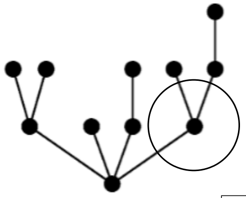
Towards a Complete AI



Complete AI

Tracks all the dependencies and interactions of the entities in your game.

Empowers Game Design

Designers can focus on high-level systems and getting the mechanics right.

Automates Many Production Tasks

Suitable AI behaviour is inferred automatically.

Asset tracking during development, including relations between entities, can be automated with AI assisted tools.

During runtime the AI uses all information it gathered during development to make the gameworld perform to the designers' specs.

What is covered here

This presentation has two parts:

#1 Lookahead: The first part introduces how lookahead can change what you can do with AI and how this tech can enhance traditional Behaviour Trees.

#2 Impact: The second part explains the impact this tech has on the production of games. It says more on the scope of the AI, how rules are central for its design, and how rules-based design streamlines the development of AI and the production of games in general.

The second section begins on page 14.

tl;dr The last slide shows the prototype AI architecture for a quick tech overview.

Original presentation

The original presentation was shown at NG23 in May 2023 to explain the basics of the technology.

You find the original presentation in the first section starting on the next page.

Contact

Stefan Gollasch – Founder Tesla Minds

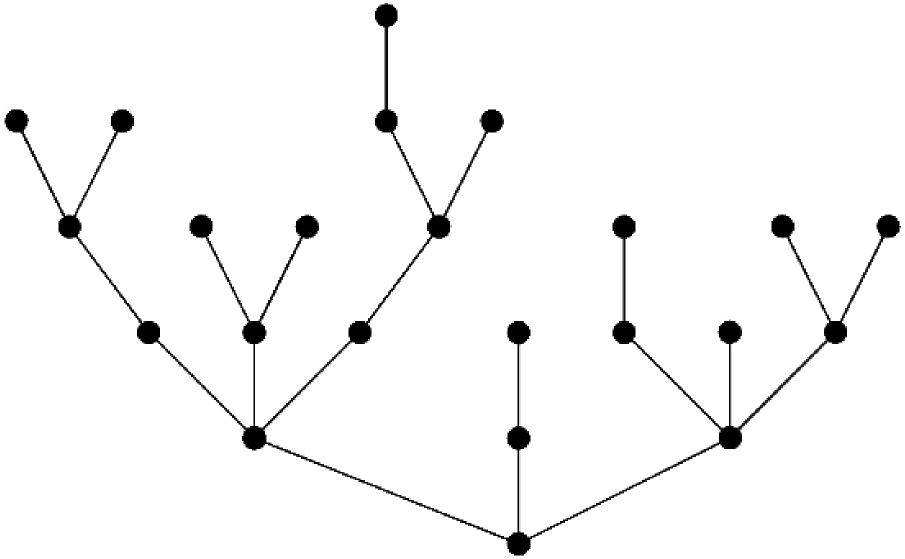
ai@teslaminds.de

www.teslaminds.de

Presentation shown at Nordic Games, Malmö May 2023

Why this technology becomes feasible

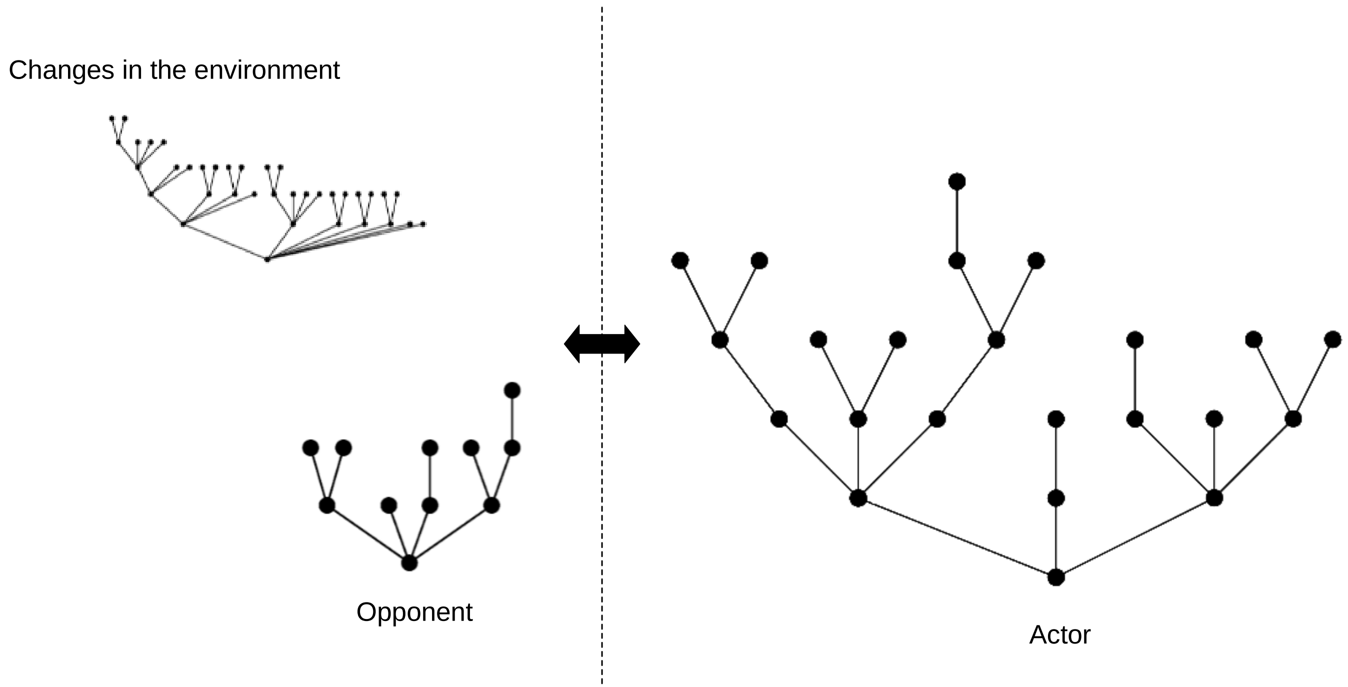
Possibilities & Lookahead



● Logic to process what to do next

— Proceed with the chosen action

Interaction



The logic what to do next and how to proceed
requires context,
for instance how the environment changes and what other actors do

Leverage

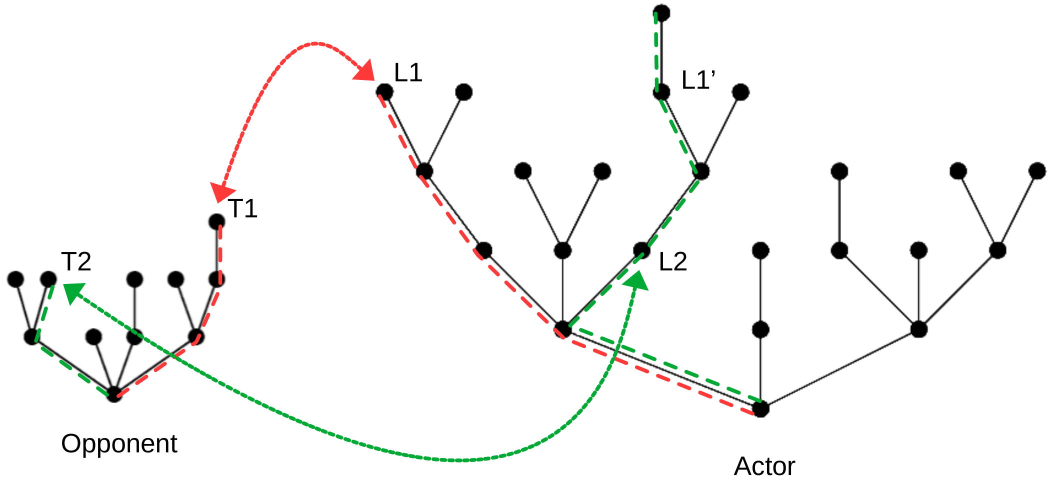
Scenario

L1 is an open area, the opponent is a sniper with an ideal vantage point at T1.

Reaching L2 forces the opponent to defend his base at T2.

With the opponent at T2, L1 can be reached before the opponent can reach T1.

"Do you employ leverage?"



Leverage logic

Rules:

#1 If the opponent is at T1 when the actor reaches L1, the actor is defeated.

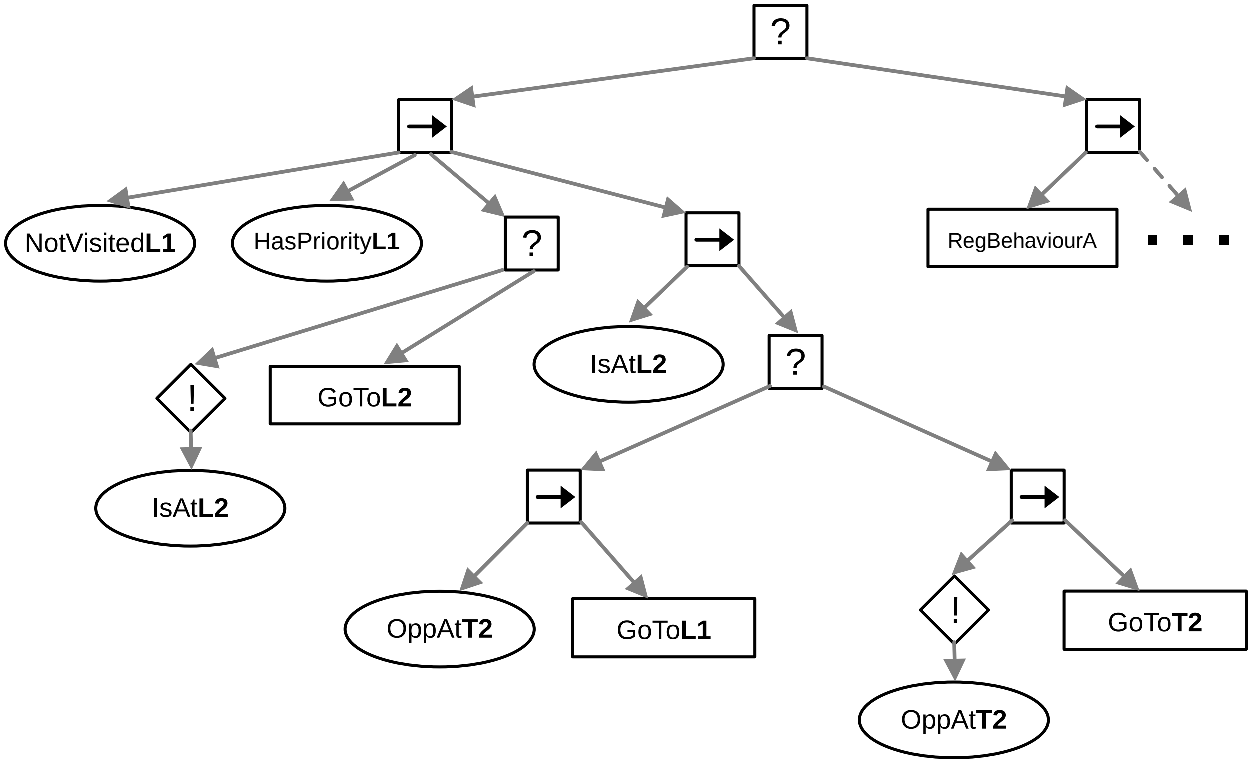
#2 Reaching L2 forces the opponent to defend T2.

#3 At T2 the opponent cannot reach T1 quickly enough to defend L1 if the actor is at L2.

Goal

How does the AI figure out to choose the better path to L1'?

Behaviour Tree



Leverage is in principle doable

But there are issues:

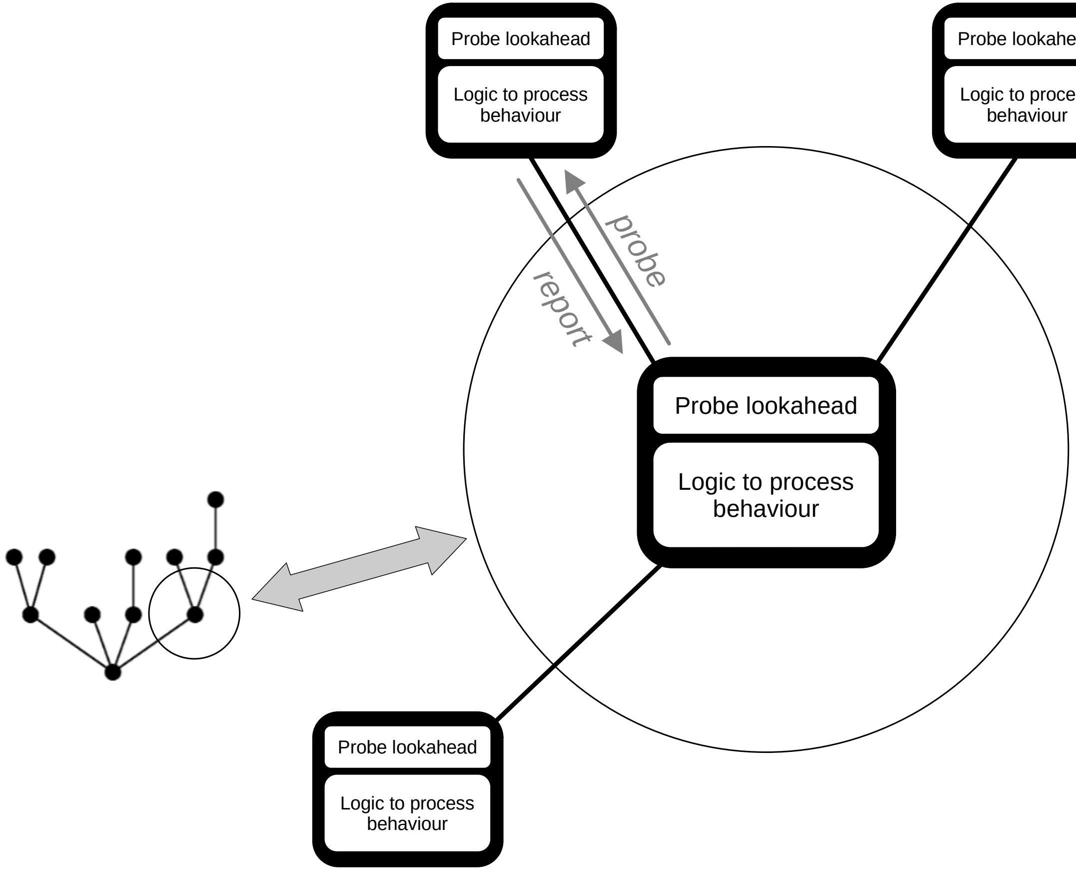
#1 Knowledge of the leverage is required from the implementer.

#2 Multiple leverage dependencies increase the complexity massively.

#3 Multiple condition checks imply increased state tracking.

#4 What happens if the movement is interrupted?

Towards Generic AI



"Small change – huge impact"

#1 Awareness of the implications of a chosen behaviour.

#2 Behaviour is context-sensitive.

#3 Recursive interconnected state processing machine.

#4 Leverage is determined automatically.

Leverage revisited

Determining the value

Each leaf node, where lookahead stops, returns a value.

This value is returned back through the chain and at branches decides which is the better choice.

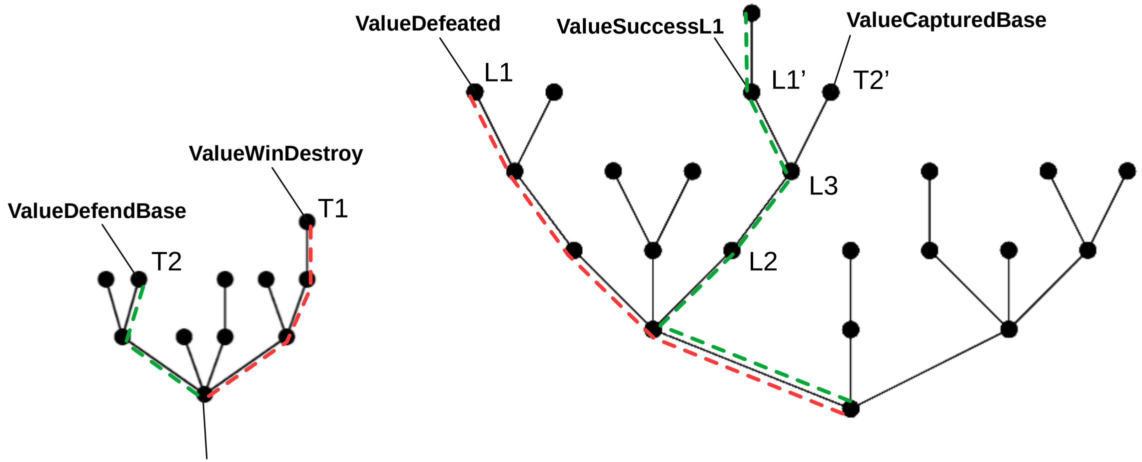
"Increased awareness vs cost"

Computing cost

#1 Each lookahead node evaluates against the assumed state it has. This requires tracking of the path leading to it and the actions taken so far.

#2 Tracking of the assumed actions of the opponent is also required.

#3 Identified leverage works as a watershed to match actions against those of the opponent.



ValueDefendBase < ValueWinDestroy

BUT

Can choose **ValueSuccessL1 | ValueCapturedBase** because opponent tracking at L2 indicates that the opponent can prevent only one of the two options. Technically the opponent has to choose between going to T2 or T1 before the actor reaches L3.

Utmost Efficiency

Mitigating the runtime cost

#1 With decisions being made based on lookahead value the actual decision making logic in a node can be very simple.

#2 Valuation of actions that happen at a node can often be based on precomputed values.

#3 Lookahead variants selected at a node can be limited to the most significant choices.

#4 The tracking of the state, what has transpired so far, can be reduced to a limited number of binary flags.

#5 Tracked data like assumed opposition behaviour can be shared and reused.

#6 Code for the lookahead can be heavily optimized, so that 100 million lookahead nodes can be processed in a second, or one million in an acceptable time frame.

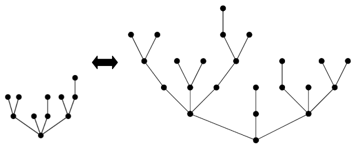
"The reason why nobody tried it so far"

Computing cost

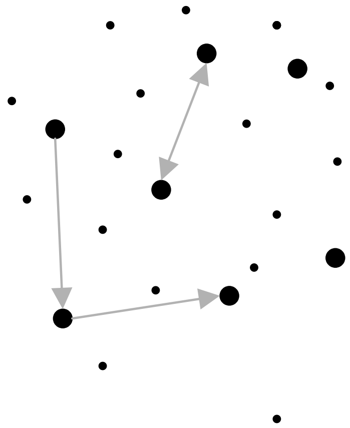
Computing the behaviour logic of n nodes of a lookahead is n-times more expensive than what a single node requires.

Moreover tracking the state for each node of a lookahead can be even more expensive because it requires data bandwidth too.

This is even more expensive if the state of the actor has to be matched against the assumed states of the environment and other actors.



Focus on the most significant actions, relations, nodes



Entity Behaviour

Relationships between entities

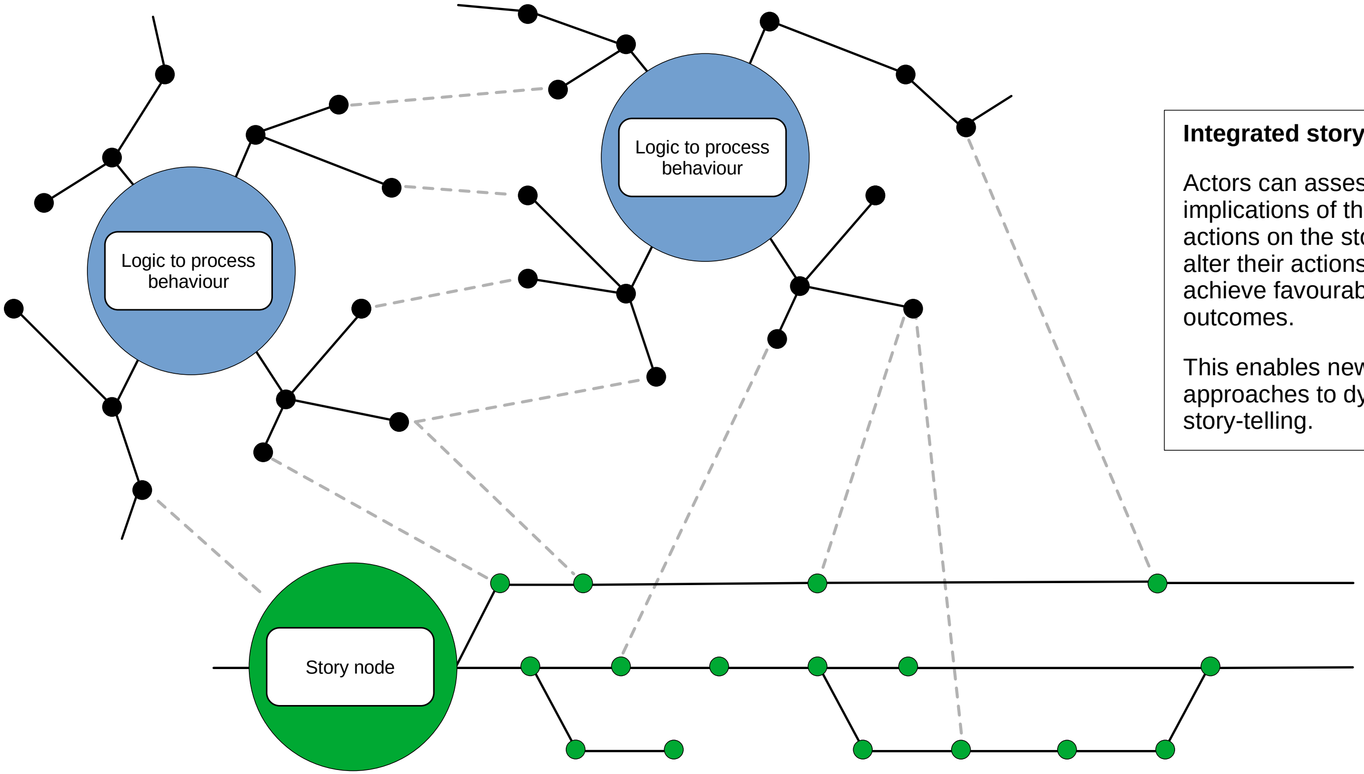
With lookahead and leverage rules that define how one entity reacts to actions of another, entities gain a measure of awareness of their environment.

#1 Entities of different type share one common interaction space.

#2 Lookahead gives entities some awareness of the implications of their choices.

#3 Leverage rules tie the actions of different entities together and align their actions logically.

"Entities gain a measure of awareness"

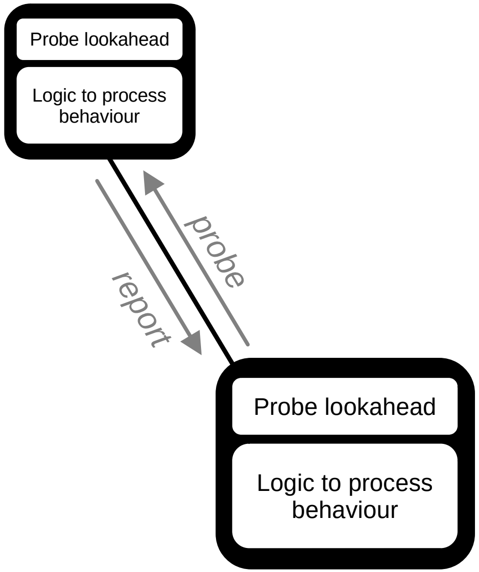


Integrated story nodes

Actors can assess the implications of their actions on the story and alter their actions to achieve favourable outcomes.

This enables new approaches to dynamic story-telling.

Development Methods and Tools



1. Implementation of behaviour trees

Initially all entities that need AI are implemented using existing behaviour tree modelling. This is done with existing tools which makes migration easy.

2. Selective addition of lookahead

For seamless integration lookahead can be implemented as a behaviour tree too .

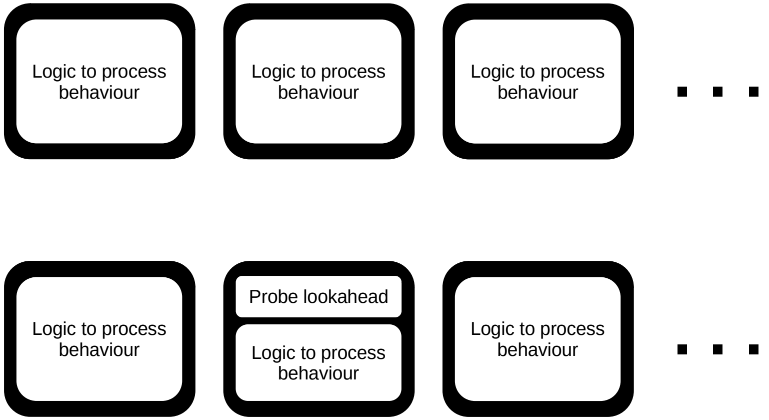
The **probe lookahead** action in the BT is then backed by a service implemented in C++.

This provides compatibility with the tools used to evaluate behaviour and the existing development environment.

3. Refinement and optimization as needed

An incremental development process that step by step interweaves the behaviour and lookahead of different entities.

This is accompanied by developing tools and high-performance algorithms as needed.



<u>Organic gameworld with unmatched reactivity</u>

#1 Entities have awareness of their environment.

#2 Actors adapt logically.

#3 Cause and effect (leverage) emerges naturally.

#4 Implemented as a rapid prototype and refined incrementally.

#5 Granularity of reactivity can be adjusted to meet performance requirements.

#6 Uses existing tools and methods, e.g. C++ and UE5.

Outlook

Organic gameworlds with unmatched reactivity

#1 Entities have awareness of their environment.

#2 Actors adapt logically.

#3 Cause and effect (leverage) emerges naturally.

Strategy games without cheats

#1 With NPCs aware of cause-and-effect, the number-crunching capacity of computers is a big advantage.

#2 Strategic planning and logistics work realistically.

#3 AI can balance and create interesting scenarios and dynamic events that keep players engaged.

Single-shard MMOs with intelligent DM

#1 Emergent systems distribute populations where needed.

#2 Players are constrained by actions rather than arbitrary limits.

Team cohesion of NPC squads

#1 Team members align and support each other.

#2 Enemy squads adapt to what the player does.

#3 The AI is aware of what matters in a fight.

Dynamic story-telling with adapting actors

#1 Actors gain awareness of potential story events.

#2 Natural competition to nudge events in a desired direction.

#3 Competitors push back while allies support.

Creative playgrounds for new game principles

#1 This AI opens up new avenues

#2 Innovative features, emergent systems, and new gameplay principles become feasible.

#3 Guidance and tutorials by the AI become a reality.

Summary

- Extension of behaviour trees
- Lookahead gives entities awareness of their environment
- Cause-and-effect emerges naturally
- Build with existing tools: BehaviourTree.CPP, C++, UE5, Groot
- Migration of existing projects and hybrid systems possible
- Organic gameworlds in which different types of entities interact
- Highly streamlined design, implementation and tests

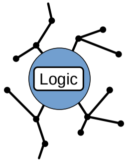
More depth on the AI, follow-up on NG23

How the AI streamlines development and compares to other recent advances in AI

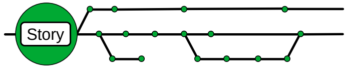
Tracking of Rules

"What does it mean?"

Many different formats of rules coexist



Behaviour of NPCs and entities



Story nodes, conditions, branches

Capitol

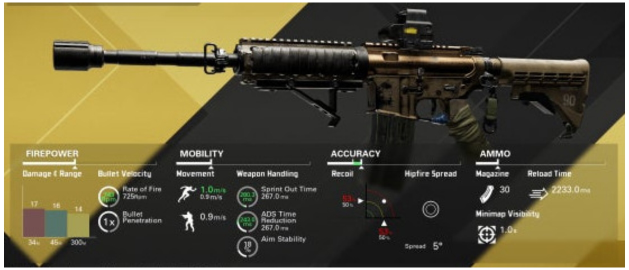
An upgrade of the City Hall.
The Capitol earns your kingdom 4000 gold per day.

» **Cost:** **10000**.
» **Requires:** Town Level 15, City Hall.

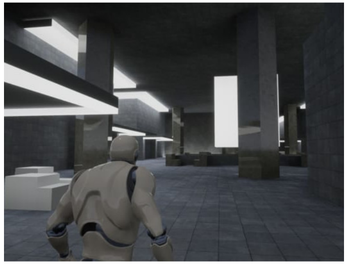
Clear-cut definitions

Data-flow & networks

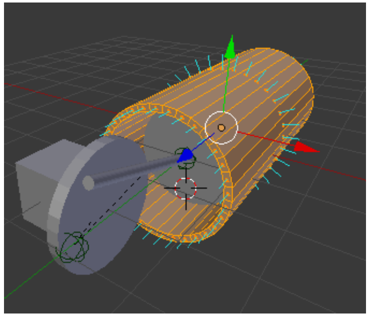
Equipment specs



Level design



Physics systems have implied rules



It doesn't replace physics systems, assets, UI and other elements

#1 Instead the AI tracks the relations between entities: how they interact and impact each other.

#2 It is the task of the designer to specify the rules, with customized tools created as needed for this purpose.

#3 A system may have inherent built-in rules, like rigid body dynamics and occlusion by opaque objects.

#4 Rules may be subsets and don't need to describe the entire world.

The AI translates all rules into a unified interaction space

#1 Natural interface between entities of different type.

#2 Designers, scripters and programmers don't have to figure out how to make different types of objects interact.

#3 Unmatched reactivity and built-in interactivity of entities.

Rules-driven AI

Emergent systems vs scripting

#1 In general this AI system is open and builds emergent systems from granular definitions (rules) that the designers specify.

#2 You can define rules that are so specific that the outcome resembles traditional scripts.

#3 While #2 is the norm today, in an open world it is only a specific case.

#4 Building persistent worlds from granular definitions is hugely empowering – if you have an AI that deals with the details.

Designers

#1 Entities, specs, properties.

#2 Mechanics and interactions.

#3 Rules that define

- **Interactions**
- **Behaviour**
- **Cause and effect**

Interface and custom tools

inspect

modify

AI designed during development

Rules Core

Lookahead and simulation, persistence

Behaviour Trees

Decision making logic

Leverage mechanic, cause and effect

Entity and story logic

Rules accumulated and iterated during development

Rules Design
Mechanics
interactions
behaviour
...

Rules Narrative
Story nodes
conditions
branches
...

Rules Other
Physics
legacy systems
runtime contraints
...

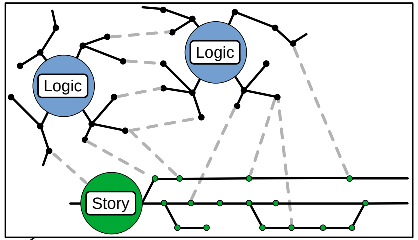
Serves as the current state of the AI

Can be inspected and modified with tools providing access to the AI state

Active observation

Deployment

Runtime



Active observation

Narrative designers

Interface and custom tools

#1 Story nodes, conditions.

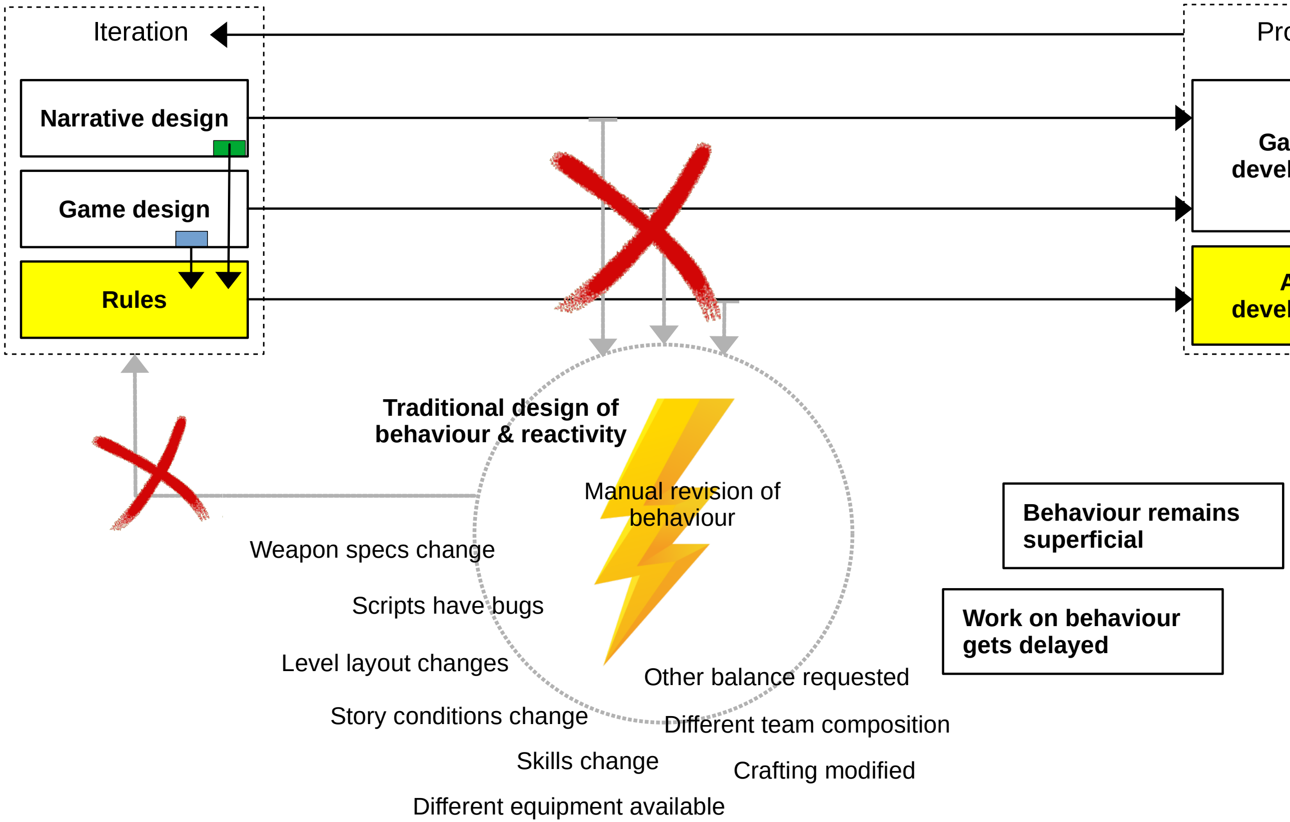
#2 Branches and narrative rules.

"Granular design, test, iteration"

Structural simulation with no/limited asset rendering

Designers can probe the runtime behaviour, including lookahead, outcomes and the interactions that occur

Huge Impact on Game Development & Design



Behaviour emerges from rules automatically

Instead of deferment of the creation of behaviours to the end of the development cycle, it can drive the development.

Rules continuously refine the reactivity and invite the designers to observe their impact.

Most of the tech dependent revision cycles of the behaviour become redundant, leaving the team free to focus on what matters.

Enhanced Reach of What a Team Can Achieve

<u>Previously</u>

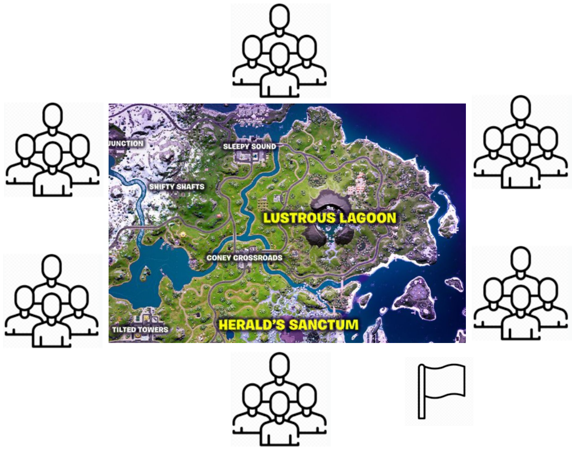
#1 Large effort spent on dealing with the technical issues of creating behaviours.

#2 The AI remains less ambitious because advanced behaviour & reactivity is not in reach.

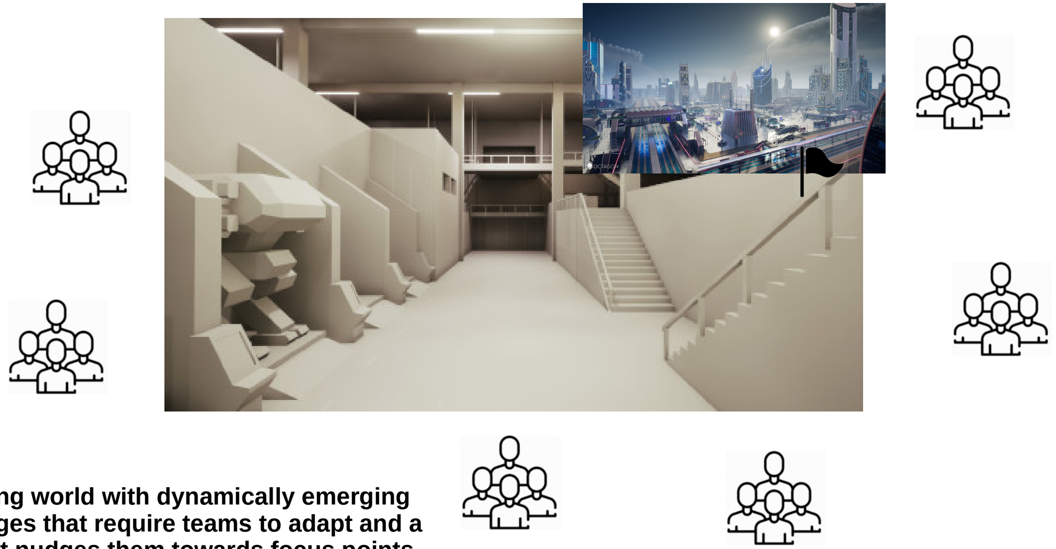
<u>Now</u>

#1 Team focuses on adaptable behaviour in a reactive environment.

#2 Team has the capability to stage competitions in highly complex settings with many different trajectories.



Straightforward squad or arena shooter



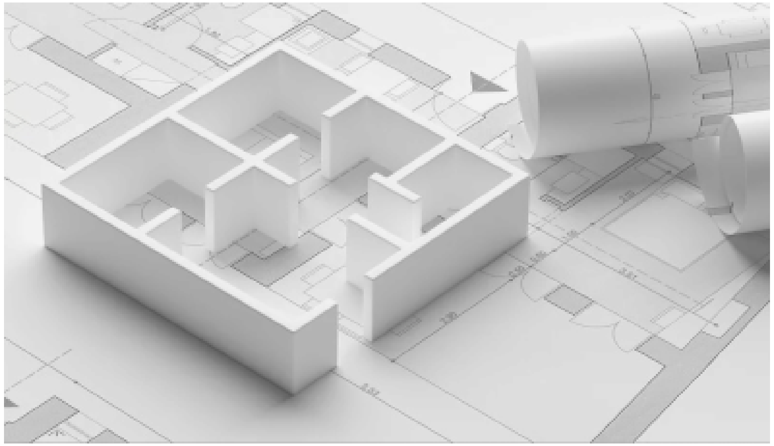
Evolving world with dynamically emerging challenges that require teams to adapt and a DM that nudges them towards focus points

Hybrid Systems

<u>You can focus on any subset of rules</u>

#1 If only specific aspects of behaviour and reactivity should be handled by the AI, work can be limited in scope.

#2 Rules can be defined that only relate to the tasks the AI should fulfil.

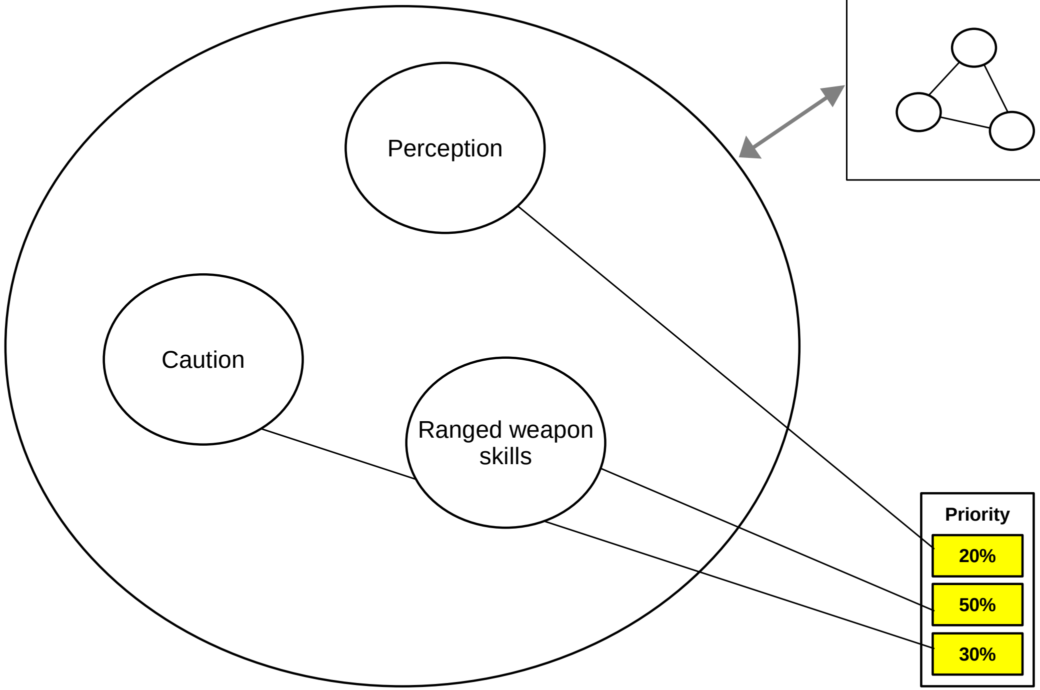


If you only desire the AI to track walls and their impact on occlusion and visibility for the actors in a level the AI can do just this.

Even a relatively simple subsystem like this can yield highly informative data what happens in a level.

Such subsystems can be augmented and customized to yield AI assisted insight in what interests you most. It is technically AI interpreted data that can be processed in a variety of ways and allows to make links visible that a human cannot perceive.

AI Focus



Complex internal model

Limit by rules

#1 Designers can specify desired skill levels.

#2 Complex mental models can adjust naturally to all kinds of situations.

#3 For example specific stimuli can enhance skills, motivation, energy and such.

Limit by computing constraints

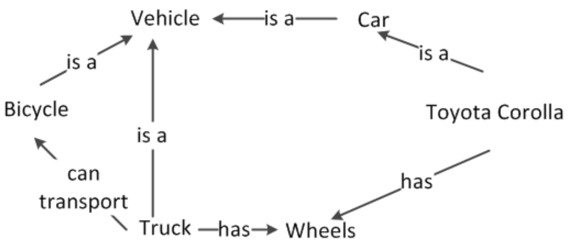
#1 The AI can automatically adjust the granularity of its processing.

#2 Time slices can be assigned to specific functions.

#3 Resulting behaviour follows defined priorities.

#4 Valuable tool to provide the best AI under variable real-time constraints.

Comparison to Contemporary AI Advances

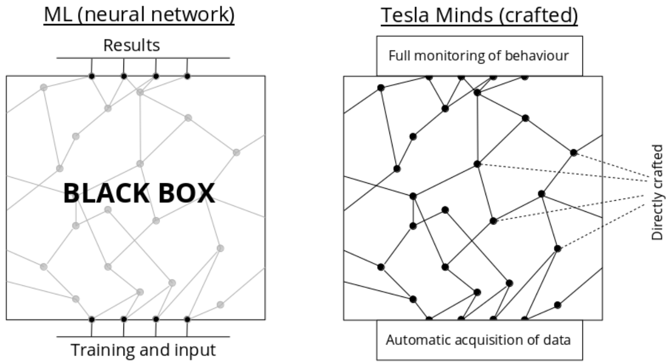


In principle recent advances like chatGPT use semantic networks (image above). To reach their advanced features they employ highly sophisticated language models, which are in a way rules.

Technically chatGPT et al are backed by neural networks that require massive amounts of computing power to train the language models on very large data volumes, like text on the web.

It is an expensive and not fully controlled process that makes the product flawed and inefficient, albeit not useless. The neural network is akin to a black box (image right, left side).

The tech itself is too inefficient to be useful in games that need smart behaviour, adaptability, and reactivity.



This AI also uses rules that could be described by semantic networks.

There are much more efficient ways to implement rules-based systems. The AI is backed by processes implemented in C++, with highly optimized algorithms evaluating only what is necessary.

At its core are methods that manage large rulesets on the one hand and employ these to arrive at logical behaviour in a given situation. Critical is the ability to determine leverage and to know what is important and what not. The key for this are the innovations on page 9. Lookahead is important for the AI to gain insight of what can happen in a specific situation, which makes it technologically a key.

The complexity of the real world is acknowledged by providing the designers with customized tools and interfaces to edit the rules that matter for their application.

In a way we use human intelligence and direct modelling backed up by sophisticated lookahead algorithms to create artificial intelligence.

It is direct, immediate, high-performance computing that is exponentially faster than neural networks (image left, right side).

Work required

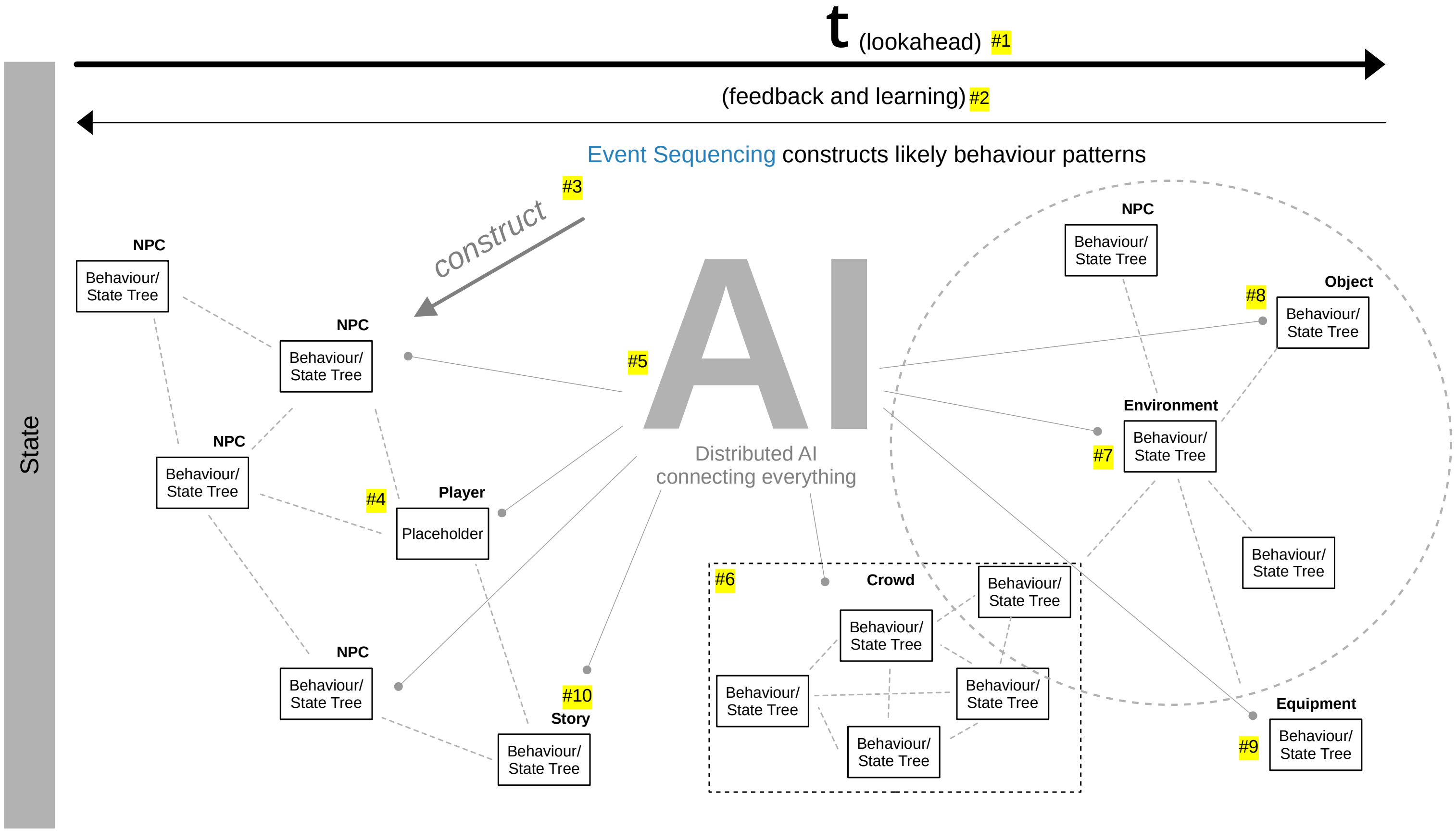
Important: doable vs feasible

→ The features presented in this tech overview are doable in the sense that we have the technological foundation to implement these.

→ However what can be implemented in a specific real world project needs to be assessed and negotiated based on the tasks at hand.

In general the features in the original presentation at the beginning are relatively straightforward to implement. In contrast the complete development cycle presented in the second part requires prototyping to arrive at the full functionality. How much is feasible depends very much on the complexity and demands of a specific project.

Prototype AI Architecture



Behaviour emerges from rules automatically

#1 Lookahead determines possible behaviour

#2 The data and patterns are fed back

#3 NPC behaviour is constructed automatically and based on the interactions that can happen

#4 Competitive and cooperative interaction between NPCs/Players is factored into the behaviour

#5 Distributed AI written in C++ connected to an entity component system; it makes the entire game world organic, systemic and intelligent

#6 Seamlessly integrated crowd behaviour

#7 Reactive environment that interacts with other entities and exhibits its own behaviour; AI placement of objects and construction of the environment is possible

#8 Objects with context-sensitive properties

#9 Equipment that affects behaviour

#10 AI story-telling as a mixture of crafted and generated content

Such an AI system has to solve formidable technological challenges to exceed existing tech

#11 Orders of magnitude more efficient lookahead to make this AI feasible in real time

#12 Sophisticated feedback and learning architecture that the AI's features improve beyond crude and easily as artificial discernible behaviour

#13 Intelligent backend architecture to master abstraction and enable uniform rulebased systems