JURNAL ILMIAH KOMPUTER GRAFIS, Vol.15, No.2, Desember 2022, pp. 254 - 266
p-ISSN : 1979-0414(print)
e-ISSN : 2621-6256 (online)
http://journal.stekom.ac.id/index.php/pixel · page 254

ANALISIS PENERAPAN METODE OBJECT POOLING PADA GAME 2D ENDLESS RUNNER

Aditya Cipta Rahadian[1], Iwan Setiawan[2], Geroge Pri Hartawan[3]
[1]Teknik Informatika – Universitas Muhammadiyah Sukabumi, adityacipta13@gmail.com
[2]Teknik Informatika – Universitas Muhammadiyah Sukabumi, metalizer515@ummi.ac.id
[3]Teknik Informatika – Universitas Muhammadiyah Sukabumi, george@ummi.ac.id

ARTICLE INFO

Article history:

Received
Received in revised form
Accepted
Available online

ABSTRACT

In this era of rapid technological development, many things are developing rapidly, one of which is the development of games in the world. Many games have been made by the developers themselves, so there are already many variations of the type of game itself, one of which is the endless runner genre game. In the endless runner game itself, there is an object that we have to create massively and repeatedly which usually applies to how to create and delete these objects massively.

However, there is a better method in optimizing the implementation for massive object creation and deletion with the object pooling method, where the difference is in this method uses how to activate and deactivate objects. So in this way, it will optimize the game so that its performance does not drop and memory usage does not take up much space.

Keywords: Game Optimization, Memory Usage, Endless Runner, Object Pooling

1. Pendahuluan

Perkembangan teknologi diera digital ini telah berkembang pesat salah satunya dibidang *game*. Yang dulunya hanya hitam putih sekarang sudah berwarna, juga *game* 2D sekarang sudah 3D. Dulu *game* hanya dapat dimainkan menggunakan console, namun sekarang dapat dimainkan di PC dan *smartphone*. Seiring perkembangan zaman *game* memiliki berbagai genre seperti *endless runner*, *arcade*, *platformer* dan lain-lain. Tak hanya beragam namun banyak juga yang sudah populer didunia, seperti berita yang beberapa bulan ini terbit membahas mengenai *game subway surfers* yang meraih penghargaan sebagai *game* paling banyak diunduh didunia pada april 2022 dikutip dari website liputan6.com, *Subway Surfers* ini digarap oleh Sybo Games yang merupakan perusahaan asal Denmark, *game* bergenre *endless runner* ini tersedia untuk android, ios, kindle dan windows phone [1].

Endless Runner adalah sebagai *game* aksi, linier dalam desain tanpa akhir, tanpa jeda atau istirahat dan tanpa tahapan atau perubahan *level*. Ini memiliki satu tingkat berkelanjutan. Kesulitan permainan mulai lambat dan mudah [2]. *Game* ini sangat menarik untuk dibahas karena didalamnya terdapat unsur objek yang akan selalu dibuat atau diinialisasi secara terus menerus sampai *player* kalah dalam kondisi tertentu [3]. Oleh karena itu peneliti mengambil *game* bergenre *endless runner* ini karena cocok dengan penelitian yang akan dibahas mengenai

Received September 23, 2022; Revised Nov 1, 2022; Accepted Des, 2022

penginialisasian ulang objek dan juga belum banyak yang membahas juga mengenai optimalisasi *game* bergenre *endless runner* ini.. Yang dimana contoh dari *game* ini yaitu seperti *Cookie Run: Oven Break*, *Subway Surfers*, *Trex Game*, dan lain sebagainya.

Gambar 1 *Game Endless Runner*

Biasanya dalam penerapan metode untuk digunakan dalam penginialisasian ulang objek menggunakan metode *instantiate & destroy*, metode yang dimana objek akan dibuat dan dihancurkan secara terus menerus. Dalam hal ini penerapan metode tersebut terbilang kurang efektif dalam kerjanya dan dampak yang terjadi dari penghancuran objek secara masif tersebut akan menambah keterbebanan kerja dalam sistem.

Dari riset pembelajaran yang telah peneliti baca dan pelajari terdapat metode yang lebih baik yaitu dengan menggunakan metode *object pooling*. dalam penerapan metode ini perbedaannya yaitu menggunakan objek kembali dengan mengaktifkan dan menonaktifkan suatu objek tersebut. Sehingga dampaknya *game* tersebut akan mempengaruhi alokasi penggunaan memori pada *game* tersebut agar tidak terbebani [4].

Berdasarkan data yang peneliti telusuri di jurnal artikel "*Fundamentals of HTML5 Game Optimization*", dalam penelitian tersebut telah melakukan suatu pengujian dalam pengoptimalan *game* menggunakan metode *object pooling*. Dalam jurnal ini juga mengujinya pada beberapa perangkat berbeda saat menerapkan metode *object pooling* tersebut dan melakukan perhitungan efektifitasnya, Hasil yang dilakukan dalam penelitan ini mempengaruhi semua perangkat dan semua perangkat menjadi lebih cepat. Angka persentasi dari setiap perangkat tersebut yaitu sebagai berikut, Sony 79,39%, Honor 7 68,22%, iPhone se 52,08, dan Dell 7010 81,81% lebih cepat ketika menggunakan penerapan metode *object pooling* dalam penginialisasian ulang objek tersebut. [5].

Penelitian ini dibuat dengan tujuan untuk menganalisis efektifitas metode *object pooling* pada *endless runner games* dalam penerapan untuk inialisasi ulang objek yang massif. Karena dari referensi jurnal yang peneliti telusuri belum ada yang membahas mengenai penerapan metode *object pooling* dalam *game endless runner* ini.

2. Tinjauan Pustaka

Terdapat beberapa penelitian yang telah dipublikasikan terkait metode *object pooling* ini dalam penerapan yang berbeda-beda, seperti contohnya pada jurnal "*Development Of FPS Defense Game Using Object Pooling*" oleh Wongyu Lim, Syoungog On dan Soo Kyun Kim [6], yang berhasil menerapkan metode *object pooling dalam game FPS* untuk mengurangi beban memori. Lalu jurnal "Endure" oleh John Albergo dan Justin C. Cullen [7], yang menerapkan *object pooling* pada *game zombie thriller*. Adapun dalam jurnal "*Development Of Mobile Defense Game Using Unity Engine*" oleh Sanghoon Park, Euiseong Shin, Changmin Han, Wongyu Lim, Soo Kyun Kim, Syungog An [8], yang berhasil menerapkan metode *object pooling* dalam *game defense* seperti *plant's & zombie*. Dan terakhir dalam jurnal "*Development of Mobile RPG using Unity 3D Engine*" oleh Horyul Kim, Jongho Yoo, Changmin Han, Syoungog An, Soo Kyun Kim [9], yang menerapkan metode *object pooling* ini kedalam game bergenre *RPG*.

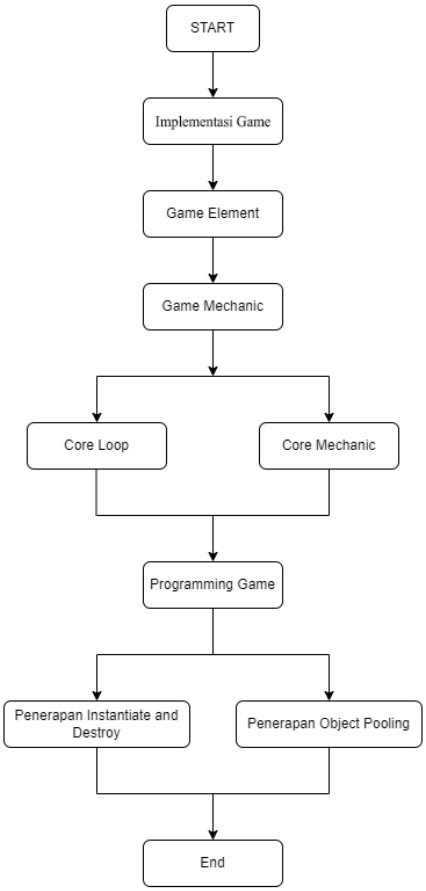
p-ISSN : 1979-0414 e-ISSN : 2621-6256

Untuk menganalisis bagaimana penerapan metode *object pooling* ini bekerja peneliti menelusuri berbagai artikel jurnal dari beberapa penelitian terdahulu, peneliti menemukan dalam jurnal "*Improving Mobile Game Performance with Basic Optimization Techniques in Unity*" oleh Georgios Koulaxidis, Stelios Xinogalos mengajarkan peneliti teknik dasar dalam optimalisasi *game* di unity yang dimana didalamnya terdapat pembahasan mengenai *object pooling* [10]. Dan pada jurnal "*Fundamentals of HTML5 Game Optimization*" oleh Petteri Pekonen peneliti menemukan data dalam angka presentasi bahwa dalam penerapan *object pooling* pada *game* dapat membuat performa game meningkat, dilakukan juga pada pengujian pada beberapa perangkat dan menghasilkan presentasi besar dalam salah satu perangkat yang dimana mendapat angka 81.81% performa *game* meningkat [5]. Maka dari itu peneliti ingin membuat analisis penerapan metode *object pooling* genre *game endless runner* yang peneliti pilih untuk menguji keefektifan atau keoptimalan dari penerapan metode ini.

3. Metodologi

3.1. Tahapan Penelitian

Pada penelitian ini dilakukan dengan membuat *game* 2D *endless runner* sederhana dengan 2 penerapan metode yang berbeda menggunakan Unity yang dimana berbahasa pemograman C# dan mengekspor *game* tersebut menjadi aplikasi *game* yang dapat dimainkan di PC/Desktop lalu melakukan analisisnya menggunakan unity profiler dan task manager. Untuk tahapan kerja dalam penelitian ini yaitu sebagai berikut.



Gambar 2 Tahapan Penelitian

Penelitian dilakukan selama 2 bulan untuk membuat *game* dan menganalisis hasil dari penerapan metode tersebut, analisis ini dilakukan juga pada beberapa laptop dengan merek berbeda yang dimana menganalisis penggunaan memori pada hasil *game* yang telah dibuat tersebut dengan menggunakan task manager.

3.2. Data dan Perangkat Penelitian

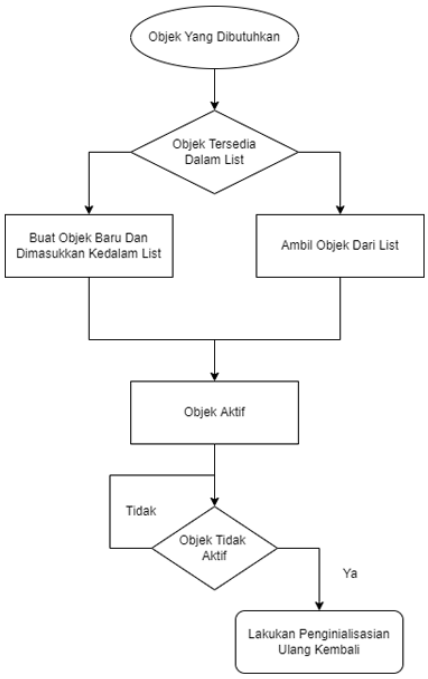
Data yang diperlukan untuk kebutuhan penelitian ini didapat dari pembelajaran akademi *game* dan terdapat beberapa referensi dari situs diskusi *website* pemograman yaitu *Unity Learn*, *Github*, *Stack Overflow* dan *GeeksforGeeks*.

Untuk hasil analisisnya didapat dari unity profiler yang merupakan *tools* dari unity, dari buku yang saya baca berjudul "*Optimizing Unity Projects*" *profiling* ini berfungsi untuk menampilkan data statistik penggunaan sistem pada *game* yang dijalankan seperti contohnya memori, cpu, gpu dan lain-lain [11]. Lalu selain analisis dari unity peneliti juga melakukan analisis penggunaan sistem menggunakan task manager untuk menganalisis hasil game yang telah dibangun/dibuat atau program yang telah *executable* atau dapat dimainkan secara langsung.

3.3. Object Pooling

Object Pooling adalah cara yang bagus untuk mengoptimalkan proyek *game* dan menurunkan beban yang ditempatkan pada CPU ketika harus dengan cepat membuat dan menghancurkan *GameObjects*. Ini adalah praktik yang baik dan pola desain yang perlu diingat untuk membantu meringankan kekuatan pemrosesan CPU untuk menangani tugas yang lebih penting dan tidak dibanjiri oleh panggilan membuat dan menghancurkan yang berulang [12].

Pada jurnal "*Implementation Of A Video Game Collection Games For Educational Use*" oleh Sami Krouvi dijelaskan bahwa metode ini akan mengoptimalkan kerja dari *game* dalam melakukan penginialisasian ulang objek dengan cara membuat dan mengaktifkan objek lalu menonaktifkan objek tersebut. Keunggulan dari metode ini adalah tidak terjadinya penghancuran objek masif, yang dimana akan mengoptimalkan kinerja *game* dalam penggunaan memori [13]. Berikut merupakan alur proses kerja dari metode *object pooling* ini.



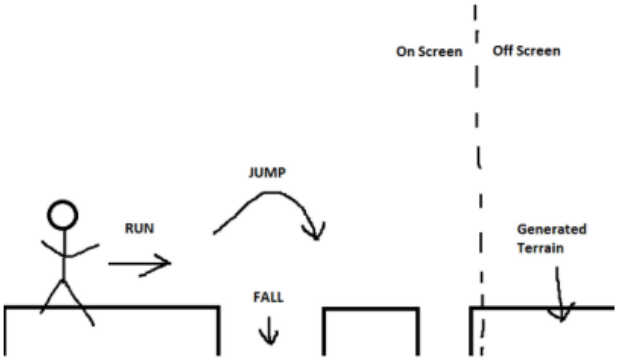
p-ISSN : 1979-0414 e-ISSN : 2621-6256

Gambar 3 Alur Metode *Object Pooling*

4. Hasil dan Pembahasan

4.1. Implementasi *Game*

Untuk skema desain *gameplay* yaitu karakter yang pengguna mainkan akan berlari otomatis dan pengguna diharuskan untuk melompat untuk tidak jatuh. Lalu kegunaan *generated terrain* ini untuk penerapan metode yang akan digunakan untuk melakukan penginialisasian ulang objek yang tak habis-habis. Yang dimana *game* akan terus berjalan sampai karakter jatuh dari *ground*.



Gambar 4 Desain *Gameplay*

Untuk hasil dari implementasi dari desain yang telah dibuat yaitu sebagai berikut.

Gambar 5 Implementasi Hasil *Game*

Untuk kebutuhan penelitian ini dibuatlah 2 *game* dengan data yang sama namun dengan penerapan metode berbeda. Yang satu menerapkan pembuatan dan penghapusan objek *ground* secara sekaligus dan satu lagi menggunakan metode *object pooling* untuk penginialisasiannya. Yang dimana dapat dilihat perbedaanya pada *hierarcy* unity yang akan dijelaskan di sub bab selanjutnya.

4.2. *Game Element*

Berikut data komponen yang digunakan pada game yang telah dibuat ini.

1. *Character*

   Komponen karakter yang dimainkan oleh pengguna sebagai berikut.

Gambar 5 Komponen Karakter

2. *Ground*

Komponen *Ground* ini terdiri dari 3 bagian diantarnya yaitu.

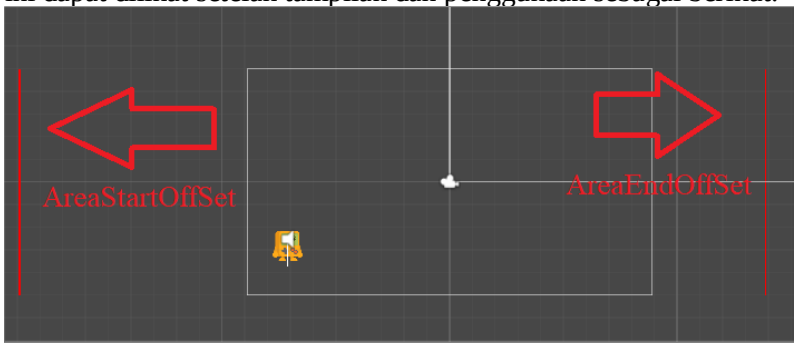
Gambar 6 *Ground Template* 1

Gambar 7 *Ground Template* 2

Gambar 8 *Ground Temp*late 3

3. *Border Line*

Batas ini berfungsi sebagai referensi untuk membuat dan menghancurkan *ground*. Hal ini dapat dilihat setelah tampilan dan penggunaan sebagai berikut.



Gambar 9 *Border Line*

Dapat dilihat diatas terdapat 2 pembatas yaitu sebagai berikut.

A. AreaStartOff

Area garis ini berfungsi sebagai referensi untuk pembuatan *ground* yang dibuat di awal permainan dan sebagai referensi untuk penghancuran *ground* saat selesai digunakan.

p-ISSN : 1979-0414 e-ISSN : 2621-6256

B. AreaEndOff

Area garis ini berfungsi sebagai *referensi* untuk batas akhir dari ground yang dibuat di awal permainan dan sebagai *referensi* untuk pembuatan kembali ground untuk digunakan.

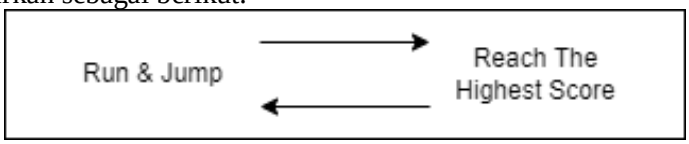
4.3. *Game Mechanic*

Mechanic adalah beragam aturan yang mengatur tindakan dan perilaku player serta keadaan (*state*) dalam sebuah *game*. Sederhananya mekanik adalah apa saja yang bisa dilakukan oleh *player*, cara melakukannya dan kenapa *player* tersebut harus melakukan hal tersebut [14].

Untuk Mekanik yang dibuat dalam *game* ini akan dijelaskan sebagai berikut.

1. Core Loop

   Objektif yang diperlukan pada *game* ini yaitu mendapatkan skor tertinggi dari skor sebelumnya dengan terus bertahan tanpa jatuh dalam *game* tersebut. Dapat digambarkan sebagai berikut.



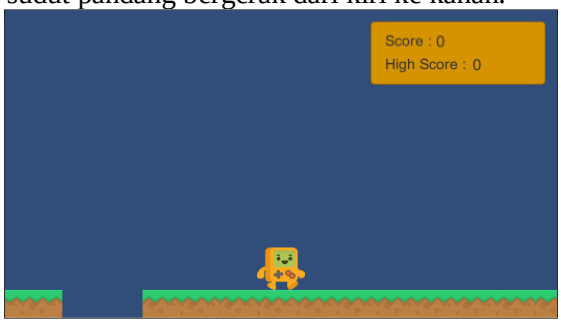
Gambar 10 *Core Loop*

2. Core Mechanic

   Untuk mekanik dalam *game* ini yaitu sebagai berikut.

   A. *Game View*

   Sudut pandang yang diterapkan pada *game* ini yaitu *side-scrolling* yang dimana sudut pandang bergerak dari kiri ke kanan.

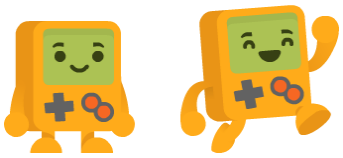


Gambar 11 *Game View*

   B. *Character*

   Untuk mekanik dalam karakter ini terdapat 2 yaitu berlari yang dilakukan secara otomatis dan melompat sesuai kehendak dari pengguna kapan akan melakukan aksi tersebut. Yang dimana animasi karakter tersebut yaitu sebagai berikut.

Gambar 12 Animasi Berlari



Gambar 13 Animasi Lompat

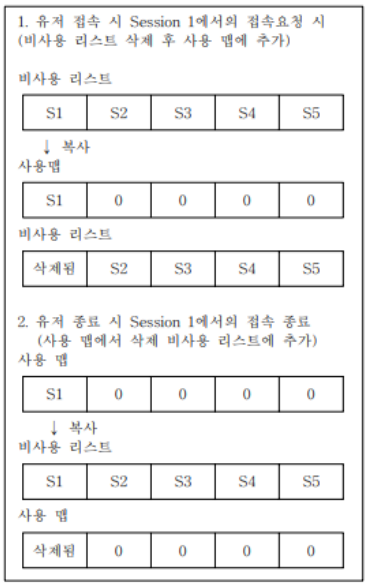
C. *Ground*

Pada mekanik ground terdapat 3 *ground* yang akan dipakai yang dipanggil secara acak untuk menjadi pijakan bagi karakter. Untuk *ground* ini sendiri akan terus menerus memanggil dan menghilangkan objek yang dimana dalam penerapannya akan menggunakan metode yang akan digunakan yaitu dengan metode *object pooling* dan pada game satu lagi menggunakan penerapan membuat dan mencurkan objek secara berulang.

Gambar 14 Mekanik *Ground*

4.4 Penerapan *Object Pooling*

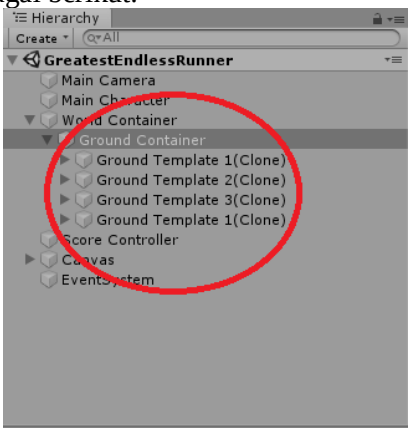
Dalam kerja penginialisasian ulang objek yang dilakukan pada objek *ground* ini diterapkan dengan metode object pooling yang dimana kerjanya itu dengan menambahkan objek lalu mengaktifkannya dan apabila tidak digunakan maka akan dinonaktifkan. Untuk konsepnya saya mengambil dari jurnal luar negeri dengan judul ”*A Study Of Object Pooling Scheme For Efficient Online Gaming Server*” [15], yang digambarkan seperti berikut.



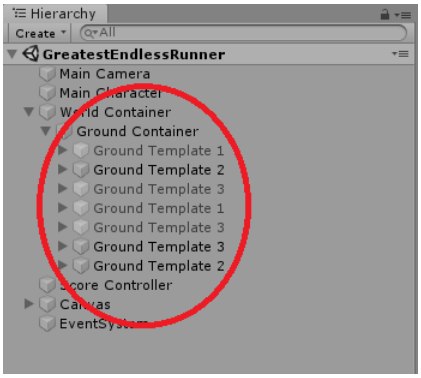
Gambar 15 Konsep Terdahulu

p-ISSN : 1979-0414 e-ISSN : 2621-6256

Perbedaan dari penerapan metode ini dapat dilihat dari *hierarcy* unity yang dimana perbedaannya dapat dilihat sebagai berikut.



Gambar 16 Terapan *Instantiate & Destroy*



Gambar 17 Terapan *Object Pooling*

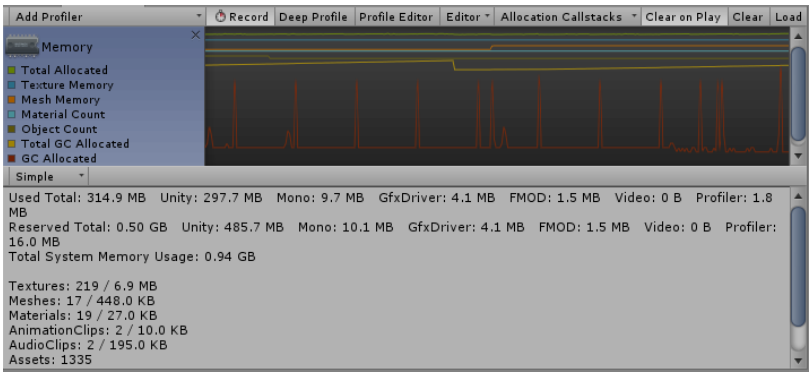
Dapat dilihat dari gambar sebelumnya yaitu perbedaanya *instantiate & destroy* ini melakukan pembuatan dan penghancuran sekaligus secara langsung sehingga pada *ground containter* yang menampung kumpulan ground digunakan 4 sampai 5 dalam kumpulannya. Sedangkan pada penerapan *object pooling* kerjanya dengan menambahkan *ground* yang tidak ada dalam dikumpulan kemudian menambahkan juga diaktifkan dan akan dinonaktifkan bila sudah tidak digunakan.

Lalu dampak yang terjadi dalam penerapan yang dilakukan tersebut dalam kerja penginialisasian ulang ini adalah pada *instantiate & destroy* ini akan membuat kerja *game* semakin terbebani karena banyaknya objek *ground* yang dhancurkan terus menerus secara masif yang berakibat pada penggunaan memori akan terbebani. Berbeda dengan object pooling yaitu dengan menonaktifkan objek yang sudah tak digunakan dan mengaktifkannya kembali ketika digunakan sehingga tak membebani kinerja *game* ketika harus melakukan penginialisasian ulang secara terus menerus dan dampaknya *game* pun tak terbebani.

4.5. Analisis Penggunaan Memori

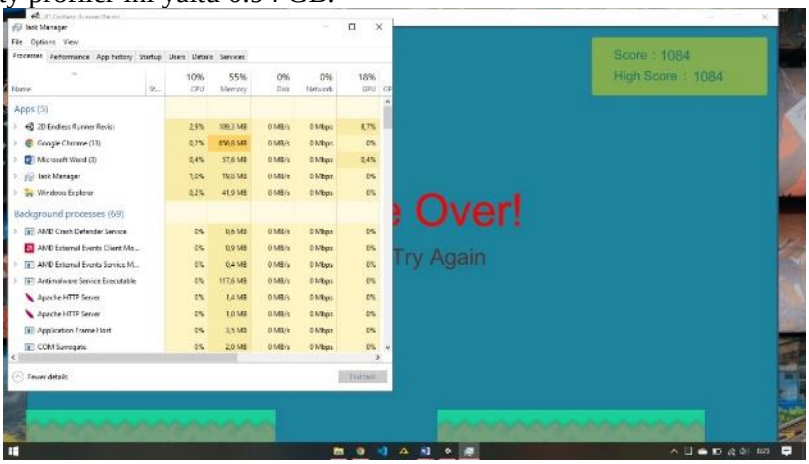
Pada tahapan analisis ini menggunakan unity profiler dan task manager sebagai medianya, yang dimana 2 *game* tersebut dijalankan lalu dilihat berapa penggunaan memori dan yang lainnya ketika game tersebut dijalankan. Dan mendeksripsikan pengalaman ketika menjalankan game tersebut apabila terdapat *lag* atau *delay*.

1. Analisis Penerapan Metode *Instantiate & Destroy*



Gambar 18 Analisis Metode *Instantiate & Destroy* Menggunakan Unity Profiler

Untuk analisis ini *game* masih dalam bentuk unity yang dilakukan dengan menggunakan *tools* unity profiler. Pada penerapan *instantiate & destroy* ini terdata total penggunaan memori yang dipakai adalah 314.9 MB dan total penyimpanan 0.50 GB. Apabila ditotalkan maka total penggunaan sistem memori dengan menggunkanan unity profiler ini yaitu 0.94 GB.

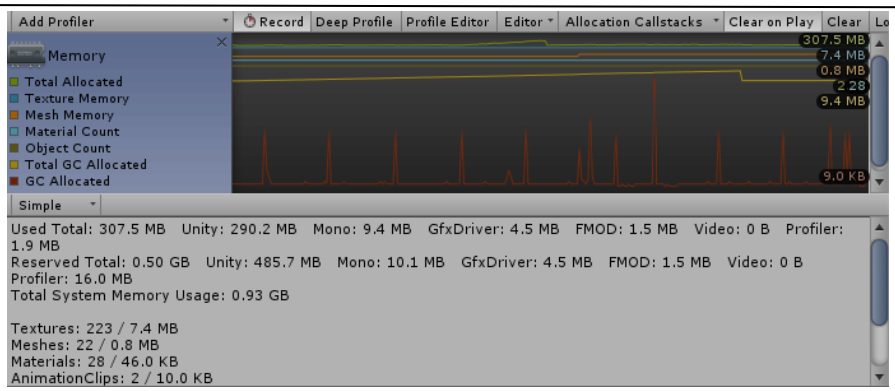


Gambar 19 Analisis Metode *Instantiate & Destroy* Menggunakan Task Manager

Untuk analisis pada task manager ini *game* telah menjadi program yang dapat dieksekusi untuk PC/Desktop. Pada penerapan *instantiate & destroy* ini menggunakan memori sebanyak 109.3 MB.

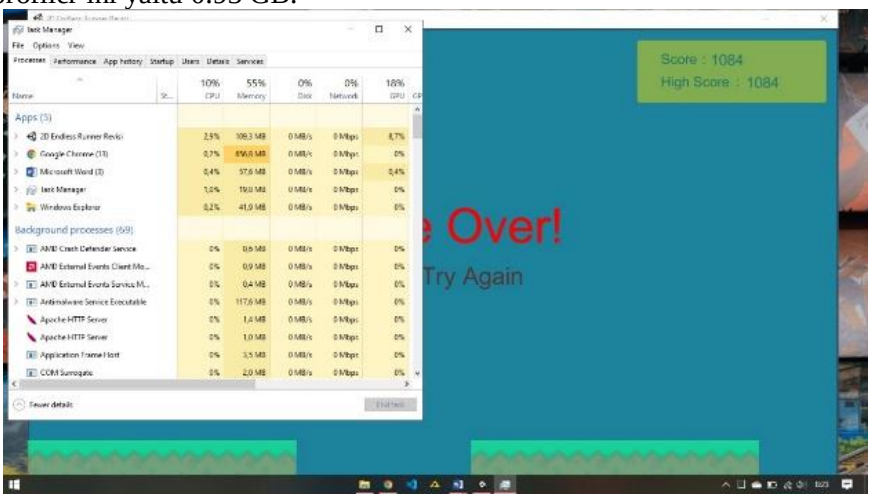
2. Analisis Penerapan Metode *Object Pooling*

p-ISSN : 1979-0414 e-ISSN : 2621-6256



Gambar 20 Analisis Metode *Object Pooling* Menggunakan Unity Profiler

Untuk analisis ini *game* masih dalam bentuk unity yang dilakukan dengan menggunakan *tools* unity profiler. Pada penerapan *object pooling* ini terdata total penggunaan memori yang dipakai adalah 307.5 MB dan total penyimpanan 485 MB. Apabila ditotalkan maka total penggunaan sistem memori dengan menggunkan unity profiler ini yaitu 0.93 GB.



Gambar 21 Analisis Metode *Object Pooling* Menggunakan Task Manager

Untuk analisis pada task manager ini *game* telah menjadi program yang dapat dieksekusi untuk PC/Desktop. Pada penerapan *object pooling* ini menggunakan memori sebanyak 104.5 MB.

4.6 Hasil Analisis Performa *Game*

Dari analisis dan pengujian penerapan metode yang berbeda dalam game 2D endless runner ini, bahwa terdapat perbedaan kinerja performa *game* ketika dilakukan penerapan metode *instantiate & destroy* dan *object pooling*. Yang dimana pada pengujian menggunakan unity profiler dalam kinerja performa *game* yang menerapkan metode *object pooling* memiliki angka 10MB lebih rendah dari penerapan menggunakan metode *instantiate & destroy* yang artinya dalam penerapannya metode *object pooling* ini lebih efektif dalam optimalisasi *game*.

Dilakukan juga pengujian menggunakan task manager untuk *game* yang telah diekspor menjadi *game executable* yang dapat dimainkan di PC/desktop. Pada analisis ini peneliti menyiapkan 3 laptop untuk pengujiannya diantaranya Lenovo G41-35 yang telah ditampilkan pada bab sebelumnya, lalu pada Asus X441U dan Acer Z-14, mendapatkan data sebagai berikut.

1. Asus X411U

   Penerapan Metode *Instantiate & Destroy*

   - Penggunaan CPU : 4,9%

- Penggunaan Memori : 106.6 MB

Penerapan Metode *Object Pooling*

- Penggunaan CPU : 3,9%
- Penggunaan Memori : 105.5 MB

2. Acer Z-14

Penerapan Metode *Instantiate & Destroy*

- Penggunaan CPU : 7,4%
- Penggunaan Memori : 64.6 MB

Penerapan Metode *Object Pooling*

- Penggunaan CPU : 6,5%
- Penggunaan Memori : 63.4 MB

Sehingga dapat disimpulkan pada hasil game yang telah dibuat bahwa dalam penerapan metode dalam penginialisasian ulang ini lebih efektif menggunakan *object pooling*. Lalu untuk pengalaman dalam bermain pun ketika memainkan *game* dengan penerapan *object pooling* lebih lancar dibanding dengan *instatiate & destroy* yang terkadang *lag* akibat proses penghapusan objek yang masif berdampak pada performa *game* yang turun.

5. Kesimpulan

Penerapan metode *object pooling* ini dapat mengurangi keterbebanan memori ketika melakukan suatu penginialisasian ulang yang berlangsung secara terus menerus atau masif. Telah dilakukan juga pengujian kepada beberapa merek Laptop lain yang telah diuji yaitu di Asus X441U, Lenovo G41-35 dan Acer Z-14 memang lebih efektif menggunakan metode *object pooling* tersebut. Dan sedikit pengalaman dari pengujian yang telah dilakukan ketika menggunakan metode *instantiate & destroy* terkadang sedikit *lag* ketika akan membuat *ground* yang baru, karena dampak dari penghapusan objek yang masif.

Daftar Pustaka

[1] D. Pratama, "Subway Surfers Jadi Mobile Game Paling Banyak Diunduh di Seluruh Dunia," 2022. https://www.liputan6.com/tekno/read/4970232/subway-surfers-jadi-mobile-game-paling-banyak-diunduh-di-seluruh-dunia (accessed Jun. 16, 2022).

[2] JERRY MOMODA, "Endless Runner Games: Evolution and Future," 2018. http://jerrymomoda.com/analysis-endless-runners/ (accessed Jun. 10, 2022).

[3] V. Černý, "Procedural Generation of Endless Runner Type of Video Games," pp. 1–82, 2018, [Online]. Available: https://dspace.cuni.cz/handle/20.500.11956/101879.

[4] M. Placzek, "Object Pooling in Unity," 2016. https://www.raywenderlich.com/847-object-pooling-in-unity (accessed Jun. 10, 2022).

[5] P. Pekonen, "Fundamentals of HTML 5 game optimization," pp. 1–61, 2019, [Online]. Available: https://lutpub.lut.fi/handle/10024/160161.

[6] W. Lim, S. An, and S. K. Kim, "Development Of FPS Defense Game Using Object Pooling," *Dep. Game Eng.*, vol. 27, pp. 77–78, 2019, [Online]. Available: https://www.koreascience.or.kr/article/CFKO201909258121494.page.

[7] J. Albergo and J. C. Cullen, "Endure," 2018. [Online]. Available: https://digitalcommons.sacredheart.edu/acadfest/2018/all/73/.

[8] S. Park, E. Shin, C. Han, W. Lim, K. Kim, and S. An, "Development Of Mobile Defense Game Using Unity Engine," pp. 29–30, 2018, [Online]. Available: https://www.koreascience.or.kr/article/CFKO201831342441109.page.

[9] H. Kim *et al.*, "Development of Mobile RPG using Unity 3D Engine," *Korean Soc. Comput. Informatio*, vol. 7, pp. 3–5, 2019, [Online]. Available: https://www.koreascience.or.kr/article/CFKO201920461757967.page.

[10] G. Koulaxidis and S. Xinogalos, "Improving Mobile Game Performance with Basic

Optimization Techniques in Unity," *Modelling*, vol. 3, no. 2, pp. 201–223, Mar. 2022, doi: 10.3390/modelling3020014.

[11] A. Lehtola, "Optimizing Unity Projects," pp. 1–31, 2018, [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/150040/Lehtola_Aleksi.pdf?sequence=.

[12] U. Technologies, "Introduction to Object Pooling," 2019. https://learn.unity.com/tutorial/introduction-to-object-pooling# (accessed Jun. 10, 2022).

[13] S. Krouvi, "Implementation Of A Video Game Collection Games For Educational Use," South-Eastern Finland University of Applied Sciences, 2019.

[14] G. Design and S. Independen, "Mechanic & Dynamic," no. March, 2022.

[15] H.-Y. Kim, D.-H. Ham, and Moonseong Kim, "A Study Of Object Pooling Scheme For Efficient Online Gaming Server," pp. 163–170, 2009, [Online]. Available: https://www.koreascience.or.kr/article/JAKO200908939909925.page.