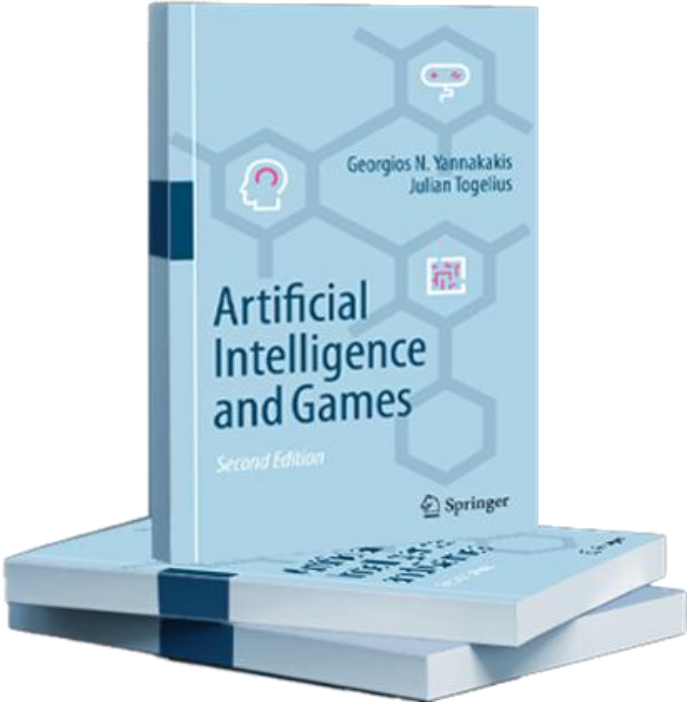
Artificial Intelligence and Games

Play



Artificial Intelligence and Games

A Springer Textbook | By Georgios N. Yannakakis and Julian Togelius

About the Book | Table of Contents | Lectures | Exercises | Resources

About the Book

Second Edition Published!

Welcome to the Artificial Intelligence and Games book (2nd edition) published with Springer Nature in 202
the first comprehensive textbook on the application and use of artificial intelligence (AI) in, and for, games
book will be used by educators and students of graduate or advanced undergraduate courses on game A
practitioners at large.

Readings: Part II

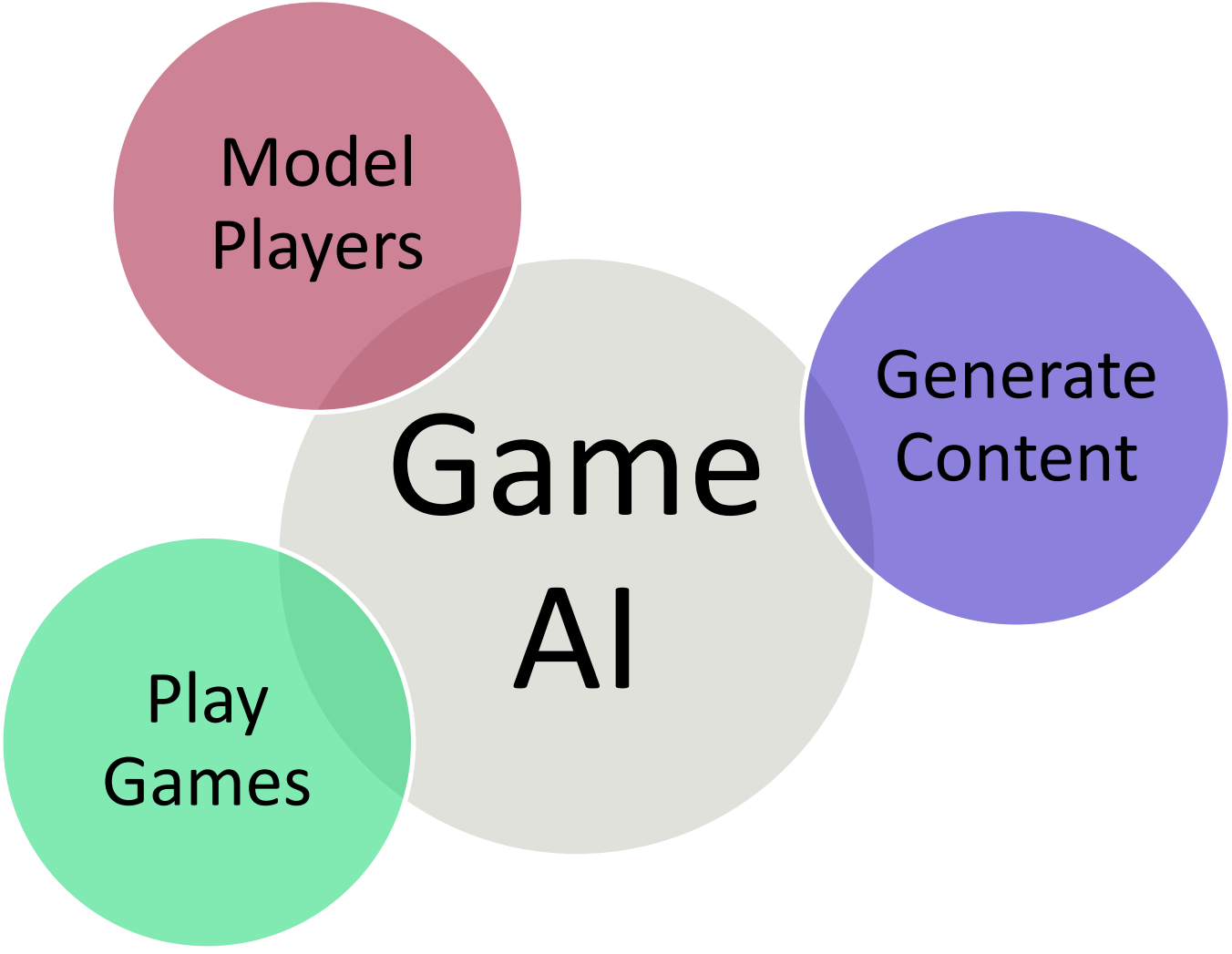
gameaibook.org

Reminder: Artificial Intelligence and Games

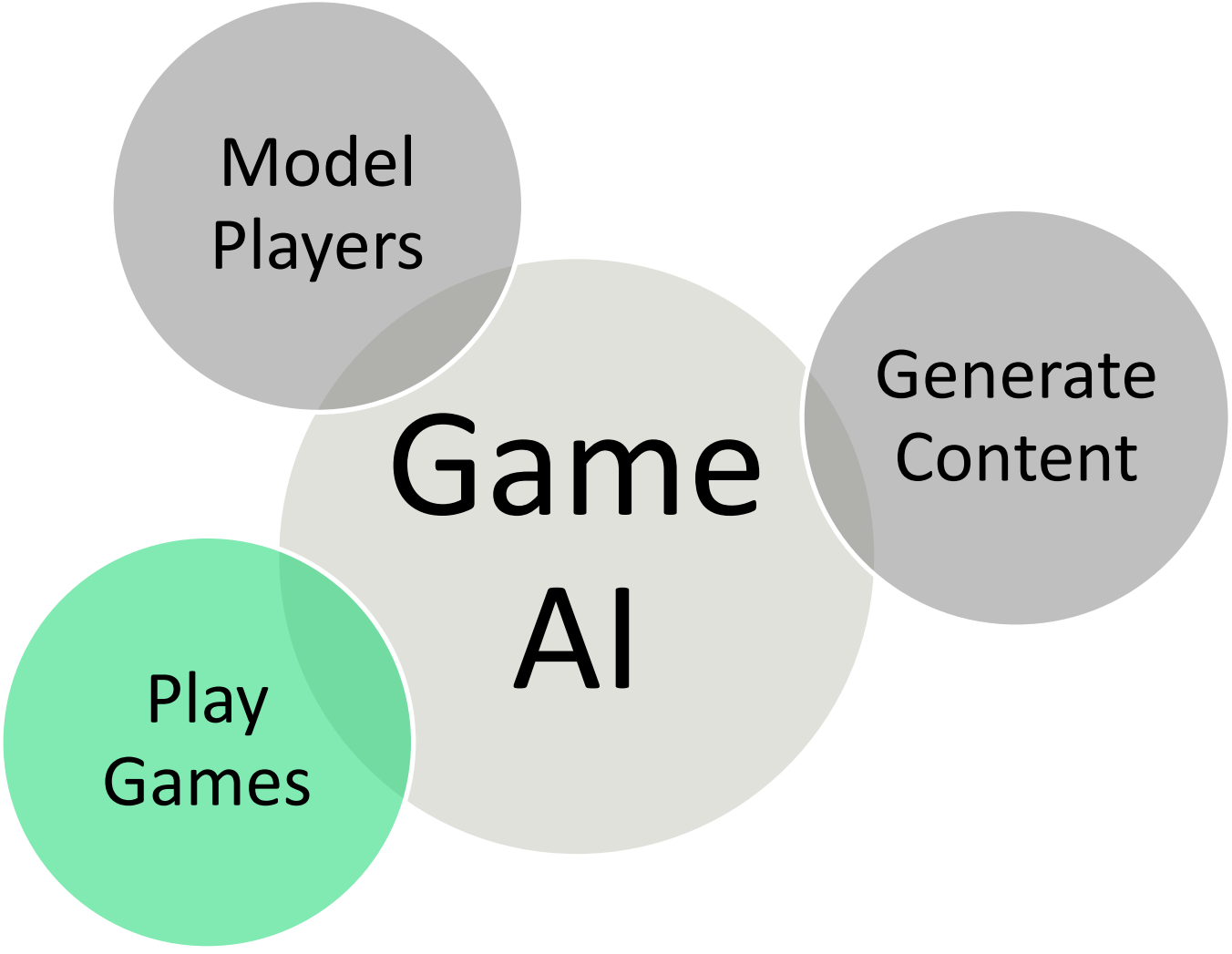
Making **computers** able to do things which currently only **humans** can do **in games**

What do humans do with games?

- **Play** them
- Study them
- Build content for them
  - levels, maps, art, characters, missions…
- Design and develop them
- Do marketing
- Make a statement
- Make money!



G. N. Yannakakis and J. Togelius, "**Artificial Intelligence and Games**," Springer Nature, 2025.



G. N. Yannakakis and J. Togelius, "**Artificial Intelligence and Games**," Springer Nature, 2025.

Chapter 4: Playing Games

Why use AI to Play Games?

- Playing to **win** vs playing for **experience**
  - For experience: human-like, “fun”, believable, predictable...?
- Playing in the **player role** vs. playing in a **non-player role**

	Player	Non-Player
Win	**Use Cases** Games as AI Testbeds AI that Challenges Players *RL Benchmarking* *General Game Playing* **Examples** TD-Gammon, Chinook, Deep Blue, Watson, AlphaGo, AlphaStar	**Use Cases** Game Balancing Playing Roles that Humans Would not (Want to) Play **Examples** Rubber Banding in Racing Games
Experience	**Use Cases** *Content Evaluation*, *Multiplayer Team Bots*, *Ghosts and Personas*, *Game Testing, Demo Mode* **Examples** Game Turing Tests (2kBot Prize) *Drivatar* in *Forza* Series	**Use Cases** Believable & Human-like Agents, *Adversarial NPCs*, *Cooperative NPCs* **Examples** Ellie in *The Last of Us,* Alyx in *Half Life 2,* Elites in *Halo* Series

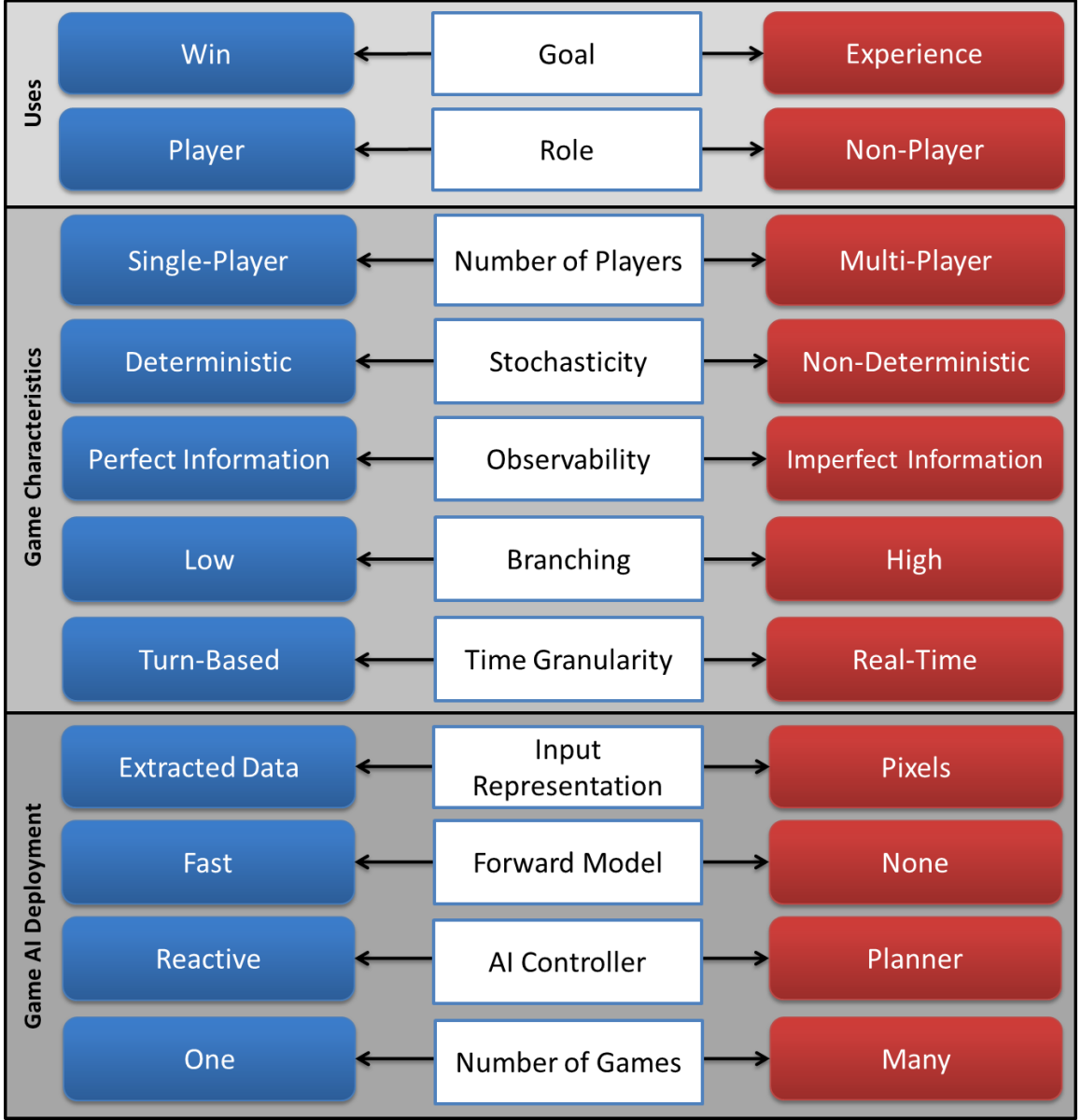
	Player	Non-Player
Win	**Use Cases** Games as AI Testbeds AI that Challenges Players *RL Benchmarking* *General Game Playing* **Examples** TD-Gammon, Chinook, Deep Blue, Watson, AlphaGo, AlphaStar	**Use Cases** Game Balancing Playing Roles that Humans Would not (Want to) Play **Examples** Rubber Banding in Racing Games
Experience	**Use Cases** *Content Evaluation*, *Multiplayer Team Bots*, *Ghosts and Personas*, *Game Testing, Demo Mode* **Examples** Game Turing Tests (2kBot Prize) *Drivatar* in *Forza* Series	**Use Cases** Believable & Human-like Agents, *Adversarial NPCs*, *Cooperative NPCs* **Examples** Ellie in *The Last of Us,* Alyx in *Half Life 2,* Elites in *Halo* Series

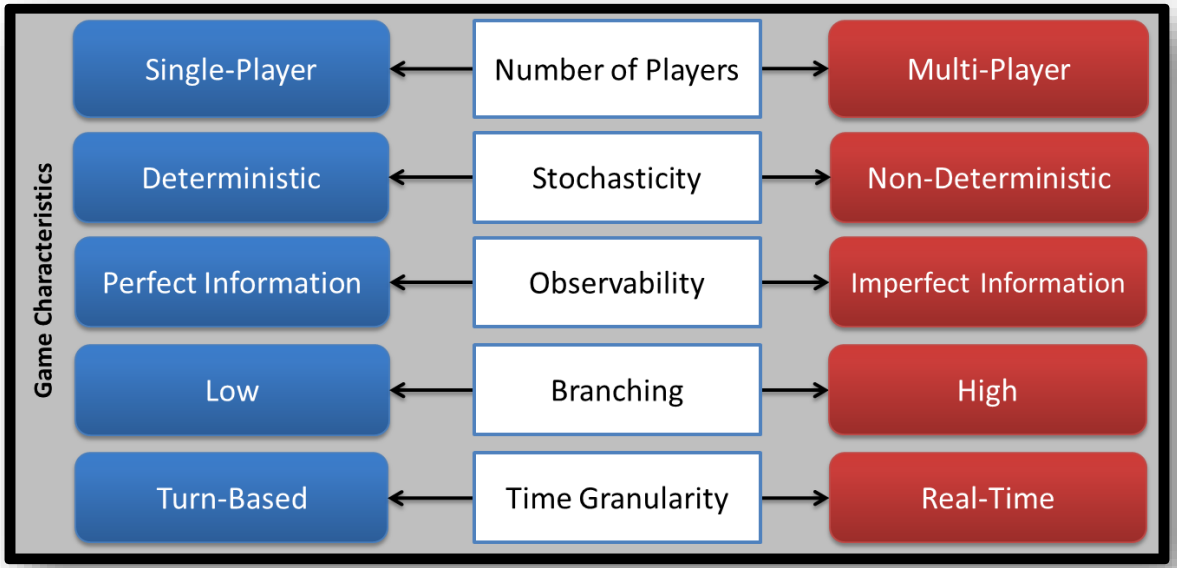
	Player	Non-Player
Win	**Use Cases** Games as AI Testbeds AI that Challenges Players *RL Benchmarking* *General Game Playing* **Examples** TD-Gammon, Chinook, Deep Blue, Watson, AlphaGo, AlphaStar	**Use Cases** Game Balancing Playing Roles that Humans Would not (Want to) Play **Examples** Rubber Banding in Racing Games
Experience	**Use Cases** *Content Evaluation*, *Multiplayer Team Bots*, *Ghosts and Personas*, *Game Testing, Demo Mode* **Examples** Game Turing Tests (2kBot Prize) *Drivatar* in *Forza* Series	**Use Cases** Believable & Human-like Agents, *Adversarial NPCs*, *Cooperative NPCs* **Examples** Ellie in *The Last of Us,* Alyx in *Half Life 2,* Elites in *Halo* Series

	Player	Non-Player
Win	**Use Cases** Games as AI Testbeds AI that Challenges Players *RL Benchmarking* *General Game Playing* **Examples** TD-Gammon, Chinook, Deep Blue, Watson, AlphaGo, AlphaStar	**Use Cases** Game Balancing Playing Roles that Humans Would not (Want to) Play **Examples** Rubber Banding in Racing Games
Experience	**Use Cases** *Content Evaluation*, *Multiplayer Team Bots*, *Ghosts and Personas*, *Game Testing, Demo Mode* **Examples** Game Turing Tests (2kBot Prize) *Drivatar* in *Forza* Series	**Use Cases** Believable & Human-like Agents, *Adversarial NPCs*, *Cooperative NPCs* **Examples** Ellie in *The Last of Us,* Alyx in *Half Life 2,* Elites in *Halo* Series

	Player	Non-Player
Win	**Use Cases** Games as AI Testbeds AI that Challenges Players *RL Benchmarking* *General Game Playing* **Examples** TD-Gammon, Chinook, Deep Blue, Watson, AlphaGo, AlphaStar	**Use Cases** Game Balancing Playing Roles that Humans Would not (Want to) Play **Examples** Rubber Banding in Racing Games
Experience	**Use Cases** *Content Evaluation*, *Multiplayer Team Bots*, *Ghosts and Personas*, *Game Testing, Demo Mode* **Examples** Game Turing Tests (2kBot Prize) *Drivatar* in *Forza* Series	**Use Cases** Believable & Human-like Agents, *Adversarial NPCs*, *Cooperative NPCs* **Examples** Ellie in *The Last of Us,* Alyx in *Half Life 2,* Elites in *Halo* Series

A Taxonomy of Play


Uses
Win
Goal
Experience
Player
Role
Non-Player
Game Characteristics
Single-Player
Number of Players
Multi-Player
Deterministic
Stochasticity
Non-Deterministic
Perfect Information
Observability
Imperfect Information
Low
Branching
High
Turn-Based
Time Granularity
Real-Time
Game AI Deployment
Extracted Data
Input Representation
Pixels
Fast
Forward Model
None
Reactive
AI Controller
Planner
One
Number of Games
Many



Game Characteristics

Game (and AI) Design Considerations

When designing AI

- It is crucial to know the **characteristics of the game** you are playing and
- the **characteristics of the algorithms** you are about to design

These collectively determine what type of algorithms can be effective

Characteristics of Games

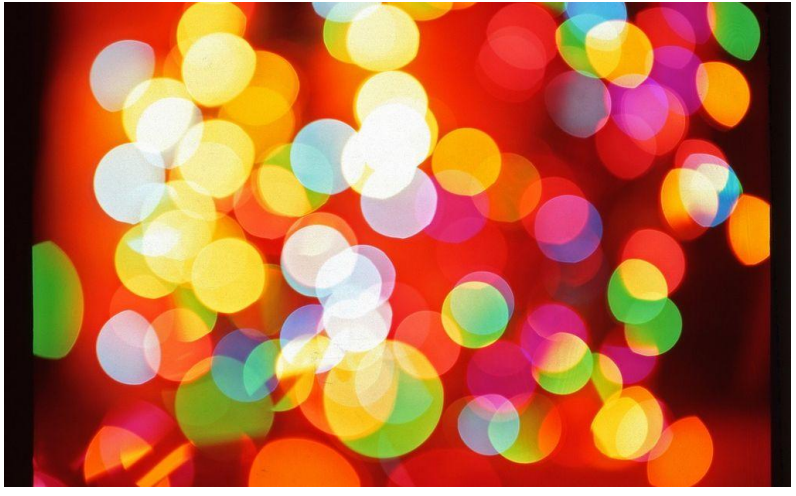
- Number of Players
  - Type: Adversarial? Cooperative? Both?
- Stochasticity
- Observability
- Branching
- Time Granularity

Number of Players

- **Single-player** – e.g. puzzles and time-trial racing
- **One-and-a-half-player** – e.g. campaign mode of an FPS with nontrivial NPCs
- **Two-player** – e.g. Chess, Checkers and *Spacewar!*
- **Multi-player** – e.g. *League of Legends* (Riot Games, 2009), the *Mario Kart* (Nintendo, 1992–2014) series and the online modes of most FPS games.

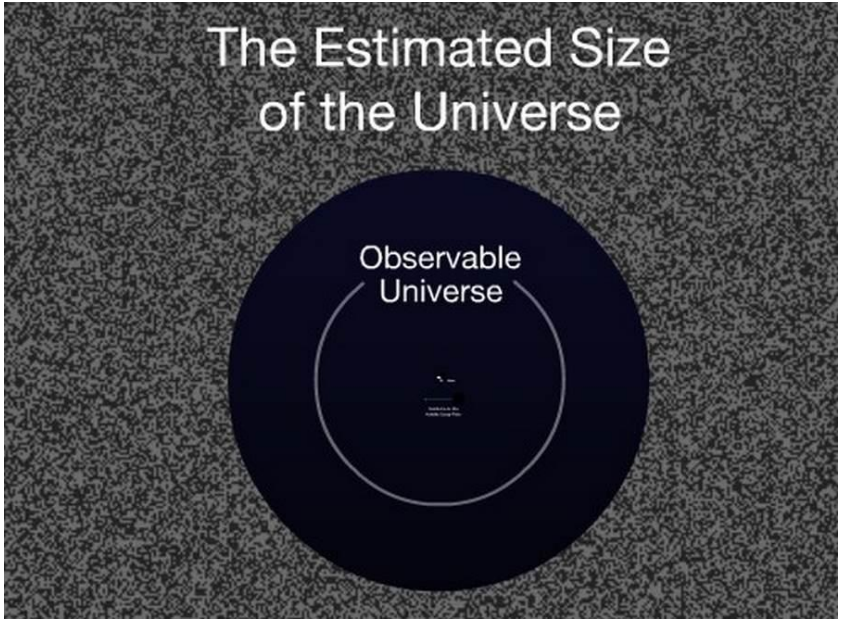
Stochasticity

- The degree of randomness in the game
- Does the game violate the Markov property?
  - **Deterministic** (e.g. *Pac-Man, Go, Atari 2600 games*)
  - **Non-deterministic** (e.g. *Ms Pac-Man*, *StarCraft, …*)



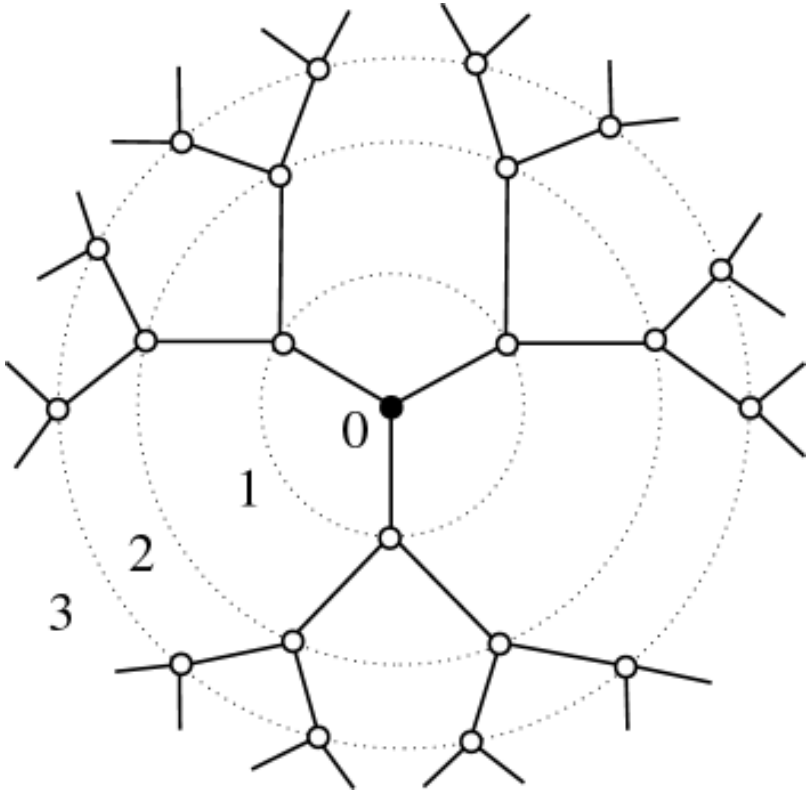
Observability

- How much does our agent know about the game?
  - **Perfect** Information (e.g. *Zork*, *Colossal Cave Adventurer*)
  - Imperfect (**hidden**) Information (e.g. *Halo*, *Super Mario Bros*)



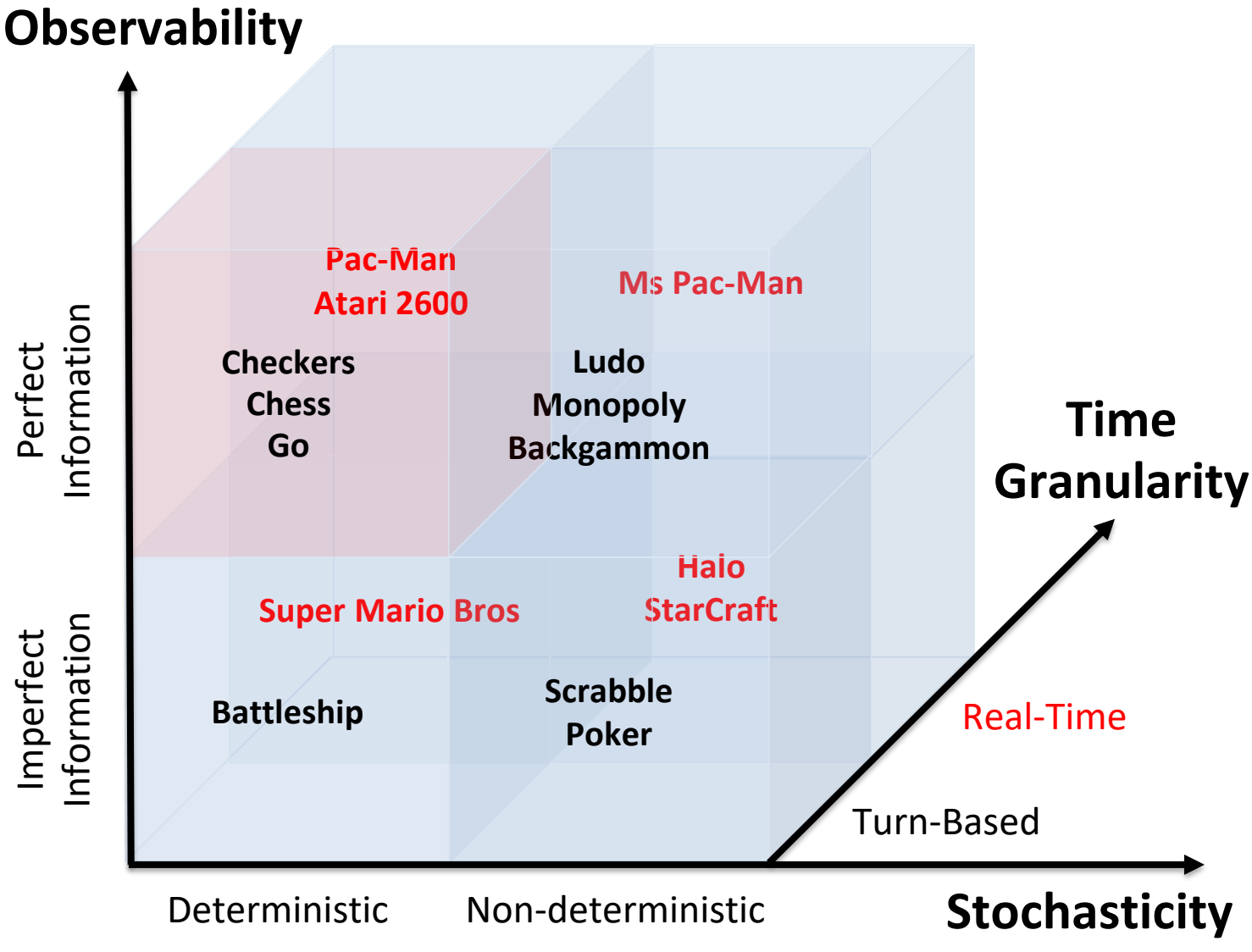
Action Space and Branching Factor

- How many actions are there available for the player?
  - From **two** (e.g. *Flappy Bird*) to **many** (e.g. *StarCraft*)….



Time Granularity

- How many turns (or ticks) until the end (of a session)?
  - **Turn-based** (e.g. Chess)
  - **Real-time** (e.g. *StarCraft*)


Observability
Perfect Information
Imperfect Information
Pac-Man
Atari 2600
Ms Pac-Man
Checkers
Chess
Go
Ludo
Monopoly
Backgammon
Halo
StarCraft
Super Mario Bros
Battleship
Scrabble
Poker
Time Granularity
Real-Time
Turn-Based
Deterministic
Non-deterministic
Stochasticity

Characteristics of Games: Some Examples

Chess

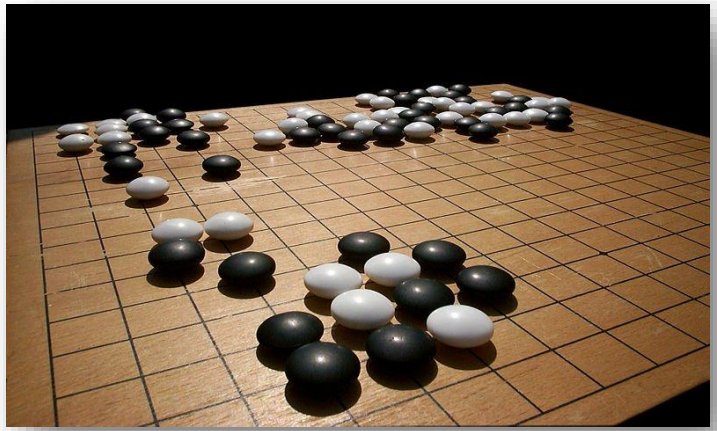
- Two-player adversarial, deterministic, fully observable, bf ~35, ~70 turns

Go

- Two-player adversarial, deterministic, fully observable, bf ~350, ~150 turns

Backgammon

- Two-player adversarial, stochastic, fully observable, bf ~250, ~55 turns



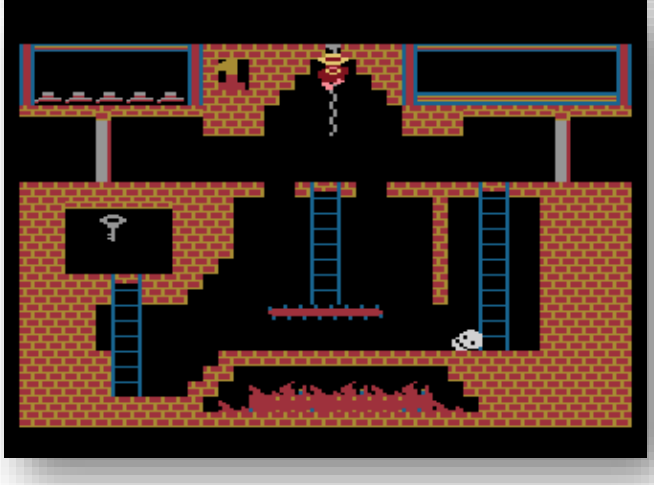
Characteristics of Games: Some Examples

Frogger (Atari 2600)

- 1 player, deterministic, fully observable, bf 6, hundreds of ticks

Montezuma's revenge (Atari 2600)

- 1 player, deterministic, partially observable, bf 6, tens of thousands of ticks



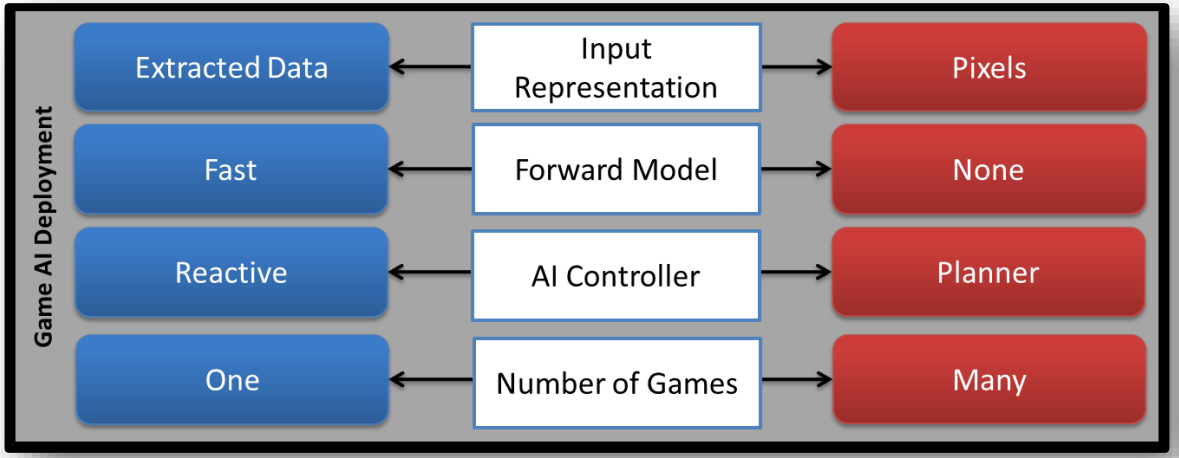
Characteristics of Games: Some Examples

Halo series

- 1.5 player, deterministic, partially observable, bf ??, tens of thousands of ticks

StarCraft

- 2-4 players, stochastic, partially observable, **bf > a million**, tens of thousands of ticks



Game AI Deployment

Characteristics of AI Algorithm Design

Key questions

- How is the game state represented?
- Is there a forward model available?
- Do you have time to train?
- How many games are you playing?

Input Representation

- Games **differ** wither regards to their output
  - Text adventures → Text
  - Board games → Positions of board pieces
  - Graphical video games → Moving graphics and/or sound
- The same game can be represented in different ways!
  - The representation matters greatly to an algorithm playing the game
- Example: Representing a racing game
  - First-person view out of the windscreen of the car rendered in 3D
  - Overhead view of the track rendering the track and various cars in 2D.
  - List of positions and velocities of all cars (along with a model of the track)
  - Set of angles and distances to other cars (and track edges)

Forward Model

- A forward model is a simulator of the game
  - Given s and $\alpha \rightarrow s'$
- Is the model fast? Is it accurate?
- **Tree search** is applicable *only* when a forward model is available!

What if…

- We **don’t have** a model (or a bad or slow model), but we have training time, what do we do?
- Train function **approximators** to select actions or evaluate states
- For example, deep neural networks

…using gradient descent…

…or evolution

Life without a forward model…

- Sad…!
- We could learn a direct mapping from state to action
- Or some kind of forward model
- Even a simple forward model could be useful for shallow searches, if combined with a state value function

AI Controller

AI distinction with regards to time:

- AI that decides what to do by examining possible actions and future states—e.g. tree search (**reactive**)
- AI that **learns** a model (such as a policy) over time—i.e., machine learning (**planner**)

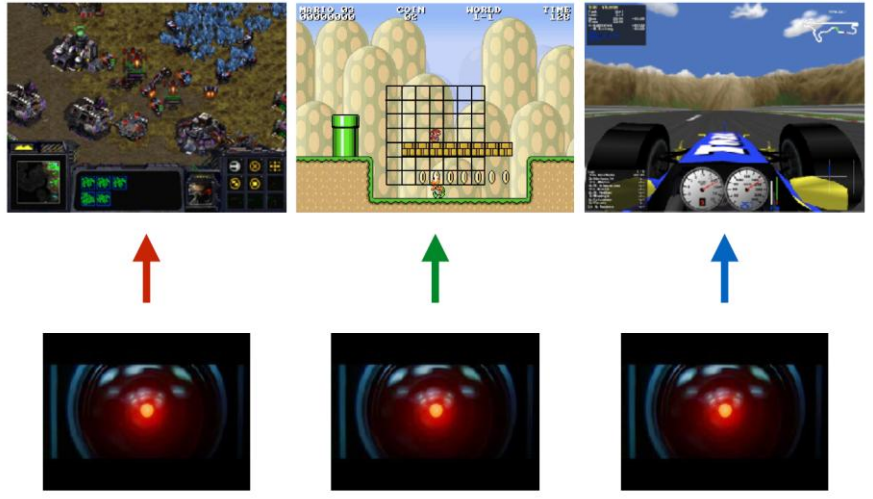
Number of Games

Will AI play one game?

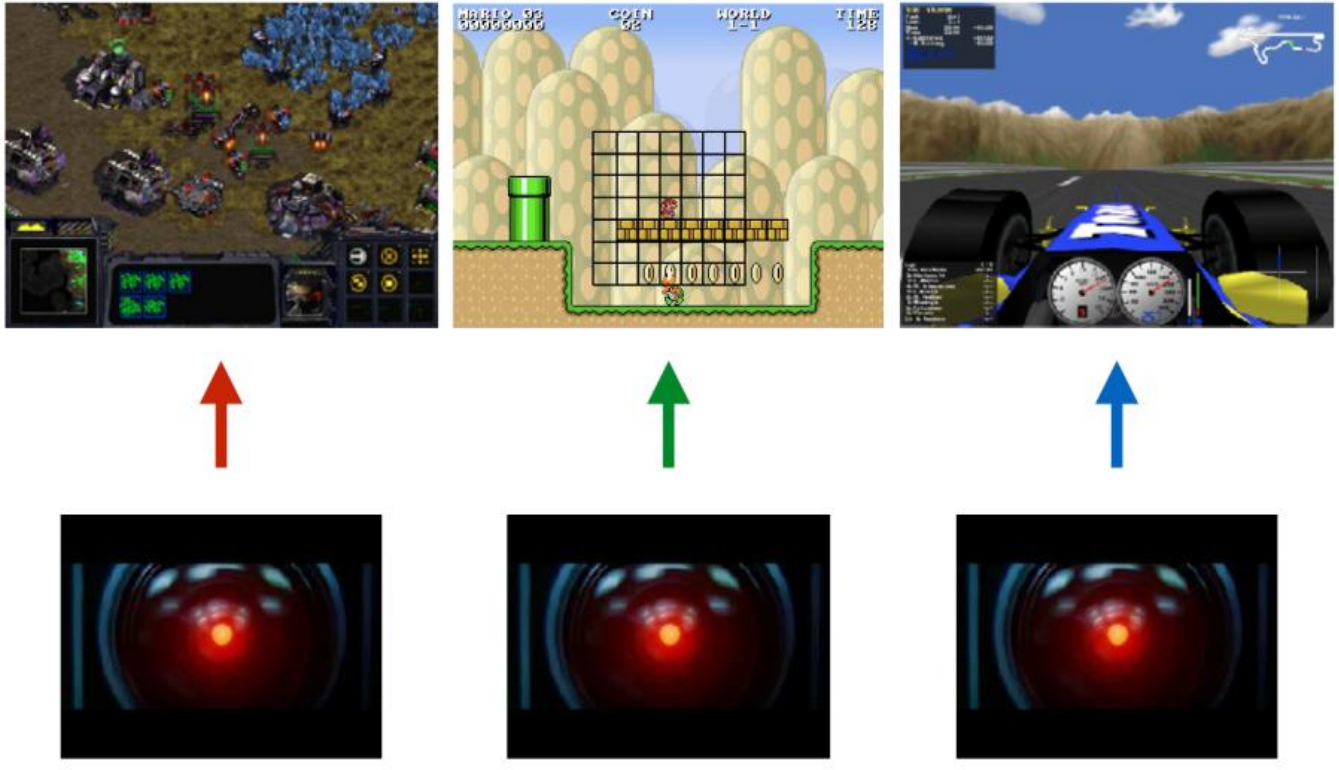
- **Specific** game playing

Will AI play more than one games?

- **General** game-playing

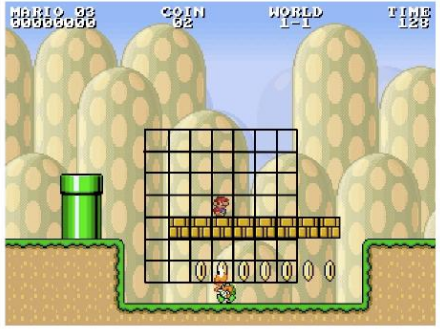


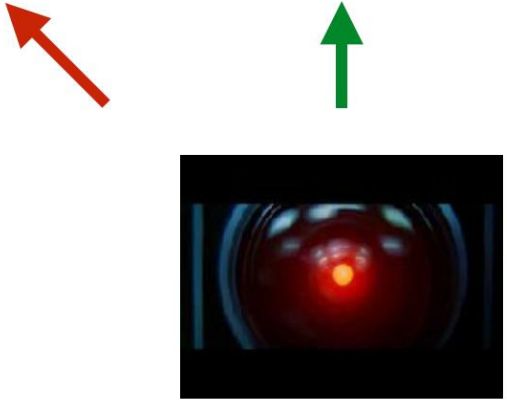
Problem: Overfitting!



Solution: General Game-playing

Can we construct AI that can play **many games**?





Chapter 5: Methods for Playing Games

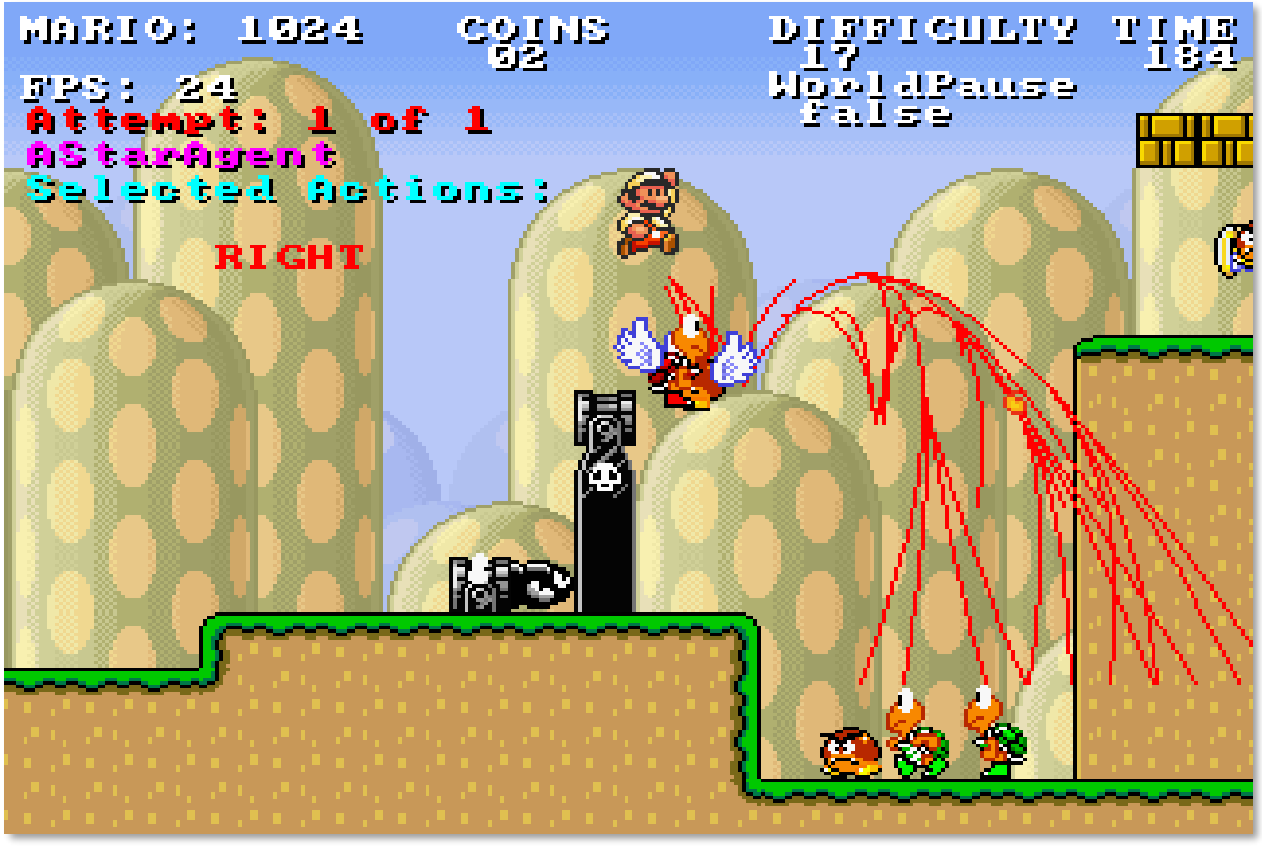
How Can AI Play Games?

- Different methods are suitable, depending on:
  - The characteristics of the game
  - How you apply AI to the game
  - Why you want to make a game-playing
- There is no single best method (duh!)
- Often, hybrid (chimeric) architectures do best

“Surely, deep RL is the **best** algorithm for playing games…”


25
Yeah, well, that's just, like,
your opinion, man.

How Would you Play Super Mario Bros?



https://www.youtube.com/watch?v=DlkMs4ZHHr8

How Can AI Play Games: An Overview

- Behaviour authoring – requires **human ingenuity and time**
- Planning-Based – requires **forward model**
  - Uninformed search (e.g. best-first, breadth-first)
  - Informed search (e.g. A*)
  - Evolutionary algorithms
- Reinforcement Learning – requires **training time**
  - TD-learning / approximate dynamic programming
  - Evolutionary algorithms
- Supervised Learning – requires **play traces**
  - Neural nets, k-NN, SVMs, Decision Trees, etc.
- Random – requires **nothing**

Life **with** a model…

How Can AI Play Games

- Planning-Based – **requires** forward model
  - Uninformed search (e.g. best-first, breadth-first)
  - Informed search (e.g. A*)
  - Adversarial search (e.g. Minimax, MCTS)
  - Evolutionary algorithms
- But path-planning does **not require** a forward model
  - Search in physical space

A Different Viewpoint

- Planning-Based
  - Classic Tree Search (e.g. best-first, breadth-first, A*, Minimax)
  - Stochastic Tree Search (e.g. MCTS)
  - Evolutionary Planning (e.g. rolling horizon)
  - Planning with Symbolic Representations (e.g. STRIPS)

Classic Tree Search

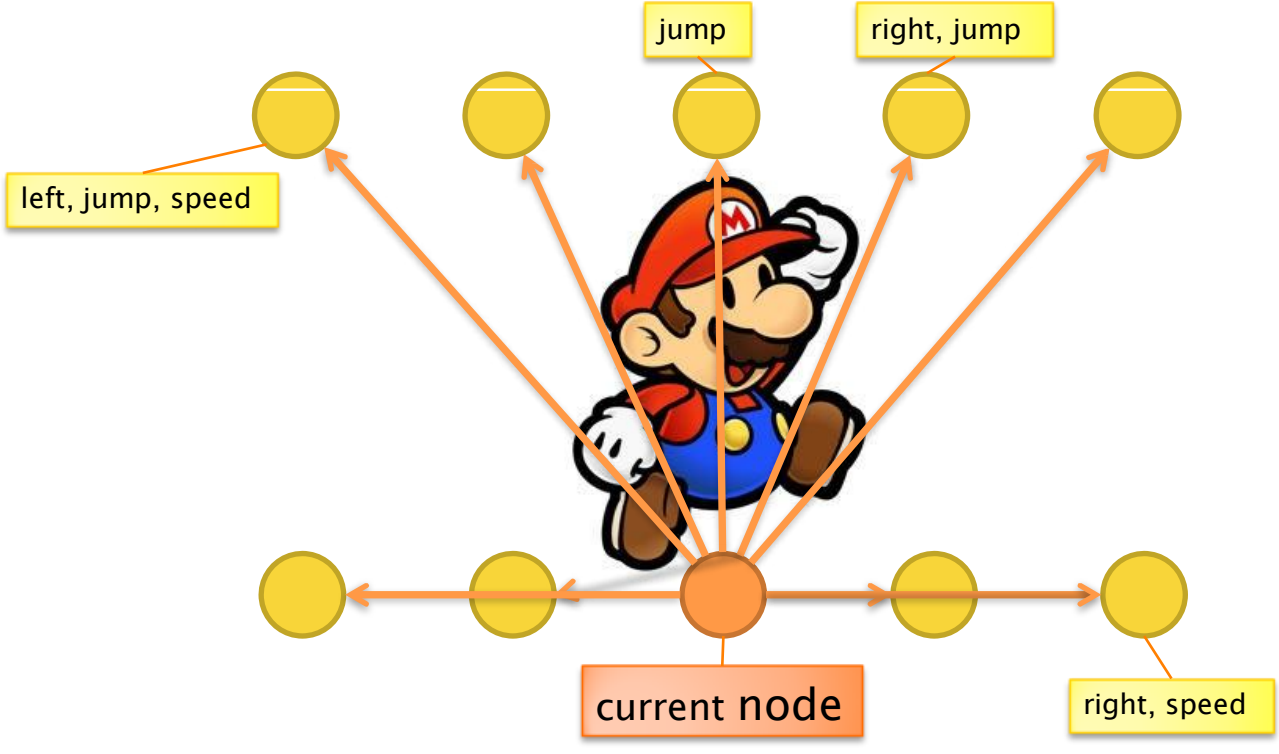
Informed Search (A*)

A* in Mario: Current Position

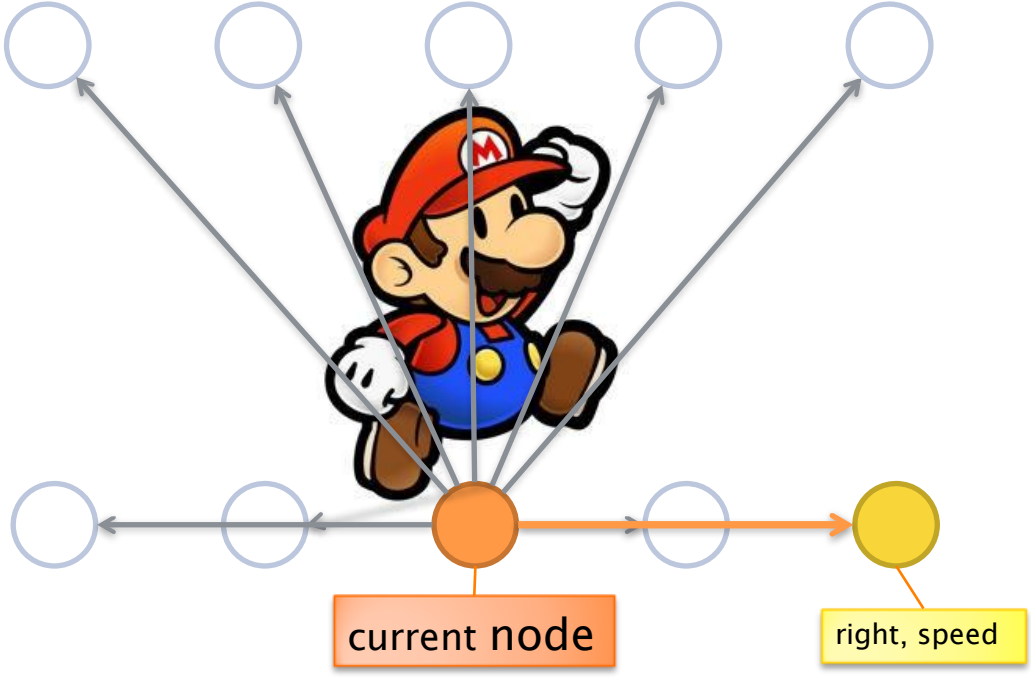
current node

Goal:
right border
of screen

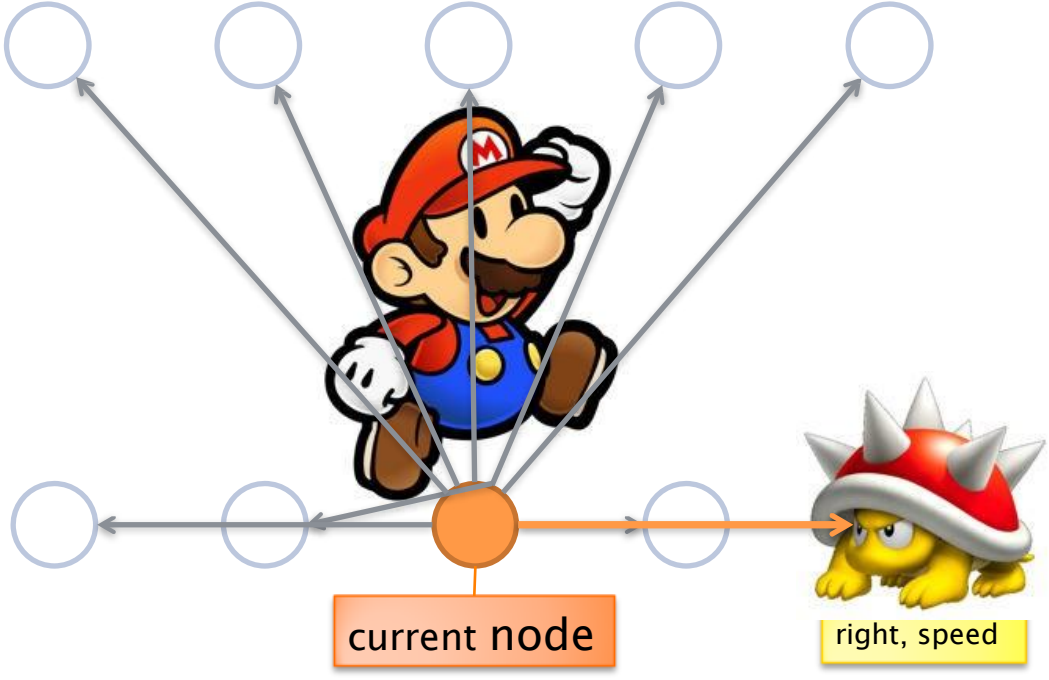
A* in Mario: Child Nodes



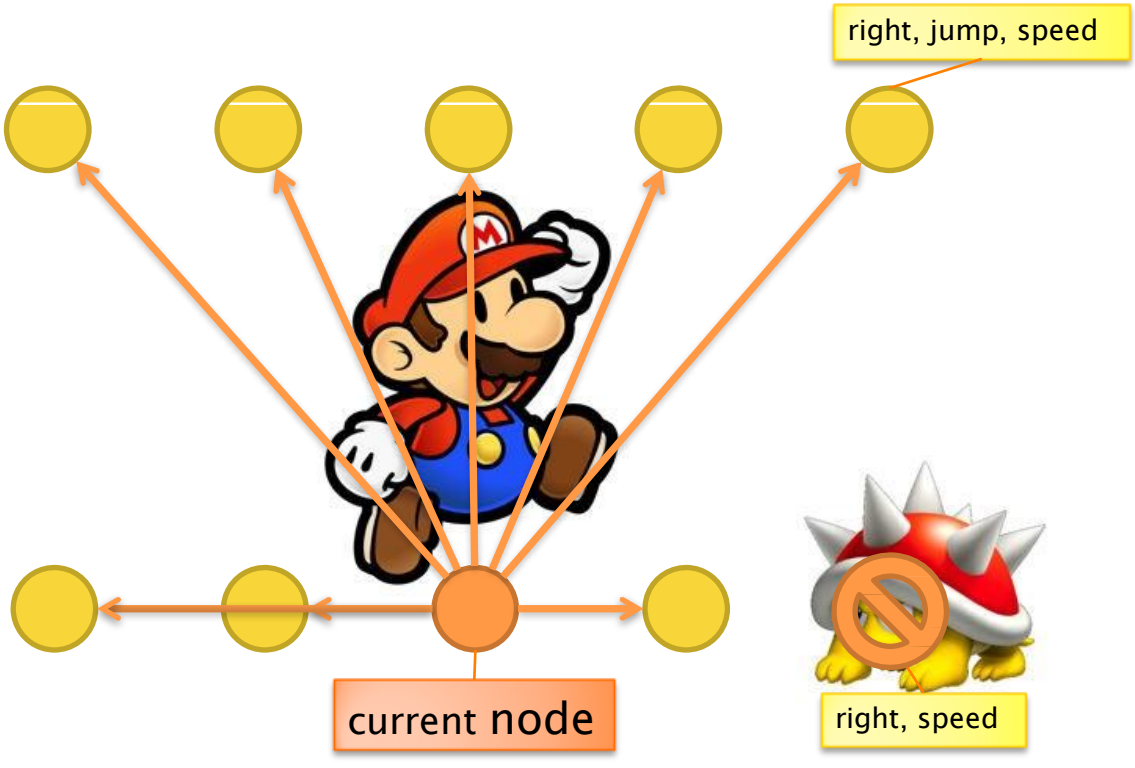
A* in Mario: Best First

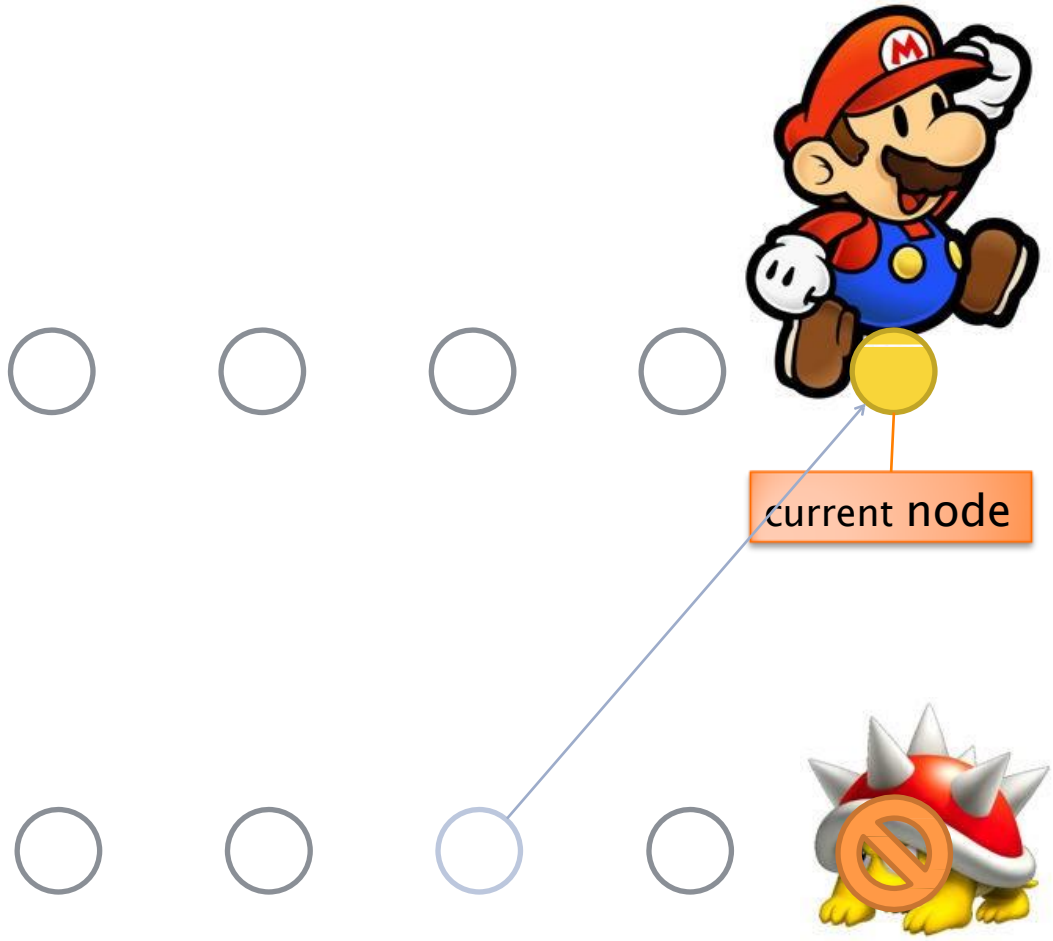


A* in Mario: Evaluate Node

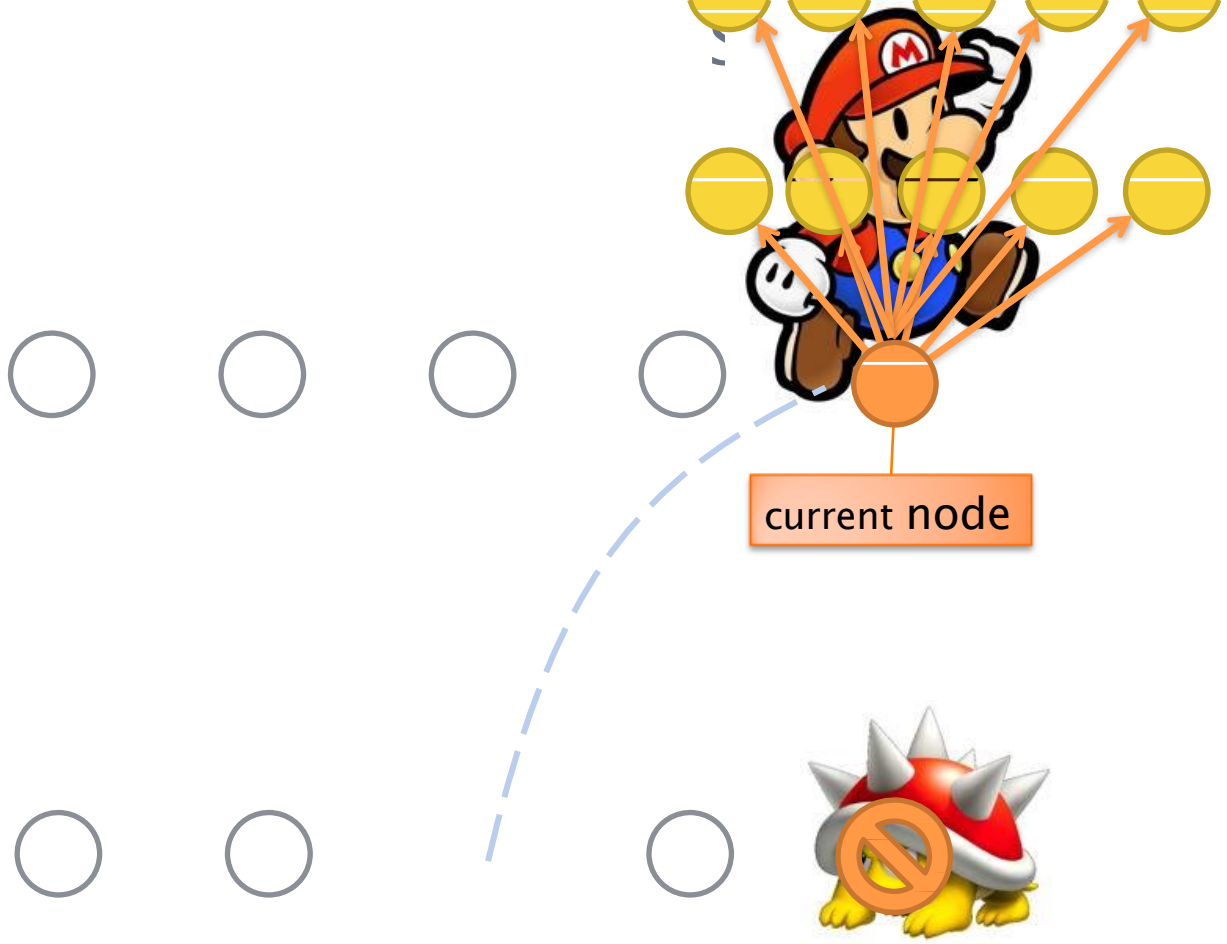


A* in Mario: Backtrack

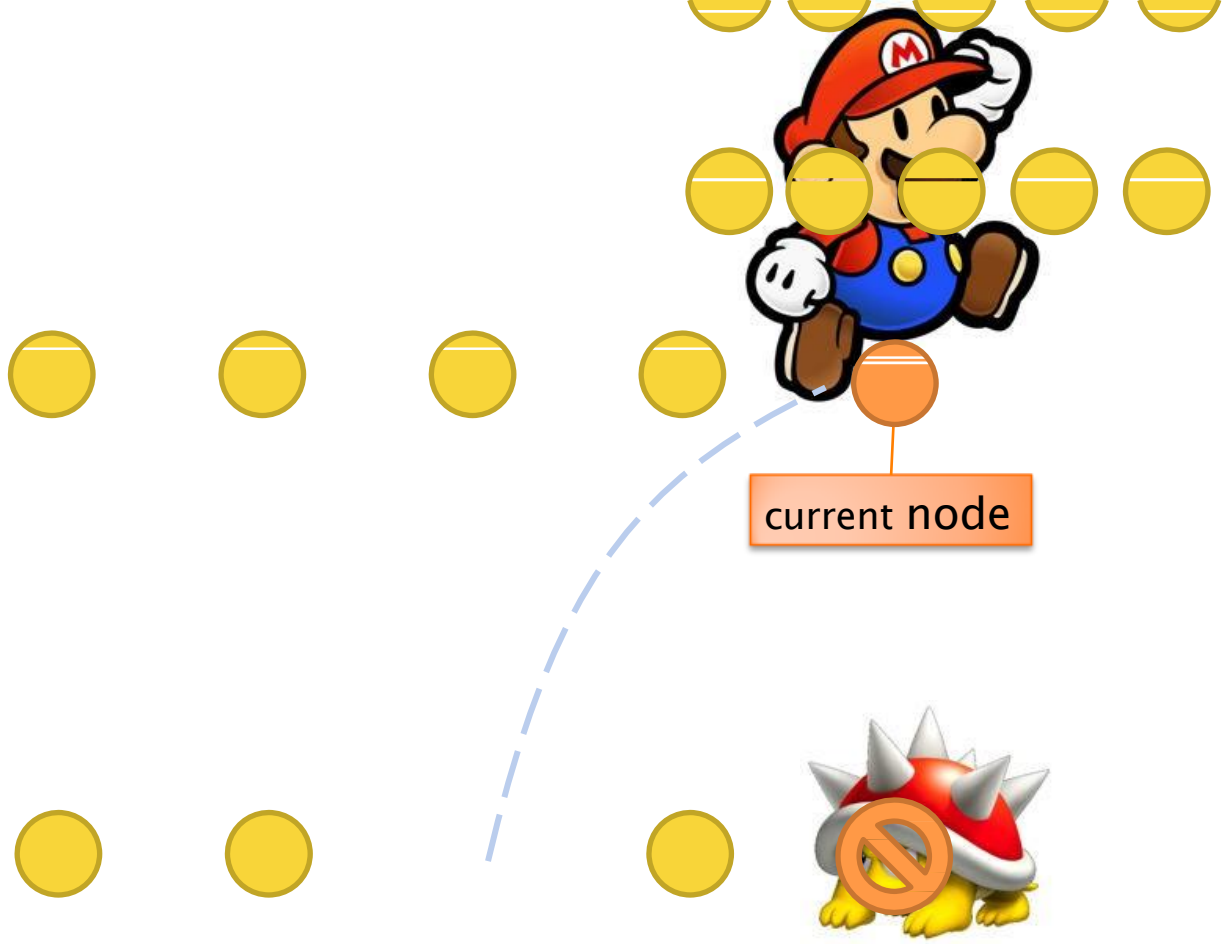




A* in Mario: Next State



A* in Mario: Create Child Nodes



A* in Mario: Best first

So why was A* successful?

Limitations of A*

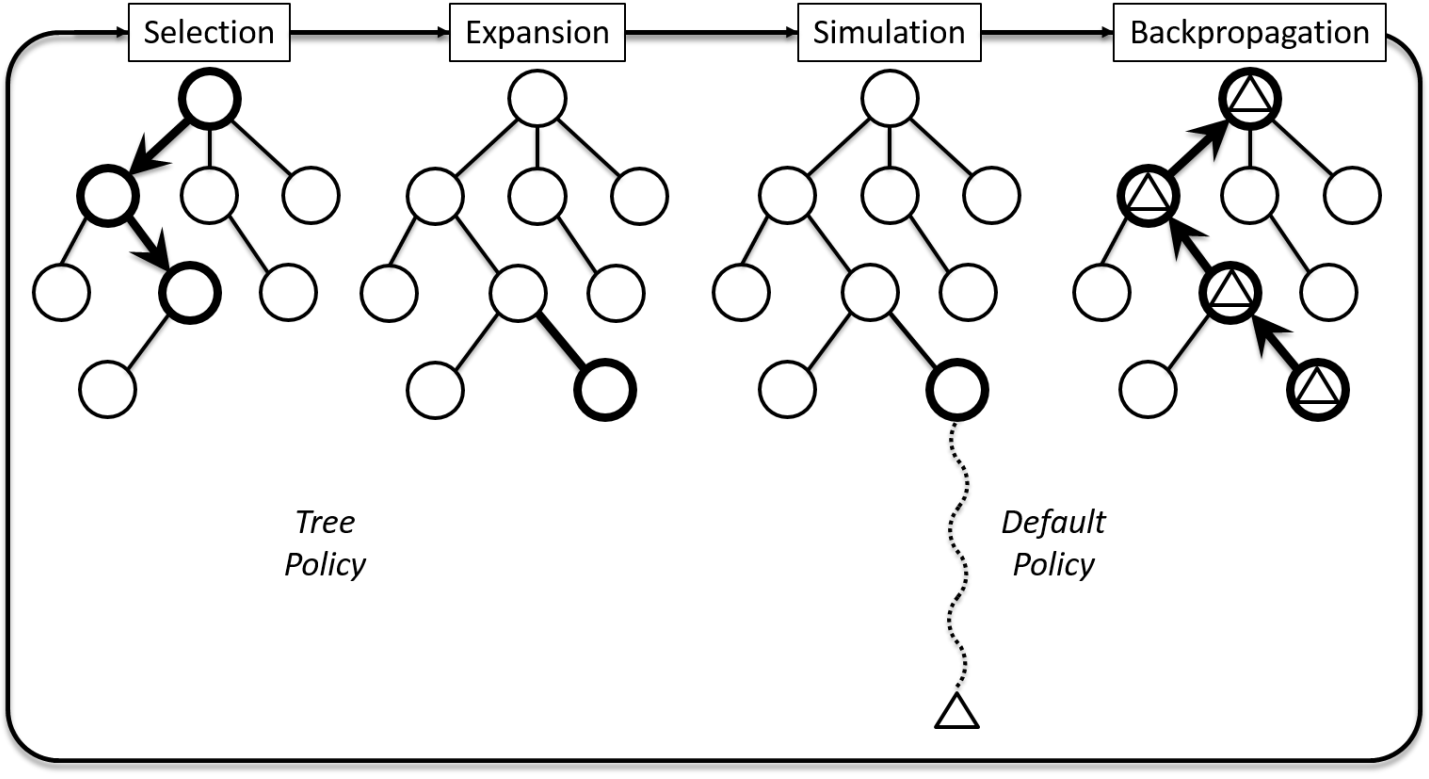
Stochastic Tree Search

Monte Carlo Tree Search

- The best new tree search algorithm you hopefully already know about
- When invented, revolutionized computer Go

Year	Program	Description	Elo
2006	INDIGO	Pattern database, Monte Carlo simulation	1400
2006	GNU GO	Pattern database, α-β search	1800
2006	MANY FACES	Pattern database, α-β search	1800
2006	NEUROGO	TDL, neural network	1850
2007	RLGO	TD search	2100
2007	MOGO	MCTS with RAVE	2500
2007	CRAZY STONE	MCTS with RAVE	2500
2008	FUEGO	MCTS with RAVE	2700
2010	MANY FACES	MCTS with RAVE	2700
2010	ZEN	MCTS with RAVE	2700

Monte Carlo Tree Search

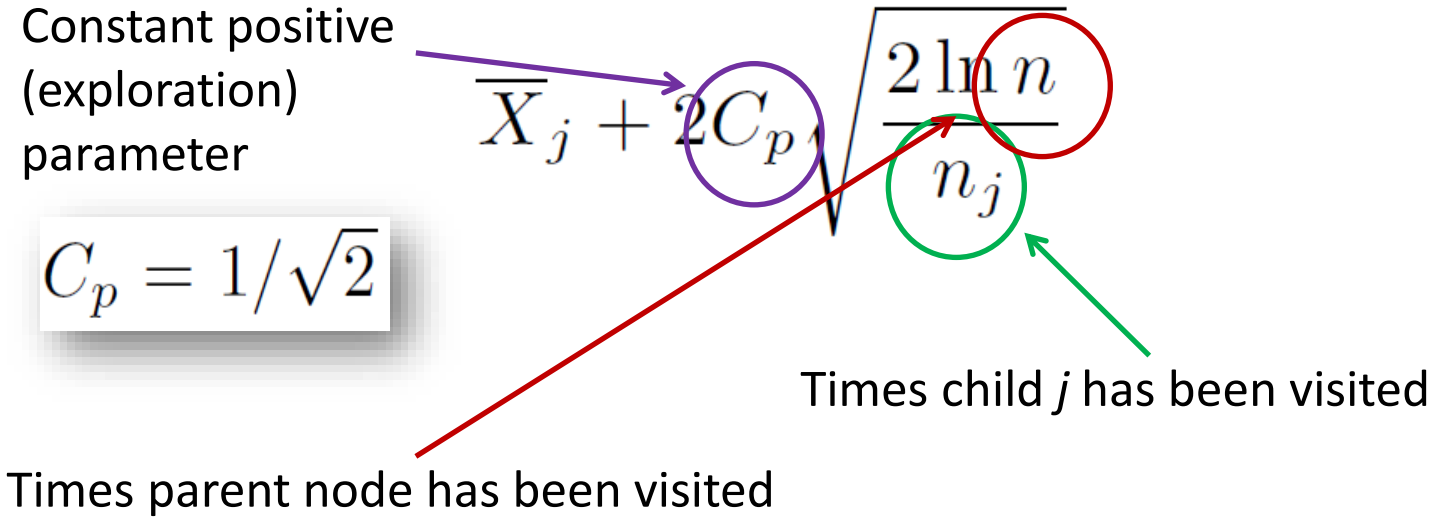


- Tree policy: choose which node to expand (not necessarily leaf of tree)
- Default (simulation) policy: random playout until end of game

UCB1 Criterion

MCTS as a multi-armed bandit problem

Every time a node (action) is to be selected within the existing tree, the choice may be modelled as an independent multi-armed bandit problem. A child node *j* is selected to maximise:



MCTS Goes Real-Time

- Limited roll-out budget
  - Heuristic knowledge becomes important
- Action space is fine-grained
  - Take *macro-actions* otherwise planning will be very short-term
- Maybe no terminal node in sight
  - Use a heuristic
  - Tune simulation depth
- Next state function may be expensive
  - Consider making a simpler abstraction

MCTS for Mario

https://www.youtube.com/watch?v=01j7pbFTMXQ

Jacobsen, Greve, Togelius: **Monte Mario: Platforming with MCTS**. *GECCO* 2014.

MCTS Modifications

Modification	Mean Score	Avg. T Left
Vanilla MCTS (Avg.)	3918	131
Vanilla MCTS (Max)	2098***	153
Mixmax (0.125)	4093	147
Macro Actions	3869	142
Partial Expansion	3928	134
Roulette Wheel Selection	4032	139
Hole Detection	4196**	134
Limited Actions	4141*	137
(Robin Baumgarten's A*)	4289***	169

A* Still Rules

- Several MCTS configurations get the same score as A*
- It seems that A* is playing essentially optimally
- But what if we modify the problem?

Making a Mess of Mario

- Introduce action noise:
  - 20% of actions are replaced with a random action
- Destroys A*
- MCTS handles this much better

AI	Mean Score
MCTS (X-PRHL)	1770
A* agent	1342**

MCTS in Commercial Games

Example: MCTS @ Total War Rome II

- Task Management System
  - Resource Allocation (match resources to tasks)
    - Typically many tasks.. but few resources
    - Large search space, little time
  - Resource Coordination (determine the best set of actions given resources & their targets)
    - Large search space
    - Grows exponentially with number of resources
    - Expensive pathfinding queries
    - MCTS-based planner to achieve constant worst-case performance

Evolutionary Planning

Evolutionary Planning

- Basic idea:
  - Don't **search** for a *sequence of actions* starting from an initial point
  - **Optimize** the *whole action sequence* instead!
- Search the space of complete action sequences for those that have maximum utility.
- Evaluate the utility of a given action sequence by taking all the actions in the sequence in simulation, and observing the value of the state reached after taking all those actions.

Evolutionary Planning

- Any optimization algorithm is applicable
- Evolutionary algorithms are popular so far; e.g.
  - *Rolling horizon evolution* in TSP
  - Competitive agents in General Video Game AI Competition
  - “Online evolution” outperforms MCTS in *Hero Academy*
  - Evolutionary planning performs better than varieties of tree search in simple *StarCraft* scenarios
  - Performs well in cooperative board games such as *Pandemic*
- A method at birth – still a lot to come!

Planning with Symbolic Representations

Planning with Symbolic Representations

- Planning on the level of in-game actions requires a fast forward model
- However, one can plan in an abstract representation of the game's state space.
- Typically, a language based on *first-order logic* represents events, states and actions, and tree search is applied to find paths from current state to end state.

- Example: STRIPS-based representation used in Shakey, the world's first digital mobile robot
- Game example: *F.E.A.R.* (Sierra Entertainment, 2005) agent planners by Jeff Orkin

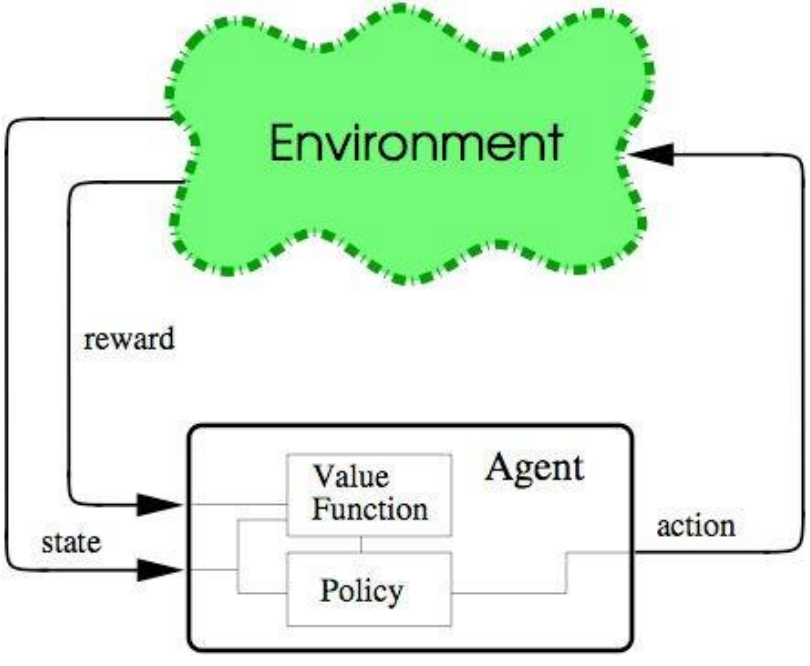
Life **without** a model…

How Can AI Play Games?

- Reinforcement learning (requires **training time**)
  - TD-learning/approximate dynamic programming
  - Deep RL/Deep Q-N, …
  - Evolutionary algorithms

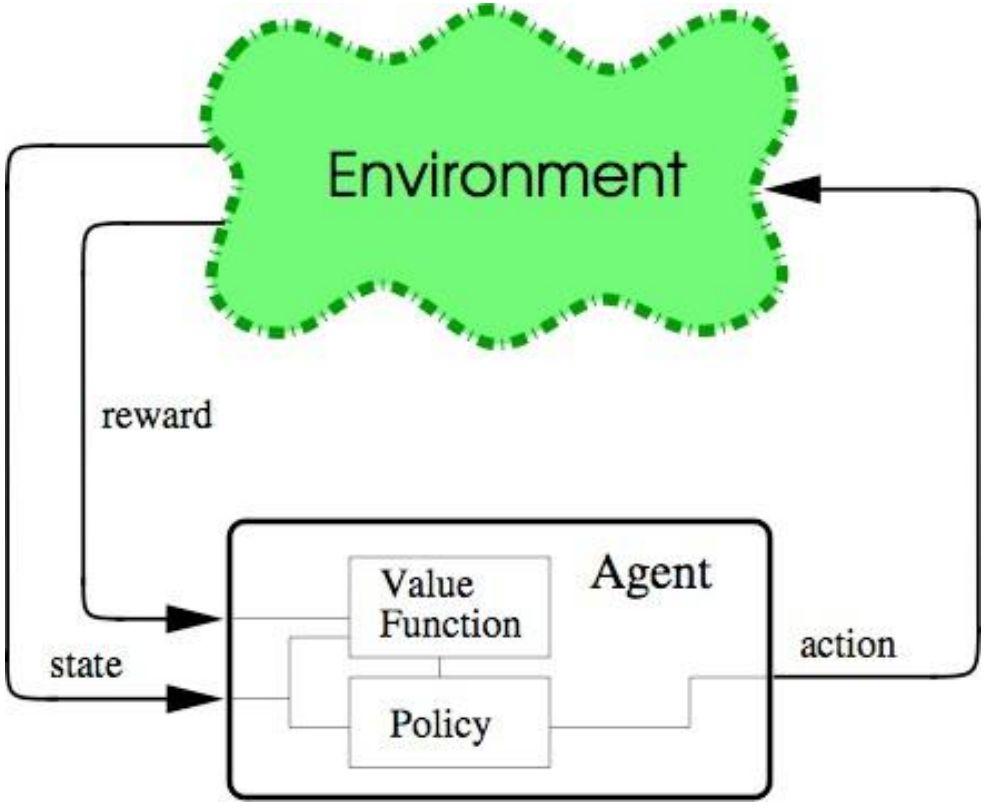
Classic and Deep RL

RL Problem


Environment
reward
Agent
Value Function
Policy
state
action

Q-learning

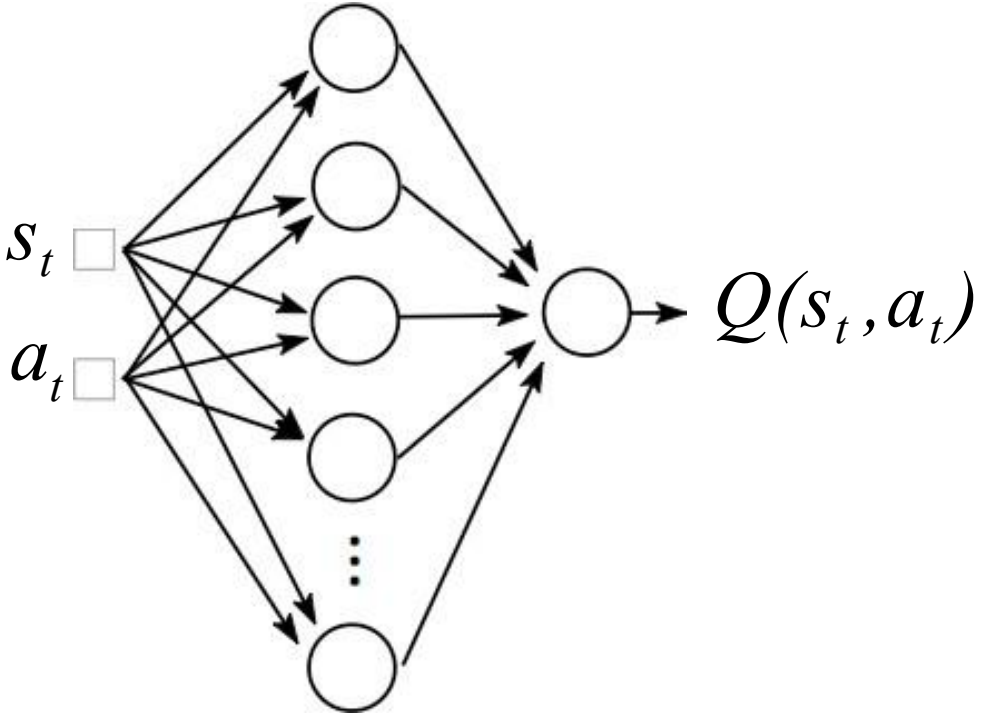
- Off-policy reinforcement learning method in the temporal difference family
- Learn a mapping from (state, action) to value
- Every time you get a reward (e.g. win, lose, score), propagate this back through all states
- Use the *max* value from each state



Agent consists of two components:

1. Value-function (Q-function)
2. Policy

Representing Q(s,α) with ANNs



Training the ANN Q-function

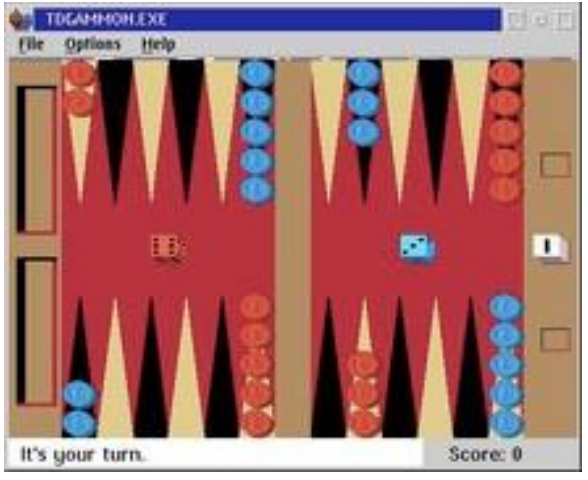
Training is performed on-line using the Q-values from the agent's state transitions

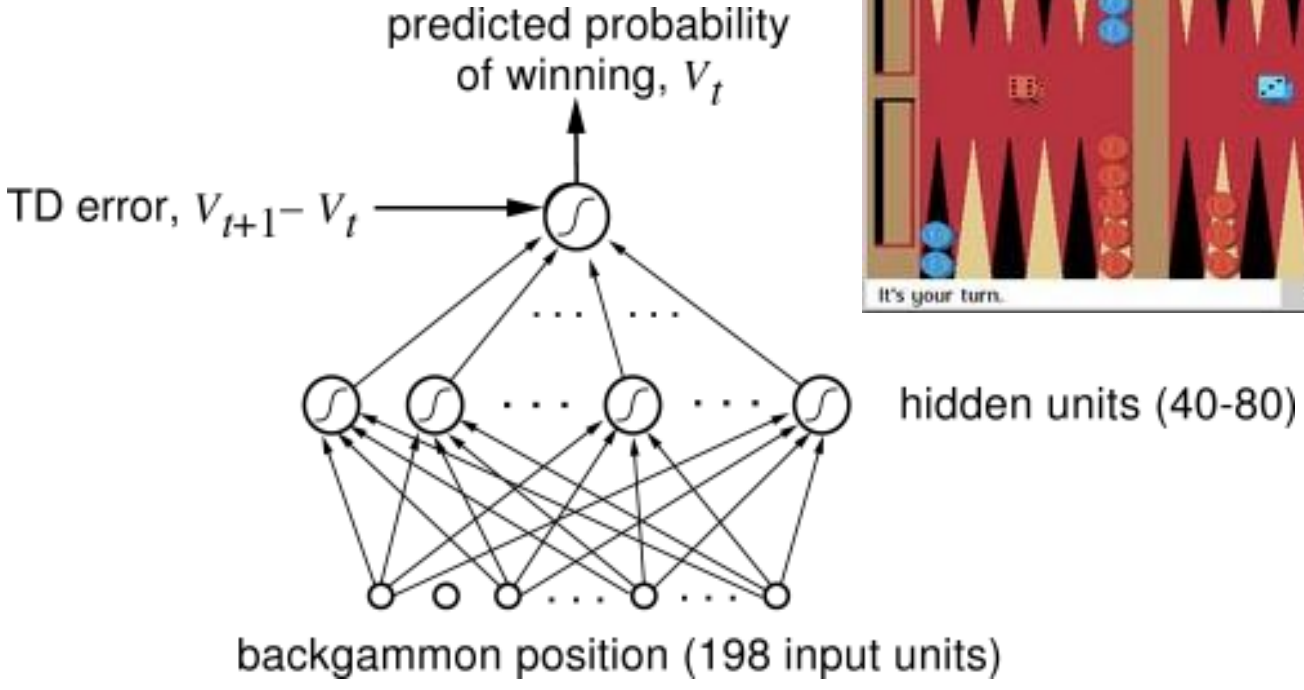
For Q-learning:

input: s_t, a_t

target: $r_t + \gamma \max_a Q(s_{t+1}, a)$

TD-Gammon (Teusaro, 1992)





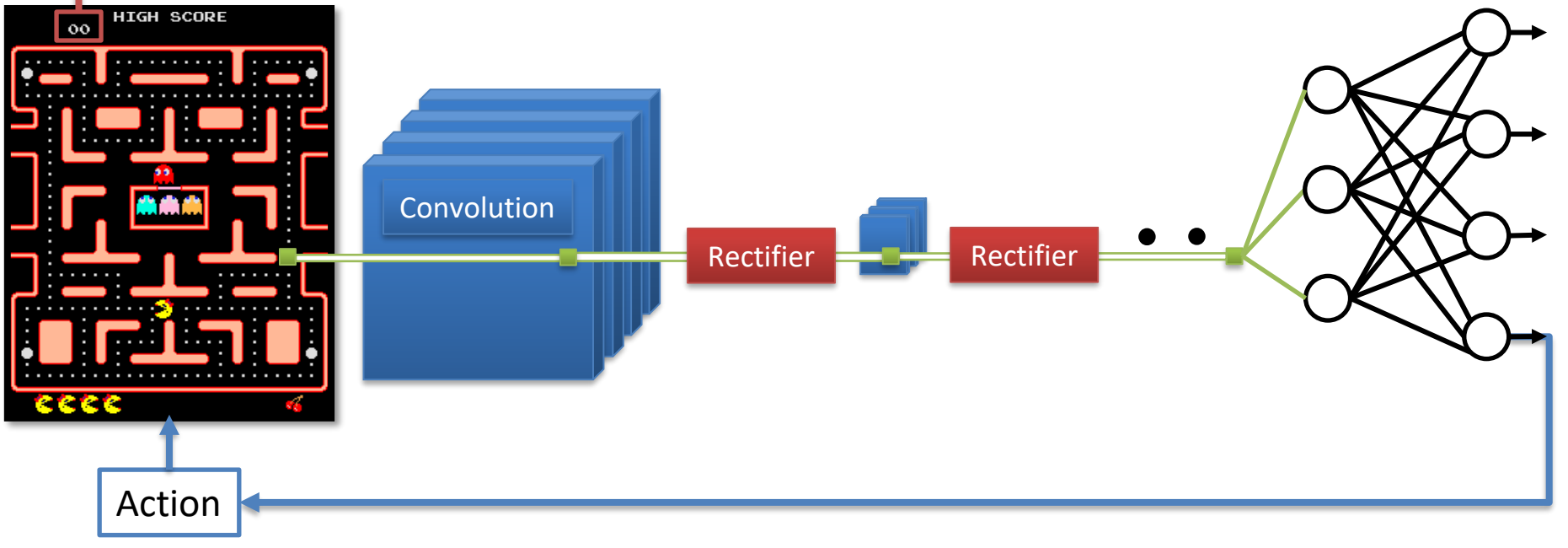
Deep Q-learning

- Use Q-learning with *deep* neural nets
- In practice, several additions useful/necessary
  - Experience replay: chop up the training data so as to remove correlations between successive states

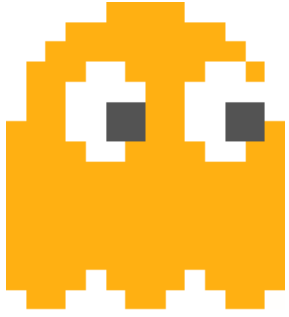
Niels Justesen, Philip Bontrager, Sebastian Risi, Julian Togelius: **Deep Learning for Video Game Playing**. ArXiv.

Deep Q Network (DQN)

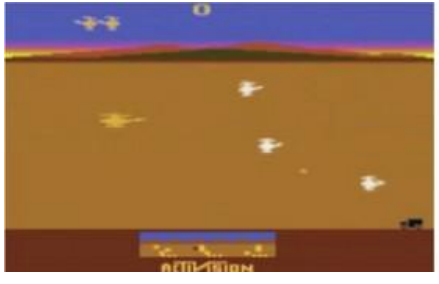
Ms Pac-Man Example



Arcade Learning Environment



ALE



Arcade Learning Environment

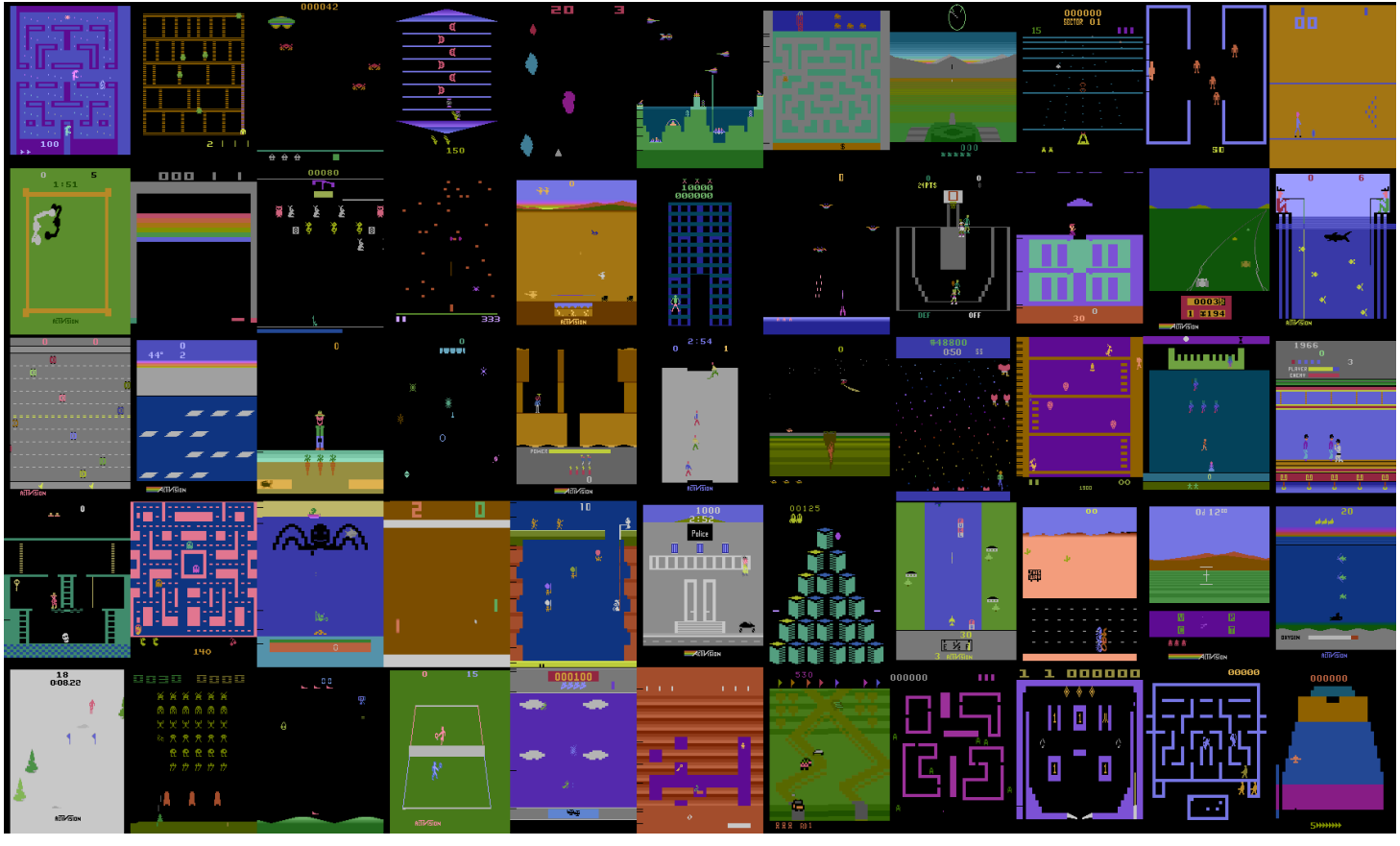
Based on an Atari 2600 emulator

- Atari: very successful but very simple
- 128 byte memory, no random number generator

A couple of dozen games available (hundreds made for the Atari)

Agents are fed the raw screen data (pixels)

Most successful agents based on deep learning



NATURE | LETTER

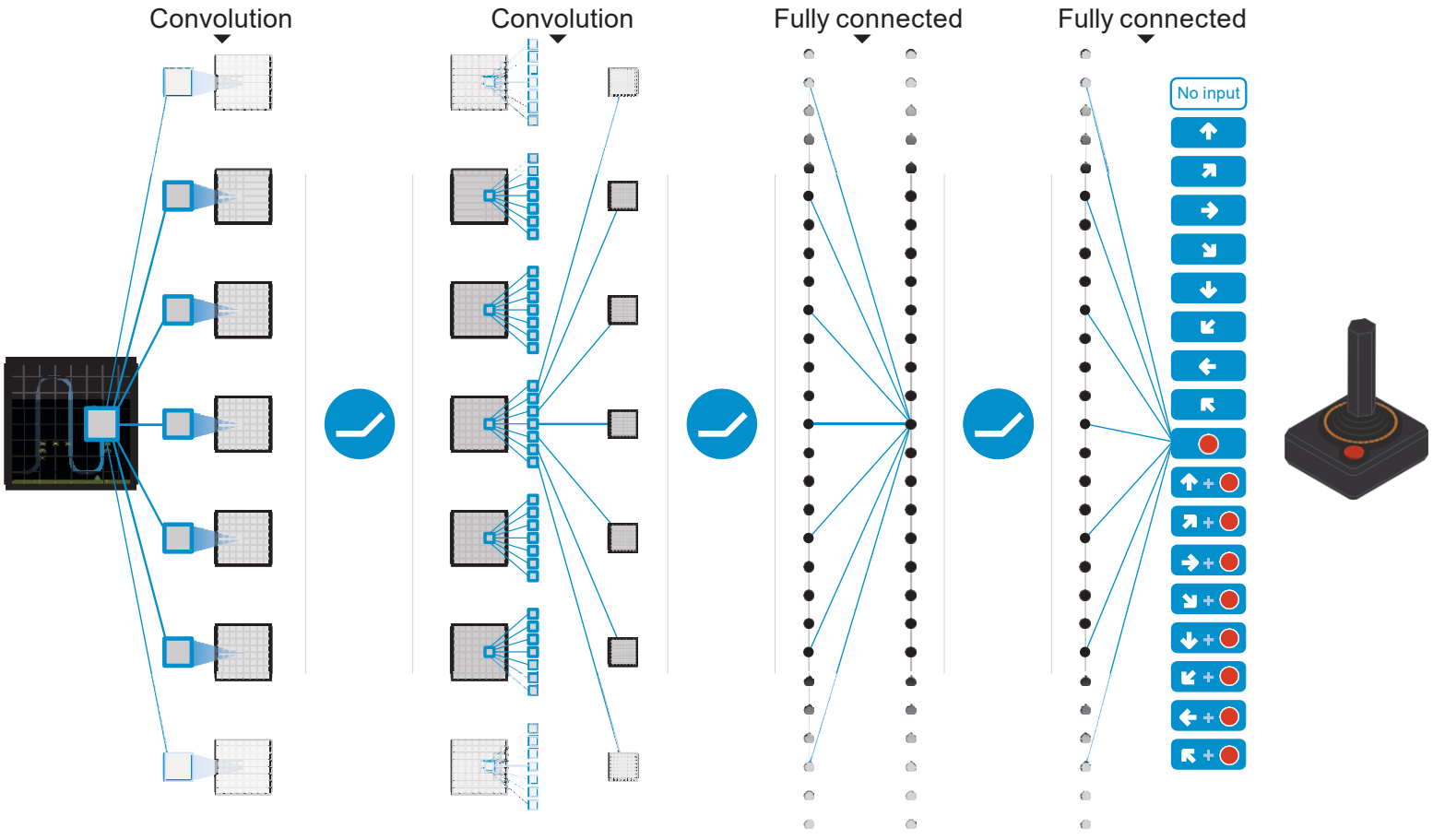
日本語要約

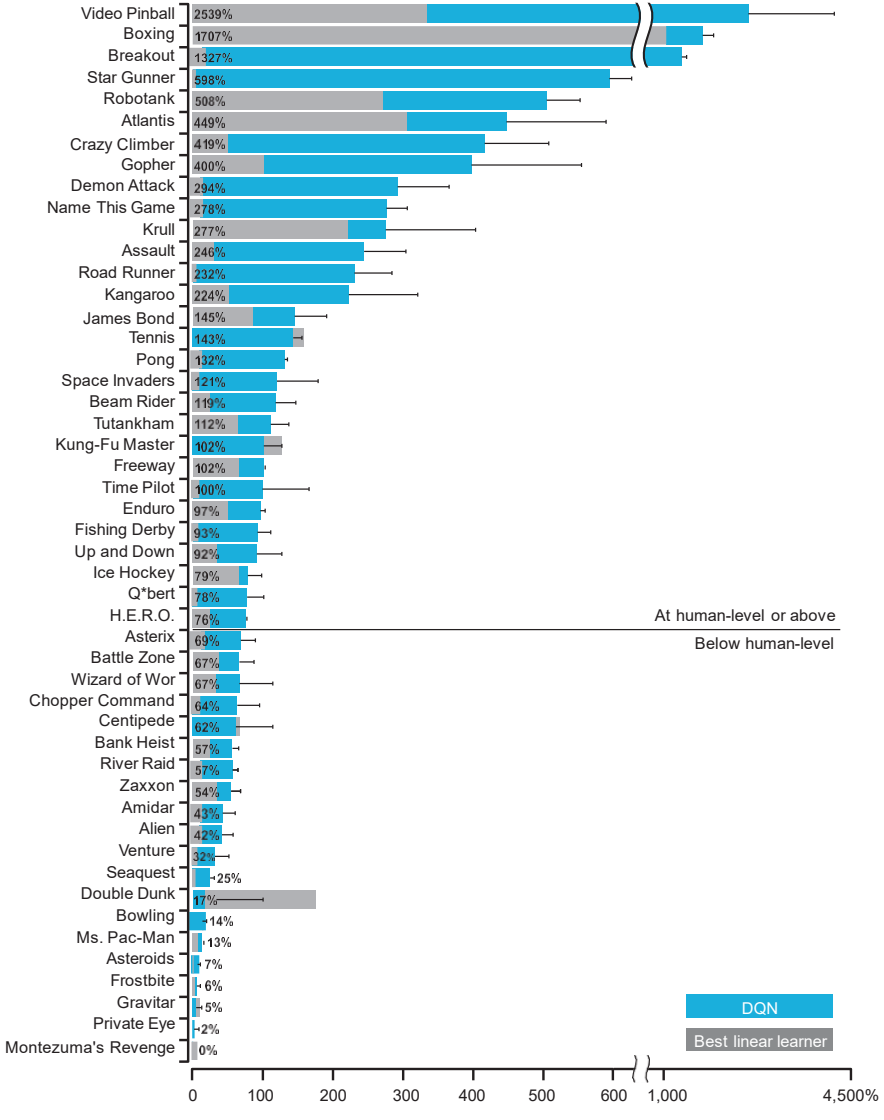
Human-level control through deep reinforcement learning

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis

Affiliations | Contributions | Corresponding authors

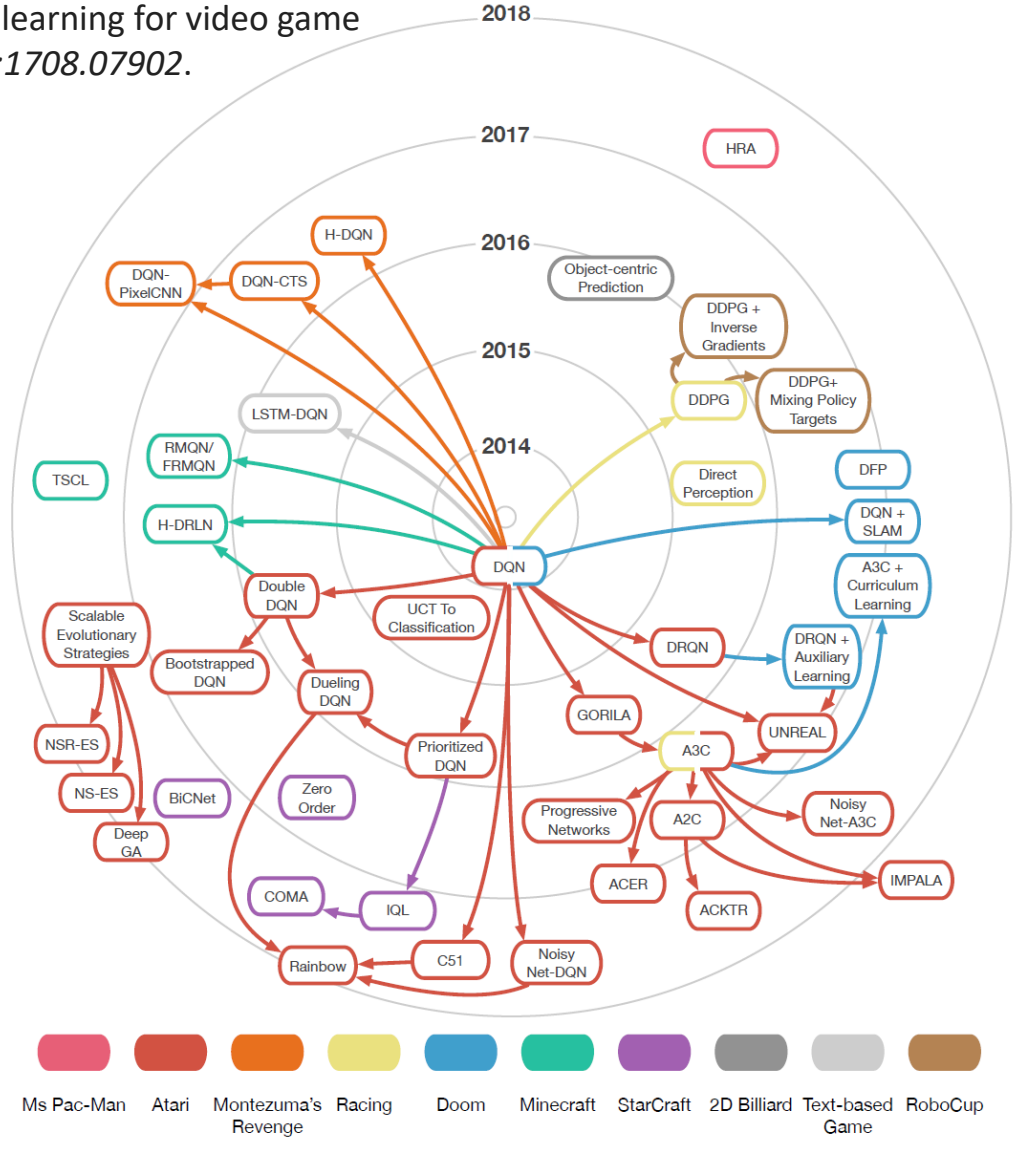
Nature **518**, 529–533 (26 February 2015) | doi:10.1038/nature14236
Received 10 July 2014 | Accepted 16 January 2015 | Published online 25 February 2015


Convolution
Convolution
Fully connected
Fully connected
No input



Results: not bad! …but not general

Justesen et al. (2017). Deep learning for video game playing. *arXiv preprint arXiv:1708.07902*.



Proximal Policy Optimization

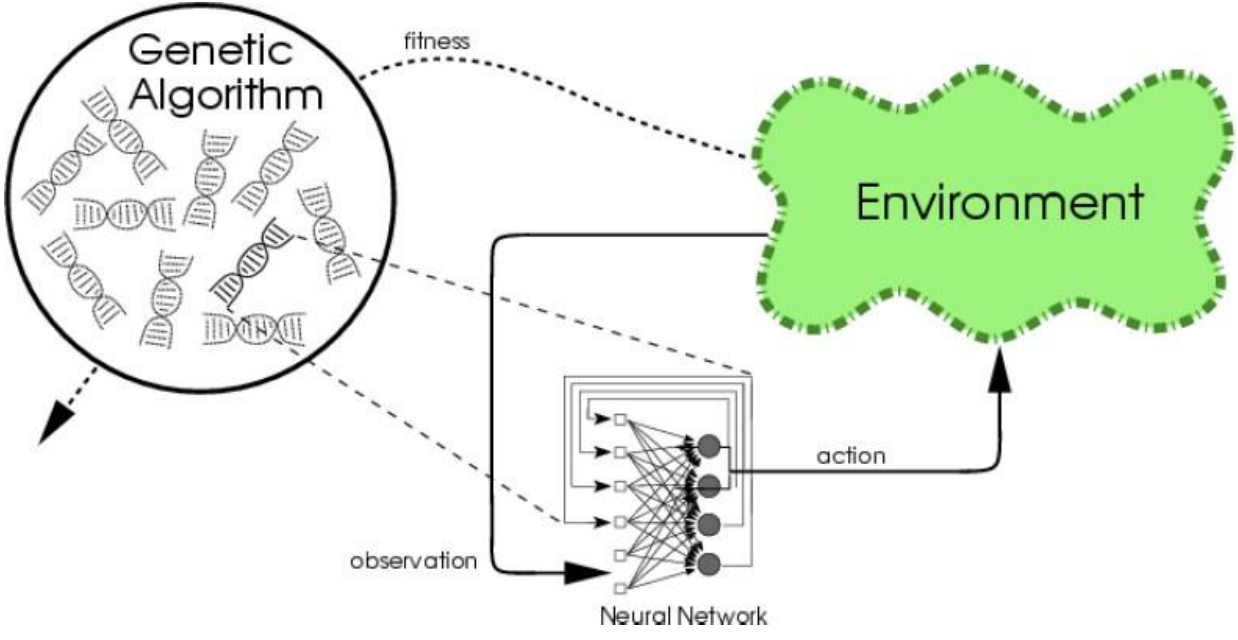
Major step forward for RL in games

- Policy gradient method; limits the changes to the policy yielding a more stable RL method (i.e., less catastrophic forgetting)
- Atari: great performance!
- Dota 2: beating the world champion!

Because of its simplicity, stability, and versatility, PPO is often used as a default RL algorithm in gameplaying

Evolutionary RL

(Neuro)Evolution as a RL Problem



Evolutionary Algorithms

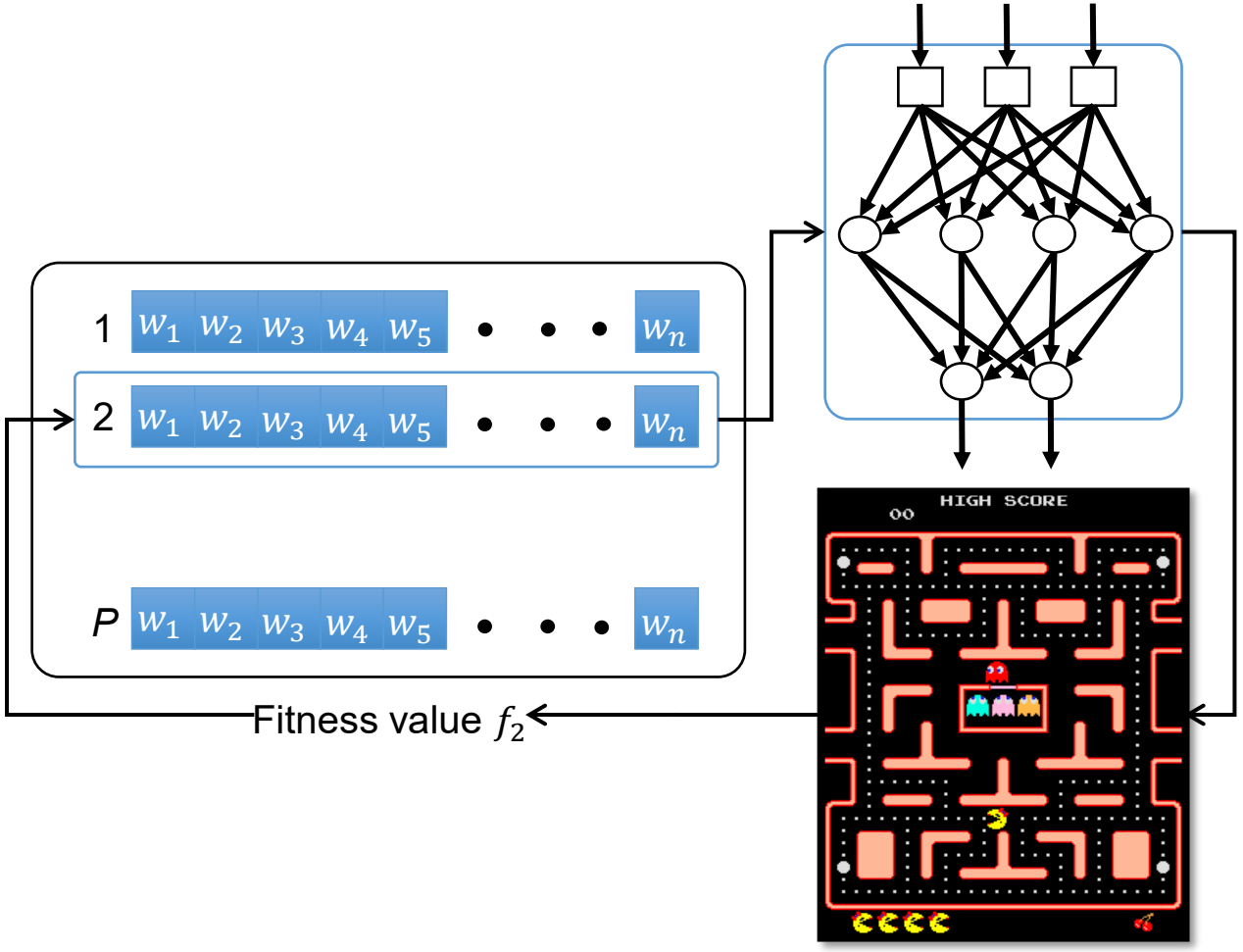
- Stochastic global optimization algorithms
- Inspired by Darwinian natural evolution
- Extremely domain-general, widely used in practice

Simple μ+λ Evolutionary Strategy

- Create a population of μ+λ individuals
- At each generation
  - Evaluate all individuals in the population
  - Sort by fitness
  - Remove the worst λ individuals
  - Replace with mutated copies of the μ best individuals

Evolving ANNs

Ms Pac-Man Example



Neuroevolution has been used broadly…

(a) (b) (c) (d)

Fig. 2. **Neuroevolution in Existing Games.** (a) NE is able to discover high-performing controllers for racing games such as TORCS [11]. (b) NE has also been successfully applied to commercial games, such as Creatures [44]. Additionally, NE enables new types of games such as GAR (c), in which players can interactively evolve particular weapons [46], or NERO (d), in which players are able to evolve a team of robots and battle them against other players [119].

Sebastian Risi and Julian Togelius (2016): **Neuroevolution in games**. TCIAIG.

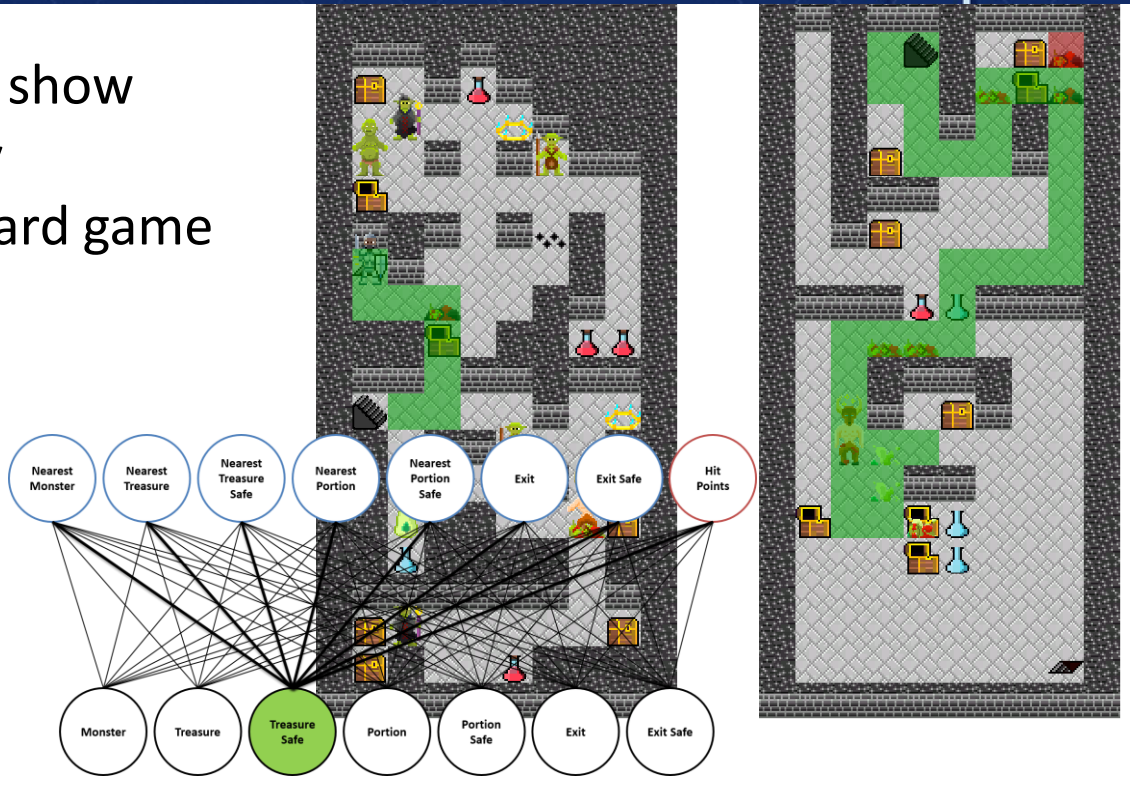
TABLE I

The Role of Neuroevolution in Selected Games. ES = evolutionary strategy, GA = genetic algorithm, MLP = multi-layer perceptron, MO = multiobjective, TP = third-person (input not tied to a specific frame of reference, e.g. number edible ghosts) , UD = user-defined network topology, PA = performance alone

NE Role (Section III)	Game	ANN Type (Section IV)	NE Methods (Section V)	Fitness Evaluation (Section VI)	Input Representation (Section VII)
State/action evaluation	**Checkers** [32]	MLP	UD, GA	Coevolution	TP (piece type)
	Chess [32]	MLP	UD, GA	PA (positional values)	TP (piece type)
	Othello [79]	MLP	Marker-based [34]	Cooperative coevolution	TP (piece type)
	Go (7×7) [38]	CPPN (MLP)	HyperNEAT	PA (score+board size)	TP (piece type)
	Ms. Pac-Man [71]	MLP	UD, ES	PA (average score)	Path-finding
	Simulated Car Racing [74]	MLP	UD, ES	PA (waypoints visited)	Speed, pos, waypoints
Direct action selection	**Quake II** [85, 86]	MLP	UD, GA	PA (kill count)	Visual Input (14×2)
	Unreal Tournament [135]	Recurrent, LSTM	UD, GA, NSGA-II	MO (damage&accuracy)	Pie-slice, way point, etc.
	Go (7×7) [118]	MLP	NEAT	Transfer Learning	Roving Eye (3×3)
	Simulated Car Racing [124]	MLP	UD, ES	Incremental Evolution	Rangefinders, waypoints
	Keepaway Soccer [122]	MLP	NEAT	Transfer Learning	Distances
	Battle Domain [105]	MLP	NEAT, NSGA-II	MO+Incremental	Angle, straight line
	NERO [119]	MLP	NEAT	Interactive Evolution	Rangefinders, pie-slice
	Ms. Pac-Man [106]	Modular MLP	NEAT, NSGA-II	MO (pills&ghosts eaten)	Path-finding
	Simulated Car Racing [29]	MLP	UD, GA	PA (distance)	Roving Eye (5×5)
	Atari [48]	CPPN (MLP)	HyperNEAT	PA (game score)	Raw input (16×21)
	Creatures [44]	Modular MLP	GA	Interactive Evolution	TP (e.g. type of object)
Selection between strategies	**Keepaway Soccer** [142, 143]	MLP	NEAT	PA (hold time)	Angle and distance
	EvoCommander [56]	MLP	NEAT	Interactive Evolution	Pie-slice, rangefinder
Modelling opponent strategy	**Texas Hold'em Poker** [66]	MLP	NEAT	PA (%hands won)	TP (e.g. size of pot, cost of a bet, etc.)
Content generation	**GAR** [46]	CPPN (MLP)	NEAT	Interactive Evolution	Model
	Petalz [97]	CPPN (MLP)	NEAT	Interactive Evolution	Model
Modelling player experience	**Super Mario Bros** [87]	MLP, Perceptron	UD, GA	PA (player preference)	TP (e.g. gap width, number deaths, etc.)

Procedural Personas

- Given utilities (rewards) show me believable gameplay
- Useful for human-standard game testing
- RL
  - MCTS
  - Neuroevolution
  - …
- Inverse RL



Liapis, Antonios, Christoffer Holmgård, Georgios N. Yannakakis, and Julian Togelius. "**Procedural personas as critics for dungeon generation**." In *European Conference on the Applications of Evolutionary Computation*, pp. 331-343. Springer, Cham, 2015.

Supervised Learning

Supervised Learning

- Supervised learning requires play traces to learn from
- Aka *behaviour cloning*
- Basic idea: record traces of human players and train some function approximator to behave like the human player.
- Methods
  - Neural networks, k-nearest neighbours, SVMs etc.
  - Generative Adversarial Imitation Learning (GAIL), Diffusion models in *CS:GO*, Dagger algorithm

Chimeric Game Players

Chimeric Methods

- While planning, RL and SL are fundamentally different approaches to playing games, it does not mean that they cannot be combined.
- Many examples of successful hybrid approaches (chimeras) such as **dynamic scripting** – a form of a learning classifier system
- Other examples
  - AlphaGo, AlphaZero and MuZero

Learning Beyond Policies

Learning Representations

Challenges of Pixel-Based Inputs

- **Native but high-dimensional**: Pixel inputs are the natural representation for most games, but require large neural networks, making them computationally expensive.
- **Real-time constraints**: At 60 FPS on CPU (GPU busy with rendering), AI has only a few milliseconds for inference → network size often limited to a few million parameters.
- **Training difficulties**: Evolutionary and stochastic search methods perform poorly in large parameter spaces.

Learning Representations

Dimensionality reduction

- **Autoencoders** can compress visuals into latent vectors for smaller policy networks.
- Must train on diverse game scenes to avoid poor performance on out-of-distribution levels.
- Example: Alvernaz & Togelius (2017) evolved small networks for *Doom* by periodically retraining autoencoders.

Alternative methods

- Dictionary vector quantizers store novel game states and use discrete encodings → enables networks with only a few thousand parameters to play Atari games effectively.

Samuel Alvernaz and Julian Togelius. **Autoencoder-augmented neuroevolution for visual doom playing**. In *IEEE Conference on Computational Intelligence and Games*. IEEE, 2017.

Learning Representations

Improving Generalizability

Many parameters go to visual processing; policies may be simpler than expected.

Challenges: Learning good representations often requires internal game state, which isn't always accessible.

Solutions:

- **Reverse engineering**: Capture rendering data (*GTA V*), or approximate internal states (*CS:GO*).
- **I/O-based capture**: BehAVE collects high-level player actions (e.g., "move forward," "aim gun") from raw inputs; uses pretrained text encoders (e.g., GPT) to align behaviors across games; demonstrates strong zero-shot gameplay transfer—even between very different games like *Minecraft* and FPS titles.
- Future: BehAVE may excel further with more computational resources!

Rasajski, Trivedi, Makantasis, Liapis, Yannakakis (2024). **BehAVE: Behaviour Alignment of Video Game Encodings**. *In Proc of ECCV.*

Learning Forward Models

- **Why?** Enables planning/training without a real game engine.
- **Issue:** Most engines aren't forward model–friendly.
- **Key example:** Ha & Schmidhuber (2018)
  - VAE compresses pixels, LSTM predicts next encoded state.
  - Works for *Doom* & *2D racing*.
  - Challenge: Prediction errors pile up → unrealistic states.

Ha and Schmidhuber. **World models**. arXiv preprint, arXiv:1803.10122, 2018.

Learning Forward Models

Recent advances and future directions

- *DreamerV3*: Discrete encoding + built-in actor-critic RL → fast, strong results.
- Neural Game Engines: Learn dynamics from pixels + actions (diffusion models); Still struggle with hidden states.
- *Genie*: Learns from videos only → short interactive 'games'. Hints at prompt-based game creation.

Bruce et al., **Genie: Generative interactive environments**. arXiv preprint, arXiv:2402.15391, 2024.

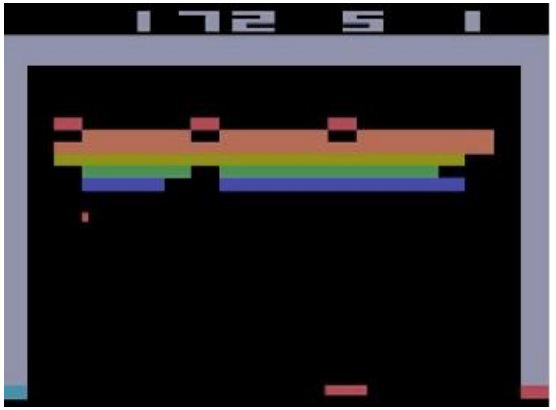
Chapter 6: Gameplaying AI by Game Genre

Board and Card Games

- Board games (Classic and Modern)
  - Adversarial planning, tree search
- Card games (Classic and Collectible)
  - Reinforcement learning, tree search

Real Time 2D Video Games

- Classic arcade games
  - Pac-Man and the like: Tree search, RL
  - Super Mario Bros: Planning, RL, Supervised learning
  - Arcade learning environment: RL
  - General Video Game AI: Tree search, RL
  - Fighting games: Tree search, RL, LLMs



Real-Time 3D Games

Racing games

- Supervised learning, RL, adversarial planning
- *Drivatar*: from lazy learning (2005) all the way to deep nets (present)
- *GT Sophy*: RL

Real-Time 3D Games

First Person Shooters

- UT2004: Neuroevolution, imitation learning
- *Doom*: (Deep) RL in VizDoom
- *Valorant* variant: Imitation learning
- *CS:GO*: Diffusion models

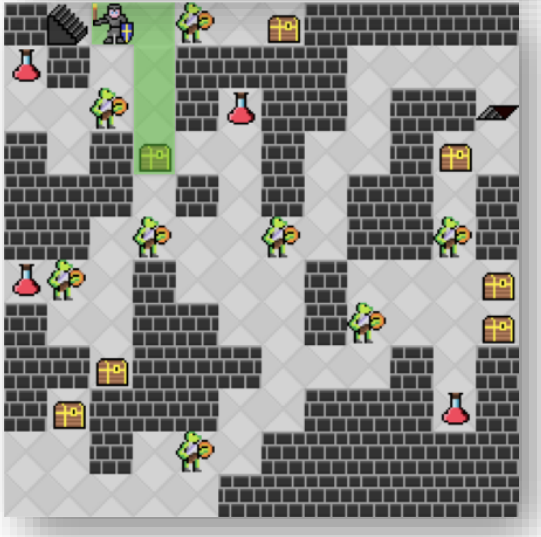
Real-Time 3D Games

Minecraft

- Sandbox game played in a 3D procedurally generated world
- *Project Malmo:* deep nets
- *Voyager:* LLM-powered “lifelong learning” agent
- Video-Pre Training (VPT):
  - First train an inverse dynamics model (IDM) on a small labelled dataset
  - The IDM labels a huge number (70k hours) of YouTube videos with actions
  - Such videos are then used to train a much larger model to play Minecraft

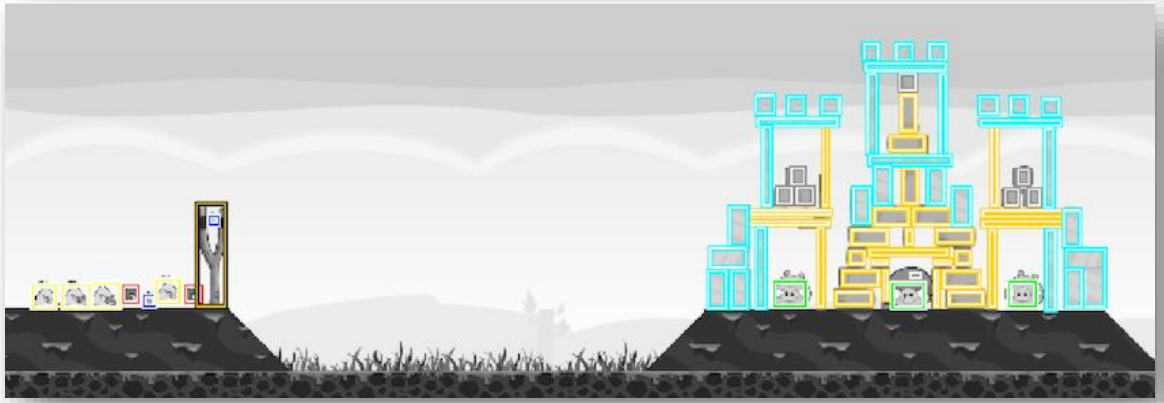
Dungeon Crawlers

- Fantasy role-playing worlds in which a player navigates through some sort of labyrinth (dungeon); e.g. Hades (2020)
- *Nethack Challenge:*
  - Game remains unsolved; standard Deep RL agents are far from solving it
  - Motif: aims to solve the problem of sparse and potentially misleading rewards in by constructing intrinsic rewards. These rewards are generated via LLMs for preference elicitation.
- *MiniDungeons*
  - Introduction of procedural personas: Q-learning, neuroevolution



Puzzle Games

- Typically, puzzle games feature complete or near-complete observability
- Angry Birds AI Competition: runs since 2012, mainly in conjunction with IJCAI. Early approaches: planning and reasoning. More recent work: deep RL, and deceptive level design.
- Match 3-games (e.g., *Candy Crush*): imitation learning, RL



Strategy Games

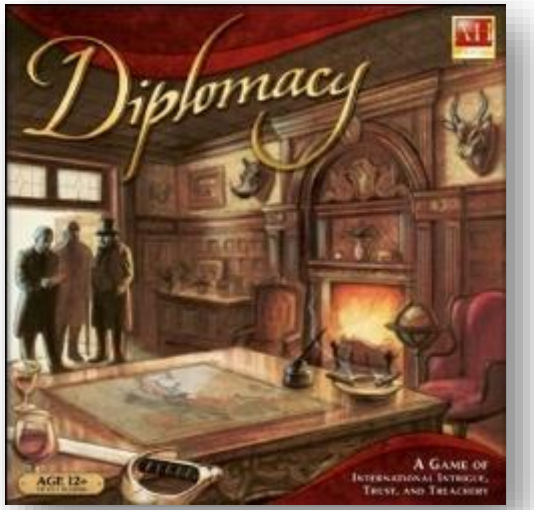
Strategy games: control multiple units, often in military-themed conquest/conflict.
Types: Turn-based (Civilization, XCOM) vs. Real-time (StarCraft, Age of Empires).
Challenges: Hidden info, long planning horizons, predicting multi-unit adversaries. Massive branching factors → millions of possible actions per turn.
Solutions:

- Decomposition: control units separately (tractable but uncoordinated).
- Sampling: Naive Sampling MCTS, non-linear Monte Carlo.
- Evolutionary planning: outperforms MCTS in high-branching games.
- NEAT & deep learning for macro-management, playtesting, balancing.

Strategy Games

Diplomacy

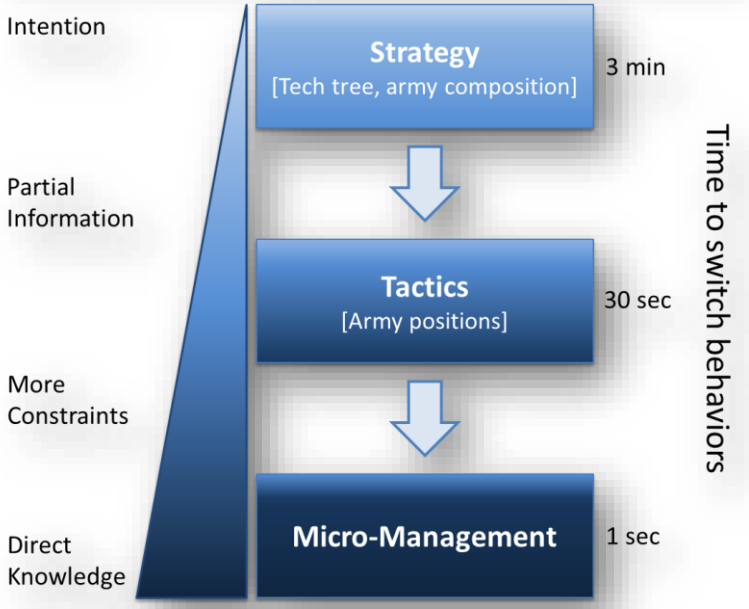
- Strategic board game (1959) with secret dealmaking as key mechanic
- **Challenges**: modeling belief spaces & unbounded player communication.
- **Early AI agents:** moderate performance due to negotiation complexity.
- Breakthrough (2022): Meta's *Cicero* – hybrid model-based RL + language model.
  - RL outputs intents → dialogue model generates natural English.
  - Dialogue model infers others' intents → feeds back to RL.
  - Emergent behavior: occasional deception, ungrounded conversational statements.



Strategy Games

StarCraft

- Strategic real-time strategy game of high complexity
- **Challenges**: real-time, high branching factor, hidden info
- **Early AI agents:** moderate performance; work focused on only part of the problem, commonly playing it at some level of abstraction; e.g. divide the different levels of decision-making (Strategy, Tactics and Micro-Management)
- **Recent AI:** AlphaStar and variants reaching grandmaster level



Strategy Games

Team Sports Games

- Coordination of multiple units that work together towards the goal of beating the other team.
- **RoboCup** (since 1996): several simulation leagues in 2D / 3D. Hard to disentangle gameplaying from problems derived from robotics.
- **Google Research Football Environment**: a recent yet increasingly popular framework for studying RL.
- **Sports10 dataset** was created to enable studies on style-invariant representation learning for gameplaying. Contains 100k gameplay images of 175 different sport games across 10 genres.

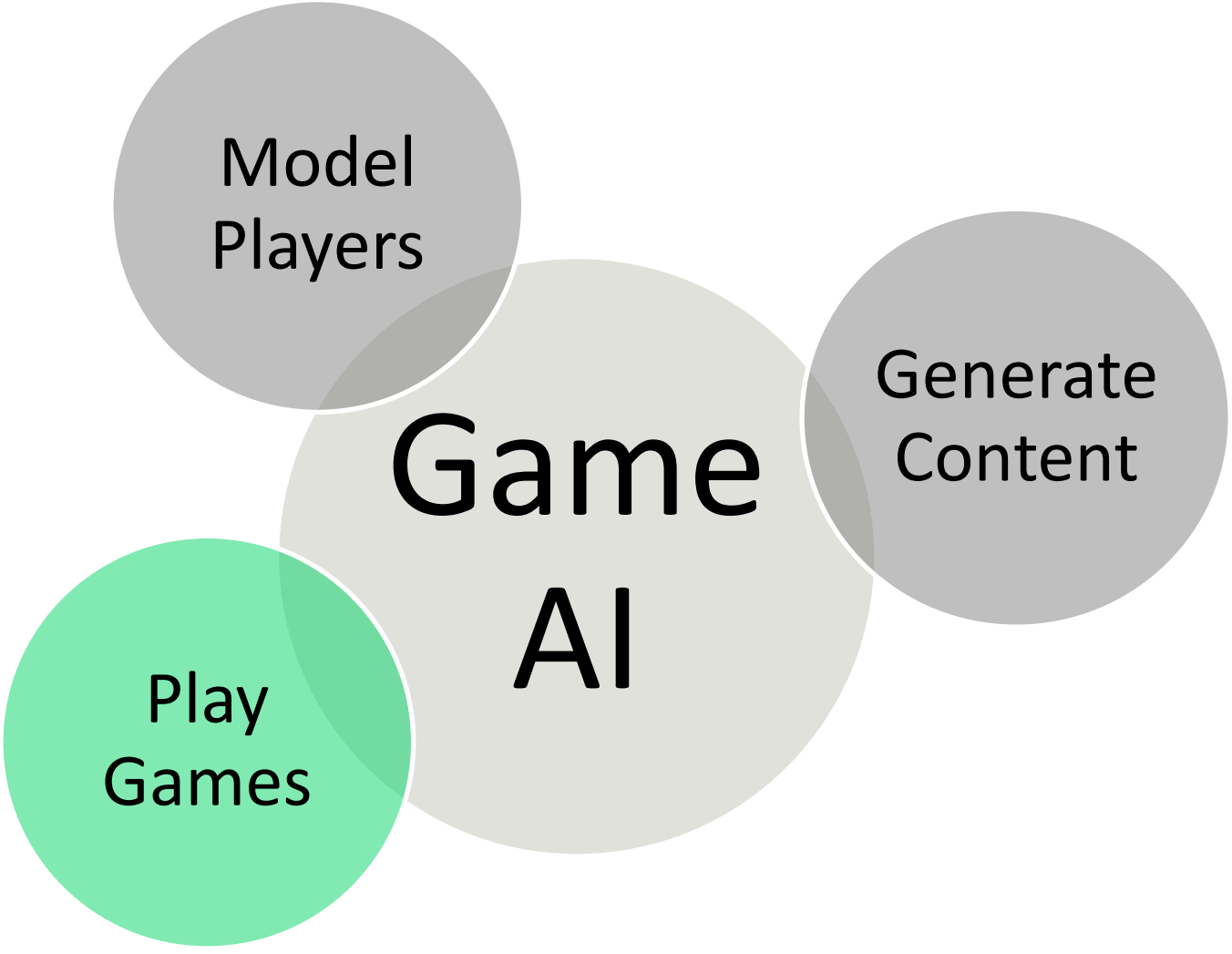
Text-Based Games

- Games in which players must use natural language commands to play, by interacting with objects and characters and solving puzzles. Fantasy is a popular genre for text-based games (aka interactive fiction), e.g., *Zork* and like *Façade*
- Interactive fiction
  - AI as NLP, Deep learning (LSTM, Deep Q networks) for text processing and generation
- Recent years: Text-based interaction dominated by LLMs; both playing such games and providing NPC interaction. e.g. *AI People*

Social Simulation Games

- Games in which players explore social interactions with multiple other game characters predominantly controlled by AI; e.g., *The Sims* series and *Animal Crossing.*
- **Popular AI Methods:** Ad-hoc designed believable agent architectures, expressive agents, conversational agents and utility-based AI
- Recent methods are LLM-dominated; e.g. *Smallville* by Park et al. (2023)

Park et al.,. **Generative agents: Interactive simulacra of human behavior**. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.



G. N. Yannakakis and J. Togelius, “**Artificial Intelligence and Games**,” Springer, 2025.

Artificial Intelligence and Games

A Springer Textbook | By Georgios N. Yannakakis and Julian Togelius

About the Book | Table of Contents | Lectures | Exercises | Resources

About the Book

Second Edition Published!

Welcome to the Artificial Intelligence and Games book (2nd edition) published with Springer Nature in 202
the first comprehensive textbook on the application and use of artificial intelligence (AI) in, and for, games
book will be used by educators and students of graduate or advanced undergraduate courses on game A
practitioners at large.

Readings: Part II

gameaibook.org