Jurnal Mantik
Volume 4 Number 1 May 2020, pp. 61-68 E-ISSN 2685-4236
https://iocscience.org/ejournal/index.php/mantik/index

Performance Analysis Of Navigation Ai On Commercial Game Engine: Autodesk Stingray And Unity3D

Moch Fachri[1], Ali Khumaidi[2], Nur Hikmah[3], Nuke L Chusna[4]

[1,2,3,4]Informatics,
Enginerring Faculty, Universitas Krisnadwipayana, Jl. Raya Jatiwaringin, Jawa Barat 13077, Indonesia

E-mail: [1]moch.fachri@unkris.ac.id, [2]alikhumaidi@unkris.ac.id, [3]nurhikmah@unkris.ac.id, [4]nukelchusna@unkris.ac.id

ARTICLE INFO

Article history:
Received: 12/02/2020
Revised: 09/03/2020
Accepted: 01/05/2020

Keywords:
Crowd Simulation,
Agent Navigation,
autodesk Stingray,
unity3D

ABSTRACT

This research establish crowd simulation on modern game engine such as Autodesk Stingray and Unity 3D. This paper explores the navigation system of both game engine. Furthermore we compare the navigation performance for each navigation system used by those engine: The gameware Navigation which is used in Stingray as its middleware for navigation AI, and Unity Navigation used in Unity3D. We simulate the crowd simulation using scenario of crossroad and narrow-passage. Experimental result demonstrates the navigation of hundreds of agents in densely populated environments.

1. Introduction

Navigation of character in modern gaming is one of many interest to game technology. Most modern games engines incorporate navigation tool to support game developers in designing the movement of the characters in the game. Many game engine designed for the creator to express their design without any hassle of programming knowledge[1].The AI included on those navigation tool has various type and algorithm. With variety option available, the developers can simply choose based on their needs.

This research mainly focused in performance analysis of navigation AI on two commercial game engine. We choose to compare the navigation performance between Autodesk Gameware navigation used in stingray game engine with the navigation tool used in unity3D[1-4].

2. System Overview

A. Unity3D Navigation System

Grand design of navigation in Unity navigation tool consist of two major navigation as shown on figure 1:

1) Global navigation is used to find the corridor across the world. Finding a path across the world is a costly operation requiring quite a lot of processing power and memory.
2) Local navigation tries to figure out how to efficiently move towards the next corner without colliding with other agents or moving objects.

In other word, global navigation is how to reason about the level to find the destination, location are described by navigation mesh component. The navigation mesh is navigable environment in the virtual world. As already illustrated on figure 1, the agent will access map location which contain the navmesh data for the environment. Next process is path finding, Unity3D uses A* For the path finding [7]. Last process involving corridor update that the agent will use for navigating through the environment. Local navigation is about agent ability to move toward destination[7]. It tries to figure out how to efficiently move towards the next corner without colliding with other agents or moving objects. Steering process

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Jurnal Mantik
Volume 4 Number 1 May 2020, pp. 61-68 E-ISSN 2685-4236
https://iocscience.org/ejournal/index.php/mantik/index

figures out a desired Velocity (direction and speed) needed to reach the destination. While Obstacle avoidance process will chooses a new velocity based on steering velocity. The avoidance process will balances between moving in the desired direction and preventing possible collisions.

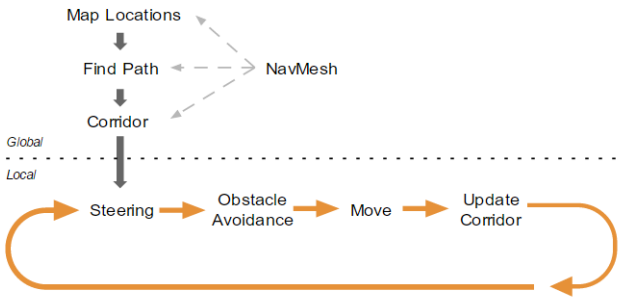

Fig 1. Navigation process in Unity navigation tool[7]

There are four major component in Unity Navigation as shown on figure 2[7]:

1) NavMesh (short for Navigation Mesh) is a data structure which describes the walkable surfaces of the game world and allows to find path from one walkable location to another in the game world. The data structure is built,or baked, automatically from your level geometry.
2) NavMesh Agent component help you to create characters which avoid each other while moving towards their goal. Agents reason about the game world using the NavMesh and they know how to avoid each other as well as moving obstacles.
3) Off-Mesh Link component allows you to incorporate navigation shortcuts which cannot be represented using a walkable surface. For example, jumping over a ditch or a fence, or opening a door before walking through it, can be all described as Off-mesh links.
4) NavMesh Obstacle component allows you to describe moving obstacles the agents should avoid while navigating the world. A barrel or a crate controlled by the physics system is a good example of an obstacle. While the obstacle is moving the agents do their best to avoid it, but once the obstacle becomes stationary it will carve a hole in the navmesh so that the agents can change their paths to steer around it, or if the stationary obstacle is blocking the path way, the agents can find a different route.

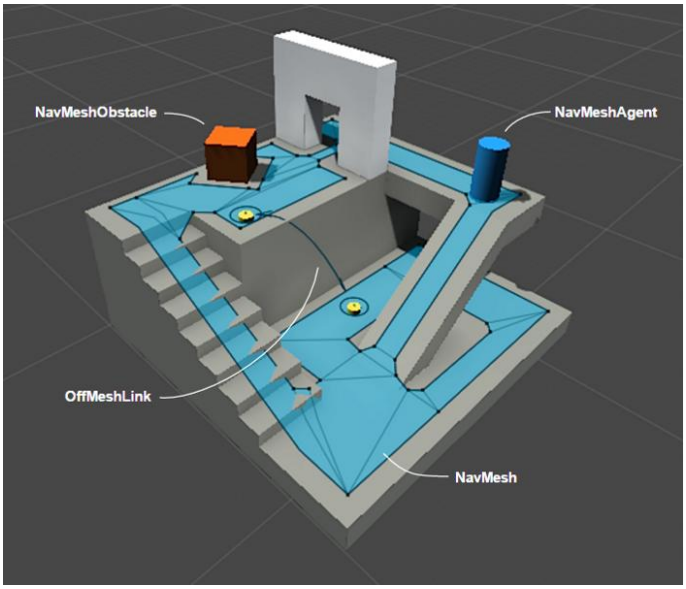

Fig 2. Component of Unity navigation tool [7]

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Jurnal Mantik
Volume 4 Number 1 May 2020, pp. 61-68 E-ISSN 2685-4236
https://iocscience.org/ejournal/index.php/mantik/index

B. Autodesk Stingray Navigation System

Autodesk Gameware Navigation is an AI middleware and NavMesh generation tools by Autodesk Inc. This middleware is successor of Autodesk Kynapse AI. Autodesk use this middleware in their game engine (Stingray) as Navigation tool[8]. According to the Autodesk Documentation, Autodesk Gameware Navigation uses Navmesh to represent what areas of the game world are navigable, just like in Unity navigation tool. Even the pathfinding employ the same algorithm, the A* algorithm.

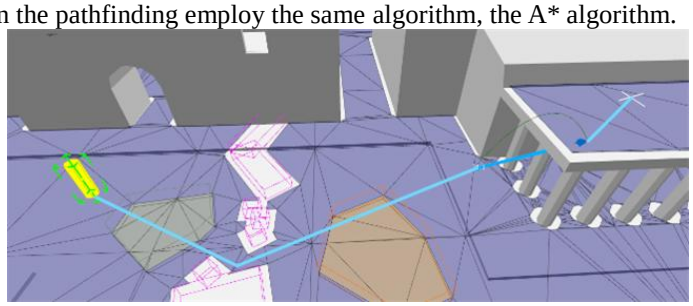

Fig 3. Pathfinding process of Autodesk Gameware Navigation [8]

However, the navigation process took different approach. In Autodesk Gameware Navigation, bot (Autodesk documentation term to calling an agent) ability to move toward it's navigation are calculated via livepath. Thus creating path following behaviour for it's bot to reach it's destination. The first step in the Autodesk Gameware Navigation system is to compute an initial path through the NavData, that connects a starting point (typically the current position of the bot) to a desired destination. As stated before, Autodesk Gameware Navigation uses A* algorithm for the pathfinding process. For example, in the figure 5[8] , the yellow Bot has planned a path to a destination on the rooftop. The path goes through the NavMesh on the ground, follows a NavGraph edge that connects the ground to the rooftop, then has a final segment on the rooftop from the end of the NavGraph edge to the destination.

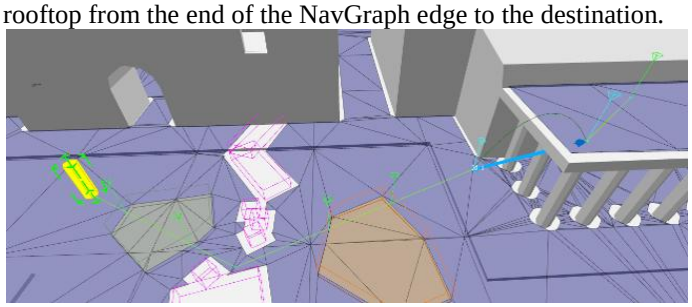

Fig 4. Livepath created for the bot to follow [8]

Once a path has successfully calculated, the next process called path following, analyzes the new path and builds a new structure called the LivePath from it. The LivePath is a dynamic structure that maintains several important items of contextual information about the path and the state of the Bot's path following process. Unlike path created from path finding which is static, livepath will change dynamicly to adjust the navigation of bot according to environment situation.

In each frame, the Bot calls a trajectory object to determine what movement it should make in the current frame. The Trajectory is responsible for evaluating the current conditions of the Bot and its path, and taking into account any other influences such as moving obstacles that may produce collisions.

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Jurnal Mantik
Volume 4 Number 1 May 2020, pp. 61-68 E-ISSN 2685-4236
https://iocscience.org/ejournal/index.php/mantik/index

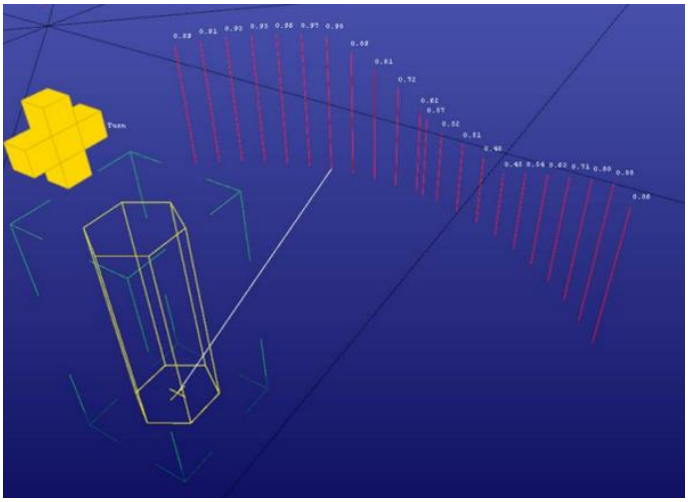

Fig 5. Collision avoidance process [8]

Bot navigation are expected to deal with collision avoidance. So the process must take account into dynamic changing to penalize the avoidance process. Based on these factors, and on a set of configuration parameters that control the character's locomotion, it produces a recommended velocity for the character to pursue in the current frame. There are three parameter are used to score velocity that will be chosen for avoiding obstacles.

1) Collision time ratio, collision time is a time limit that serves as a horizon, and compute a collision time with the collected colliders and navmesh boundaries for each sample. The ratio takes account into the minimal Time To Collision with the safety Distance
2) Desired velocity distance, the path following already produced a desired velocity. This velocity used to score the samples based on how far they are relative to this velocity.
3) Previous velocity distance, to stabilize even further, Bot can perform a similar computation relative to the previously computed velocity

3. Test and Simulation Scenario

In order to test the capability of both navigation system, we create application to simulate certain crowd condition in both game engine. This application created using prototype method[9]. As shown on figure 6 we use the same flow in both game engine to create exact application with only difference in navigation system used.

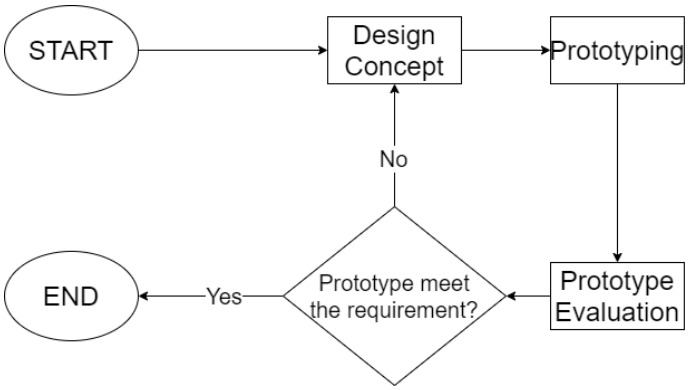

Fig 6. Prototype Method

1) Design Concept

Our simulation concept is based on two simulation skenario, which is crossroad[5] and narrow passage scenario[7]. We replicate both scenario environemt in both game engine with same scale. Agents variation and initial potitioning for each scenario are also replicated. The requirement for agents navigation already provided in the game engine itself, either Unity3D or Autodesk Stingray. Both of them

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Jurnal Mantik
Volume 4 Number 1 May 2020, pp. 61-68 E-ISSN 2685-4236
https://iocscience.org/ejournal/index.php/mantik/index

also provide other requirement such as Graphics dan Physics engine.

2) Prototyping

Application prototype designed based on crowd scenario. In order to create exact environment for both game engine, we create the 3D environment using third party program called Sketchup. The 3D environment model created then imported to both game engine in order to maintain consistent scaling. As shown in figure 7, the enviroment model will be integrated in each simulation created by each navigation tool at their respective game engine.

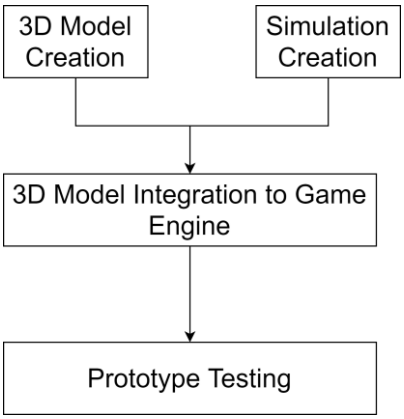

Fig 7. Prototyping Process

3) Prototype Evaluation

Prototype produced were evaluated to ensure the functionality of each application in Unity3D and Autodesk Stingray. prototype functionality must suitable for crowd simulation purposes. This application must be able to map the area/building where the movement of the crowd, also able to calculate the movement to be carried out by a group of crowds to get to the specified point, and able to display the results of crowd modeling in the form of movement.

If the prototype produced by both game engine meet the above requirements, then the prototype has been formalized as the final prototype. If there is functionality that has not yet been fulfilled, a revamping process will begin, starting from the concept design to the protoype-making section. This process is repeated until the prototype requirement are met. This applies as long as there is no limitation from the game engine.

4. Result and Discussion

The prototype built on both game engine were able to replicate the simulation scenario as shown in figure 8 and 9. Both program able to perform similarly for the simulation to occured. While there are noticeable difference in navigation performance as shown on figure 10 to 13, all agents in both scenario and both programs were able to complete the scenario.

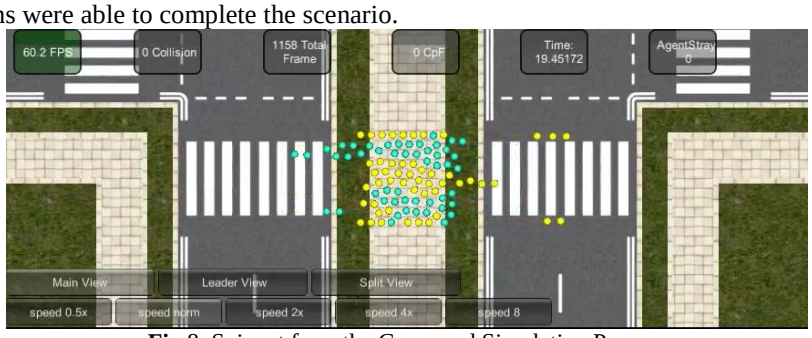

Fig 8. Snippet from the Crossroad Simulation Process

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Jurnal Mantik
Volume 4 Number 1 May 2020, pp. 61-68 E-ISSN 2685-4236
https://iocscience.org/ejournal/index.php/mantik/index

Fig 9. Snippet from the Narrowpassage Simulation Process

The data on figure 10 and 11 is a graph that shows collisions occured in crossroad scenario. As simulation runs with increased number of agents, the number of collision is increase as well. However Autodesk Gameware Navigation used in Autodesk Stingray produce less collisions that Unity Navigation. There are 16.49% less collisions occured in crossroad scenario for Autodesk Gameware Navigation compared to Unity Navigation, also 23.93% less for narrow passage scenario.

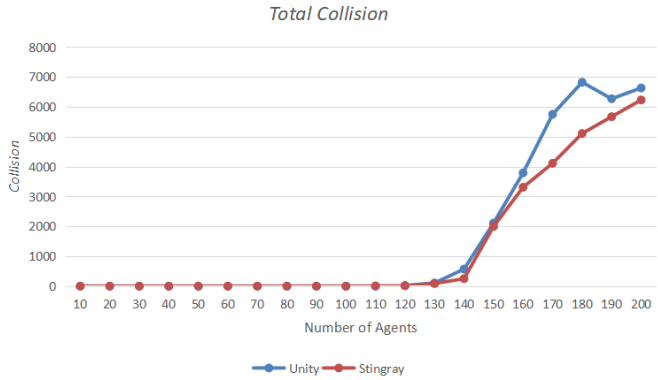

Fig 10. Collision Occured in Crossroad Scenario

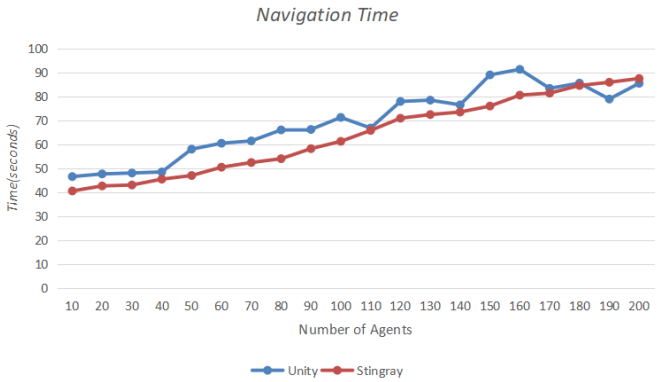

Fig 11. Collision Occured in Crossroad Scenario

Collision occured because the simulation agents fail to move safely as calculated by the navigation system. Due to the fact each navigation system use different approach in predict and avoid collision, such result is as expected. More collision lead to more time required for all agents to reach their destination thus end the scenario. Simulation in Unity3D produce more collision hence the simulation time. Compared to simulation in Autodesk Stingray as shown on figure 12 and 13, Unity3D need 8.2% more time to finish the crossroad scenario, and 17.83% for the narrowpassage scenario.

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Jurnal Mantik
Volume 4 Number 1 May 2020, pp. 61-68 E-ISSN 2685-4236
https://iocscience.org/ejournal/index.php/mantik/index

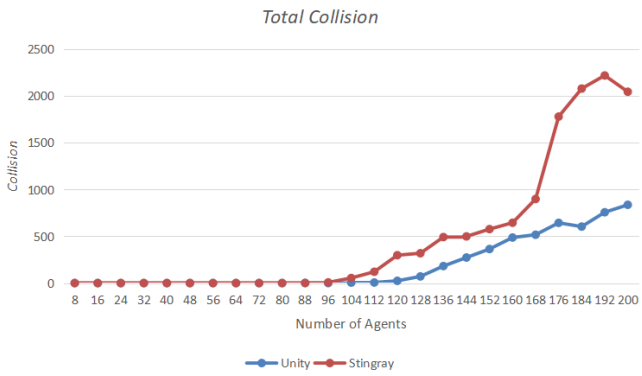

Fig 12. Collision Occured in Narrowpassage Scenario

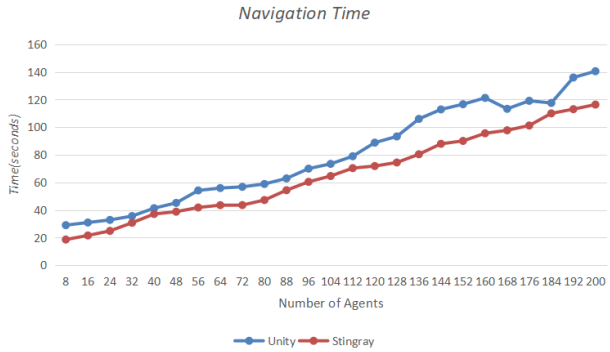

Fig 13. Collision Occured in Crossroad Scenario

5. Conclusion

Our research show how different navigation system used by two game engine may lead to different perfomance in scenario for crowd simulation. Autodesk Gameware navigation used in Autodesk Stingray perform better than Unity Navigation used in Unity3D. Agents navigation in Autodesk Stingray produce 16.49% less collisions and 8.2% less time to finish the crossroad scenario compared to Unity3D. As for the narrowpassage scenario, Autodesk Stingray produce 23.93% less collisions and 17.83% less time compared to Unity3D.

6. References

[1] S. H. M. Hashim and N. Mat Diah, "Game engine framework for Non-Programming background," 2015 IEEE Conference on Open Systems (ICOS), Bandar Melaka, 2015, pp. 18-21.
[2] B. Cowan and B. Kapralos, "A Survey of Frameworks and Game Engines for Serious Game Development," 2014 IEEE 14th International Conference on Advanced Learning Technologies, Athens, 2014, pp. 662-664.
[3] Z. Ali and M. Usman, "A framework for game engine selection for gamification and serious games," 2016 Future Technologies Conference (FTC), San Francisco, CA, 2016, pp. 1199-1207.
[4] P. Petridis, I. Dunwell, S. de Freitas and D. Panzoli, "An Engine Selection Methodology for High Fidelity Serious Games," 2010 Second International Conference on Games and Virtual Worlds for Serious Applications, Braga, 2010, pp. 27-34.
[5] J. van den Berg, S. Pati, J. Sewall, D. Manocha, and M. Lin, Interactive Navigation Of Multiple Agents in Crowded Environment. USA: University of North California, 2008.
[6] J. van den Berg, M. Lin, and D. Manocha, Reciprocal Velocity Obstacles For Real-time Multiagent Navigation. USA: University of North California, 2008.
[7] Unity Technologies, Inner Workings of the Navigation System, Unity3D Online Documentation, accessed 8 January 2020, https://docs.unity3d.com/Manual/Navigation.html.

Jurnal Mantik
Volume 4 Number 1 May 2020, pp. 61-68 E-ISSN 2685-4236
https://iocscience.org/ejournal/index.php/mantik/index

[8] Autodesk Inc, Autodesk Gameware Navigation, Autodesk Online Documentation, accessed 28 January 2020, http://help.autodesk.com/view/GWNAV/2016/ENU/
[9] Banoriya D., Purohit R., Dwivedi R.K. (2015). Modern Trends in Rapid Prototyping for Biomedical Applications. Materials Today: Proceedings, 2(4-5): 3409-3418.

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.