# Collision Detection and Pathfinding

https://github.com/MichalKlukas/Game-development

## Scene settings

A screenshot of a computer

Description automatically generated

Figure Scene in Unity

We have a player (red ball) and NPCs (green cubes). In the map, which has clear borders and few obstacles like stationary walls and pillars, moving walls or reappearing walls and pillars.   
In the running test there is a clear objective and that is for all NPCs on the map to navigate to the player using a simple NavMesh. I am using a script to spawn 200 NPCs at a moment, script for a Player to teleport to semi random (Map is divided into set number of areas and it scans through all of them and tries to teleport to the least occupied one) position every few seconds.

## Significance and relevance

Pathfinding and collision detection are fundamental aspects of game development, directly impacting gameplay, realism, and player experience.

**Pathfinding** ensures that non-player characters (NPCs) and game objects navigate the virtual environment intelligently. Algorithms allow NPCs to find the shortest or most optimal path between points while avoiding obstacles. This creates immersive and believable gameplay, where NPCs exhibit behaviors that feel lifelike, such as evading enemies, pursuing players, or collaborating with allies. Efficient pathfinding also affects game performance, as computationally expensive algorithms can strain system resources, especially in dynamic or large-scale environments.

**Collision detection**, on the other hand, ensures physical interactions between objects are realistic and prevent unintended overlaps or glitches. It underpins core mechanics, such as ensuring a player cannot walk through walls, detecting hits in combat, or triggering events when objects collide. Accurate collision systems enhance gameplay by supporting realism, fairness, and interactivity.

## Parameters

In my tests I have decided to work with 3 parameters that change throughout the tests. These and: number of NPCs in the scene at a given moment, frequency of player’s teleportation and size of NPCs. I didn’t create different scenes but only changed all the parameters manually. I did 36 tests, 6 for each number of NPCs (1, 200, 400, 600, 800, 1000) – these 6 tests were combinations of two possible frequencies of player teleportation (2 or 4 seconds) and size of NPCs (scale 0.3, 0.45 and 0.6).   
In the beginning I tried to play with the frequency of map changes but with faster changes, my laptop was not able to handle more than 600 NPCs and at some point, the tests crashed.   
There were a lot of parameters to choose from, because pathfinding and mostly collision detection are affected by many settings, for example obstacle density, frequency of pathfinding update, collider types (different shapes of NPCs) or Speed of NPCs.

## Optimization of the scene

Before I started running the tests I did some basic optimization to get closer to what one would do if he was trying to optimize these technical issues in real game.   
I have lowered the Default Solver Velocity Iterations to 1, which basically allows the NPCs to collide in less precise ways. Then I froze the Axes in rigid bodies of NPCs so there is no extra movement such as rotation or jumping. I have changed the quality of nav mesh agent to medium, so the navigation is less precise, but since my settings were nothing complex it allowed to save some unnecessary computations. I have removed auto-breaking for NPCs. Since there is a change in path every 1-3 seconds it would only make all the movement slower because of the slowing and acceleration. At the same time, it is not what is usually used when operating large numbers of NPCs on the map.  
Last thing I did is that I have made the avoidance radius NPCs bigger. As I realized later, this is not something I should have done, since my map is not that big and with large numbers of NPCs there is sometimes not enough space to get through some areas, so it takes more time for them to move.

## Metrics

In the beginning of the project, I wanted to test the CPU, GPU, Memory usage and FPS. After some tests I realized that my problem is only CPU heavy and there is not much happening on GPU or with the memory even with higher number of NPCs. After the observation I ran the tests only with focus on the CPU usage and the changes in FPS throughout the tests.

Since I use a script to load 200 NPCs at a time there is always a spike in the usage, so for all the tests I did I have waited for some time after the start of the scene, then I observed several thousand frames and then at one point stopped recording. I took the 2000 frames (which is the maximum profiler lets you see at a given time) and pulled them to profile analyzer. There I was able to export the data of the 2000 frames. With the help of a python script, I was able to calculate the average of the frames. From running this for every setting (36 times) I was able to create a table of CPU usage in milliseconds. From this table I did a copy with the difference that every cell used formula: 1000/ (original value). That gave me a table of FPS for every setting from my tests.

## Methods used for evaluation

I started with observation of how the scene handles different settings and that helped me to decide which parameters I want to focus on and what metrics I want to focus on.   
Then I used Profiler to clearly see the effects the settings have on the computational demand. I have set the profiler to show 2000 frames instead of default 300. After each test I used Profile Analyze to export the data in .csv format. After that I used python script to compute the average and last but not least I used another python scripts to create some graphs to be able to observe the changes not just in numbers but also visually.

## Results

The results showed that there is a severe decline in FPS when you add more and more NPCs to the scene. With lower numbers of NPCs, the pathfinding is playing a bigger role in the decline than collision detection. With higher numbers of NPCs the trend changes and collision detection exponentially rises.

A screen shot of a computer screen

Description automatically generated

Figure Others (AI - pathfinding) takes a bigger part of the CPU load, 500 NPCs

A screenshot of a video game

Description automatically generated

Figure Physics (collision detection) severely rises in general (different situation in the few spikes), 1000 NPCs

At the same time there is an observable change in the influence of size of NPCs in comparison to the frequency of Player’s teleportation. In the table below, we can see the average FPS for different settings (tests). There we can observe that the first three rows, tests with up to 400 NPCs, have different trend the other 3 rows. In the first 3 rows there is obvious decline from left to right, which means that in the middle where we change from big NPCs to small but from lower frequency of teleportation to higher, FPS still drop lower. That means that the fact that the path has to be calculated has bigger effects than the size of the NPCs.   
When we look at the part of the table with higher numbers of NPCs, we can see that it is different, now FPS go up when we switch from big NPCs with lower frequency to small NPCs with higher frequency. So, between 400 and 600 NPCs there is a turning point where collision detection becomes bigger problem for CPU than pathfinding.

A screenshot of a graph

Description automatically generated

Figure Table of FPS for all the tested settings.

A screenshot of a graph

Description automatically generated

Figure Table of ms of CPU usage for all tested settings.

On the graph below, we can see how FPS drops during different settings for each number of NPCs. As I mentioned before, we can see that the lines with higher number of NPCs are not “linear”, but there is a jump in the middle little bit up.

A graph with different colored lines

Description automatically generated

Figure Graph showing FPS drops for each number of NPCs during different settings

To make my point clearer, here is the same graph with visual presentation of the observed behavior. Here we can see that collision detection is more demanding than pathfinding with higher number of NPCs.A chart with a yellow circle

Description automatically generated

Figure Graph with visual explanation

This is how the FPS drop for different settings while adding more and more NPCs.

A graph of different colored lines

Description automatically generated

Figure Graph showing FPS drops for different settings while adding NPCs