UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

FACULTAT D'INFORMÀTICA DE BARCELONA

COMPUTER SCIENCE

IMPROVEMENT OF THE LOCAL MOVEMENT ALGORITHMS TO SIMULATE MORE REALISTIC CROWDS

Author: José Martín Nogueira
Director: Alejandro Ríos Jerez
Co-director: Nuria Pelechano Gómez

April 9, 2018

I want to thank everyone that supported me during the development of this project. To my friends that help me carry this endeavour without knowing it, to my family that had to deal with me when I got stuck on the project, to my grandpa that kept me fighting when I needed the most. Finally, I want to dedicate this project to my grandma, she was my lighthouse in my darkest night.

"All that is gold does not glitter,
Not all those who wander are lost;
The old that is strong does not wither,
Deep roots are not reached by the frost.

From the ashes a fire shall be woken,
A light from the shadows shall spring;
Renewed shall be blade that was broken,
The crownless again shall be king"

- J.R.R. Tolkien, The Fellowship of the Ring

Index

Figures' Index

Tables' Index

0 Abstract

In this project we will try to improve the behaviour of Unity's[1] navigation system[2] in order to obtain a more realistic crowd movement.

Throughout the development of this project we will observe the difficulties that developers face when the try to accomplish such endeavour due to the lack of data and functions provided to them. Also, the possible solutions to avoid such nonces.

Finally, we will analyse a leading technology in comparison to the traditional at a superficial level but enough to see why it can be considered revolutionary.

1 Introduction

1.1 Project formulation

In a moment in time where programming has become a necessity for almost every project, programmers are in need for more tools to make their redundant coding faster, so they can focus on more important or difficult aspects of their work.

The video game industry has become one of the most important inside the entertainment industry and the most profitable of them. These later years, a lot of games were developed by small companies, group or even a single person all alone. So these people tend to use free platforms to work, and they need tools and guidance in some aspects during the development of their project due to their limited resources and knowledge.

Unity is a game engine which is used to develop 3D and 2D video games and simulations. It is widely used among students and people who want to build their own game. This game engine provides a tool to create navigation paths for the characters to follow, but when there are several characters or objects moving towards the same destinations, their behaviour is not very realistic.

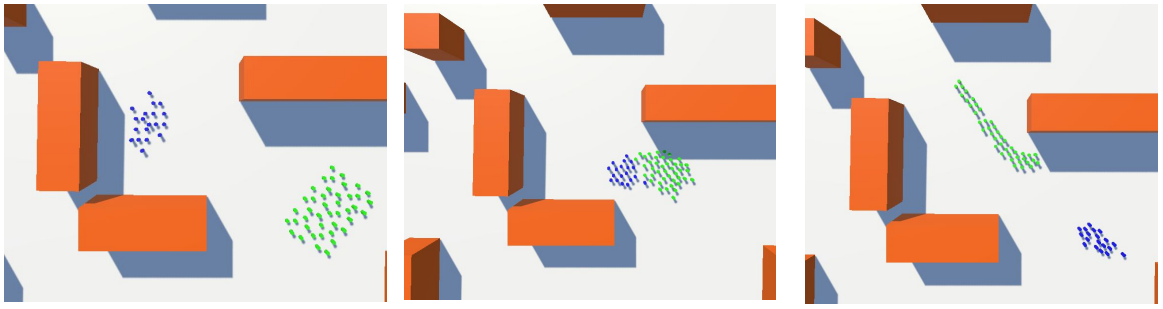

Figure 1: Unrealistic movement

As we can see in the figure above, the current implementation as an unrealistic behaviour, the crowd ends up forming a line because all of them try to go through the same point. What we want to achieve with this project is to improve the current algorithms and avoid behaviours like this.

1.2 Personal motivation

Since I was a child I always had curiosity about how video games were developed and when I started this degree, I had the chance to take a glance on it through some of the courses I did. But it was during my Erasmus that I realized that maybe it is time to get more into it and see if this is the right field of work for me, so I took a course about video game development. I had such a great time creating my own video game that when I came back I started to ask professors if the had a video game related project available to be my thesis. Alejandro Ríos

and Nuria Pelechano answered me and explained to me the project. I accepted it the following reasons:

- **It is related to the field I would like to work at.** It can give me experience on the tools I might use in the future.
- **It covers part of my curiosity and interest.** I always found the pathfinding algorithms interesting.
- **It can be helpful for other people project.** Having as a final goal to create a tool for developers, people could use it for their own projects.

1.3 State-of-the-art

We can trace this theme until 1970 when John Horton Conway [3] created the game of life [4]. This “game” is interesting due to the fact that we can observe complex patterns by introducing simple instructions and rules. What we get from Horton’s work is the use of graphic computation to simulate different patterns and interaction between them.

Nowadays, this is an interesting field for several science disciplines such as psychology and medicine due to their interest in human interaction between individuals and crowds to analyse the spread of diseases [5] or migrations [6]. It is also useful to study traffic jams.

2 Context

In this section the context of the project will be treated. It will be divided in 2 sections: stakeholders and means and technologies.

2.1 Stakeholders

- **Developer:** is the main responsible and also the main beneficiary of this project, due to it will be his thesis.
- **Director and Co-director:** they have the responsibility of orientating the developer, give him advice. This is very important in order to help the developer complete his work.
- **Direct beneficiaries:** all developers that may need this project as a tool for their work or personal interests. In this group we can differentiate two sub-groups between amateur and professional developers. The first could be formed by students and people who want to create their own game, so this group can take advantage of the project due to their lack of knowledge and experience. The second group are people that work on the game industry, from smaller to bigger companies (especially smaller companies) and also standalone developers, this group can use this project to make easier and faster their projects.

- **Indirect beneficiaries:** People who play video games created by the direct beneficiaries.

2.2 Means and Technologies

For the development, execution and testing of this project is only needed a PC (personal computer). The technologies needed and the reasons to choose them are:

- **Unity:** The project is focus on this IDE (Integrated Development Environment) so it is essential.
- **C#**[7]**:** It is a multi-paradigm programming language (strong typing, imperative, declarative, functional, generic, object-oriented and component-oriented). It was developed by Microsoft. It will be the programming language of the project because Unity use it as its primary scripting language.

- **Visual Studio** [8]**:** It is another IDE that supports multiple programming languages. It has several features but the ones that interest us are:
    - The debugging tool: Visual Studio can be linked with Unity in order to debug the code.
    - Guide and auto-complete: It supports C# so whenever we use a function from a C# library we have a description of it without searching for it in an Internet browser.

  Also, the developer is familiarized with it.

- **GitLab**[9]**:** it is a web-based Git and Internet hosting service. It will be used to keep track of the code and version management.

- **Scrumme**[10]**:** it is a web that allow us to use the scrum methodology (explained in section 3.2).

3 Scope of the project

First of all, I will study how Unity navigation tool works by creating a simple scenario with an obstacle, a goal and several avatars. This way I will have a better understanding of the algorithm currently implemented and I will get familiarized with the IDEs and the language.

After that, the development part will start with the improvement of the current algorithms in order to get a more realistic crowd movement. Once we get the desired results, the psychological behaviours for the avatars will be implemented.

3.1 Project Objectives

The main objectives of this project are:

- Improve the current navigation algorithms implemented in Unity to achieve a more realistic crowd movement.
- Give the avatars (known as agents in Unity) the possibility of having psychological behaviour (secondary goal).
- Upload the project as a tool at the Unity store (secondary goal).

3.2 Methodology

The methodology used for this project will be Scrum[9]. It is an agile methodology that follows an iterative and incremental model. Also, provides flexibility, fast development and is result-oriented. Even though it is a team-oriented methodology, it can be applied for individuals.

Scrum uses sprints to keep the project on schedule, receive feedback and keep a constant work. A meeting every two weeks will be arranged in order to keep the process of the methodology.

In every meeting we will review the work done in the previous sprint as a form of validation method due to the impossibility of doing it automatically.

4 Project planning

4.1 Planning and scheduling

The estimated time for this project is 4 months long. It started in mid February and will end around mid June 2018. Even if a planning is done, it will probably be affected and modified due to the agile methodology used

4.2 Task description

- **GEP course (~50 hours):**

  During this course I will study the implications of the project, previous works, the real costs of a project of these dimensions and the sustainability of it through the next deliverables:
  - **Deliverable 1:** Formulation of the project, state-of-the-art and scope of the project.
  - **Deliverable 2:** Temporal planning, task description, estimation and Gantt chart.
  - **Deliverable 3:** Budget and sustainability matrix.
  - **Deliverable 4:** Draft of the presentation.
  - **Deliverable 5:** Justification of the academic competences.
  - **Deliverable 6:** Final delivery.

- **Familiarization with Unity (~40 hours)**

  During this task I will get familiarized with the IDEs used in this project in order to know how to use them properly and study how the current navigation algorithms work. This is an important task because it will make easier the following tasks.

  The resources spent in this task will be Unity, Visual Studio and GitLab (all of them free) and the amortized costs of personal computer, electricity and human cost.

- **Implementation of the improvements (~200 hours)**

  This is probably the most important task of the project, the whole of it is based on this so the success of this task is crucial to finish it successfully. It will last longer than any other task for this reason.

  Here the improvements of the current algorithms will take place and any other implementation of these if it is needed. It is hard to divide this task into sub-task due to we don't really know which problems we will have to face.

  Again the resources spent here will be Unity, Visual Studio and GitLab and the amortized costs of the personal computer, electricity and human cost.

- **Testing the improvements (~40 hours)**

  In this task the testing of the previous task will take place and the correction of any bug or wrong behaviour too. I don't expect much time spent here due to the fact that the errors that may appear are in my opinion easily spotted but it can take longer than expected to correct them.

- **Implementation of the psychological behaviours (~150 hours)**

  This task fulfils a secondary goal of the project so the estimated time for this is calculated based on the 3rd task described in this section, if that task takes longer this task will have less. In this task some psychological behaviours for the avatars or agents will be implemented. I will use the thesis of a colleague who worked with the director and co-director of this project (Alejandro Martínez Zamorano) as a base for my implementation.

- **Final testing (~40 hours)**

  The final testing and corrections will be made in this task. Again, the errors may be easy to spot but harder to correct than the previous testing task due to the fact that the interactions with the two implementation task can produce more errors than expected.

4.3 Estimated time

Task	Estimated Hours
GEP	50
Familiarization with Unity	40
Implementation of the improvements	200
Testing the improvements	40
Implementation psychological behaviour	150
FInal testing	40
Total	**520h**

Table 1: Estimated hours

4.4 Gantt Chart

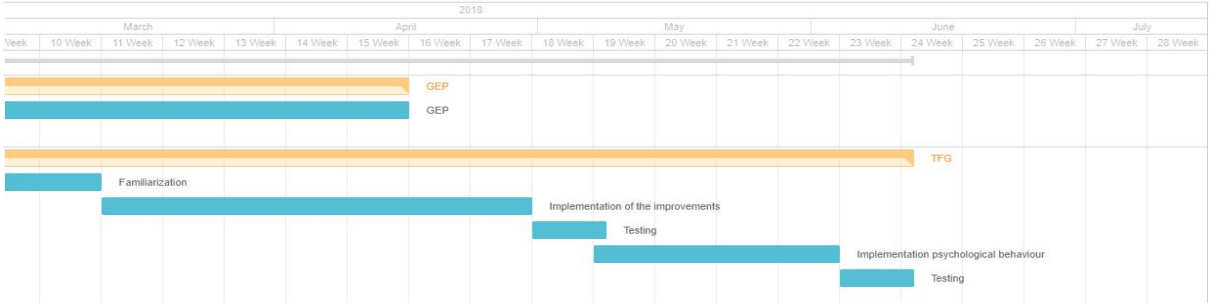

Figure 2: Gantt chart

4.5 Alternatives and action plan

The only thing that will change on each task it will be the human cost due to unpredictable problems that may appear. These problems can be:

- **New implementations:** So if the current algorithms do not provide us with the data we need I will have to implement them in a way that they are useful for us.
- **Unexpected interactions:** The psychological behaviours may interact with the navigation improvements in a way we don't expect generating problems.

However, this two problems can be solved with the help of the director and co-director of this project that far way more experience than me.

Other general problems that may occur are easily avoid by using technologies like GitLab that, for example, makes more difficult to lose the project.

5 Budget

An estimation of the costs will be presented calculating an approximation of the hardware, software and human costs and the amortizations too.

5.1 Cost identification

The costs identified are the following:

- **Human resources budget:**

  This project will be developed by only one person, the estimated time is 520 hours that is more than the expected for a final degree project. Only two roles are needed to complete this project, a Project Manager and a Software engineer.

Role	Estimated Hours	Aprox price/hour	Total estimated cost
Project Manager	60h	50€/h	3000€
Software engineer	460h	20€/h	9600€
Total			**12600€**

Table 2: Role costs

The amount of hours of each role in each task are approximately the following

Task	Hours	Soft. Eng.(h)	P.M.(h)
GEP	50	10	40
Familiarization	40	38	2
Implementation of the improvements	200	193	7
Testing the improvements	40	37	3
Implementation psychological behaviours	150	145	5
Final Testing	40	37	3
Total	**520h**	**460h**	**60h**

Table 3: Work distribution

- **Hardware resources budget:**

Product	Price	Units	Useful life	Total estimated amortization
Desktop computer	900€	1	5 years	60€
Asus FF52L	400€	1	4 years	33€
Total	**1300€**			**39€**

Table 4: Hardware cost

- **Software resources budget:**

Product	Price	Units	Useful life	Total estimated amortization
Unity	0€	1	-	0€
Visual Studio	0€	1	-	0€
GitLab	0€	1	-	0€
Google Drive	0€	1	-	0€
Total	**0€**			**0€**

Table 5: Software cost

- **Indirect costs:**
  We will have to take into consideration costs that are not related to the project like electricity and internet.

Product	Usage	Price	Months	Cost
Electricity	10%	50€/month	4	20€
ADSL	25%	36€/month	4	36€
Total				**56€**

Table 6: Indirect cost

Taking into account all the costs previously estimated and adding a 15% for contingencies the total expense is the following:

Expense name	Cost
Human	12600€
Hardware	39€
Software	0€
Indirect	56€
Subtotal	**12695€**
Contingency	1904,25€
Total	**14599,25€**

Table 7: Total cost

5.2 Budget Control

To control the possible budget deviations, an update will be made at the end of each task with the amount of hours dedicated so the real cost can be calculated, although the indirect costs are not possible to calculate precisely. The formulas used to calculate this are:

Cost deviation = (Estimated cost - Real Cost) * Real Hours
Consumption deviation = (Estimated Hours - Real Hours) * Estimated cost

At the end of each task, these formulas will be applied and if a high deviation is appreciated, the reasons of it will be analysed and a solution will be searched for the next ones.

6 Sustainability

6.1 Sustainability matrix

In this section it will be analysed the impact of this project from an economic, environmental and social dimensions to evaluate its sustainability. From this analysis we get the following matrix:

	Project Development	**Exploitation**	**Risks**
Environmental	Consumption Design (0:10)	Ecological footprint (0:20)	Environmental risks (-20:0)
	7	14	-1
Economic	Project Bill (0:10)	Viability plan (0:20)	Economic risks (-20:0)
	7	17	0
Social	Personal Impact	Social impact (0:20)	Social risks (-20:0)
	10	8	0
Sustainability range	**24**	**39**	**-1**
	62/90		

Table 8: Sustainability matrix

We got 62/90 points , it is not really high mark due to the fact that this project barely has a social impact.

6.2 Economic dimension

The non-human requirements for this project are very low, only a pc is required and the rest is the electricity bill and the internet. All the software used is free. Also, the human cost is low taking into account that the project is developed by only one person. The extra hours needed to complete it, if needed, are covered by the 15% contingency.

The project is Open Source so anyone who want to use it can use it for free. Probably there will be no updates so the cost of them is ignored.

6.3 Social dimension

This project belongs to the video game field so the social benefits of it are low. Some can argue that the current state of the industry and a huge part of the games released can be put into the culture group next to the books and films but since this is a debate issue right now for this project I will consider them as they have been considered since they appeared, just entertainment.

As is it mentioned before, this project is Open Source and anyone can use it, extend it or modify it.

6.4 Environmental dimension

Being a computer science project it has only one footprint, the electricity used. Although, in exchange we reduce the paper used, also because I have a whiteboard at home to work and all the paper used is recycled. Moreover, the code will be reusable and modularized so it will be easier for everyone to manipulate it.

7 Justification of the project speciality

7.1 Subjects useful for the project

The most relevant are the following:

- **Programming Languages:** This subject gives some insights about different programming languages and paradigms that help the students to adapt to new languages faster. So this helps me to work easier with C# which is a language I'm no used to it because I barely used it
- **Algorithmics:** It gives an overall view of the most important algorithms, some of them probably will be used for this project. It is also relevant for the efficiency analysis helping me to see if the algorithms implemented are good enough.
- **Computational Geometry (GEOC):** Algorithms like triangulation and pathfinding are taught in this subject and they are the base of this project.

7.2 Speciality of the project

All the knowledge needed to carry on this project are taught in this speciality or in subjects related to it. It deals with complex algorithms and efficiency which are the distinctive of this speciality.

7.3 Academic objectives

The academic objectives are labeled with: None, A little bit, Enough and In depth. All objectives with the label None are ignored and will not be presented here:

- **CCO1.1:** To evaluate the computational complexity of a problem, know the algorithmic strategies which can solve it and recommend, develop and implement the solution which guarantees the best performance according to the established requirements. **[In depth]**.
- **CCO1.3:** To define, evaluate and select platforms to develop and produce hardware and software for developing computer applications and services of different complexities.**[Enough].**
- **CCO2.3:** To develop and evaluate interactive systems and systems that show complex information, and its application to solve person-computer interaction problems. **[A little bit]**.
- **CCO2.6:** To design and implement graphic applications, virtual reality, augmented reality and video games.**[In depth]**.
- **CCO3.1:** To implement critical code following criteria like execution time, efficiency and security. **[In depth]**.

Once completed the project, they will be accomplished in the following way:

- **CCO1.1:** This objective will be covered through the implementation of a solution with the best performance possible.
- **CCO1.3:** By choosing software during the development such as Visual Studio and developing a tool for other developers.
- **CCO2.3:** The objective of this project is to recreate a crowd movement that resembles the reality.
- **CCO2.6:** The goals of this project are directly related with video games.
- **CCO3.1:** A crowd of people contains a huge amount of avatars which each of them has their own path, so to make this possible we will have to implement an efficient code.

8 Unity

8.1 Introduction

Unity is a game engine that provides access to the development of games (2D and 3D) regardless the experience of the developer. It has a good learning curve, allowing people to learn while they develop their project. Also, professional developers can use it to its full potential.

In Unity, you work with Scenes, each element of the scene is a GameObject and each of them have their own components that describes them, such us the Transform, that provides the position, rotation and scale of the GameObject, and the Scripts attached to it. All these components can be seen at the Inspector tab of the GameObject. There is also another tab that we will focus later, the Navigation tab.

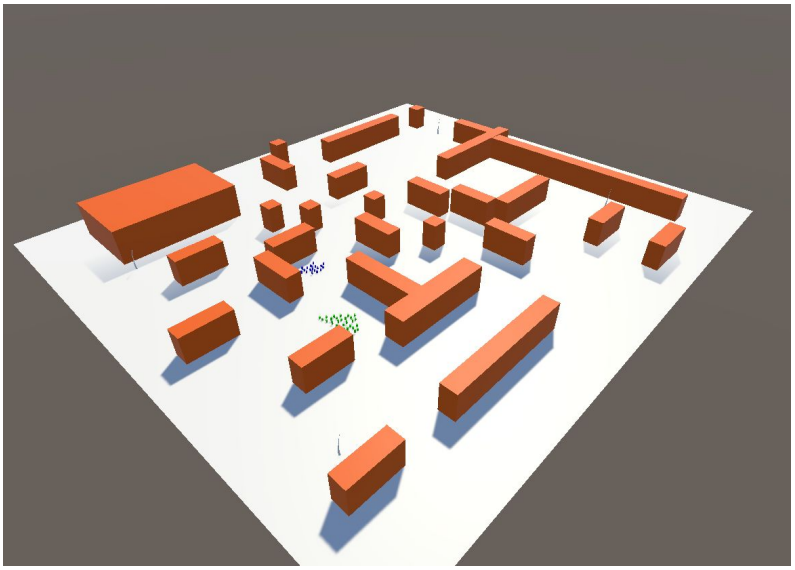

Figure 3: Scene

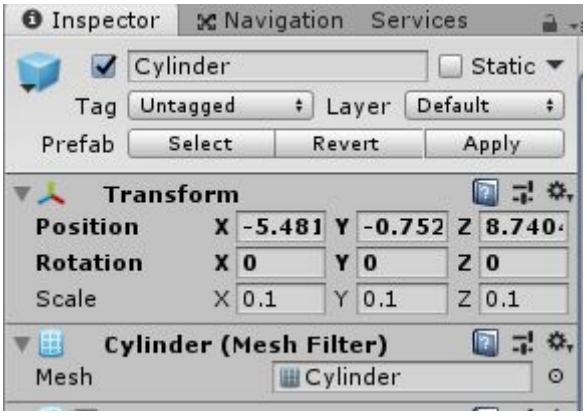

Figure 4: GameObject tabs and Transform

The geometric models that unity provides is what we are going to use to build our scene of the project. Cylinders will act as persons, trees as goal to reach and prisms as walls.

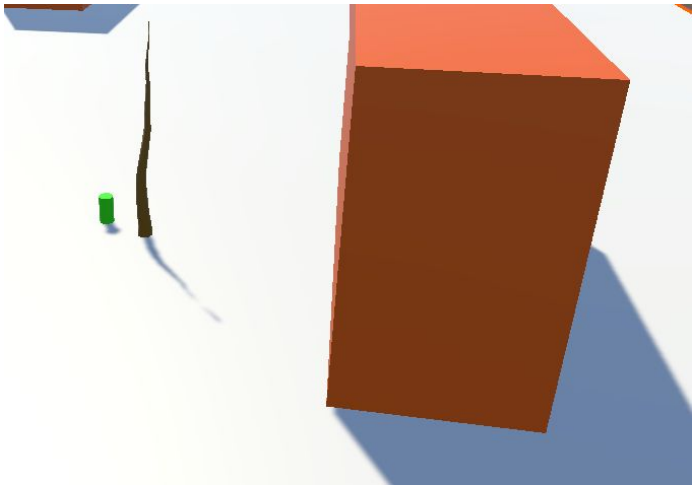

Figure 5: Cylinder, tree, prism.

An important component to the project is the Collider. It is a box, capsule, sphere or a more complex and precise structure that can cover a GameObject and give it certain properties. It is used as its name says to perform collisions and give the GameObject a collision surface.

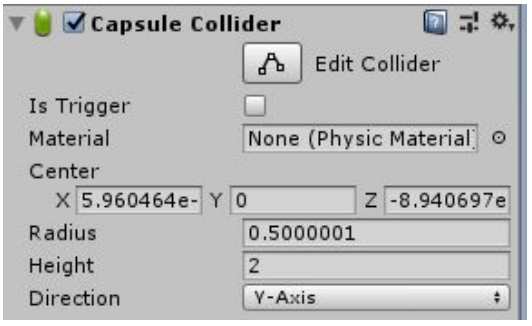

Figure 6: Collider component

Another concept to understand this project is the mesh. A mesh is a data structure that contains triangles that create the impression of a solid object. A triangle is defined by three integers. For example the first triangle is represented by 0, 1 and 2 and in these positions of the array there is an integer linked to the vertex array that contains the coordinates. This way you don't have multiple repetitions of a vertex all over the place. In our case, we will use a kind of mesh that covers the ground.

The most important feature of Unity for this project is Navigation. It allows us to build a Navigation Mesh.

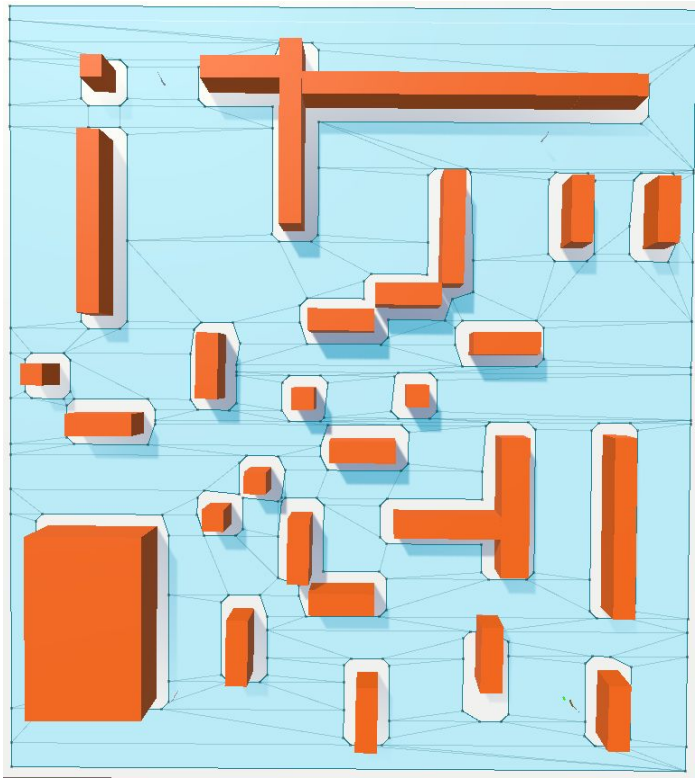

Figure 7: Navigation Mesh

The blue zone of the picture above is the walkable zone were the cylinders will move through and it is the Navigation mesh. Adding the Navigation Agent component to a cylinder and assigning it a goal with a script will make it move towards the goal. It is a really simple process and only requires two lines of code to work. The Navigation Agent component contains several values such as its acceleration.

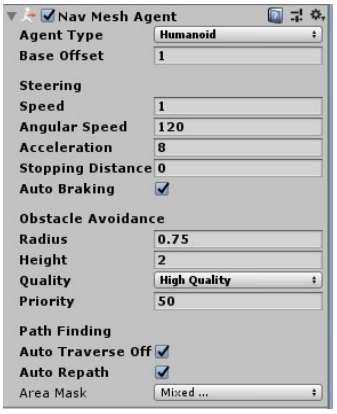

Figure 8: Navigation Agent Component

It has more features but they won't be used for this particular project.

8.2 Technologies

The scripts in Unity are used to give the GameObjects behaviours. These are coded using C#. Unity provide us with a really good API that makes programming way easier due to the amount of libraries it has. There are two functions worth mentioning: Start and FixedUpdate. All the functions called in the Start function will run one time just at the start of the simulation. Those called in FixedUpdate will run once, none or several times per frame, depending on how many frames per second are fixed in the time settings making it better, in our case, than the Update function that runs every frame.

ECS[10](Entity Component System) is a new way of programming in Unity that combined with the new C# Job System[12] allows you to use the new multithreaded systems that run on a variety of hardware and take full advantage of great performance by default. This way you can create games with a lot of units and complex simulations.

ECS has shown a great improvement to Unity allowing us to create games and simulations with a tremendous amount of characters or units with an incredible performance. In exchange, so far, it is really complex to code. Knowing this Unity is developing Hybrid ECS that merges the two worlds, ECS efficiency and the traditional Unity coding simpleness.

9 Development: problems faced and solutions

Through the development of this project I was dealing with health problems that didn't let me work as I should so it took way more time than needed to be completed. For that reason the end of the project was delayed.

In this section all the problems and solutions will be explained in detail.

9.1 Identification of the problem

The project started by creating a Scene in Unity, putting a prism in the middle of the scene acting as an obstacle, a tree as the goal and a few cylinders as persons. In the Navigation tab we set to true the static checkbox of the prism and the plane so when we use the bake button it generates the navigation mesh.

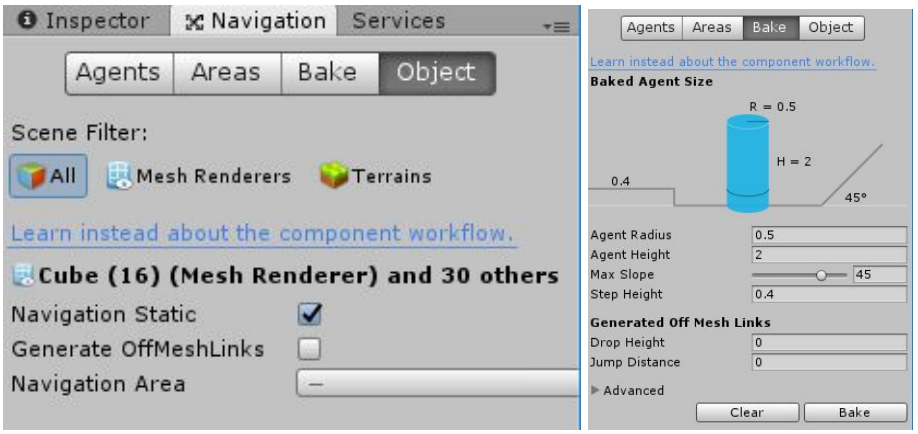

FIgure 9: Building a Navigation mesh

The objective of this was to check how they move using the navigation system that Unity provides.

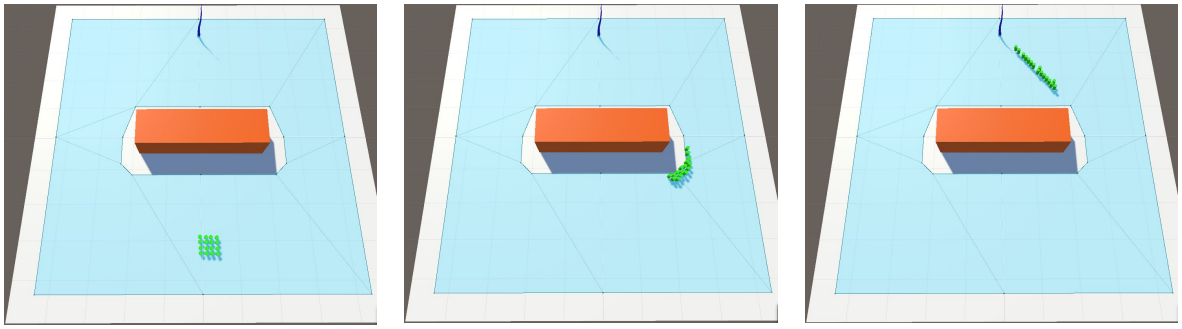

Figure 10: Identification of the problem

As we can see in the figure above all the characters try to go through the same vertex and to achieve that they form a line.

9.2 Problem: All through one vertex

To solve this may seem easy, find the triangle edge and make the characters cross it along it. The problem with this is that the only data that Unity provide us about the path are the vertices that will be crossed, so we can't know the other vertex that forms the edge. A vertex is shared between 2 or more triangles. To solve this we decided to implement our own A*[13] algorithm, so we can take any data from the path so instead of having only a few vertices we would have every triangle that the cylinders had to cross.

9.3 Solution: A*

The A* algorithm is widely used in path finding and graph traversal, which consist in finding the path between points called nodes. In order to pick the nodes a heuristic is needed to discern between all the possible options.

Implementing this algorithm is quite easy, the only difficulty is to make a good heuristic due to that the triangles have different shapes and dimensions, so we have different points from a triangle that can act as a node. So to select a good node we calculate the distance between the all the options (the neighbours of the current treated triangle) and the goal, then we take the closest. These points are the 3 vertex, the 3 middle points of the edges and the barycentre. This way we can have a more or less precise A*.

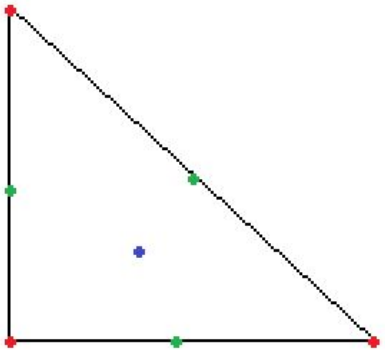

Figure 11: Possible nodes

So at the end, A* will return a list of triangles that will be the path for a character. These triangles have an identification, so we can search for their vertex in a data structure that contains all the vertex of the navigation mesh without repetitions.

With this algorithm we never intended to have the shortest path, we want to emulate the human behaviour, and we don't always take the shortest path because we are not able to calculate it perfectly.

This solves the lack of data given to us but now we have to solve the vertex problem. This algorithm will be used, with other calculations, in the Start function which is activated right at the start of the simulation. This way, it won't affect performance while characters are moving towards the goal.

9.4 Solution: Furthest vertex and orthographic projection

The solution we thought to solve the one vertex problem is to make the cylinders pass through the orthographic projection on the edge. This way, some of them will cross through different points of the edge. If the point to cross does not belong to the edge, they will move towards the furthest vertex of the pair. Then, while they move, we will check if at some point, the orthographic projection returns a valid point.

The idea behind this was to make them stay away from the walls, and for that we need a reference point away from the walls. The most reliable point to do it is the furthest vertex of the edge which is going to be crossed. In order to avoid the full movement towards that point, we interrupt when the orthographic projection returns a valid point.

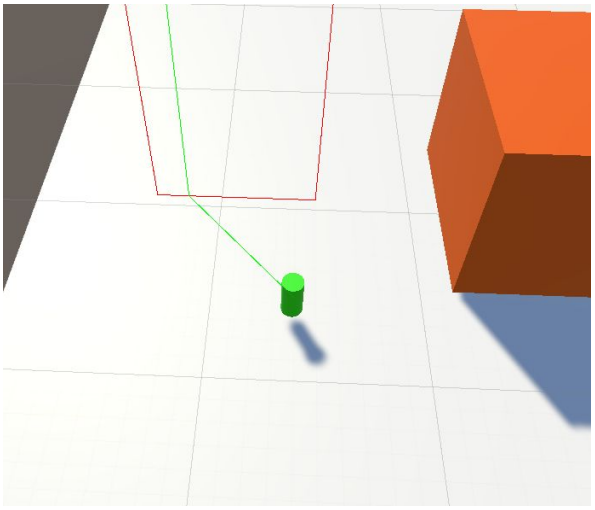

Figure 12: Furthest vertex with shrink edge

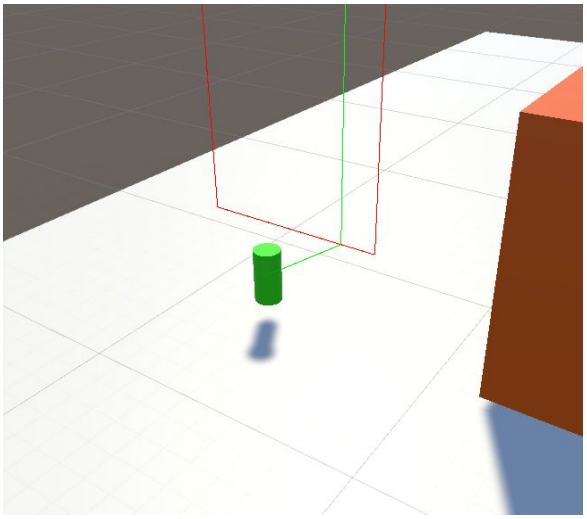

Figure 13: Orthographic projection

All this conditions and decisions take place in the FixedUpdate.

Moreover, to avoid even more the walls, we reduced the length of the edges by a 20% on each side so the characters can only cross the 60% of it that is in the centre of it. Making this, the cylinders will occupy more centric positions along the path.

9.5 Problem: Robotic movement

The problem of using the orthographic projection with every edge we cross is that you get as a result a movement behaviour closer to a robot than a person. The triangles of the mesh are very different from each other and even if they were all the same we will still have the same problem, they won't move in a straight line.

9.6 Solution: See ahead

To solve the robotic moment, we implemented a loop in the code where the next point in the path was assigned to the character so it can move. If this loop we check if there is a vertex of the path visible form the current position of the cylinder by using a RayCast function. This function traces a ray between two points and if it hits anything will return true. It was also added that whenever they see the goal they walk straight to it.

So the movement behaviour implemented before stays the same but with this solution added to it.

This caused a little issue that remains unsolved, again because of the lack of data and functions. The RayCast does not detect the not walkable path between the navigation mesh and the obstacles, so it returns false when it should return true because it "saw" a point that shouldn't be visible because it goes through a not walkable zone. This behaviour was sort out in most cases by shrinking more the edges.

9.7 Problem: Steering behaviour

So we advanced new problems appeared. Now the characters move more or less properly through the map, but they do not interact and stumble with each other. To correct this the implementation of the steering behaviour was needed, so they can avoid collisions. On paper implementing it sounds easy but in fact it is not. A lot of forces calculations must be taken into account, ray casting, threat evaluations, etc. The implementation would take long and the tuning up as well. This part could be, and in fact it is, a whole thesis.

9.8 Solution and Problem: Navigation Agent Steering behaviour

The lack of time and skills was solved by using the navigation agent by changing its goal every time we reach a point. This way we don't have to code the steering behaviour but it comes with a price. By using the NavAgent, each time we change the goal it runs an A*, turning the whole project inefficient. This was the only way it could be done because Unity does not allow developers to create a path and give it to an agent. The path array can be access through a read only function so it can't be modified in any way, it was a dead end.

Even though the solution we got at this point is an inefficient one, we reach one of our goals, to get a better movement behaviour than Unity's.

9.9 ECS

When we heard about ECS we thought that it could be a great addition to the project and given the fact that we got an inefficient solution it seemed a great idea to include it. When the developer was trying to use ECS with Unity's navigation system realized that it was way more difficult than it seemed and failed to implement it. It was like start all over again with a completely new programming language and given the time left it was impossible. Also, the Hybrid ECS was at the time not in a usable stage of the development and was only advised to use for experimentation.

Anyway, a project[14] found in ECS very similar to what we had in mind was found, so we could do a performance comparison between Unity's navigation system without ECS and with it. The results were astonishing, without ECS we could hold around 8000 cylinders moving around 35 fps and decreasing each time we add more. With ECS at 10100 we got an average of 150 fps, at 20100 was of 75 fps and it was at 30100 that we got less fps than without ECS.

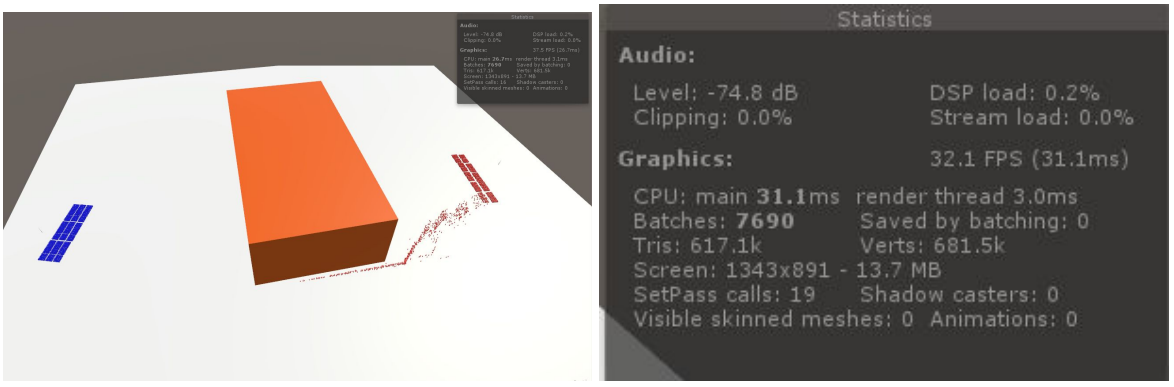

Figure 14: Without ECS stats 8000 units approx.

Figure 15: ECS stats 10100 units

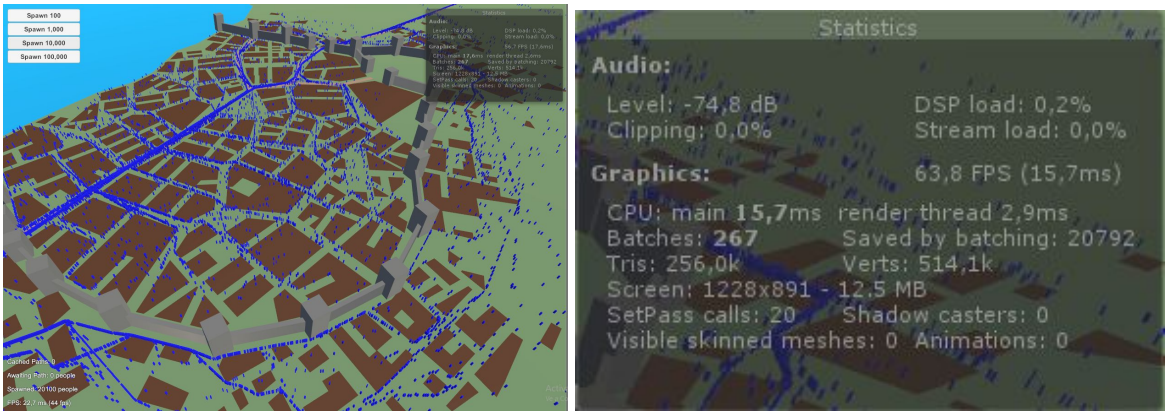

Figure 16: ECS stats 20100 units

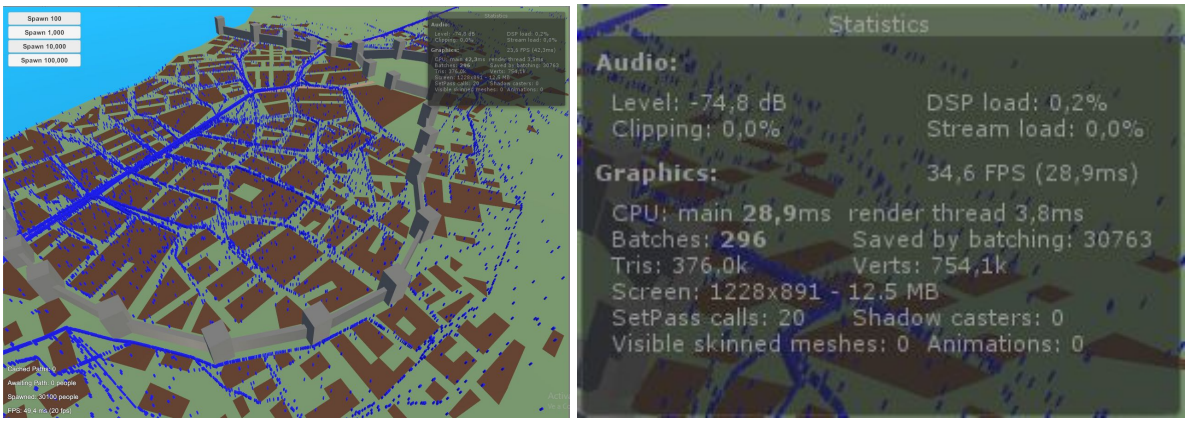

Figure 17:: ECS stats 30100 units

Taking into account that the scene and the path was more complex without ECS we could see the tremendous potential of ECS in projects of this kind, and we could say with little doubt that it could be the future of the game industry.

9.10 Other functionalities implemented

- A function to find the triangle where the goal or the cylinders are in order to run the A* algorithm. The algorithm used is the Barycentric Technique[15] that is basically check if the point is on the same side of every edge, if it is, then the point is inside the triangle.
- A function to get the next two vertex of the edge that will be crossed by the characters.

9.11 Final Result

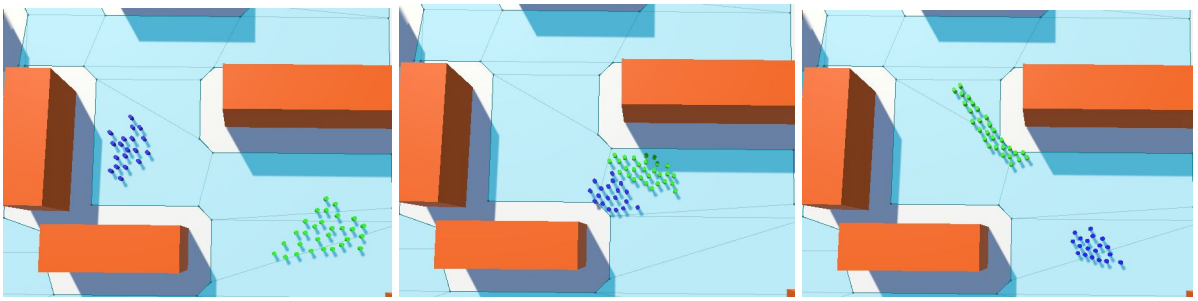

Figure 18: Unity's navigation system

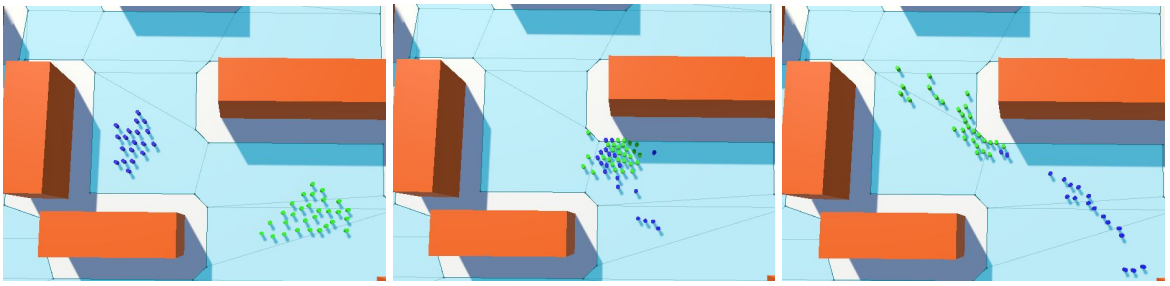

Figure 19: Final solution

As we can see, there is an improvement on the behaviour, they are not trying to go through the same point, instead they spread out and cover more surface. When two crowds of people "collide" they do not avoid each other by stepping aside, they usually merge. Also the do not wait to cross the same point and spread along the edge.

10 Conclusion

In the current state of Unity and its API I think this project can't go further. The objective of getting a better movement behaviour has been reached but at a high cost and the essence of it was lost. What I mean with this is that from the moment we had to implement a new A* we went from "improving the current algorithm in Unity" to rebuild the navigation system.

Unity is a great platform but it has its issues. From the navigation system perspective, they give us, the developers, very little to work, to customize due to the read only functions. As mentioned in previous sections, the lack of data and API functions give us very little things to work with. For example the only function to detect not walkable zones is a function that tells you the distance and the point to the closest edge that limits the walkable zone, this is useless to our work because most of the time, the closest point is not in the movement direction. Again, there is nothing that show us which edge limits the walkable zone or the other way around. Also, we can't access to their code which is a reasonable thing but it causes us more headaches when the only thing you want is to improve the current code.

Even though we reached on of our goals, this project leaves me a bitter taste, I like to face problems like those I found during this work and solved them trying to find multiple solutions but given the circumstances I think I couldn't do more. I believe that the best solution of this was to remove the "improving" part of the project and rebuild the whole thing but with that amount of work it will fill several theses.

On the other hand, I believe that I've learnt more from all these problems I faced than what I could have learnt if everything went as intended.

11 Future work

As mentioned in the conclusions, the future work of this project goes through rebuilding the whole navigation system from start using ECS. As we could see, the performance of ECS it so good that any project of this kind should be considered to be programmed with it.

Putting aside ECS, the navigation mesh should be rebuilt in order to get more information of the edges that mark the frontier between the walkable zone, this way we can make a better use of the RayCast.

Following this, the implementation and tuning up of the steering behaviour. Since the navigation agent's path it is not modifiable, we can't afford the solution found in the section 9.8 so a completely new steering behaviour must be done.

References

[1] Unity https://unity3d.com/es

[2] Navigation System https://docs.unity3d.com/Manual/nav-NavigationSystem.html

[3] John Horton Conway, Wikipedia, https://en.wikipedia.org/wiki/John_Horton_Conway

[4] Game of life, Wikipedia, https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

[5] https://www.bi.vt.edu/research/Network-Dynamics-and-Simulation-Science-Laboratory

[6] Scott Edwards, The Chaos of Forced Migration: A Means of Modeling Complexity for Humanitarian Ends, 2009

[7] C# https://docs.microsoft.com/es-es/dotnet/csharp/

[8] Visual Studio https://www.visualstudio.com/es/

[9] GitLab https://about.gitlab.com/

[10] Scrum Alliance, https://www.agilealliance.org

[10] ECS https://unity3d.com/es/learn/tutorials/topics/scripting/introduction-ecs

[11] C# Job System https://docs.unity3d.com/Manual/JobSystem.html

[12] A* algorithm https://en.wikipedia.org/wiki/A*_search_algorithm

[13] ECS project https://github.com/zulfajuniadi/unity-ecs-navmesh

[14] Barycentre Technique http://blackpawn.com/texts/pointinpoly/