampuan bertarung yang berbeda-beda serta terdapat kemampuan khusus dalam setiap karakter yang berbeda dari karakter yang dipilih untuk mengalahkan lawan yang biasanya disajikan secara *one on one* (satu lawan satu) dalam sebuah arena terbatas. Contoh *game* jenis ini adalah Bloody Roar, Tekken, Street Fighter dan lainnya.

Sumber: apkpure.com & pinterest.com

Gambar II.3 Contoh *Fighting Game*

4. *Shooting Game*

Shooting game adalah jenis *game* yang bergenre tembak-menembak. Dalam *game* jenis ini terdapat dua jenis tipe berdasarkan sudut pandangnya yaitu *first person view* (sudut pandang orang pertama), di mana pemain memainkan karakter dengan cara pandang orang pertama sehingga yang ditampilkan biasanya hanya dalam bentuk tangan yang dipersenjatai, contohnya seperti Counter Strike dan Point Blank, dan *third person view* (sudut pandang orang ketiga), di mana pemain dapat melihat keseluruhan karakter yang sedang dikendalikan di dalam *game*, dengan contoh *game* seperti PUBG dan sebagainya.

Sumber: m.blanja.com & deals.weku.io

Gambar II.4 Contoh *Shooting Game*

5. *RPG (Role Playing Game)*

Game jenis *Role Playing Game* atau yang biasa disebut *game* RPG ini merupakan *game* yang memainkan peran suatu karakter berdasarkan *class* (peran) dan *job* yang dimainkan dalam menjalankan cerita atau misi-misi tertentu yang harus diselesaikan. Contoh dari *game* jenis ini adalah Breath of Fire, Final Fantasy dan sebagainya.

Sumber: m.emuparadise.me & youtube.com

Gambar II.5 Contoh *RPG*

6. *RTS (Real Time Strategy)*

Game jenis ini lebih menitikberatkan pada unsur strategi, yang mana memerlukan kemampuan pemain untuk memimpin sebuah pasukan yang sudah ditentukan serta mengelola sumber daya yang ada, sehingga membutuhkan kecerdasan pemain dalam mengatur strategi untuk mengalahkan lawan. Contoh dari *game* jenis ini adalah KKND Krossfire, World of Warcraft, Clash of Clans dan lainnya.

Sumber: gog.com & romsmania.cc

Gambar II.6 Contoh *RTS Game*

7. *Simulation Game*

Game dengan genre ini yakni *game* yang mengambil konsep dengan mirip seperti kenyataan serta keadaannya dan biasanya diajak untuk menciptakan lingkungan sesuai dengan keinginan pemain. Contoh dari *game* jenis ini antara lain Harvest Moon, The Sims, Truck Simulator dan sebagainya.

Sumber: jalantikus.com & loop.co.id

Gambar II.7 Contoh *Simulation Game*

8. *Sports Game*

Game jenis ini menitikberatkan pada olahraga yang memiliki *gameplay* seperti berbagai jenis olahraga di dunia, yang di dalamnya pemain akan melakukan pertandingan olahraga secara virtual. Contoh dari *game* jenis ini adalah Pro Evolution Soccer, NBA, Golf dan lainnya.

Sumber: youtube.com & m.emuparadise.me

Gambar II.8 Contoh *Sports Game*

2.1.3 *Idle Game*

Idle game adalah genre *video game* yang relatif baru, *game* pertama dari genre *idle game* pertama kali diciptakan pada awal tahun 2000-an. Seperti namanya, *idle game* dirancang untuk memungkinkan pemain untuk bermain saat *idle* (menganggur), saat pemain tidak berinteraksi dengan apapun. Sebagian besar *game* jenis ini ketika dijalankan, waktu akan dihabiskan dengan sedikit interaksi dari pemain untuk mempercepat *progress* (kemajuan) dengan hasil yang tidak terbatas.

Dalam interaktivitas spektrum *idle game* menurut (Alharthi, Alsaedi, Toups, Tanenbaum, & Hammer, 2018) mengidentifikasi 3 pola umum *idle game*, yaitu:

1. *Clicker*

   Interaksi yang dilakukan dalam permainan bergenre *idle game* tidak jauh hanyalah permainan mengklik-klik saja hingga ke tingkatan yang lebih tinggi. *Game* jenis ini hanya melibatkan mengklik, menggosok, atau mengetuk sebagai mekanik inti, *damage* yang disebabkan serta sumber daya dihasilkan oleh beberapa siklus mengklik pada suatu objek dalam permainan yang di dalamnya terdapat periode hanya menunggu.

2. *Minimalist*

   Game yang minimalis mengurangi jumlah aksi atau interaksi yang tersedia dalam permainan menjadi sebagian kecil pilihan untuk berinteraksi, baik melalui mekanisme permainan yang mengotomatiskan *gameplay* atau fase *gameplay* yang mengurangi pemain untuk berinteraksi dalam *game*.

3. *Zero-Player*

   Di sisi lain, gaya permainan *idle game* adalah permainan yang di dalamnya dapat dikatakan tanpa pemain. Beberapa *idle game* tidak membutuhkan keterlibatan pemain ketika bermain atau hanya terdapat tombol *input* (masukan) yang diatur secara otomatis sehingga tidak terdapat interaksi dalam *game*, yang tetap mempengaruhi *gameplay* dalam mode latar belakang sehingga mempermudah pemain untuk kembali hadir kapan saja.

Sumber: medium.com

Gambar II.9 Contoh *Idle Game*

Idle game telah menjadi semakin populer dikarenakan sejumlah aspek *game* termasuk kemudahan aksesibilitas dalam *game* dan pemenuhannya dalam kebutuhan akan *game* di zaman modern seperti sekarang ini yang memudahkan untuk melakukan banyak tugas atau pekerjaan dan menggunakan waktu secara optimal.

2.1.4 Kecerdasan Buatan *(Artificial Intelligence)*

Menurut John McCarthy "*Artificial Intelegence: is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to understand human intelligence, but AI does not have to confine itself to methods that are biological observeable"* (Kaliman, 2019).

"*Artificial Intelligence* ialah perpaduan ilmu pengetahuan dan teknologi dalam pembuatan mesin pintar, khususnya program komputer pintar. *Artificial Intelligence* berhubungan dengan tugas yang serupa dalam menggunakan komputer untuk memahami kecerdasan manusia, tetapi AI tidak memiliki batasan tersendiri dalam metode-metode yang tampak, dengan kata lain tidak bertindak secara alamiah atau insting tetapi AI membutuhkan sebuah kecerdasan buatan yang diterapkan oleh si pembuat program" (Kaliman, 2019).

2.1.5 Bahasa Pemrograman C# (Visual C-Sharp)

Bahasa pemrograman C# atau Visual C-Sharp merupakan salah satu bahasa pemrograman berorientasi objek yang dikeluarkan Microsoft yang merupakan bahasa pemrograman modern berorientasi objek yang menjadi bahasa pemrograman utama dalam platform Microsoft.NET Framework. Visual C# dianggap sebagai kombinasi

antara efisiensi pemrograman C++, kesederhanaan pemrograman Java, dan penyederhanaan dari pemrograman Visual Basic (Hakim, 2018).

2.1.6 Animasi 2D (Dua Dimensi)

Disebut animasi dua dimensi, karena 2D mempunyai ukuran panjang *(X-axis)* dan *(Y-axis)*. Realisasi nyata dalam perkembangan dua dimensi yang cukup revolusioner yakni film kartun. Dan animasi 2D adalah animasi yang menggunakan sketsa gambar, lalu sketsa gambar ini digerakkan satu persatu, maka tidak akan terlihat seperti nyata. Disebut animasi 2D karena dibuat melalui sketsa yang digerakkan satu persatu sehingga nampak seperti nyata dan bergerak. Animasi 2D hanya bisa dilihat dari depan saja. Animasi sendiri berasal dari bahasa latin yaitu *"anima"* yang berarti jiwa, hidup, semangat (Zulkifli, Jundro, & Rangan, 2017).

2.2 Peralatan Pendukung

2.2.1 Metode *Agile Game Development*

Dalam (Wijaya & Utomo, 2017) kata *agile* berarti bersifat cepat, ringan, bebas bergerak, dan waspada, kata ini digunakan sebagai kata yang menggambarkan konsep model proses yang berbeda dari konsep model yang sudah ada. Konsep *Agile Game Development* adalah cara membangun aplikasi *game* dengan melakukannya dan membantu orang lain membangunnya sekaligus. *Agile Game Development* memungkinkan model proses yang toleransi terhadap perubahan kebutuhan sehingga perubahan dapat cepat ditanggapi, namun di sisi lain menyebabkan produktifitas menurun.

Selain itu, menurut Widodo *Agile Game Development* memiliki salah satu metodologi dalam rekayasa perangkat lunak yang memiliki 12 *practices* utama (Wijaya & Utomo, 2017), yaitu:

1. *Planning Game*

   Merupakan *practice* yang digunakan untuk melakukan perencanaan dan melakukan prioritas terhadap fitur-fitur yang dituliskan pada *index card* oleh *customer*.

2. *Small Releases*

   Rilis yang dihasilkan untuk setiap iterasi sangat pendek dan dengan umpan balik terhadap perubahan dari *customer* juga sangat cepat.

3. *Metaphor*

   Adalah semacam *simple guidance* bagi proses pengembangan dari fase paling awal hingga terakhir.

4. *Simple Design*

   Merupakan rancangan yang sederhana untuk di-*deliver* pada setiap iterasi.

5. *Testing*

   Adalah *testing* yang dilakukan setiap saat, bahkan terdapat *testing* awal.

6. *Refactoring*

   Adalah proses untuk memperbaiki *code* selain untuk menghindari berbagai redundansi yang mungkin terjadi.

7. *Pair Programming*

   Adalah proses pengembangan dengan memprogram di satu mesin komputer yang dilakukan oleh dua orang *programmer*.

8. *Collective Ownership*

   Adalah keadaan di mana semua anggota tim harus dapat menanggulangi semua hal yang berkaitan dengan proses pengembangan.

9. *Continuous Ownership*

   Adalah bahwa proses pengembangan setiap hari bahkan setiap saat terdapat perubahan harus segera diintegrasikan.

10. *40-hour Week*

    Adalah jumlah jam kerja selama satu minggu. Hal ini sebenarnya tidak mutlak 40 jam, namun intinya adalah bahwa proses pengembangan tidak mengenal lembur,

semua harus diselesaikan pada saat jam kerja. Semua masalah dan pekerjaan harus dioptimalkan pada waktu jam kerja tersebut.

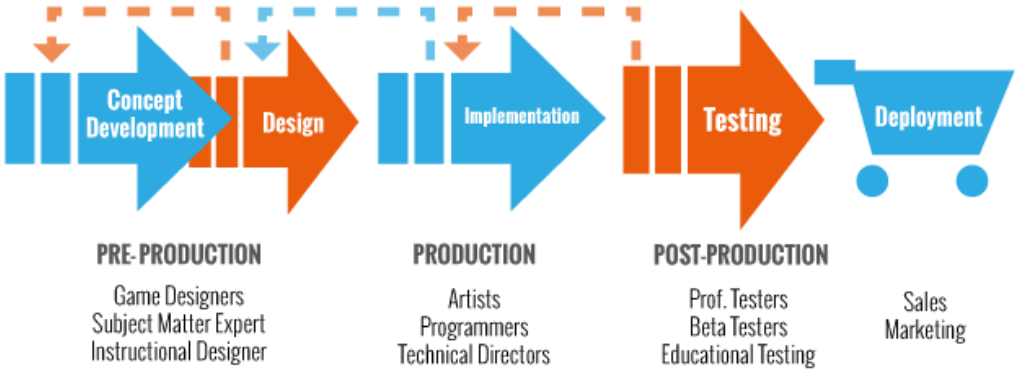
11. *On-site Customer*

Merupakan salah satu keunggulan bagi metodologi ini karena memerlukan satu orang dari pihak bisnis yang akan dibawa dalam proses pengembangan dari awal sampai berakhir. Hal ini akan cepat dalam mengatasi berbagai *requirements* yang mungkin akan berubah atau bertambah, namun akan mengurangi produktifitas pada pihak bisnis.

12. *Coding Standard*

Adalah menstandarkan proses *coding* bagi *programmer* terutama karena hal ini berkaitan dengan *practice pair programming*.

Dalam (Akbar, Damayanti, & Sulistiani, 2018) dengan tahapan model pengembangan sistem *Agile*, penelitian harus dilakukan dengan perencanaan yang teratur dan sistematis untuk mewujudkan tujuan. Berikut tahapan pembangunan *Agile Game Development*:



Sumber: BitBar.com

Gambar II.10 Metode *Agile Game Development* (Ville, 2015)

1. *Concept Development*

Pada tahap ini merupakan konsep awal dari pembuatan *idle game* yang akan dibuat, meliputi pengumpulan ide-ide untuk mendapatkan data dan keperluan lain.

2. *Design*

Pada tahapan atau fase ini akan dilakukan pendesainan yang dibutuhkan dalam aplikasi *idle game* yang akan dikembangkan. Desain sistem dan tampilan aplikasi sebagai rancangan yang dapat dijadikan acuan dalam pembuatan aplikasi *idle game* yang digunakan adalah Adobe Photoshop CS6 serta desain rancangan sistem yang akan dirancang dengan *Unified Modelling Language*, yakni:

a. *Use Case Diagram*

Dalam (Akbar, Damayanti, & Sulistiani, 2018) *Use Case Diagram* adalah diagram yang mendeskripsikan interaksi antara pengguna dengan sistem aplikasi. Fungsionalitas pada sistem akan digambarkan dengan menggunakan *use case diagram* desain dalam pembahasan berikutnya.

b. *Activity Diagram*

Dalam (Akbar, Damayanti, & Sulistiani, 2018) *Activity diagram* berfungsi untuk menggambarkan *workflow* (aliran kerja) atau aktivitas proses dari sebuah sistem. *Activity diagram* ini akan dibahas dalam pembahasan berikutnya.

c. *Flowchart*

Dalam (Kaliman, 2019) Diagram Alir *(Flowchart)* adalah sebuah jenis diagram yang mewakili algoritma, alir kerja atau proses, yang menampilkan langkah-langkah dalam bentuk simbol-simbol grafis, dan urutannya dihubungkan dengan panah. Diagram ini mewakili ilustrasi atau penggambaran penyelesaian masalah. Diagram alir digunakan untuk menganalisis, mendesain, mendokumentasi atau memanajemen sebuah proses atau program di berbagai bidang. Diagram alir *(flowchart)* dalam rancangannya akan dibahas pada pembahasan selanjutnya.

a. Rancangan Tampilan Sistem

Perancangan struktur navigasi yang menggambarkan hubungan antar menu pada aplikasi *idle game* yang akan dibangun yakni menggunakan model hierarki. Selain itu, perancangan *Story Board* merupakan salah satu konsep dari rancangan *interface* atau tampilan antar muka dari aplikasi yang dilengkapi dengan spesifikasi seperti gambar, UI (*User Interface*), teks dan sebagainya yang akan dibahas dalam pembahasan berikutnya.

3. *Implementation*

   Tahap implementasi yakni mengimplementasikan perancangan desain sistem ke dalam situasi nyata. Desain sistem yang telah didesain dan dirancang akan diimplementasikan ke dalam *tools* pembuatan *game* menggunakan Unity Game Engine.

4. *Testing*

   Pengujian dilakukan untuk memastikan apakah aplikasi dapat berjalan dengan baik pada lingkungan sistem operasi dari *user*. Pengujian *alpha* ini dilakukan oleh pengembang aplikasi sendiri dengan cara menjalankan aplikasi pada perangkat Android.

5. *Deployment*

   Di dalam tahap *deployment* ini selain sebagai peluncuran ataupun sekaligus promosi program dalam tahap ini terdapat rancangan sistem *deployment* guna dapat terpasangnya program ke dalam perangkat yang sesuai digunakan yang akan di rancang dalam bentuk *deployment diagram* yang akan dibahas dalam pembahasan berikutnya.

2.2.2 Unity Game Engine

1. Sejarah Unity

   Dalam (Kaliman, 2019) *Unity Technologies* dibangun pada tahun 2004 oleh David Helgason (CEO), Nicholas Francis (CCO), dan Joachim Ante (CTO) di Copenhagen, Denmark setelah *game* pertama mereka GooBall, gagal lagi dalam meraih sukses. Ketiganya menyadari nilai sebuah *engine* dan *tool* dalam sebuah pengembangan *game* dan berencana untuk membuat sebuah *engine* yang dapat digunakan oleh semua dengan harga terjangkau. *Unity Technologies* mendapat bantuan dana dari Sequoia Capital, WestSummit Capital, and iGlobe Partners. Kesuksesan Unity terletak pada fokus mereka untuk memenuhi kebutuhan *indie developer* yang tidak dapat membangun *game engine* mereka sendiri atau membeli lisensi *game engine* yang terlalu mahal. Fokus perusahaan ini adalah "*Democratize Game Development*" atau diterjemahkan sebagai "Demokrasi

Pembangunan *Game*" dan membuat sebuah pembangunan *game* baik 2D maupun 3D bisa dicapai oleh banyak orang, oleh siapa pun.

Pada tahun 2008, Unity melihat kebangkitan iPhone dan menjadi *game engine* pertama yang melakukan dukungan penuh pada *platform* tersebut. Unity sekarang digunakan oleh 53.1% *developers* (termasuk *mobile game developer*) dengan ratusan *game* yang dirilis baik untuk iOS maupun Android. Pada tahun 2009, Unity mulai meluncurkan produk mereka secara gratis. Jumlah *developer* yang mendaftar melonjak drastis sejak pengumuman tersebut. Pada April 2012, Unity mencapai popularitas yang sangat tinggi dengan lebih dari 1 juta *developer*.

Unity sangat mampu melihat berbagai peluang dan perubahan. Hal inilah yang menjadikannya sebagai *game engine* "termurah" yang paling banyak digunakan oleh seluruh orang di dunia. Unity bisa digunakan untuk perorangan dan tidak selalu harus digunakan oleh sebuah studio *game* yang berjumlah ratusan orang. Tampaknya, demokrasi yang diusung sebagai slogan Unity memang benar adanya. Apalagi semenjak dirilisnya Unity dengan lisensi *free*.

2. Fitur-fitur dalam Unity
   a. *Rendering*

      Dalam (Kaliman, 2019) *Graphics engine* yang digunakan adalah Direct3D (Windows, Xbox 360), OpenGL (Mac, Windows, Linux, PS3), OpenGL ES (Android, iOS), dan proprietary APIs (Wii). Ada pula kemampuan untuk *bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion* (SSAO), *dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects*.

      Unity dapat mengambil format desain dari 3DS Max, Maya, Softimage, Blender, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks and Allegorithmic Substance. *Asset* tersebut dapat ditambahkan ke *game project* dan diatur melalui *graphical user interface* Unity.

      ShaderLab adalah bahasa yang digunakan untuk *shaders*, di mana mampu memberikan deklaratif "*programming*" dari *fixed-function pipeline* dan program *shader* ditulis dalam GLSL atau CG. Sebuah *shader* dapat

menyertakan banyak varian dan sebuah spesifikasi *fallback declarative,* di mana membuat Unity dapat mendeteksi berbagai macam *video card* terbaik saat ini, dan jika tidak ada yang *kompatibel*, maka akan dilempar menggunakan *shader alternatif* yang mungkin dapat menurunkan fitur dan performa.

Pada 3 Agustus 2013, seiring dengan diluncurkannya versi 4.2, Unity mengijinkan *developer indie* mengunakan *realtime shadows* hanya untuk *directional lights*, dan juga menambahkan kemampuan dari DirectX11 yang memberikan *shadows* dengan *resolusi pixel* yang lebih sempurna, *texture* untuk membuat objek 3D dari *grayscale* dengan lebih *graphic facial*, animasi yang lebih halus dan mempercepat FPS *(Frame Per Second).*

b. *Scripting*

Script game engine dibuat dengan Mono 2.6, sebuah implementasi *open-source* dari .NET Framework. *Programmer* dapat menggunakan UnityScript (bahasa terkustomisasi yang terinspirasi dari sintax ECMAScript, dalam bentuk JavaScript), C#, atau Boo (terinspirasi dari sintax bahasa pemrograman Phyton). Dimulai dengan dirilisnya versi 3.0, Unity menyertakan versi MonoDevelop yang terkustomisasi untuk *debug script*.

c. *Asset Tracking*

Unity juga menyertakan *Server Unity Asset* – sebuah solusi terkontrol untuk *developer game asset* dan *script*. Server tersebut menggunakan PostgreSQL sebagai *backend*, sistem audio dibuat menggunakan FMOD Library (dengan kemampuan untuk memutar *Ogg Vorbis compressed audio*), video *playback* menggunakan *Theora codec*, *engine* daratan dan *vegetasi* (di mana mendukung *tree billboarding*, *Occlusion Culling* dengan *Umbra*), *built-in lightmapping* dan *global illumination* dengan *Beast*, *multiplayer networking* menggunakan RakNet, dan *navigasi mesh* pencari jalur *built-in*.

d. *Platforms*

Unity *support* pengembangan ke berbagai *platform*. Di dalam *project, developer* memiliki kontrol untuk mengirim ke perangkat *mobile, web browser, desktop, and console*. Unity juga mengijinkan spesifikasi kompresi *texture* dan pengaturan resolusi di setiap *platform* yang didukung.

Saat ini *platform* yang didukung adalah BlackBerry 10, Windows 8, Windows Phone 8, Windows, Mac, Linux, Android, iOS, Unity Web Player, Adobe Flash, PlayStation 3, Xbox 360, Wii U and Wii. Meskipun tidak semua terkonfirmasi secara resmi, Unity juga mendukung PlayStation Vita yang dapat dilihat pada *game Escape Plan* dan *Oddworld: New 'n' Tasty*.

Rencana *platform* berikutnya adalah PlayStation 4 dan Xbox One. Dan juga rumor untuk kedepannya mengatakan HTML akan menjadi *platform*-nya, dan *plug-in* Adobe baru dimana akan disubstitusikan ke *Flash Player*, juga akan menjadi *platform* berikutnya.

e. *Asset Store*

Diluncurkan November 2010, Unity *Asset Store* adalah sebuah *resource* yang hadir di Unity *editor*. *Asset store* terdiri dari koleksi lebih dari 4.400 *assets packages*, beserta 3D *models*, *textures* dan *materials*, sistem *particle*, musik dan efek suara, tutorial dan *project, scripting package, editor extensions* dan *services online.*

f. *Physics*

Unity juga memiliki *support built-in* untuk *PhysX physics engine* (sejak Unity 3.0) dari Nvidia (sebelumnya Ageia) dengan penambahan kemampuan untuk simulasi *real-time cloth* pada *arbitrary* dan *skinned meshes, thick ray cast,* dan *collision layers*.

2.2.3 Android

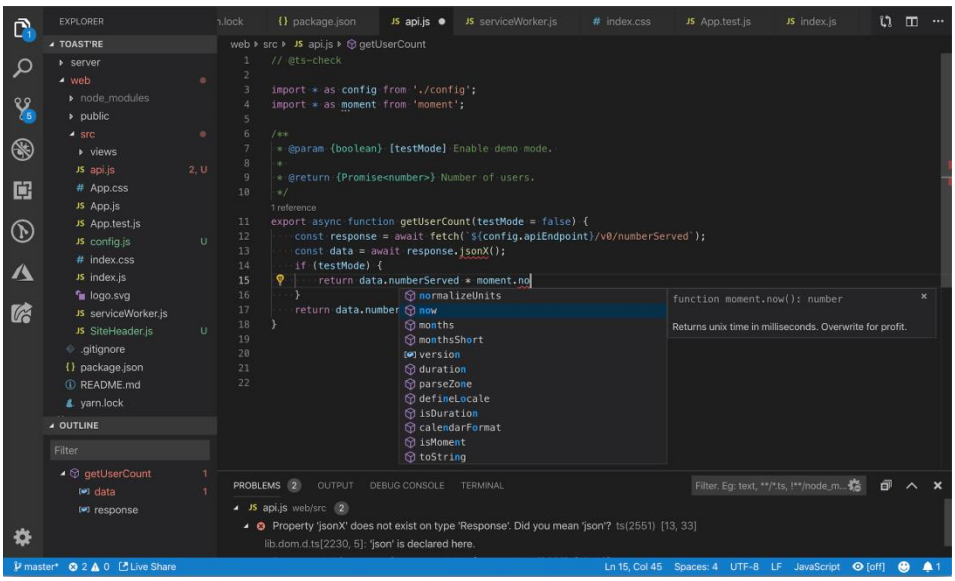
Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat seluler layar sentuh seperti *smartphone* dan *tablet*. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007.

> Menurut Jubilee Android adalah sistem operasi *open source*, dan Google merilis kodenya di bawah Apache. Kode *open source* dan lisensi perijinan pada Android memungkinkan perangkat lunak untuk dimodifikasi secara bebas dan didistribusikan oleh para pembuat perangkat, operator nirkabel, dan pengembang aplikasi (Wijaya & Utomo, 2017).

2.2.4 Microsoft Visual Studio

Dalam (Kaliman, 2019) Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi *console*, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, Integrated Development Environment (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam bentuk Microsoft *Intermediate Language* di atas .NET Framework).



Sumber: Visual Studio Code

Gambar II.11 Visual Studio Code

Salah satu alasan penggunaan Microsoft Visual Studio 2017 dalam pembuatan proyek adalah sudah terintegrasinya visual studio ini dengan Unity, dengan terdapatnya semacam *Plugin* berupa "*Game development with* Unity" sehingga memudahkan pengguna Unity dalam penulisan kode-kode program.

2.2.5 Adobe Photoshop CS6

Menurut Haka "Desain grafis adalah suatu bentuk komunikasi visual yang menggunakan teks atau gambar untuk menyampaikan informasi atau pesan seefektif mungkin" (Musmuliadi & Purmadi, 2018).

Seni desain grafis mencakup kemampuan kognitif dan keterampilan visual, termasuk tipografi, pengolahan gambar, dan *page layout*. Desain grafis diterapkan dalam desain komunikasi dan *fine art*. Seperti jenis desain lainnya, desain grafis dapat merujuk kepada proses pembuatan, metode merancang, produk yang dihasilkan (rancangan), ataupun disiplin ilmu yang digunakan (desain) (Musmuliadi, 2018).

Selain itu, menurut Andi "Adobe Photoshop merupakan aplikasi yang memang digunakan untuk memanipulasi foto, mengedit gambar, menciptakan sebuah karya *original*, dan masih banyak lagi yang berhubungan dengan seni gambar dan foto" (Musmuliadi & Purmadi, 2018).

Menurut Talajan "Kreativitas dapat diartikan sebagai pola pikir atau ide yang timbul secara spontan dan imajinatif, yang mencirikan hasil artistik, penemuan ilmiah, dan penciptaan secara mekanik. Kreativitas meliputi hasil sesuatu yang baru, baik sama sekali baru bagi dunia ilmiah atau budaya maupun secara relatif baru bagi individunya sendiri walaupun mungkin orang lain telah menemukan atau memproduksi sebelumnya" (Musmuliadi & Purmadi, 2018).

2.3 Penelitian Terdahulu

Di bawah ini adalah beberapa jurnal utama yang dijadikan sebagai penelitian terdahulu dalam bentuk tabel sebagai berikut:

Tabel II.1

Penelitian Terdahulu dari Jurnal Utama

No.	Jurnal	Judul	Metode	Kelebihan dan Kekurangan
1.	Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) Vol. 7,	GAME EDUKASI PENGENALAN HEWAN LANGKA BERBASIS ANDROID	Metode SDLC Agile	- Kelebihan Metode: Proyek menjadi lebih cepat terselesaikan atau rilis

	No. 2, Tahun 2018	MENGGUNAKAN CONSTRUCT 2		- Kekurangan Metode: Sulit diimplementasikan dalam proyek berskala besar
2.	Jurnal JIIFOR, Volume 1, No. 1, Tahun 2017	RANCANG BANGUN GAME EDUKASI ANAK USIA DINI MENGGUNAKAN METODE EXTREME PROGRAMMING BERBASIS ANDROID	Metode Agile Software Development Model Extreme Programming	- Kelebihan: Tim dapat meninjau tingkat keberhasilan dan kegagalan yang dihadapi - Kekurangan: Membutuhkan manajemen tim yang terlatih
3.	Jurnal Ilmu Komputer dan Sistem Informasi (JIKSI) Vol. 6, No. 2, Tahun 2018	PEMBUATAN GAME MATCH 3 “BE A HERO” MENGGUNAKAN UNITY	Metode Game Design Second Edition	- Kelebihan: Desain dan implementasi disusun sesederhana mungkin - Kekurangan: Waktu perencanaan proyek yang sangat singkat

Berdasarkan pembahasan tabel di atas penulis dapat merangkum kesimpulan yang dapat dijadikan landasan pembuatan program *game* yang akan dibuat dengan metode yang penulis pilih yakni metode *Agile Game Development* dengan menerima segala kelebihan dan kekurangan yang ada di dalamnya demi terealisasikannya pembuatan program *game* yang akan dibuat dalam beberapa poin sebagai berikut:

1. Penggunaan metode *Agile Game Development* akan sangat tepat untuk dalam pembangunan *game* bergenre seperti *idle game* di mana ketepatan dalam metode yang menggunakan sedikit waktu untuk sebuah *idle game* yang dapat dikatakan membutuhkan kesegeraan dalam hal rilis.
2. Selain itu, metode ini memiliki perubahan kebutuhan akan perkembangan *game* kedepannya yang dapat sering dilakukan secara berangsur setiap waktunya demi perubahan tertentu yang dapat dibilang signifikan.

3. Seiring berjalannya waktu segala kekurangan yang ada dalam *game* yang terdapat dalam metode ini dapat terpecahkan karena pergerakan sistem perkembangan yang dilakukan berangsur-angsur setiap waktunya.
4. Metode *Agile Game Development* dapat dikatakan metode yang fleksibel penggunaannya terhadap pembangunan maupun pengembangan program.