Thinking Before Acting

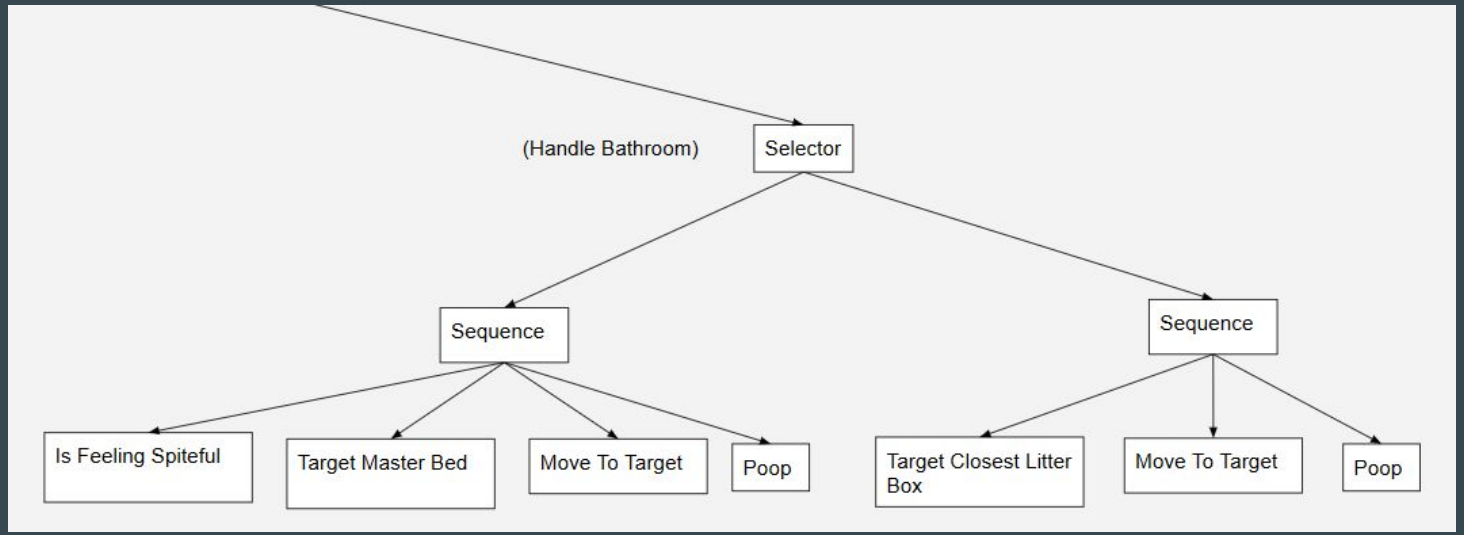
AI Planning In Games

The Plan

- How AI Planners work and when to use them
- Some tricks and optimizations
- Tips for controlling agent behaviours

State Machines and Behaviour Trees

State Machines (SM) and Behaviour Trees (BT) are commonly used to control the behaviours of individual characters.



(Part of a cat simulator behaviour tree)

State Machines and Behaviour Trees

These have a lot of advantages in games:

- Simple
- Easy to control
- Predictable
- Low resource requirements
- Easy to test and debug
- etc.

State Machines and Behaviour Trees

However they also have drawbacks:

- Every situation must be authored
- Not capable of improvising
- Changes to your game might require major AI changes
- Predictability isn't always a good thing
- etc

What is an AI Planner?

- Unlike SM and BT planners are not specifically hand authored

- Planners enable AI to solve problems using information about how the game works

- They are especially useful when your AI needs to solve problems that require multiple steps

Common Types of Planners

- Goal Oriented Action Planning (GOAP)

- Hierarchical Task Network (HTN)

(We'll be focused on planners similar to GOAP)

Games Using Planners

- F.E.A.R
- Fallout 3
- Empire: Total War
- Just Cause 2
- Deus Ex: Human Revolution
- Etc.

Common AI Planning Terms

- World state
- Actions
- Plan
- Goal

What is a World State?

- A simplified version of your game world
- Typically only holds information relevant to decision making
- This info will often be held in a simpler form than the game

Example World State

Objects In World State:

- Agent
  - Hit points: 100
  - Hunger: 45
  - Comfort: 27
- Wood
  - On fire: true
- Cat
  - Hit points: 20
  - On fire: false
- Chicken Leg
- Chair
  - Hit points: 20
  - On fire: false
- Axe

What is an Action?

- Something your agent can do to change the world state

- Sometimes the world state can affect what actions are available
  - Example: You can’t eat unless there are edible objects

What is a Plan?

- A series of actions that can be executed one after another to achieve a goal

- Can vary in how specific they get depending on your needs.

- Plans can become invalid while they are being executed
  - Need to check for failure and replan

What is a Goal?

- A condition that you want to be true in a world state.
  - Example: Agent Hunger is less than Initial Agent Hunger

- There is potentially a lot of world states that could satisfy the given condition.
  - I typically stop the search at the first world state that satisfies the goal.

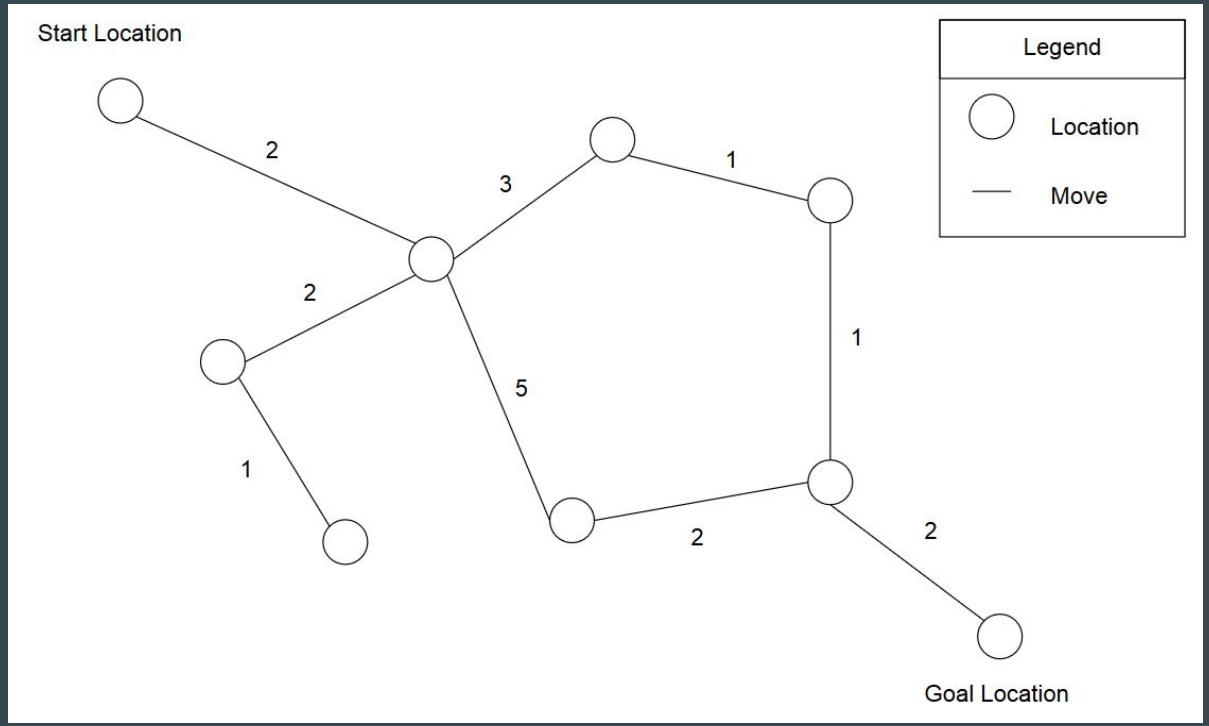
How Do Planners Work?

- Planners use very similar search techniques as A* pathfinding.

- In a sense, A* pathfinding is a very specialized form of planner.

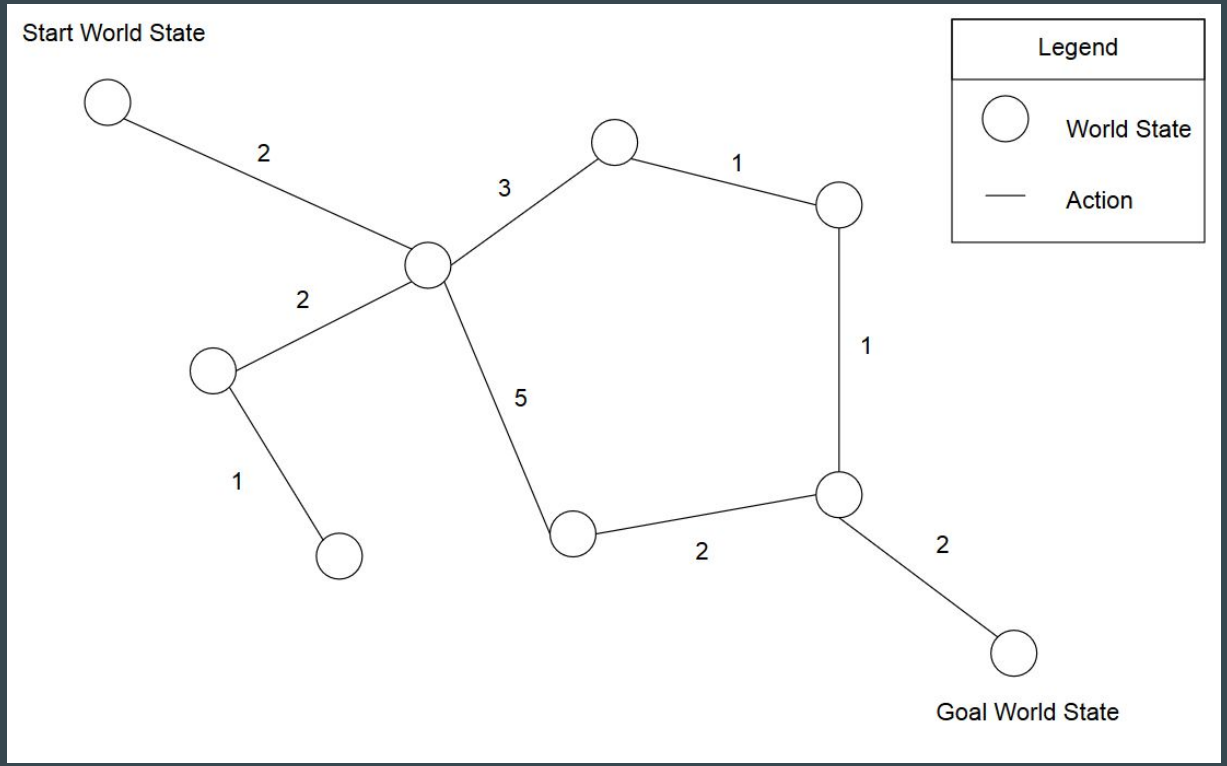
A* Pathfinding

- Distinct locations that the agent can be
- Can move between these locations
- Each move has a cost
- Use a heuristic search to produce a low cost path.



Planner

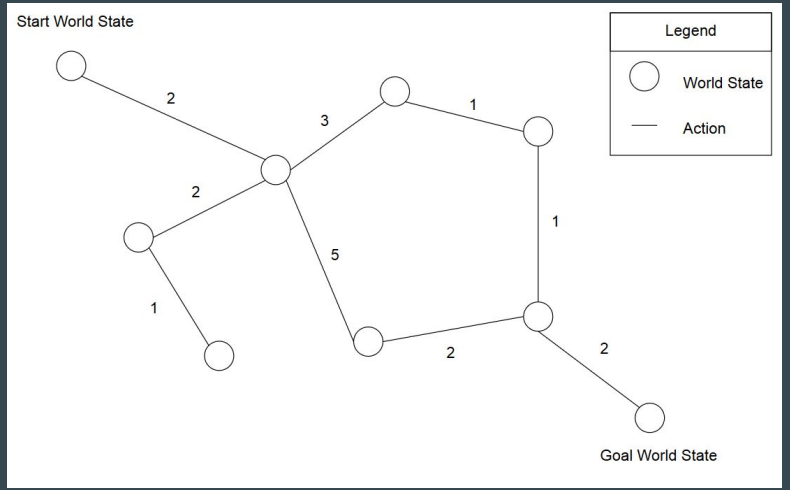
- Distinct states that the world can be in.
- Can action can produce a new world state.
- Each action has a cost
- Use a heuristic search to produce a low cost plan



Planner Search Algorithm

1. While there are states in the Queue:
    a. Pop the best state from the queue
    b. If this state satisfies the goal: break
    c. For each possible action:
        i. Create a new world state
        ii. If new state hasn't been visited, put it in the queue

2. If goal was found, construct plan by backtracking over the actions that produced the goal



Planner Search Algorithm

- You can either search forward from your start state to your goal state, or vice versa.
- Try to choose the direction that will narrow down your search the quickest

Challenges for Planner Searches

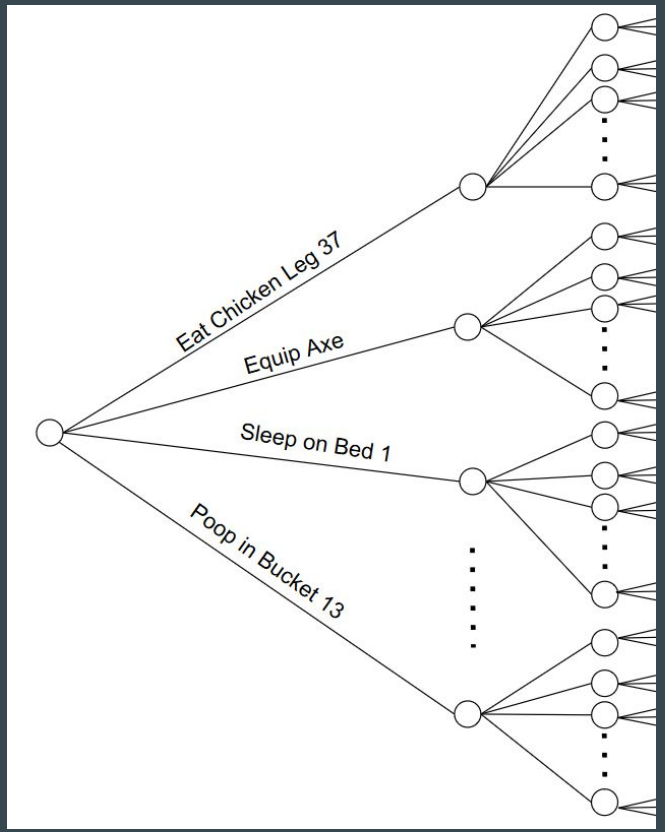
- High amount of branching

- Huge number of possible world states

Challenges: High Amount of Branching

- Every choice you can make is essentially a branch that the search needs to consider.

- The amount of possible branching can get quite high.
  - Number of Objects X Number of Actions Per Object



Challenge: Huge Number of Possible World States

- Unlike path finding, the states that the search can visit are not predetermined.
- The number of states that can be visited, can be effectively limitless.
- Need a way to terminate the search early if the search is having trouble finding the goal.

Optimizing Techniques

- Simplify your world state
- Object grouping
- Make planning more abstract
- Heuristics

Optimizing Planners: Simplify Your World State

- Ignore objects that are far away from your AI
- Only include details that are necessary for planning
- The details you chose can be different depending on the goal
- And more...

Examples:

  - Ignore beds if you want to satisfy your hunger.
  - Ignore all food if want to satisfy sleep.
  - Ignore cats if you want to go to the bathroom.
  - Choose a random subset of the objects nearby.

Optimizing Planners: Object Grouping

- When planning you don’t usually need to distinguish between objects of the same type.
  - If there are fifty burgers nearby, does it really matter what you choose to eat?

- Equivalent objects can be grouped up to greatly reduce the number of objects that need to be considered.

- You might even be able to get away with removing duplicate items completely from your world state.

Optimizing Planners: Make Planning More Abstract

- You don’t necessarily need to make every single decision when planning.

- Things can be simplified by focusing on the high level details, and then let another part of the AI figure out the rest.

- You may also be able to use abstract types to represent a wide range object types.
  - Ex. (Mashed Potatoes, Avocado, Mystery Meat, etc.) => Food

Optimizing Planners: Heuristics

- Heuristics will help narrow your search towards your goal faster

- I’ve used these two approaches:
  - Action-Goal look up table.
  - Plan tree

Heuristic Lookup Table

- 2D where action and goal pairs are given a heuristic value.

- The more appropriate an action is towards achieving a goal the lower the number.

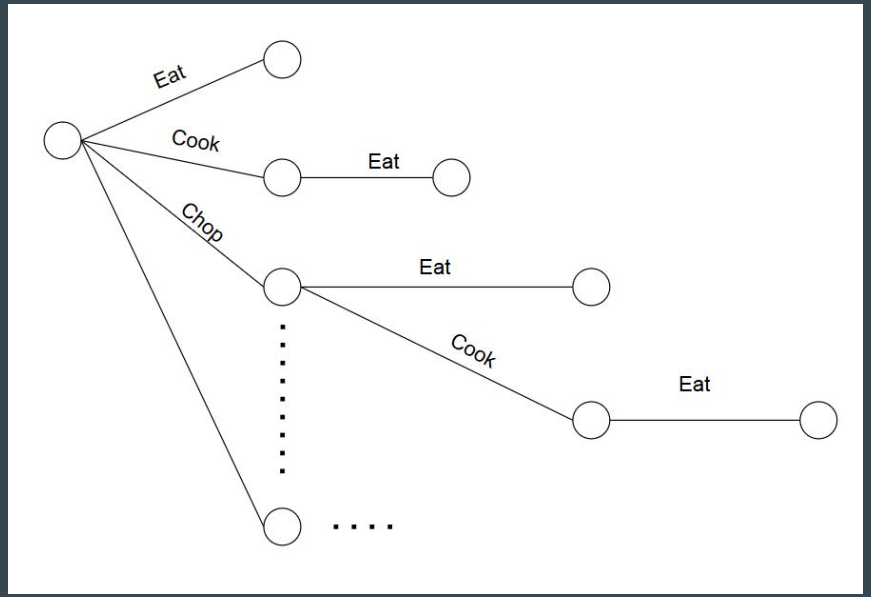
  Examples:

  - Eating action when hungry gets a really low heuristic
  - Sleeping action when hungry gets a really high heuristic.

Plan Tree

- Cache valid plans and combine them into a tree

- Planner can follow along this tree, and use it to narrow things down on the goal much faster.
  - If the action you are considering is next in the plan tree, it's guaranteed to be part of a possible valid plan.



Quick Planing using Plan Trees

- You can plan things way faster by only considering plans in the plan tree.

- One technique that works fairly well is to start with quick planing, and then move on to the “slow” variety if nothing suitable is found.

- You can precompute plans in your plan tree to make this more effective

Improving your AI With Plan Trees

- It’s easy to add on to the plan tree during runtime.
- Once you successfully plan something once, you can remember it in your plan tree forever
- Next time, it’ll be way faster to reach the same conclusion.

- It’s also possible to put plans that are “too hard” to figure out in your plan tree to give the AI more capabilities.

Plan Tree Problems

- Without careful tuning your planner could greatly favour plans in your plan tree, over unknown, better plans.

- Precomputing a wide range of common good plans can help

- Also make sure you aren't overestimating your lookup table heuristics too much if you're using both

The Unintended Results of Planning

Prototype Game

- Sandbox “Sims-like” prototype
- Every agent has stats that change over time
  - Hunger
  - Sleepiness
  - Comfort
  - etc.
- Goals typically involve satisfying these stats

Prototype Game

Agent actions can change objects properties, or even create / destroy objects

- Eating decreases hunger
- Chopping up a chair, will decrease its hit points, and eventually produce wood.
- Petting cats increases comfort

Prototype Game

Some of these actions include:

- Eat
- Sleep
- Poop
- Pick up objects
- Drop objects
- Equip tool
- Warm hands on fire
- Light things on fire
- Cook things
- Pet cats
- Chop up objects
- Wait

Prototype Game

The “best” solutions to achieve a goal sometimes require a lot of steps.

For example, potential steps to become less hungry:

1. Get axe
2. Slaughter chicken
3. Get meat
4. Chop up chair
5. Light wood on fire
6. Cook chicken
7. Eat

Prototype Game

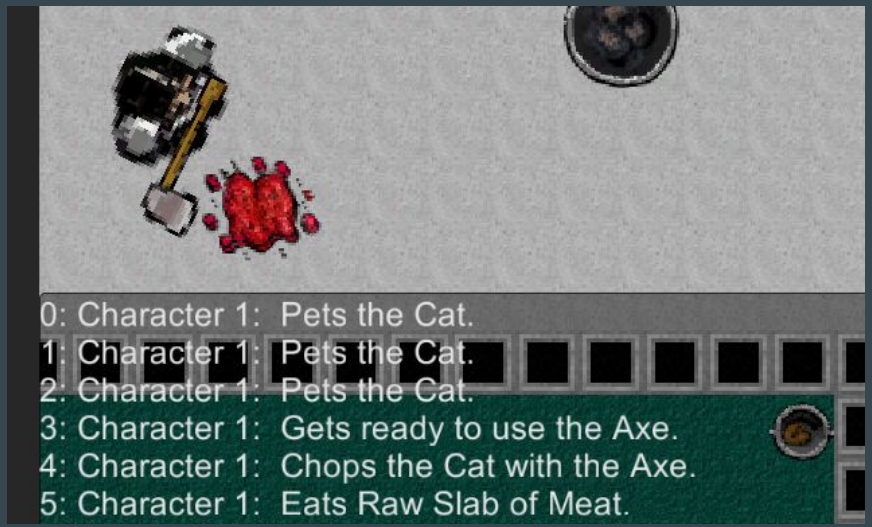
- Typically all of objects support all actions that seem to make sense.

- This sometimes had unintended results...

Prototype Game - Unintended Results

People started eating poop. Sometimes immediately after pooping.

Prototype Game - Unintended Results



People would walk up to cats and start petting them for comfort. Then kill them and eat them without hesitation.

Prototype Game - Unintended Results

As soon as the area ran out of normal food, people would immediately resort to cannibalism.

Prototype Game - Unintended Results

People occasionally cooked chicken by lighting another chicken on fire and cooking over top of it.

Influencing the Results

- Try tweaking costs of actions
- Remove or prevent actions
- Tweak the effects of actions
- Dynamic Cost Maximums

Unintended Results - Solutions

Problem:

- People started eating poop. Sometimes immediately after pooping.

Possible Solutions:

- Make the cost of eating poo really high.
- Remove hunger decrement amount from the poo.
- Remove the eat action from poo.

Unintended Results - Solutions

Problems:

- People would walk up to cats and start petting them for comfort. Then kill them and eat them without hesitation.
- Sudden cannibalism

Possible Solutions:

- Higher cost of killing things you have a social connection with?
- Higher cost of killing people and pets?
- Empathy modeling?

Unintended Results - Solutions

Problem:

- People lighting live chickens on fire. Then started cooking another chicken over the flaming chicken.

Possible Solutions:

- Don’t make chickens flammable?
- Make it so different cooking tools have different costs

Other Improvements

- It’s harder to find resources when looking for advice or ways to improve your planner.
- However, A* pathfinding can often be a source of advice / improvements.
- Ex:
  - Using heuristics to arrive at the solution faster
  - Precomputing paths to speed things up
  - Changing costs to influence path results
  - Processing path results
  - etc.

When to use Planners?

- Planners aren't suitable for every game.

- For the right games and situations though they can work out really well.

AI Planner - Pros

- AI will be capable of chaining actions together in an effective way
- Can sometimes come up with novel situations
- AI can often work with new actions and objects without additional work.
- Works well with unpredictable content like procedural and user generated content.

AI Planner - Cons

- Harder to control
- Requires more resources than FSM and BT
- Results can sometimes be harder to predict
- Harder to test and debug

Questions when Considering Using Planners

- Do you know what situations your AI needs to handle?
- Is is overwhelming to think about all of the cases your AI needs to handle?
- Will your AI require a lot of special case authoring?
- Does your AI need to work with unpredictable content?

Questions when Considering Using Planners

- Do you want your AI to be able to come up with its own solutions?
- Do you have tightly scripted events that require very specific behaviours?
- How predictable do you want your AI to be?
- What is your memory and processor budget?

Random Thoughts

- Even though the AI can be unpredictable the solutions are still logical
- Unexpected results can sometimes highlight holes in your design.
- Being surprised by your own game can be both scary and magical

Final Thoughts

- AI has the potential to open up new possibilities for gameplay and design
- It's possible that planners can help with this.
- Possibly a gateway technique towards something really cool.

Questions?

Here's my contact info if you want to get in touch:

Evan Hahn

evan.hahn@snowedin.ca

http://snowedin.ca/

@ehahnda