A picture containing tool

Description automatically generatedSoenen Tristan

What is the impact of switching from a Finite State Machine to Goal Oriented Action Planning (GOAP) in a simulation game?

Supervisor: Vandenberghe Pieter-Jan

Coach: Mores Alexandra

Graduation Work 2024-2025

Digital Arts and Entertainment

Howest.be

A close up of a card

Description automatically generated

Contents

[Abstract & Key words 3](#_Toc187609018)

[Preface 4](#_Toc187609019)

[List of Figures 5](#_Toc187609020)

[Introduction 6](#_Toc187609021)

[Literature Study / Theoretical Framework 7](#_Toc187609022)

[Finite State Machines 7](#_Toc187609023)

[What is a Finite State machine? 7](#_Toc187609024)

[How does a finite state machine work? 9](#_Toc187609025)

[State machines are useful when: 10](#_Toc187609026)

[Where are finite state machines used? 10](#_Toc187609027)

[Goal Oriented Action Planning (GOAP) 11](#_Toc187609028)

[What is Goal Oriented Action Planning (GOAP)? 11](#_Toc187609029)

[How does Goal Oriented Action Planning work? 12](#_Toc187609030)

[The comparison finite state machines vs Goal Oriented Action Planning 13](#_Toc187609031)

[Differences 13](#_Toc187609032)

[Research 14](#_Toc187609033)

[Research Question: 14](#_Toc187609034)

[Hypothesises: 14](#_Toc187609035)

[Methodology: 15](#_Toc187609036)

[Experiment Set-Up: 15](#_Toc187609037)

[Agent Behaviors 15](#_Toc187609038)

[Hardware: 16](#_Toc187609039)

[Data Collection: 16](#_Toc187609040)

[Checking agent behavior: 17](#_Toc187609041)

[Experiments: 18](#_Toc187609042)

[Testing the level for performance: 18](#_Toc187609043)

[Data Collection: 18](#_Toc187609044)

[Prediction: 18](#_Toc187609045)

[Conclusion: 19](#_Toc187609046)

[Testing Agent behaviour: 21](#_Toc187609047)

[Data Collection: 21](#_Toc187609048)

[Prediction: 21](#_Toc187609049)

[Results: 22](#_Toc187609050)

[Conclusion: 23](#_Toc187609051)

[Discussion 25](#_Toc187609052)

[Conclusion 26](#_Toc187609053)

[Future work 27](#_Toc187609054)

[Critical Reflection 28](#_Toc187609055)

[References 29](#_Toc187609056)

[Acknowledgements 31](#_Toc187609057)

[Appendices 32](#_Toc187609058)

# Abstract & Key words

In a world where players want smarter non-playable characters as their allies or enemies in video games. The industries favourite way to do AI for non-playable characters is with Finite State Machines and Behaviour trees. But we can challenge these AI behaviour systems with Goal Oriented Action Planning(GOAP) that creates a Smart agent. This paper seeks to address why Goal Oriented Action Planning isn’t used more in comparison to the industry standard of Finite State Machines. I will create a hospital simulation where the patients and nurses use one of the two AI behaviour systems. I will check each AI behaviours performance based on how much time is spent running the scripts per frame. I will also look at if the agent is a smart agent. I will keep track of the agents their states or actions and see if there are different outcomes between the agents. This paper presents a comparison between the two AI behaviour systems. This paper concludes that Finite State Machines are better for performance and are faster at executing their behaviour as a non-playable character. But Goal Oriented Action Planning has lot more variety in how it executes their behaviour.

In een wereld waar spelers slimmere, niet-speelbare personages willen als hun bondgenoten of vijanden in videogames. De favoriete manier van AI voor niet-speelbare personages is met Finite State Machines en behaviour trees. Maar we kunnen deze AI-systemen uitdagen met Goal Oriented Action Planning (GOAP) die een slimme agent creëert. In deze bachelor proef wordt besproken waarom Goal Oriented Action Planning (GOAP) niet meer wordt gebruikt in vergelijking met de industriestandaard van Finite State Machines. Ik ga een ziekenhuissimulatie maken waarbij de patiënten en verpleegkundigen één van de twee AI-systemen gebruiken. Ik zal de prestaties van elk AI-gedrag controleren op basis van hoeveel tijd er per frame wordt besteed aan het uitvoeren van het script. Ik zal ook kijken of de agent een slimme agent is. Ik zal de non-playable characters, hun toestanden of acties bijhouden en kijken of er verschillende uitkomsten zijn tussen de agenten. Dit artikel presenteert een vergelijking tussen de twee AI-systemen. Deze bachelor proef concludeert dat Finite State Machines beter zijn voor prestaties en sneller zijn in hun plan uitvoeren als een non-playable character. Maar dat er veel meer variatie zit bij de non-playable characters van Goal Oriented Action Planning.

Key Words: Finite State machine, Goal Oriented Action Planning, AI behaviour systems, Smart agent, Non-Playable Characters.

# Preface

During my time at Digital Arts and Entertainment (DAE) I have learned about AI behaviour systems during gameplay programming. The AI behaviour systems we learned are Finite State Machines and Behaviour Trees. I really started enjoying creating the AI behaviour for non-playable characters during the Final exam of gameplay programming “The Zombie game”. The goal of the zombie game is you create an AI behaviour with Blackboards and Behaviour Trees and that AI has to survive in the survival game. The AI had to wander around the map, gather drinks and food, healing and weapons to protects itself from the oncoming zombies and survive.

So after that I wanted to further my knowledge in what other AI behaviour systems exist. So this research started with the goal to compare multiple AI behaviour systems. I have looked at Finite State Machines, Behaviour Trees, Goal Oriented Action Planning and Utility AI. I learned a lot by just doing research about what AI behaviour systems are just out there. But this was a little bit out of scope so it became this research where I compare Finite State Machines with Goal Oriented Action Planning.

# List of Figures

[Figure 1 Traffic light states [2] 7](#_Toc187596373)

[Figure 2 Traffic light diagram [2] 8](#_Toc187596374)

[Figure 3 example of Finite State Machine in video games [3] 9](#_Toc187596375)

[Figure 4 Example states for a vending Machine 10](#_Toc187596376)

[Figure 5 Example off Goal Oriented Action Planning actions 12](#_Toc187596377)

[Figure 6 example of a plan created with the actions 12](#_Toc187596378)

[Figure 7 example of how a Finite State Machine is Coupled [5] 13](#_Toc187596379)

[Figure 8 Example of how GOAP is Decoupled [5] 13](#_Toc187596380)

[Figure 9 Example of Patient behaviour 15](#_Toc187596381)

[Figure 10 Example Of Nurse behaviour 16](#_Toc187596382)

[Figure 11 Finite State Machine performance check results 19](#_Toc187596383)

[Figure 12 GOAP performance check results 20](#_Toc187596384)

[Figure 13 Table of the collected behaviour data for the agents with Finite State Machine as their AI behaviour system 22](#_Toc187596385)

[Figure 14 Table of the collected behaviour data for the agents with GOAP as their AI behaviour system 22](#_Toc187596386)

[Figure 15 Comparisons of the collected Patient Data of both AI-systems 23](#_Toc187596387)

[Figure 16 Comparisons of the collected Data for the nurses 24](#_Toc187596388)

# Introduction

Non-playable characters and their AI-systems are important for video games. It’s frustrating for players when their ally who is a non-playable character in the video game makes some questionable moves or gets stuck with something. Or even if the strong boss their AI becomes too predictable or it becomes too easy to beat.

The current way the video game industry does it is with Finite State machines and Behaviour trees that have some sort of Utility AI (scoring system for the best next possible action) with them so the non-playable characters become smarter non-playable characters. This makes it interesting for the player because if the AI system is too predictable like for example with Finite State Machines. The player can start predicting the next moves of the AI and we don’t want that.

This research paper will attempt to explain how Finite State Machines and Goal Oriented Action Planning work with some examples of how they are used. I will put both AI systems to the test and explain the results that come from the tests.

The setting of the simulation game: the game is a simulation of the hospital where I spawn patients that go to the hospital and get treated by nurses before they go back home but if they have to wait to long for treatment they leave angry.

This paper seeks to answer “**What is the impact of switching from a Finite State Machine to a Goal Oriented Action Planning (GOAP) in a simulation game?**”

These are the hypotheses that have been made:

**H0**: After comparing the data graphs from both Finite State Machines and Goal Oriented Action Planning, there is no real difference in performance. Both take the same amount of time per frame or there is less than a 5% difference in average script execution time.

**H1**: The data graph of the Goal Oriented Action Planning (GOAP) shows a longer average script execution time per frame of more than 5% compared to the average execution time per frame of State Machines.

**H2**: After a 100 000 patients have passed through the hospital the Goal Oriented Action planning nurses take more coffee breaks.

**H3**: After a 100 000 patients have passed through the hospital the Goal Oriented Action Planning Patients leave the hospital more angry.

**H4**: After a 100 000 patients have passed through the hospital the Finite State machine has more Patients treated.

# Literature Study / Theoretical Framework

## Finite State Machines

### What is a Finite State machine?

“A finite-state machine, or FSM for short, is a model of computation based on a hypothetical machine made of one or more states. Only a single state can be active at the same time, so the machine must transition from one state to another in order to perform different actions.” [1]

Example: Traffic Lights

Afbeelding met Kleurrijkheid, cirkel

Automatisch gegenereerde beschrijving

Figure 1 Traffic light states [2]

A traffic light has 4 States:

1. Green: you can go.
2. Amber: you should stop if you can do so safely because soon it will be red.
3. Red: you have to stop.
4. Amber flickering: this means go with caution because something is wrong with the traffic light.

But how do the traffic lights change states?

It has some triggers or events that makes the states switch because it’s a traffic light.

It has three types of triggers:

1. Power: the electricity powers the whole system this starts off the first state.
2. Timers: 3 out of the four states have timers when a timer expires we switch state.
3. Reset button: when the reset button is pressed we go back into the first state.

Afbeelding met diagram, lijn, cirkel, Lettertype

Automatisch gegenereerde beschrijving

Figure 2 Traffic light diagram [2]

When the power is on for the traffic light he starts he goes into the red state. The red light turns on, the red timer starts ticking. After the red timer expires we transition into the green state. The red light turns off but now because we switched state the green light turns on and the green timer starts ticking. After the green timer expires we go into the next state which is amber. The green light turns off the amber timer starts ticking and the amber light turns on. After the Amber timer expires we go into the red state again and start the whole process over. But something went wrong the traffic light has a system error. We transition from any off the three previous states into the red flashing state. The red light starts flickering the traffic light stays in this state until we restart the traffic light system.

### How does a finite state machine work?

This example goes over how a Finite State Machine works in video games.

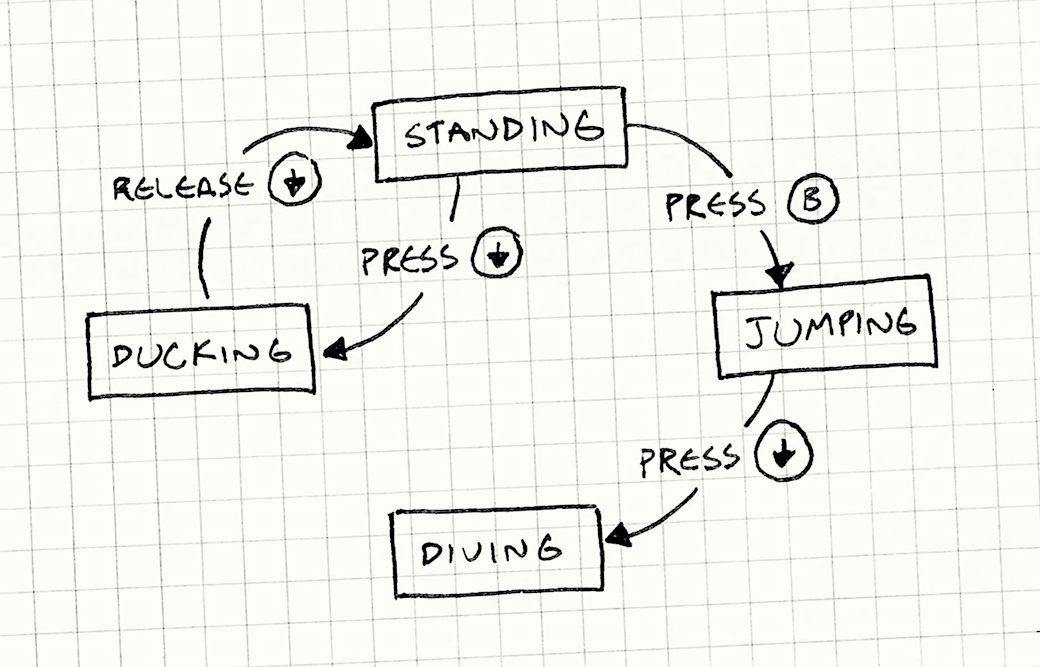


Figure 3 example of Finite State Machine in video games [3]

There are a fixed amount of states that the machine can be in. In the example above that is standing, ducking, jumping and diving. The Finite State Machine can only be in one state at a time. This means you can’t be standing and ducking at the same time. When down is pressed for example we transition from the standing state into the ducking state. When down is released we go back to the standing state. Each state has its transitions for example when we are in the ducking state and the input of B is pressed comes in we will not jump. In the ducking state we are only checking if down gets released to go back to the standing state. When we are in the standing state and B is pressed we transition into the jumping state. If we press down in the jumping state we then transition into the diving state. So per state we check an input and if that input matches what we are looking for we transition into the next state. These inputs can be from an controller or events from the world around the character.

### State machines are useful when:

* You need to respond to a series of inputs, events and signals over time. [3]
* You can split the work into multiple states. [3]
* The behaviour changes based on an internal state. [3]

### Where are finite state machines used?

Finite State Machines are used when you need to react to a signal, event or input from the external world.

Examples:

|  |  |  |
| --- | --- | --- |
| * Vending machines[4] | * Alarm Clock[4] | * Video games |
| * Traffic lights[4] | * Microwave[4] |  |
| * Elevators[4] | * Cash Registers[4] |  |

Explaining the vending machine example:

The following diagram was made with the inspiration of the diagram of a vending machine from this study: [5]

Afbeelding met diagram, lijn, Plan, Technische tekening

Automatisch gegenereerde beschrijving

Figure 4 Example states for a vending Machine

The diagram above shows the states of a vending machine and how they work. I walk up to the vending machine and press the start button. The vending machine starts up and transitions into the select product state. I press the buttons with a valid number for the item I want from the vending machine. The machine transitions into his next state. The machine checks is the item that I selected in stock? It isn’t The machine shows a message with item out of stock and asks to select an other product or just end. After I give the number of an item that is in stock the machine transitions back to the state where it checks if the item is in stock. This time I pass the check. The machine transitions to the pay state. It now has a message with the amount you have to pay. We pay with our card if it is accepted. The vending machine transitions into the next state that gives the product prints a thank you message and ends. If the payment is not accepted the machine transitions into the next state and prints out a message that says payment not accepted and end the transaction.

## Goal Oriented Action Planning (GOAP)

### What is Goal Oriented Action Planning (GOAP)?

“Goal oriented action planning is an artificial intelligence system for agents that allows them to plan a sequence of actions to satisfy a particular goal. The particular sequence of actions depends not only on the goal but also on the current state of the world and the agent. This means that if the same goal is supplied for different agents or world states, you can get a completely different sequence of actions., which makes the AI more dynamic and realistic.” [6]

Goal Oriented Action Planning is used in the game F.E.A.R. The Non-Playable Characters use Goal Oriented Action Planning in combination with a small Finite State Machine for their AI.

GOAL oriented action planning is useful when:

* you want to decouple the actions from each other so you can focus on each action individually.[6]
* you want code that is easy to test and maintain.[6]

### How does Goal Oriented Action Planning work?

Goal Oriented Action Planning has Actions. Action have preconditions and effects so first we need to check if we have met the precondition. Then we do the action and get the effect. Example our goal is to be hydrated these are the actions we have:

Afbeelding met tekst, Lettertype, cirkel, diagram

Automatisch gegenereerde beschrijving

Figure 5 Example off Goal Oriented Action Planning actions

Based on the preconditions this is the plan that gets created.

Afbeelding met tekst, cirkel, lijn, diagram

Automatisch gegenereerde beschrijving

Figure 6 example of a plan created with the actions

So we want to drink water first we search for a water bottle since it has no precondition. The effect we get is we have now a water bottle. So we go to the next action Open the water bottle. The precondition of that action is that we need a water bottle we got that from our last action so now we can start our new action. The effect we get is that the water bottle is now open. The next action is to drink water. We have the precondition so we drink water the effect is our goal to be hydrated. Good our plan is complete.

## The comparison finite state machines vs Goal Oriented Action Planning

### Differences

“Let’s compare FSMs to planning. An FSM tells an A.I. exactly how to behave in every situation. A planning system tells the A.I. what his goals and actions are, and lets the A.I. decide how to sequence actions to satisfy goals.”[7]

Finite State Machine (coupled):

Afbeelding met schermopname, cirkel, lijn, Lettertype

Automatisch gegenereerde beschrijving

Figure 7 example of how a Finite State Machine is Coupled [6]

The image above shows how a Finite State Machine is coupled Each state knows what other states it can transition into. They know their next states.

Goal Oriented Action Planning (decoupled):

Afbeelding met schermopname, cirkel, Lettertype, tekst

Automatisch gegenereerde beschrijving

Figure 8 Example of how GOAP is Decoupled [6]

The image above just shows the actions based on these action the planner now has to create a plan to satisfy its goal. Example: our goal is to chop a log to satisfy this goal we could have these preconditions we need a log and something to chop the log with. This could be the plan: 1) pick up the axe. 2) Get a log. 3)Chop the log.

We created a plan to satisfy the goal. we can also have two goals where we want to drop of fire wood and chop the log. So we would have the previous plan done first to now start on our next goal to drop off fire wood. The plan goes as follows: 1) pick up the axe. 2) Get a log. 3)Chop the log. Goal 1 is met new plan gets created: 1) pick up the fire wood. 2) drop the fire wood off.

# Research

## Research Question:

**What is the impact of switching from a Finite State Machine to a Goal Oriented Action Planning (GOAP) in a simulation game?**

The simulation game consists of a hospital the patients get spawned into the game and have to go to the hospital to receive treatment. After waiting to long at the hospital the patient gets angry and leaves. The nurses are at the hospital and the treat the patients. The simulation game is used to check what performance impact the agents their behaviour have on the time spent per frame on scripts alone. The simulation game is also used for just checking the differences in the agents their behaviour for Goal Oriented Action Planning and Finite State Machines.

## Hypothesises:

**H0**: After comparing the data graphs from both Finite State Machines and Goal Oriented Action Planning, there is no real difference in performance. Both take the same amount of time per frame or there is less than a 5% difference in average script execution time.

**H1**: The data graph of the Goal Oriented Action Planning (GOAP) shows a longer average script execution time per frame of more than 5% compared to the average execution time per frame of State Machines.

**H2**: After a 100 000 patients have passed through the hospital the Goal Oriented Action planning nurses take more coffee breaks.

**H3**: After a 100 000 patients have passed through the hospital the Goal Oriented Action Planning Patients leave the hospital more angry.

**H4**: After a 100 000 patients have passed through the hospital the Finite State machine has more Patients treated.

## Methodology:

### Experiment Set-Up:

There is one level where I just switch the Goal Oriented Action planning agents with the Finite State Machine agents and build. It’s important that there are no meshes and other clutter so we can just see the performance of the scripts. The agents with the Goal Oriented Action Planning are created following a guide from unity learn[8]. For the agents of the Finite State Machine I recreated the agents their behaviours of the Goal Oriented Action Planning making sure they behave in the same way.

### Agent Behaviors

**Patient:**

Afbeelding met tekst, diagram, lijn, schermopname

Automatisch gegenereerde beschrijving

Figure 9 Example of Patient behaviour

The Patient can do 6 things:

1. The patient spawns and goes to the hospital.
2. Once at the hospital the patient registers himself.
3. After registering the patient goes to the waiting room.
4. If the patient has to wait too long he gets angry and leaves the hospital.
5. The patients goes with an available nurse to a cubicle and gets treated.
6. The patient leaves the hospital and goes home

**Nurse:**

Afbeelding met tekst, lijn, diagram, schermopname

Automatisch gegenereerde beschrijving

Figure 10 Example Of Nurse behaviour

The nurse can do 3 things:

1. Get a patient from the waiting room and see if a cubicle is available.
2. Treat the patient.
3. Coffee break.

### Hardware:

Information over the hardware that the build is tested on:

* Processor: AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz
* Ram: 16GB
* Type: system: x64-processor

### Data Collection:

To test the performance of each build I used the Unity Profiler. The unity profiler is used with the target set to 60 Frames Per Second. I check from when the agents are spawned until they get destroyed and explain their graphs. Because the unity profiler runs in development mode I did the tests with less agents.

### Checking agent behavior:

Important data for the level set up:

* The are only 9 cubicles that can be used to treat patients.
* There are 9 nurses in the level.
* There is a patient spawner that spawns a 1000 agents every second until the total number of a 100 000 agents have been spawned.
* Once an agent can switches from one action to another it takes 0.01 seconds if there isn’t another condition that has to be met.
* After 0.25 seconds in the waiting room the patient can leave angry for both Finite State Machines and Goal Oriented Action Planning.
* After 0.25 seconds the nurse can take a coffee break for both Finite State Machines and Goal Oriented Action Planning.

This is a list of data that I will look at to see if the agents behave differently:

* The amount of patients that get treated.
* The amount of patients that left because they had to wait too long.
* The amount of coffee breaks taken by the nurses

All the data is kept in states where I just add to the state each time something happens.

After a 100 000 patients have gone home or left angry all the data get written to a txt file and then the application quits. I do this five times for Finite State machines and five time for Goal Oriented Action Planning then I take the averages and compare them.

## Experiments:

## Testing the level for performance:

I compared the Finite State Machine with the Goal Oriented Action Planning build. Both builds spawn a 2500 patients in total. Each second from play 250 patients gets spawned until we hit the total. After the last patient has been treated or left angry the application stops. I used the Unity Profiler on each build to get the performance data from it.

### Data Collection:

The application keeps running until all 2500 patients have been treated or left angry. I ran each application and captured the moment 250 patients get spawned and check the profiling data until the 250 agents get destroyed.

### Prediction:

My prediction is that the Finite State machine will better for the performance of the overall simulation. So **H0**  *“After comparing the data graphs from both Finite State Machines and Goal Oriented Action Planning, there is no real difference in performance. Both take the same amount of time per frame or there is less than a 5% difference in average script execution time. “* should be true.

This then makes **H1**: *“The data graph of the Goal Oriented Action Planning (GOAP) shows a longer average script execution time per frame of more than 5% compared to the average execution time per frame of State Machines.”* False*.*

### Conclusion:

Finite State Machine:

Afbeelding met schermopname

Automatisch gegenereerde beschrijving

Figure 11 Finite State Machine performance check results

The first peak is of the 250 patients spawning. After that they go into their first state and start switching to their next states until the patients are destroyed this graph shows a really good performance for the Finite State Machine build. The average from the second peak and the next 33 frames is 0.34s of time spent on scripts per frame.

Goal Oriented Action Planning:

Afbeelding met tekst, schermopname, Lettertype, ontwerp

Automatisch gegenereerde beschrijving

Figure 12 GOAP performance check results

The first peak of the Goal Oriented Action Planning graph is just the 250 patients spawning. After the dip Its just The Goal Oriented Action Planning agents just creating plans to satisfy their goals. Until the dip at the end which is caused because all the remaining patients that are waiting get angry and are destroyed. Since it’s easy to track the G agent update with the profiler I can just look at their times spent on running scripts per frame. The average from the second peak and the next 33 frames is 6.05ms spent on scripts per frame.

So the Finite State Machines average is almost 18 times faster at running their scripts per frame compared to the Goal Oriented Action Planning average.

This than means that **H0**  *“After comparing the data graphs from both Finite State Machines and Goal Oriented Action Planning, there is no real difference in performance. Both take the same amount of time per frame or there is less than a 5% difference in average script execution time. “* is false*.*

This then makes **H1**: *“The data graph of the Goal Oriented Action Planning (GOAP) shows a longer average script execution time per frame of more than 5% compared to the average execution time per frame of State Machines.”* is true*.*

## Testing Agent behaviour:

I created a build for both Finite State Machines and Goal Oriented Action Planning. I run each application five times. The applications close after a 100 000 agents have gone through the hospital and write a txt file with data. Then I take the averages of both their five runs and conclude which one is better at what.

### Data Collection:

This is how I collected my data for patients treated, patients that left angry and the amount of coffee breaks a nurse took.

Patients treated: Once a patient exits the get treated it counts as a treated patient.

Coffee breaks: each time a nurse takes a coffee break it counts as a coffee break.

Patients that left angry: once the counter in the Go To Waiting Room action hits 0.25 seconds for a patients he counts as a patient that left angry.

Each time I count something I keep track off it with a world state and after a 100 000 patients have been treated I write the states and their values to a txt file.

### Prediction:

I predict that the state machine will treat more patients and take more coffee breaks. This means that **H2***: “After a 100 000 patients have passed through the hospital the Goal Oriented Action planning nurses take more coffee breaks.”* Is false.

**H3**: *“After a 100 000 patients have passed through the hospital the Goal Oriented Action Planning Patients leave the hospital more angry.”* Is true.

**H4**: *“After a 100 000 patients have passed through the hospital the Finite State machine has more Patients treated.”* Is true.

### Results:

Results for the simulations with the Finite State Machine as the agents behaviour:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1st run | 2nd run | 3rd run | 4th run | 5th run | Average run |
| Patients Treated | 78 870 | 72 793 | 72 625 | 76 133 | 74 488 | 74 981.8 |
| Patients Angry | 21 130 | 27 216 | 27 375 | 23 887 | 25 512 | 25 024 |
| Coffee breaks | 900 | 909 | 900 | 900 | 900 | 901.8 |

Figure 13 Table of the collected behaviour data for the agents with Finite State Machine as their AI behaviour system

Results for the simulations with Goal Oriented Action Planning as the agents behaviour:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1st run | 2nd run | 3rd run | 4th run | 5th run | Average run |
| Patients Treated | 1 818 | 1 935 | 1 881 | 1 863 | 1809 | 1 861.2 |
| Patients Angry | 98 182 | 98 065 | 98 119 | 98 137 | 98 191 | 98 138.8 |
| Coffee breaks | 117 | 1845 | 702 | 1 800 | 1 314 | 1 155.6 |

Figure 14 Table of the collected behaviour data for the agents with GOAP as their AI behaviour system

### Conclusion:

Figure 15 Comparisons of the collected Patient Data of both AI-systems

Based on the averages the Goal Oriented Action Planning agents leave more angry. A lot more patients get treated with the Finite State Machines. But why are the Finite State Machine Patient getting treated more? Thiscan be because the Finite State Machine patient already knows his next State and just waits until he can transition. The Goal Oriented Action Planning agent on the other hand still has to create new plans.

Example the Goal Oriented Action Planning patient has 3 goals:

1. Be at the hospital
2. Receive Treatment
3. Go home

For all the goals a new plan has to be made and their preconditions also have to be met. So Goal Oriented Action Planning get just left behind.

So I can conclude that:

**H3**: *“After a 100 000 patients have passed through the hospital the Goal Oriented Action Planning Patients leave the hospital more angry.”* Is true.

**H4**: *“After a 100 000 patients have passed through the hospital the Finite State machine has more Patients treated.”* Is true.

Figure 16 Comparisons of the collected Data for the nurses

The Goal Oriented Action Planning nurses take more coffee breaks on average.

So I can conclude that:

**H2***: “After a 100 000 patients have passed through the hospital the Goal Oriented Action planning nurses take more coffee breaks.”* is true*.*

# Discussion

Testing the level for performance:

The experiment where I compared the performance off all the agents with a Finite State Machine to the agents who have Goal Oriented Action Planning as their behaviour. The results clearly show that Finite State Machines are better for performance. I did not think the difference would be this big. I think this experiment is good if you only want to look at performance because you need an Ai behaviour system for a lot of agents. However this experiment lacks when you don’t need to watch for performance. example: if I need a lot of dumb agents and just one really good boss. The dumb agents should be tested more for performance since we update that large amount of agents in comparison with just one update from a Smarter more interesting agent.

Testing Agent Performance:

The Finite State Machine Agents where a lot faster because they already had a plan/ planned next state while the Goal Oriented Action Planning agents still need to think about their next goal and how they are going to get there. The big amount of angry patients on the Goal Oriented Action planning side is just because they could not keep up and the patient would just leave angry. While I was predicting that the Finite State machine agents would have an higher amount of agents treated I was not expecting the difference to be this high. But I also think this experiment should be done again when the agents have more choice because we weren’t able to show any of Goal Oriented Action Planning their strengths. Goal Oriented Action Planning isn’t supposed to be the faster agent but a smarter agent. This means that the Goal Oriented Action Planning agent should show a lot more variety in the data. Which here is just a big difference because there really isn’t a lot of choice.

# Conclusion

The research question was: “**What is the impact of switching from a Finite State Machine to Goal Oriented Action Planning (GOAP) in a simulation game?**” The two experiments show what happens when you switch from a Finite State Machine to Goal Oriented Action Planning.

Because of the comparison I made with the experiments I can conclude the following:

Finite State Machines are better for performance. Goal Oriented Action Planning takes a lot more time running scripts per frame because they have to keep making plans for characters that have nothing to do this creates the performance hit.

The Finite State Machine Patients get treated more because they will always know what their next states are and there is less room for error.

The Goal Oriented Action Planning nurses take more coffee breaks but this can just be because if there are not patients to treat you just keep taking coffee breaks.

The Goal Oriented Action Planning nurses have more patients leaving the hospital angry. This is because the timer for leaving the hospital is very low 0.25of a Second.

While a lot of agents with Goal Oriented Action Planning is bad for performance maybe one agent with Goal Oriented Action Planning can be a good thing for example a boss. Because GOAP showed more variety with the coffee breaks.

# Future work

While this experiment was good to check the differences between Goal Oriented Action Planning and Finite State machines it still was limited in the amount of choice it gave to the agents. So when there is a limited amount of choice you can make I would say that gives a little bit of an unfair advantage to the Finite State Machines. Because with the Finite State Machines you already know the next state. But with Goal Oriented Action planning he would just plan for goals he already knows that are going to be the next step. So if I were to run these experiments again I would make them a lot bigger so there is an abundance of choice.

If I were to run this research again I would change these things. I would make one level where you fight a boss. The boss uses an AI behaviour system Finite State Machines or Goal Oriented Action Planning. We keep the first and second experiment the same. But I would create a third experiment where you ask people to play the boss fight and ask which boss was more interesting to fight. The third experiment could show one of the strengths of Goal Oriented Action Planning. The Goal Oriented Action Planning could show the player a smarter boss compared to the Finite State Machine Boss. The conclusion would not just be the performance and data from a simulation but have the player also decide for themselves if they like smarter non playable characters in video games.

Another way to create a smart agent is with Utility AI. I looked into Utility AI during the Literature Study, but chose Goal Oriented Action Planning over Utility AI. So making a comparison between an AI behaviour system and the same AI behaviour system but with Utility AI can be interesting. The experiments I did in this project can also be used to check if Utility AI is better than the normal behaviour system.

The industries standard way to make non-playable characters systems are Finite State Machines and Behaviour Trees. This project compared Goal Oriented Action Planning with Finite State Machines but you could also compare it with Behaviour Trees.

# Critical Reflection

I started this research project to learn more about AI behaviour systems for non-playable characters. After doing some research I was wondering why Goal Oriented Action Planning is not used more by the gaming industry. After learning and working with Goal Oriented Action Planning, I have gotten a better idea of when you want to use Goal Oriented Action Planning and when you don’t want to use Goal Oriented Action Planning. I have already learned so many new things about Goal Oriented Action Planning. After learning about Goal Oriented Action Planning I want to learn more about Utility AI.

On the other hand I hoped this research project might be able to get on my portfolio. But there is not really a lot to show. The project is not flashy so I think not a lot of people will be interested in it since also the rendering is turned off and you can’t see anything. However this project helped me acquire the skills for creating a flashy project with a fully working Goal Oriented Action Planning Boss.

I have also grown in many other ways for example I’ve got the opportunity to work with unity version 6. Learn more about the unity profiler and profiler analyser. But I really need to work on my writing skill and presenting skills.

After doing this research paper I am happy with what I got done and what I have accomplished. Sadly there wasn’t more time I was thinking of maybe adding a third experiment but this is now written in the future work section.

# References

[1] Title: Finite-State Machines: Theory and Implementation.

Author: Fernando Bevilacqua

Release Date: 24 October 2013

Type: Online Article

Link: <https://code.tutsplus.com/finite-state-machines-theory-and-implementation--gamedev-11867t>

[2] Title: State Machines for Everyone — Part 1 Introduction

Author: Alex Dodge

Release Date: 8 September 2020

Type: Online Article

Link: <https://medium.com/well-red/state-machines-for-everyone-part-1-introduction-b7ac9aaf482e>

[3] Title: State

Author: Robert Nystrom

Release Date: 2014

Type: a Chapter of the book “Game Programming Patterns”

Link: <https://gameprogrammingpatterns.com/state.html>

[4] Title: 2.3 Finite State Machine (FSM) Concept and implementation

Author: Kyong-Sok(KC) Chang & David Zhu

Release Date: /

Type: PDF from a university lecture.

Link: <https://web.stanford.edu/class/cs123/lectures/CS123_lec07_Finite_State_Machine.pdf>

[5] Title: Design of Vending Machine using Finite State Machine and Visual Automata Simulator

Author: Ashwag Alrehily, Ruqiah Fallatah and Vijey Thayananthan

Release Date: 18 April 2015

Type: Online Research PDF

https://www.researchgate.net/publication/276136463\_Design\_of\_Vending\_Machine\_using\_Finite\_State\_Machine\_and\_Visual\_Automata\_Simulator

[6] Title: Goal Oriented Action Planning for a Smarter AI

Author: Brent Owens

Release Date: 23 April 2014

Type: Online Article

Link: https://gamedevelopment.tutsplus.com/goal-oriented-action-planning-for-a-smarter-ai--cms-20793t

[7] Title: Three States and a Plan: The A.I. of F.E.A.R

Author: Jeff Orkin

Release Date: 2006

Type: PDF from the Game Developers Conference

Link: <https://www.gamedevs.org/uploads/three-states-plan-ai-of-fear.pdf>

[8] Title: Goal Driven Behaviour

Author: Penny de Byl

Last Updated: 24 January 2023

Type: guide on the Unity Learn Platform

Link: <https://learn.unity.com/project/goal-driven-behaviour>

# Acknowledgements

I would like to thank my supervisor Pieter-Jan Vandenberghe and my coach Alexandra Mores for helping me throughout this project. I also would like to thank the DAE teachers for teaching me valuable skills and a big thank you to my family for supporting me all the way.

# Appendices

The code is available on Github: <https://github.com/TristanSoenen/GradWork>

On Github there is also a zip file with the four builds of the experiments. The profiler build have profiler auto connect so you just have to run those builds and have the Unity profiler open to get the profiling data. For the other experiment you just have to run the application it closes on its own and check the data file for the results.