

Digital Reasoning and Creativity

GOAP Algorithm

Introduction

Goal-Orientated Action Planning (as known as **GOAP**) is a basic and simple way to explain why people's take actions. In this presumption, we consider that everyone's determination to take action is an outcome of a series of planning. In addition, Why one takes actions is to achieve each specific **Sub Goal**.

- Example

Node 1 : Having a stomachache.

Node 2 : Going to the hospital (Sub Goal : To feel better)

Node 3 : Doing the final project (Sub Goal : To complete it before deadline)

Accordingly, we can analyze one's thought with GOAP algorithm.

- If a person choose to go to the hospital, he/she attempted to feel better after taking action.
- If a person choose to do the final project, he/she attempted to complete it before deadline.

In the planning procedure, we assume that each link (from a node to another node) exist a variable called **Cost** to help one plan next action.

- Example

Node 1 → Node 2 (Total Cost 10)

Node 1 → Node 3 (Total Cost 5)

Then, we go through some other algorithm (this is not the critical part in the GOAP introduction, we assume that we always choose the minimal cost to take action.), and we would obtain the result of Doing the final project.

Digital Reasoning

Code Concept

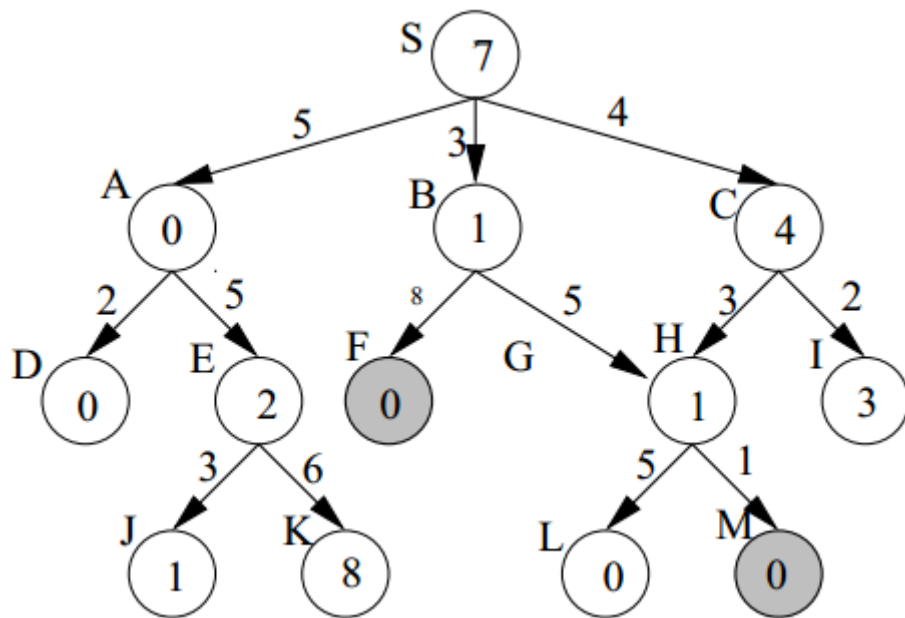
GOAP (Goal-Orientated Action Planning) is a simple algorithm to allow us to control the whole game AI system easier. In other words, after we have established codes about our game world as well as working logic, **we made use of a "Planner"**(code we created) to **figure out the plan for AI behavior**.

We apply two parameters (**Precondition** and **Postcondition**) on AI behavior. That is, all behaviors have their cause and effect, and we just need to control these parameters on the surface.

- **Build a Graph**

We use A* algorithm to find the cheapest plan.

We build nodes of temporary goals, and **Tree** of every actions. Every **Leaf** (End Node) refers to each **Sub Goal**.



Concepts

Trying to split a mission/action into fractions.

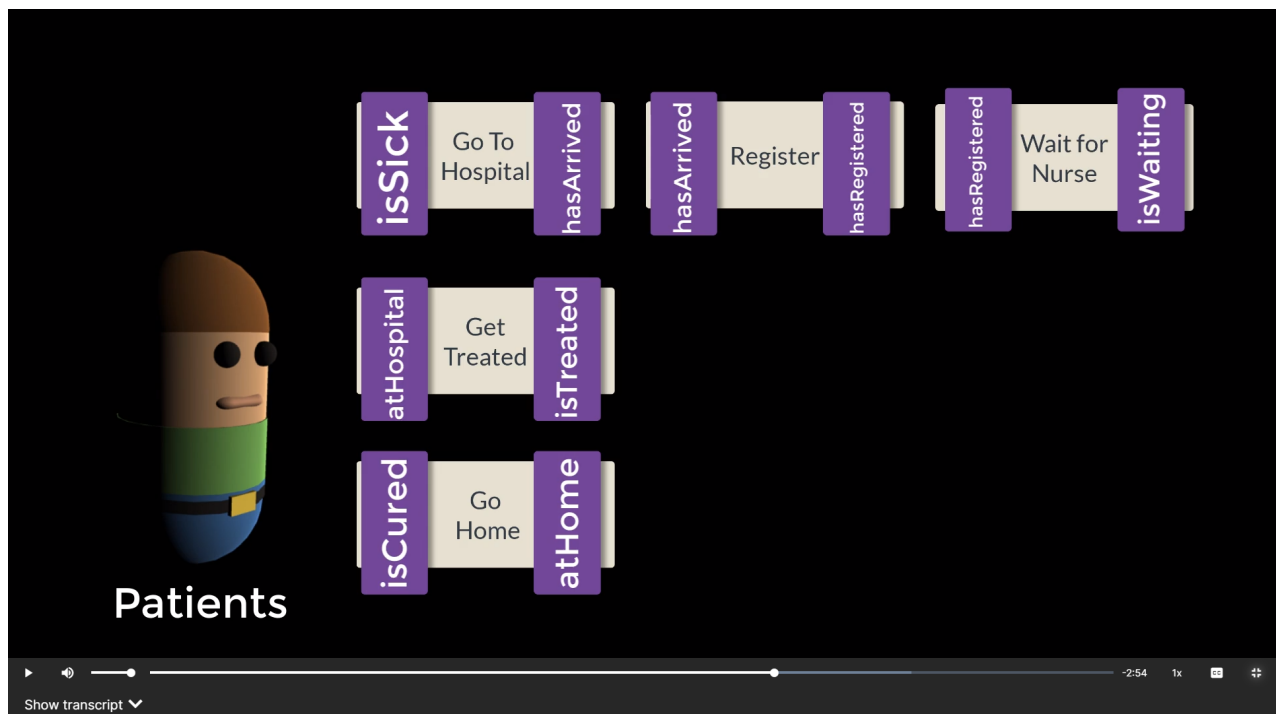
- Without interacting

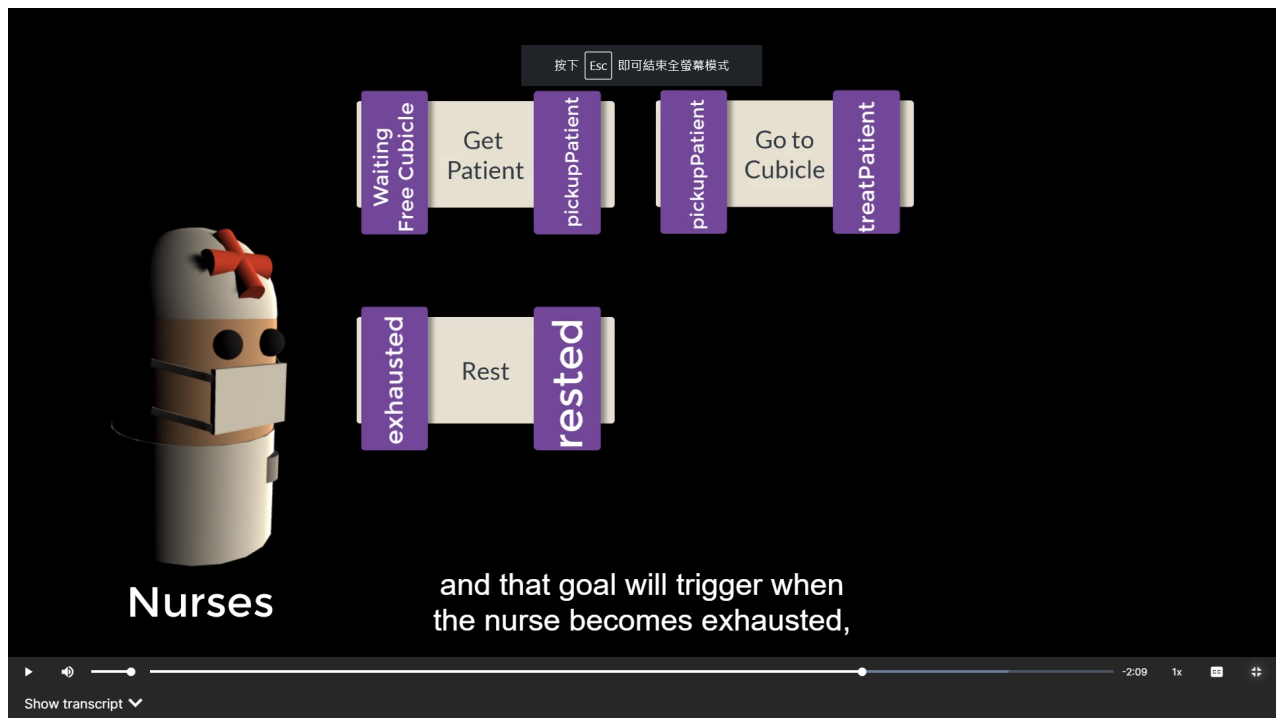
However, if you do not need dynamic interactions with other agents, you can just put them into a procedure chain and complete every steps.

- Interacting

Now you split them into fractures. Then you need to organize those steps with structures(precondition and effect chain).

For example :





History

GOAP (Goal-Orientated Action Planning) is a simple algorithm to allow us to control the whole game AI system easier. Before then, the game designer had difficulty coding the scripts of AI. Because it was considered complex and maintainable; therefore, we need to a more simple way to code with game's AI.

Novelty

- It create a simple thought to program.
- GOAP is **scalable**, **maintainable** and **systematic** way to make AI.
- Bringing the concept of letting AI plan their actions by themselves to increase changeability in games.

Novelty Value

- To whole game forum and build a fundamental thought to create the afterword AI logic.
- As we can see, GOAP has the categorical value (*good in itself*)




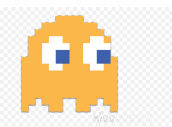
Implementation

F.E.A.R. (X360/PS3/PC) - Monolith Productions/VU Games, 2005
Condemned: Criminal Origins (X360/PC) - Monolith Productions/Sega, 2005
S.T.A.L.K.E.R.: Shadow of Chernobyl (PC) - GSC Game World/THQ, 2007
Mushroom Men: The Spore Wars (Wii) - Red Fly Studio, 2008
Ghostbusters (Wii) - Red Fly Studio, 2008
Silent Hill: Homecoming (X360/PS3) - Double Helix Games/Konami, 2008
Fallout 3 (X360/PS3/PC) - Bethesda Softworks, 2008
Empire: Total War (PC) - Creative Assembly/SEGA, 2009
F.E.A.R. 2: Project Origin (X360/PS3/PC) - Monolith Productions/Warner Bros, 2009
Demigod (PC) - Gas Powered Games/Stardock, 2009
Just Cause 2 (PC/X360/PS3) - Avalanche Studios/Eidos Interactive, 2010
Transformers: War for Cybertron (PC/X360/PS3) - High Moon Studios/Activision, 2010
Trapped Dead (PC) - Headup Games, 2011
Deus Ex: Human Revolution (PC/X360/PS3) - Eidos Interactive, 2011

PAC-MAN

We take a typical game as a instance.

AI

AI	Image	Action
Blinky		Follows Pac-Man directly during Chase mode, and heads to the upper-right corner during Scatter mode.
Inky		During Chase mode, his target is a bit complex. His target is relative to both Blinky and Pac-Man, where the distance Blinky is from Pinky's target is doubled to get Inky's target. He heads to the lower-right corner during Scatter mode.
Pinky		Chases towards the spot 2 Pac-Dots in front of Pac-Man. Due to a bug in the original game's coding, if Pac-Man faces upwards, Pinky's target will be 2 Pac-Dots in front of and 2 to the left of Pac-Man. During Scatter mode, she heads towards the upper-left corner.
Clyde		Chases directly after Pac-Man, but tries to head to his Scatter corner when within an 8-Dot radius of Pac-Man. His Scatter Mode corner is the lower-left.

Player Reactions

There were many players who had completed all levels (255 in total) in PAC-MAN agreed that if you made one mistake after level 25, you could have not achieved level 255.

In the other words, because the tracing logic is fixed, you only have one route to pass the game after level 25. Therefore, you need to memorize all route from level 25 to level 255.

In my opinion, if you have to play the game in this way, then the game will be more like a **Mechanism verse Mechanism** than a **Player verse Mechanism**.

PAC-MAN with GOAP

We can allow AI to plan the tracing mode to navigate to the player's position.

Sub Goals

- Chasing player
- Obstructing player's motion.
- Not queuing in one line.
- Preventing player eat the yellow dots or pink dots.

In every frame, AI would consider how to move and which action to adopt. That will not be the fixed formula to pass each level, and will get a greater changeability in games.

Opinion

Constraint and Free

Constraint

It is too simple and not capable to plan actions the same as how people do in reality. The most convenient point is also the major setback of this algorithm is **Cost**. How to determine cost value? How to change cost value while playing? How to use the minimal usage of hardware in players' computers?

Although we have the efficient way and manage to determine value within game. There is still a problem, **Cost for what?**

In reality, we consider not only one factor to plan, such as, time, money or emotion. All these variables can not replace by simply add one variable **Cost** in GOAP algorithm.

Free

Initially, I just wanted to adopt a simple algorithm on the Enemy AIs in my game. Then I learned GOAP algorithm. It help me use another aspect to program with AI, NPC or even the character of Player. This algorithm brings me the creative thought of how to make my roles in games deed more reasonable and humanize these code-made roles.

Frankly speaking, the concept of combining PAC-MAN with GOAP algorithm is my idea. This algorithm may seem to be too simple and nothing to notice. GOAP still provide programmers with a lot of concepts and structures. I start to have the ability to come up with the others creative idea and implement them on my games.

Reference

[GOAP MIT alumni](#)

[Unity Learn - GOAP](#)

[GOAP HackMD](#)

[Applying Goal-Oriented Action Planning to Games](#)

[PAC-MAN AI Behavior](#)

[Blinky image](#)

[Inky image](#)

[Pinky image](#)

[Clyde image](#)