CS 380: Artificial Intelligence

Lecture 16: Game AI

Artificial Intelligence

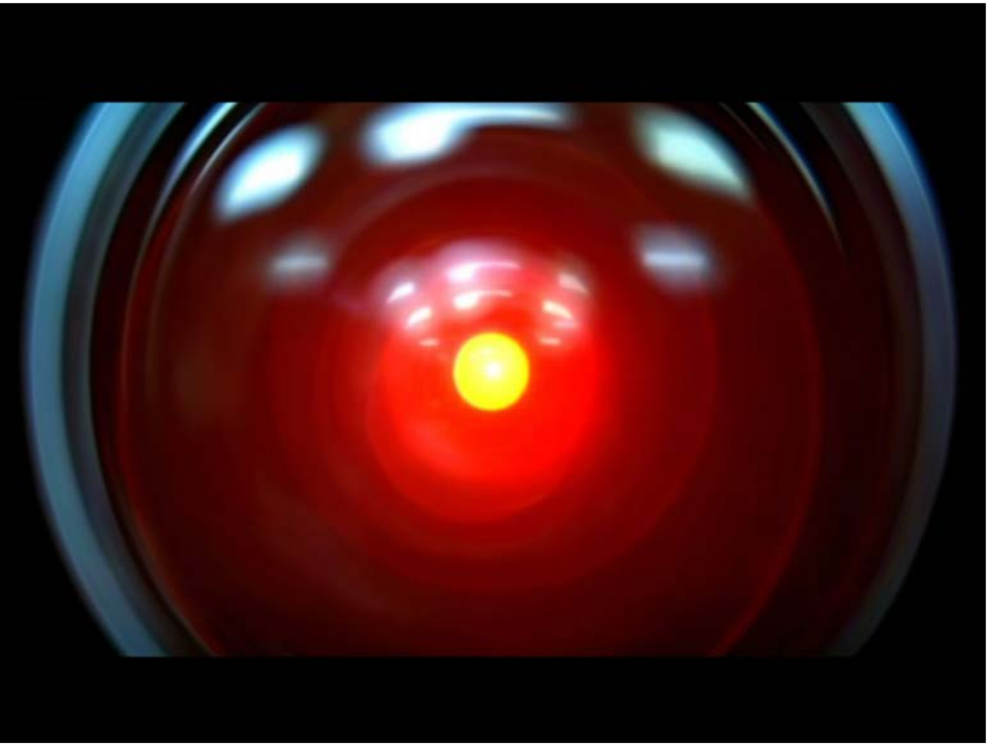
We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire.

The study is to proceed on the basis of the conjecture that **every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it**. An attempt will be made to find **how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves**.

- J. McCarthy et al.; Dartmouth AI Project Proposal; Aug. 31, 1955.

What is Game AI?

- Artificial Intelligence for Computer Games
- Different from traditional AI

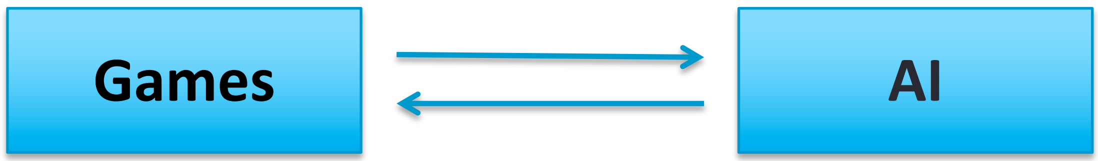


Traditional AI: Optimality, efficiency

Game AI: Fun, artificial "stupidity"

What is Game AI?

- Intersection of games and AI:



- Two (or three) main communities working on it:
  - Academics:
    - AI community: how can games improve AI? (AI-centric)
    - Computer Game scholars: how can AI help us have better/more interesting/new forms of games? (game-centric)
  - Game industry:
    - How can we make better games that sell more? (game-centric)

First Ever Game to Feature AI?

First Ever Game to Feature AI?

- Dr. NIM from the 1960s?

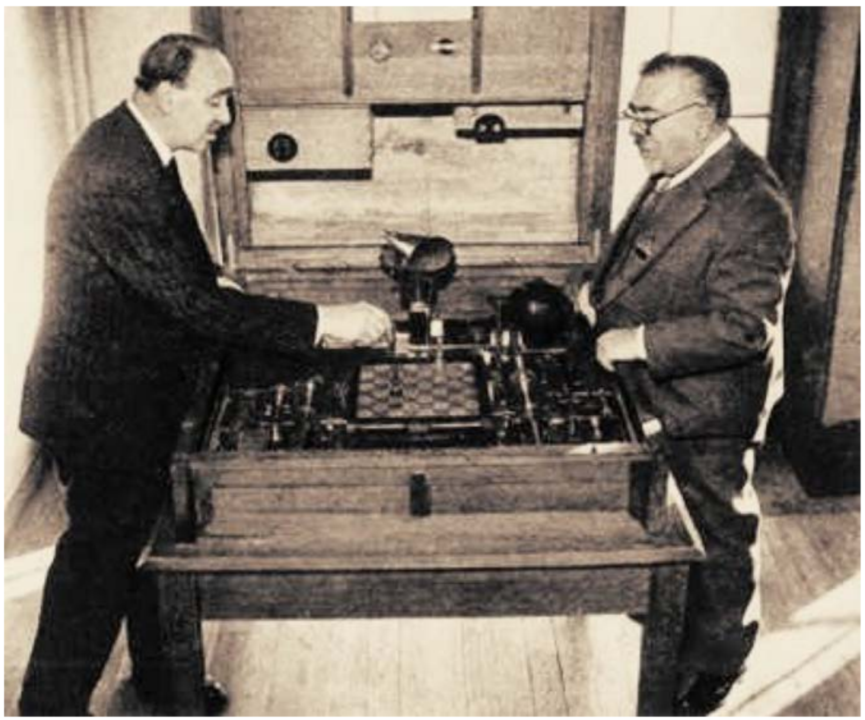
  - NIM (1 version of the game anyway): 12 coins; each player takes 1-3 coins; player who takes last coin loses
  - https://www.youtube.com/watch?v=9KABcmczPdg
  - https://www.youtube.com/watch?v=oxBghtQ8McA

First Ever Game to Feature AI?

- Dr. NIM from the 1960s?
  - NIM (1 version of the game anyway): 12 coins; each player takes 1-3 coins; player who takes last coin loses
  - https://www.youtube.com/watch?v=9KABcmczPdg
  - https://www.youtube.com/watch?v=oxBghtQ8McA

First Ever Game to Feature AI?

- "El Ajedrecista" ("The Chess Player), 1914
  - by Leonardo & Gonzalo Torres y Quevedo



"In the first version, the pieces were plugged into the board, and the **game states of check and checkmate were signaled with light bulbs**. Leonardo's son Gonzalo made an improved chess automaton based on El Ajedrecista in 1920, which **made its moves via electromagnets located under the board**. It also included a sound effect, with a **voice recording announcing checkmate** when the computer won the game."

[https://en.wikipedia.org/wiki/El_Ajedrecista]

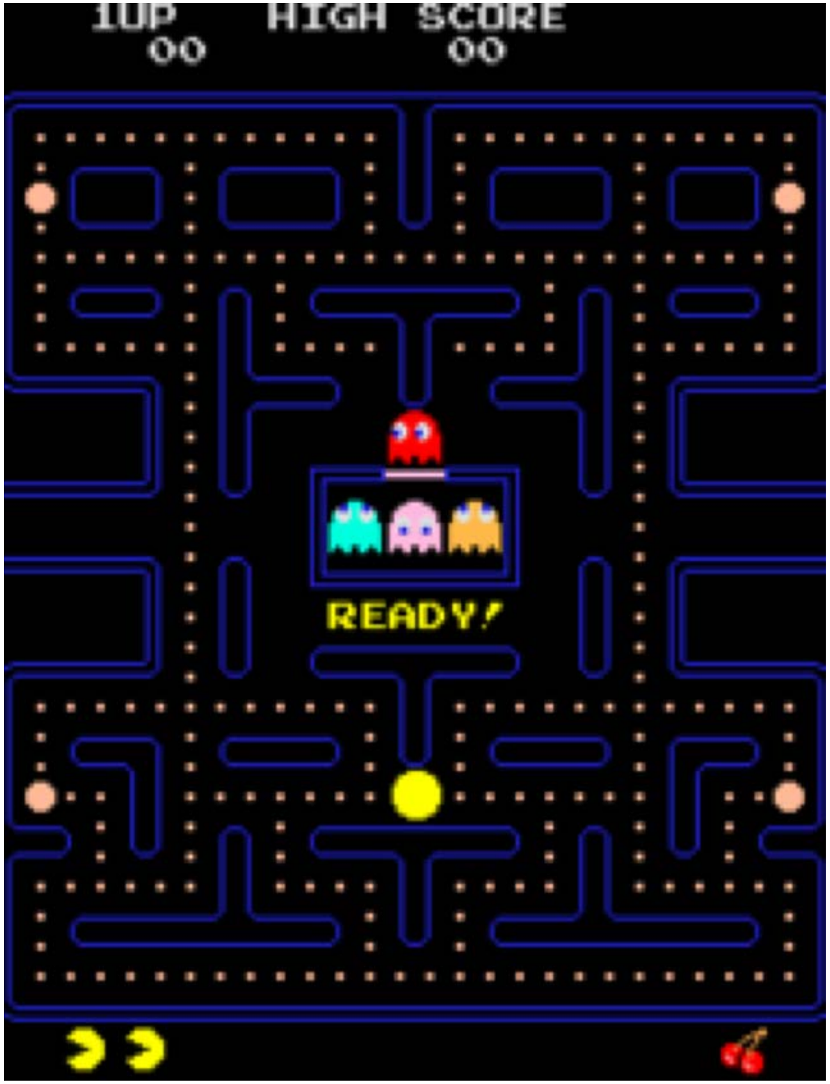
Examples of Game AI

Board Games

We've seen how AI is used to challenge the world's top players.

But for mere mortals, AI needs to provide a **variety of difficulty levels**.

Examples of Game AI

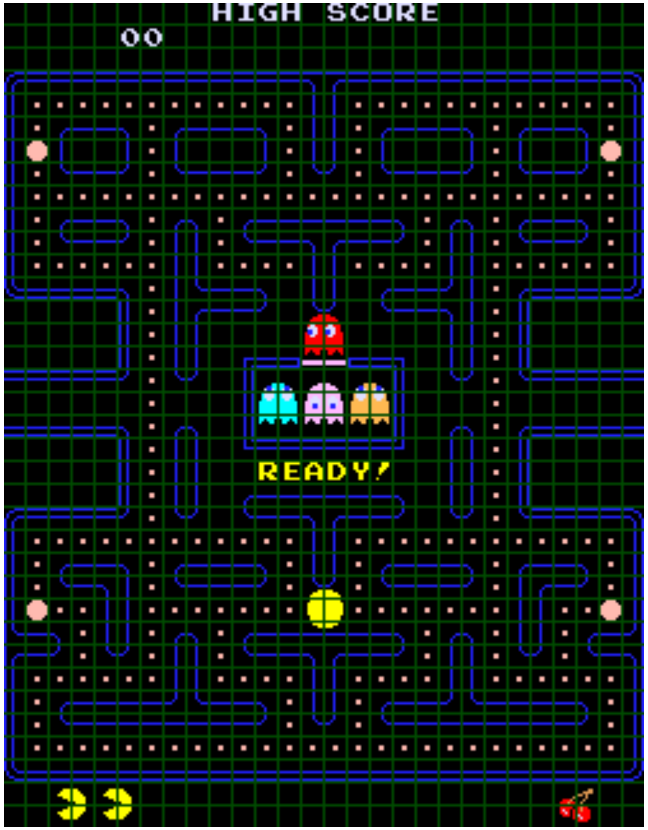


"Pac-Man" (1980)
First ever video game to feature AI

Where was the AI?

Pac-Man

- Let's first look at the engine for the AI ghosts in the old-school game of Pac-Man
- First, the seemingly complex board is divided into tiles, namely a set of "target tiles" that the ghosts use to navigate...



Details on this and subsequent slides derived from
http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior

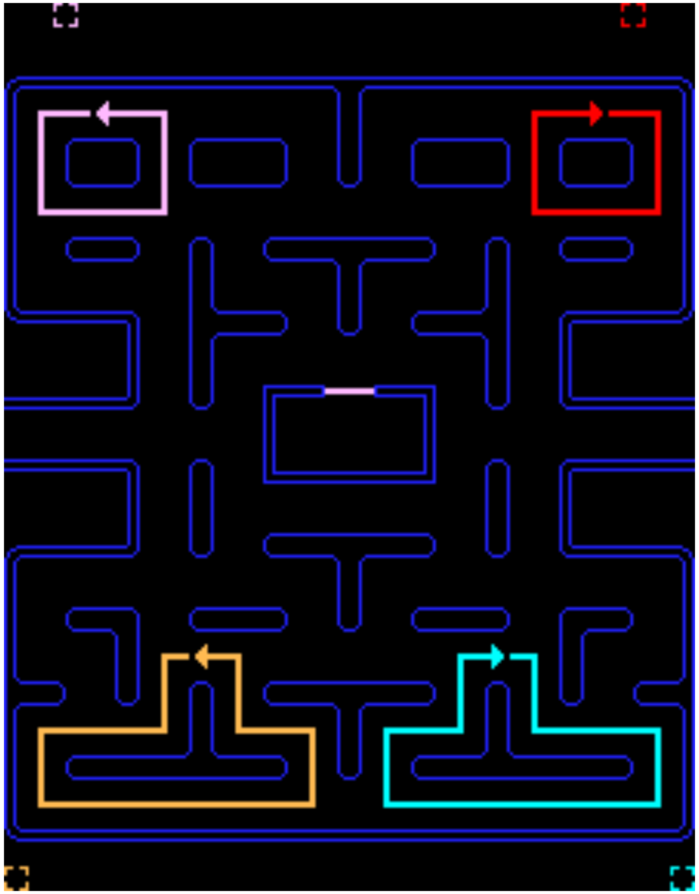
Pac-Man

> *"To give the game some tension, I wanted the monsters to surround Pac Man at some stage of the game. But I felt it would be too stressful for a human being like Pac Man to be continually surrounded and hunted down. So I created the monsters' invasions to come in waves. They'd attack and then they'd retreat. As time went by they would regroup, attack, and disperse again. It seemed more natural than having constant attack."*
> *— Toru Iwatani, Pac-Man creator*

- Ghosts always follow one of 3 modes:
  - Chase: pursue the Pac-Man
  - Scatter: disperse by moving toward a home tile (a different home for each ghost)
  - Frightened: pseudo(!)-randomly turn at each intersection

Pac-Man

- Scatter mode…



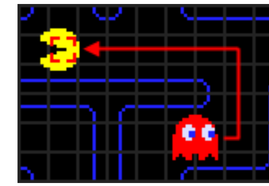
Pac-Man

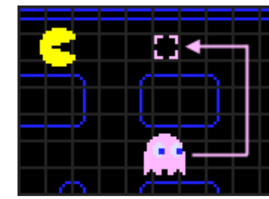
- The distribution of modes changes at each level
  - Level 1
    - Scatter for 7 seconds, then Chase for 20 seconds.
    - Scatter for 7 seconds, then Chase for 20 seconds.
    - Scatter for 5 seconds, then Chase for 20 seconds.
    - Scatter for 5 seconds, then switch to Chase mode permanently.
  - Level 2+
    - Shorten Scatter mode
    - Lengthen Chase mode

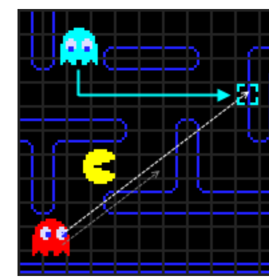
Pac-Man

- Ghost personalities

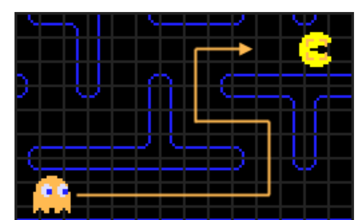
"This is the heart of the game. I wanted each ghostly enemy to have a specific character and its own particular movements, so they weren't all just chasing after Pac Man in single file, which would have been tiresome and flat."
— Toru Iwatani, Pac-Man creator

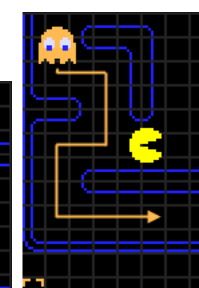






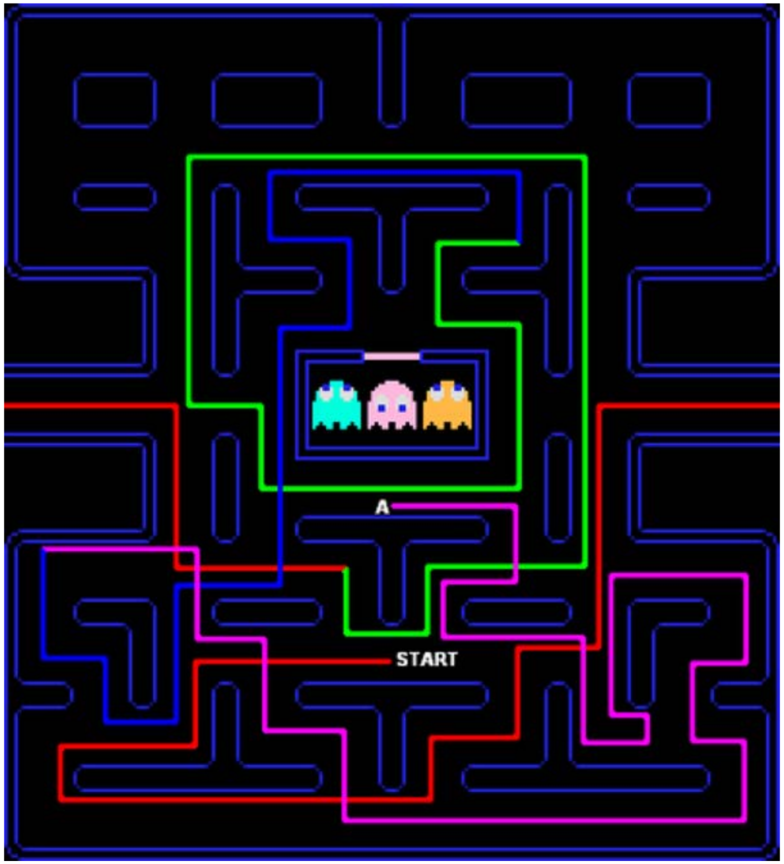
  - Red (Blinky): "chaser"
  - Pink (Pinky): "ambusher"
    - Target 4 squares ahead of Pac-Man
      (you can "play chicken" by forcing him away!)
  - Blue (Inky): "whimsical"
    - Considers Blinky's position in addition to Pac-Man's
  - Orange (Clyde): "feigning ignorance"
    - chase/scatter alternation





Pac-Man

- Because there’s no randomness (only pseudo-random), there are routes that work every time!



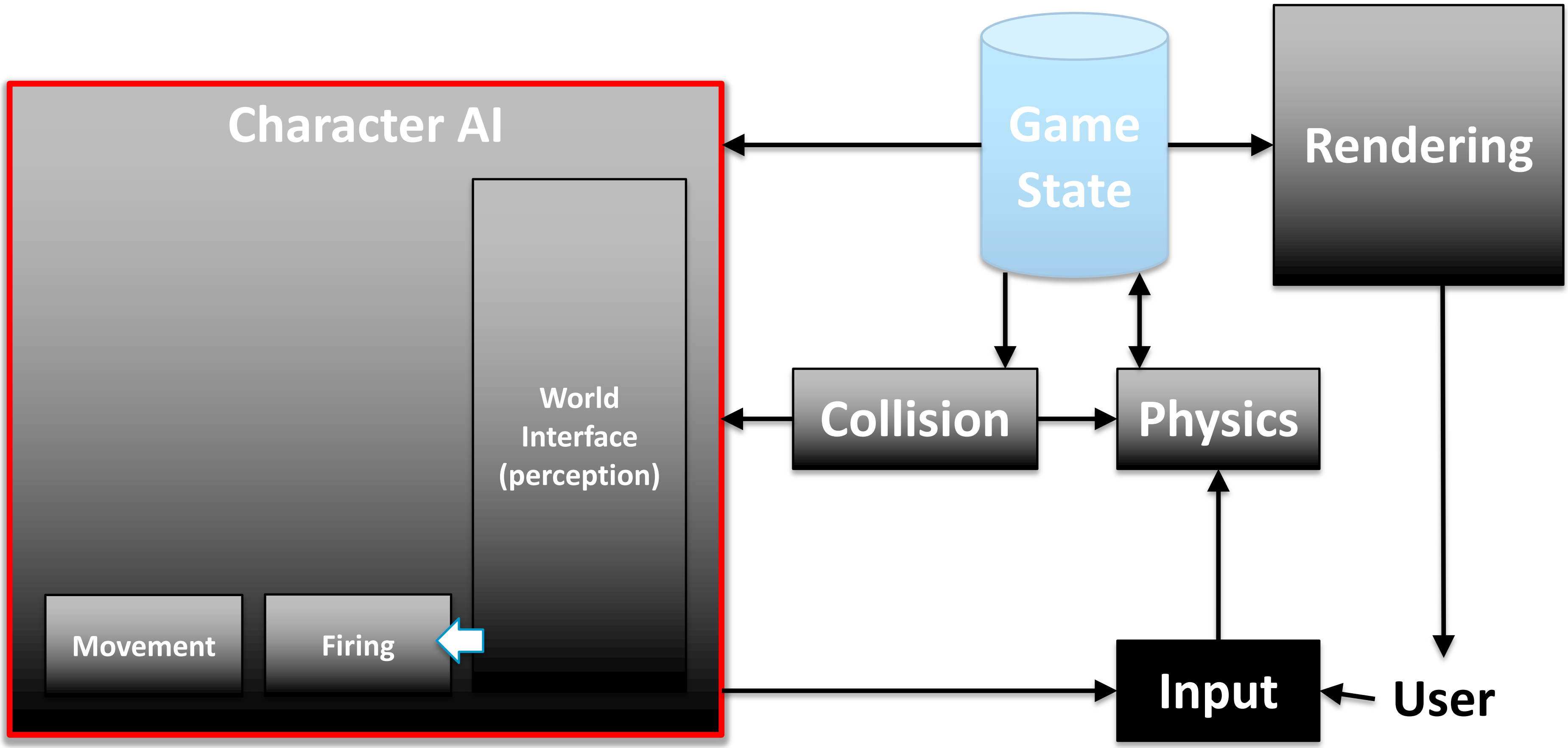
Types of Game AI

- **Inside** the game:
  - Character control
  - Director (drama management)
- **During** game development:
  - Help in behavior/content design
- **After** game deployment:
  - Analysis of game data

AI in Modern Games

- What kinds of AI components are standard in today's games?
- For example, consider "first-person shooter" (FPS) games…
  - Two basic needs: Movement + Firing
  - With those, you already have a basic FPS running
- But you need much more than that for modern game AI…

Game Engine



Character AI
World
Interface
(perception)
Movement
Firing
Game
State
Rendering
Collision
Physics
Input
User

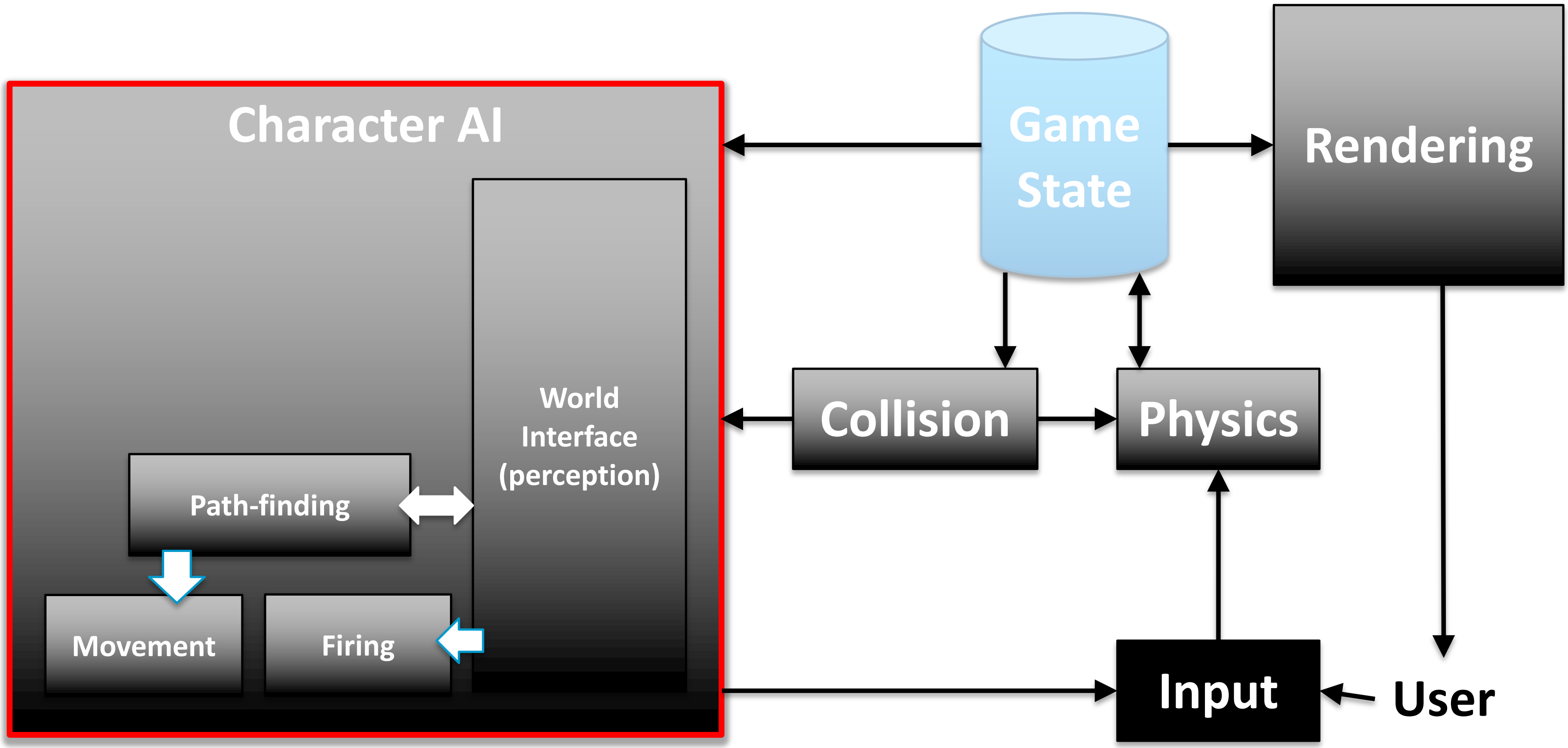
AI in Modern Games

- In modern games, you don’t expect:
  - Enemies to get stuck behind a wall (enemies in old games like Wolfenstein do)
  - To stand still while you kill their comrades (they do in some old games)
  - To be trapped just because you closed a door!
  - To move independently of each other
  - etc.

Path-Finding

- Not all enemies are given path-finding capabilities:
  - Path-finding makes characters look smart; the ones that should not look smart should not have path-finding
- Representation of the navigation graph?
  - Navigation meshes is the most popular
- Embedded navigation:
  - Annotate links in the navigation graph with the action required to traverse it (walk, climb, open door, etc.)
  - Characters look smart knowing they have to perform those actions, but it’s all hand-annotated or precomputed

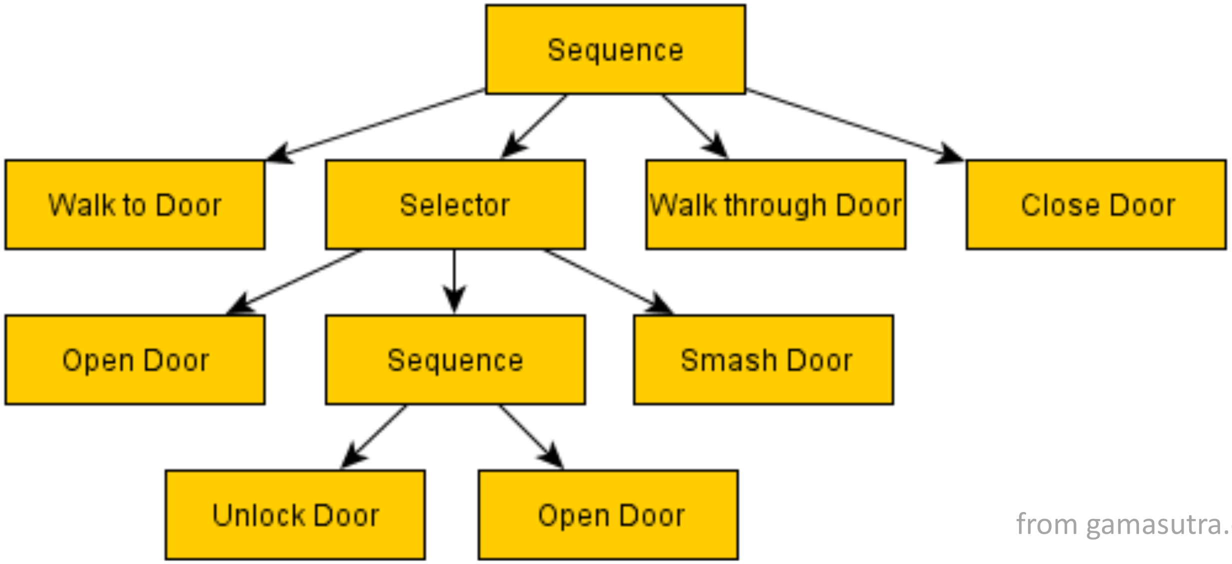
Game Engine



Character AI
World Interface (perception)
Path-finding
Movement
Firing
Game State
Rendering
Collision
Physics
Input
User

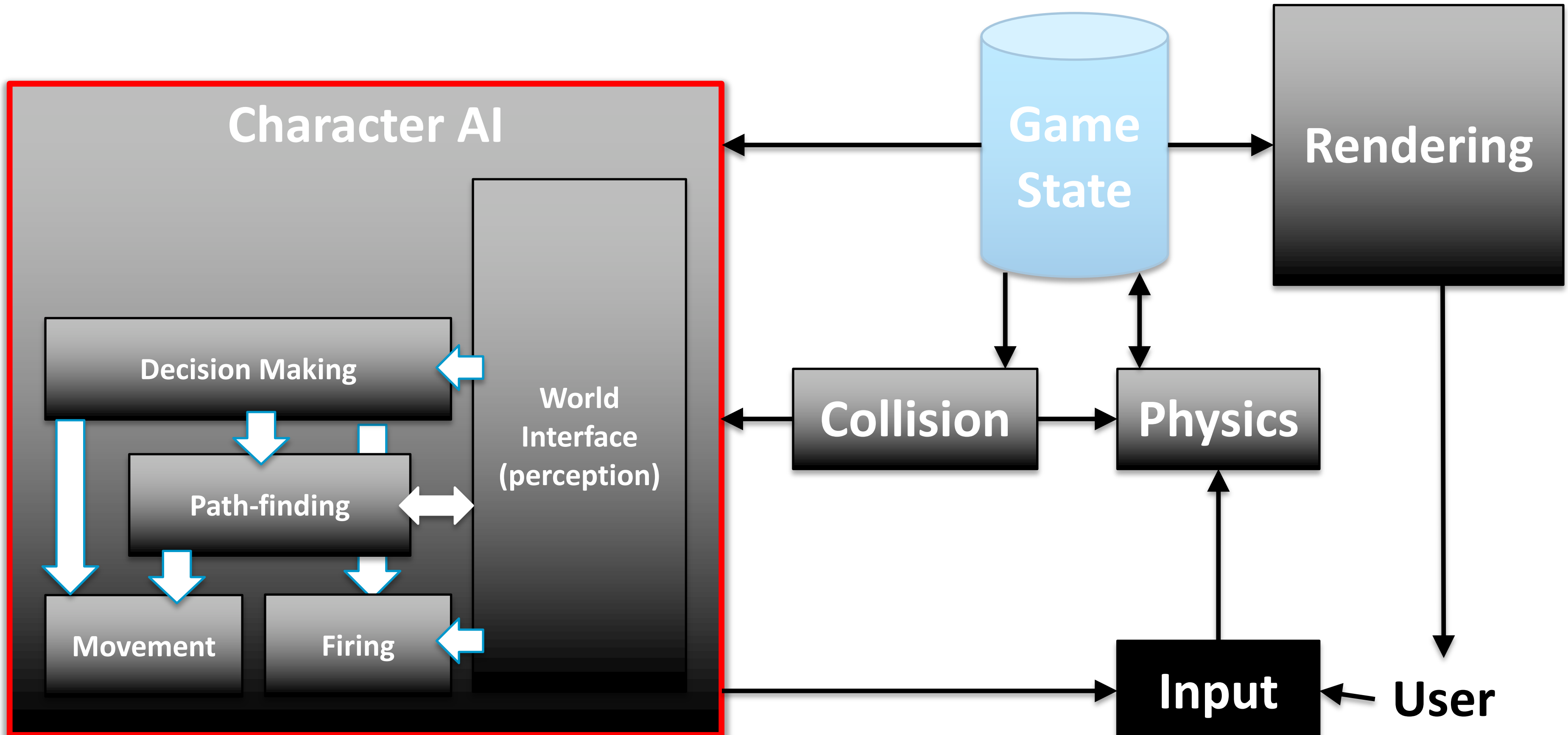
Decision-Making

- Decision-making controls what characters do at a high level: Do they attack? Do they retreat? Do they execute other actions?
- Classic games: Finite-state machines
- Modern games: Behavior Trees



from gamasutra.com

Game Engine



Character AI
Decision Making
Path-finding
Movement
Firing
World Interface (perception)
Game State
Rendering
Collision
Physics
Input
User

Group Control

- Coupled with better decision-making, this is the largest difference between classic and modern game AI.
- In old games (Doom, etc.), enemies always rush!
- Modern games use group tactics:
  - Simplest is “kung-fu style” (one enemy at a time)
  - Complex attack tactics: some characters attack while others cover or wait or …

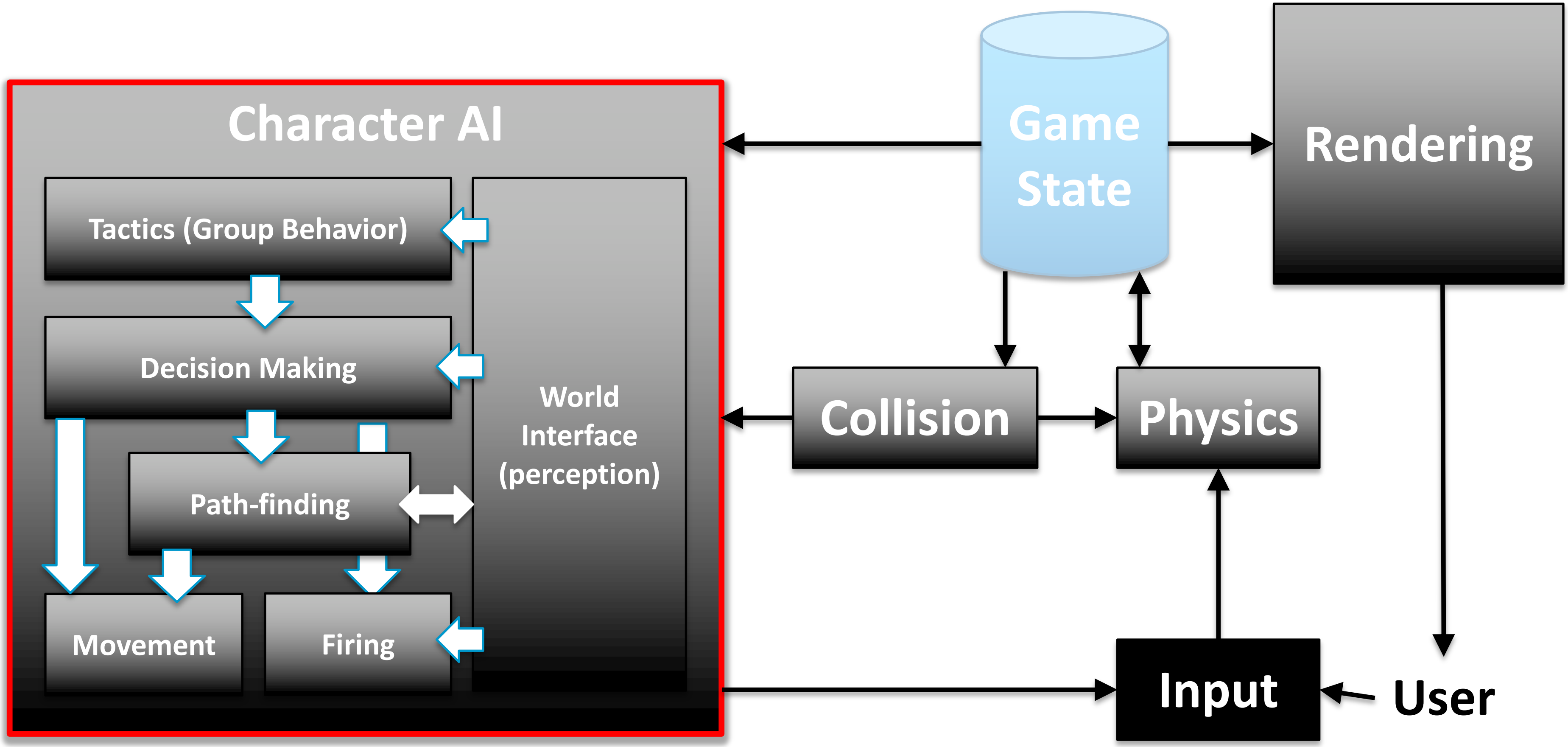
Examples of Game AI

- FIFA {*n*} (soccer)
  - https://www.youtube.com/watch?&v=RWXIdfj-9QA

Group Control

- Can be also defined using behavior trees
- Most common approach:
  - Each tactic defines a set of “roles”
  - A role is filled by one character
  - Each tactic has a behavior tree that executes it
- For example:
  - A “flank” tactic can have 3 roles for “left attack”, “front attack”, and “right attack”
  - Only 3 enemies will execute it at once (if there are more, they will wait, kung-fu style)
  - Enemies look smart, but **do not overpower the player**

Game Engine

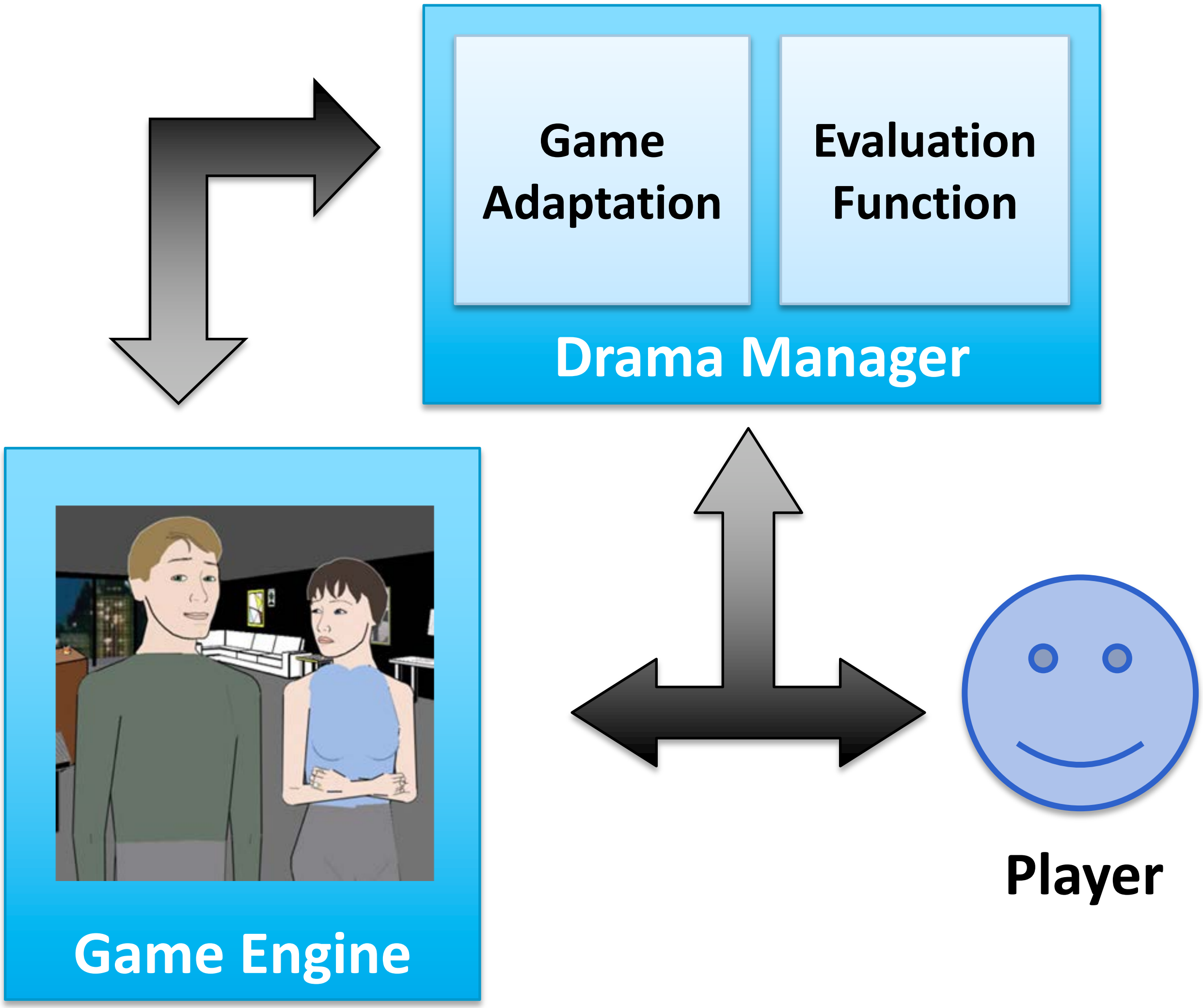


Character AI
Tactics (Group Behavior)
Decision Making
Path-finding
Movement
Firing
World Interface (perception)
Game State
Rendering
Collision
Physics
Input
User

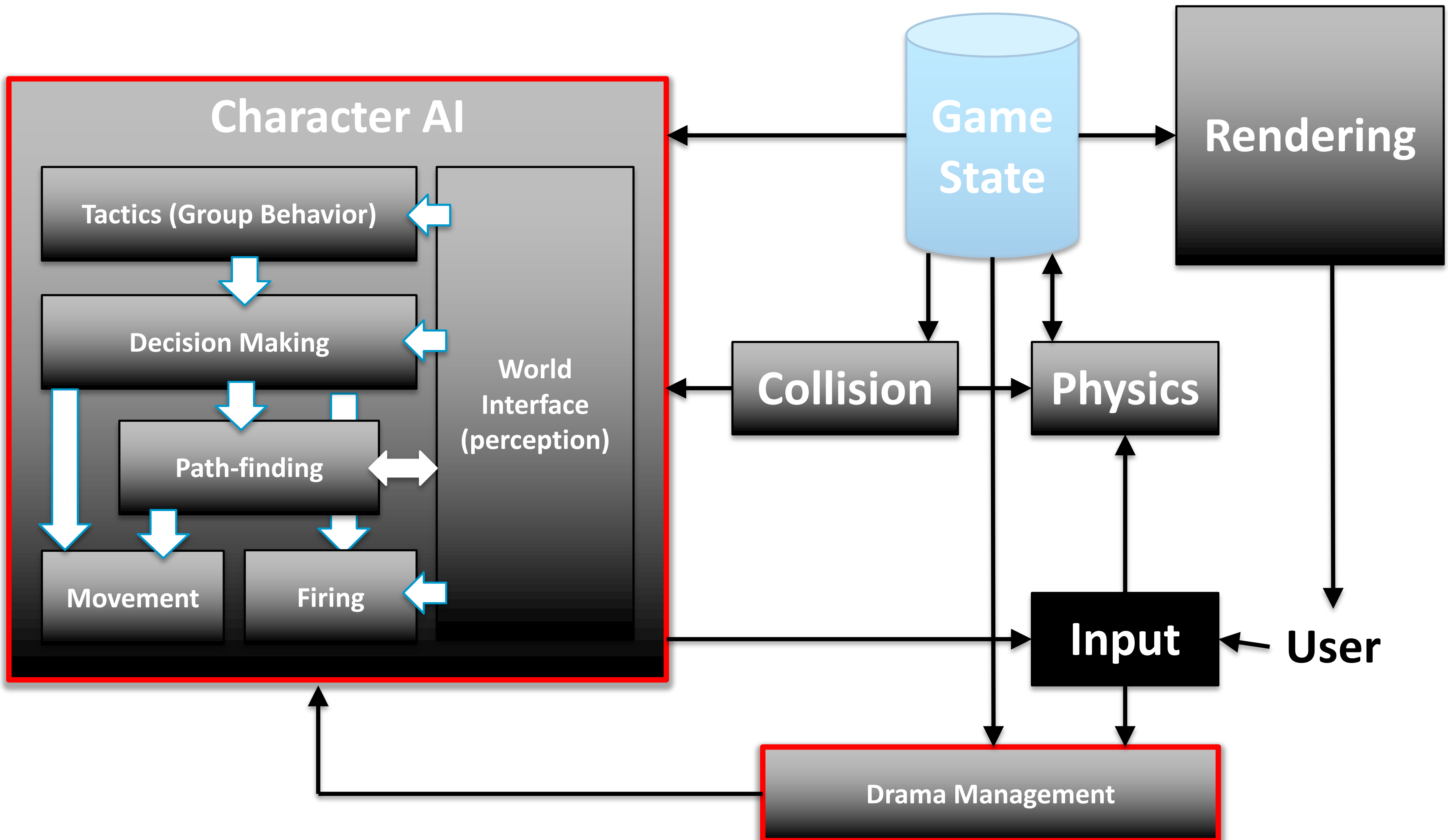
Drama Management / Directing

- Premise:
  - Different players have different preferences
  - Each player enjoys a different subset of the game
- Problem:
  - How can a game automatically adapt to match the player?
    - Maintain the dramatic arc expected by the game designer
    - Prevent the players from dealing with unappealing game areas
    - Adjust difficulty level, etc.
- Solution:
  - Drama Management / Experience Management / AI Director

Drama Management



Game Engine


Character AI
Tactics (Group Behavior)
Decision Making
Path-finding
Movement
Firing
World Interface (perception)
Game State
Rendering
Collision
Physics
Input
User
Drama Management

Intelligent Camera Control

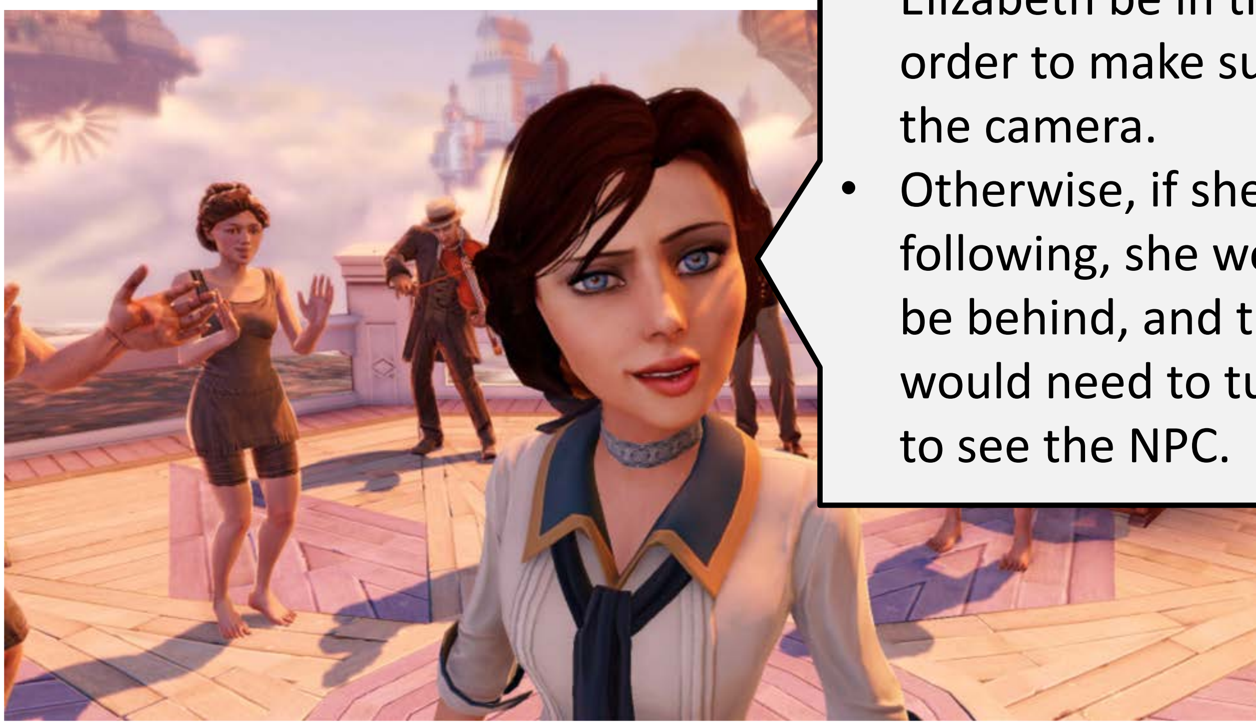
- Recent games have an emphasis on story-telling
- Camera control is an essential component in story telling, since it provides **focus**
- A lot of recent work on automatic camera control:
  - E.g., http://www.youtube.com/watch?v=8G0xfyDGLsk&feature=related
- For example, for non-player characters (NPCs)…

NPCs and Camera Control

- What if you put a lot of work into NPC AI, but the player never "looks" at the NPCs?
- Example: Elizabeth in *Bioshock Infinite*

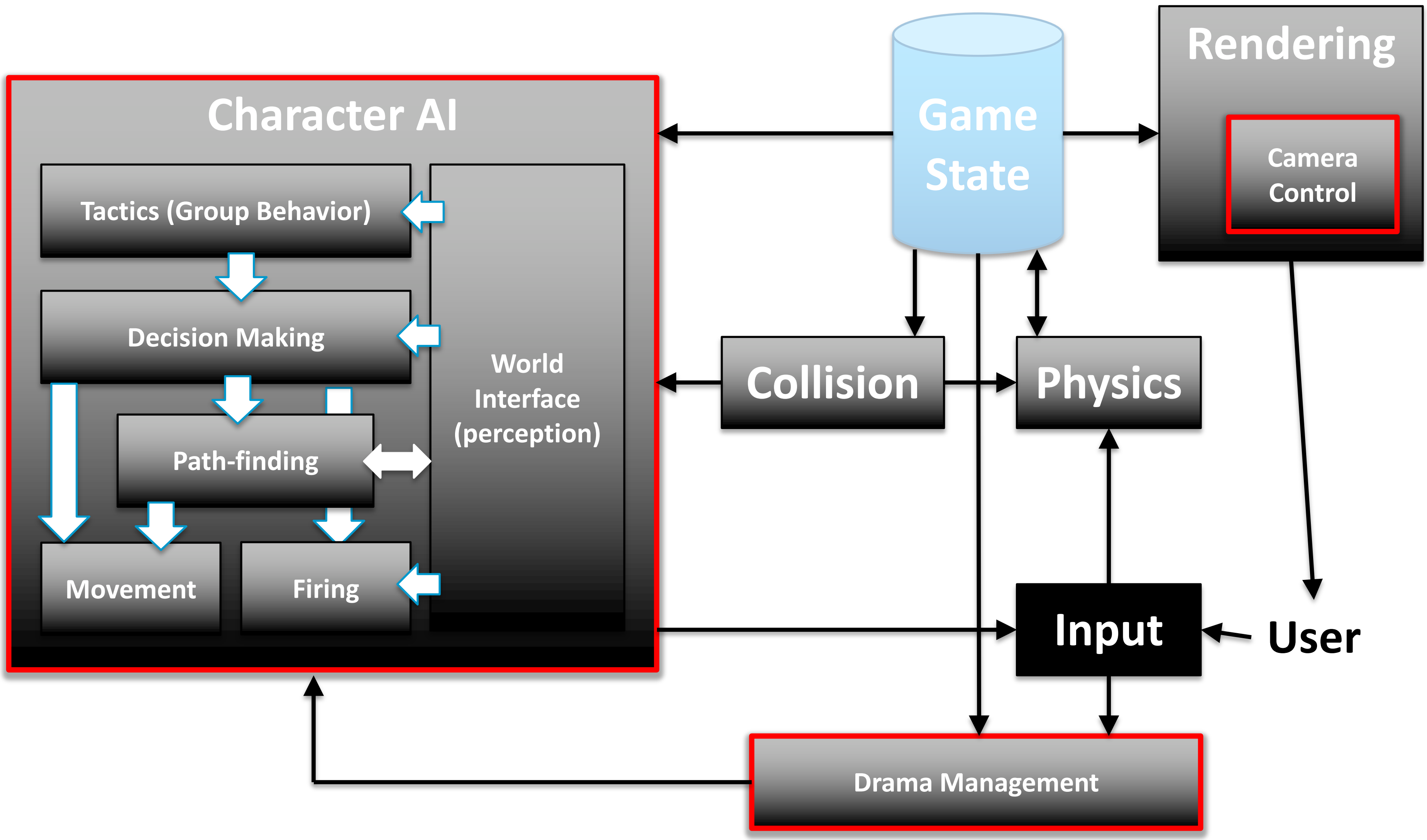
NPCs and Camera Control

- What if you put a lot of work in
  player never “looks” at the NPC
- Example: Elizabeth in *Bioshock*

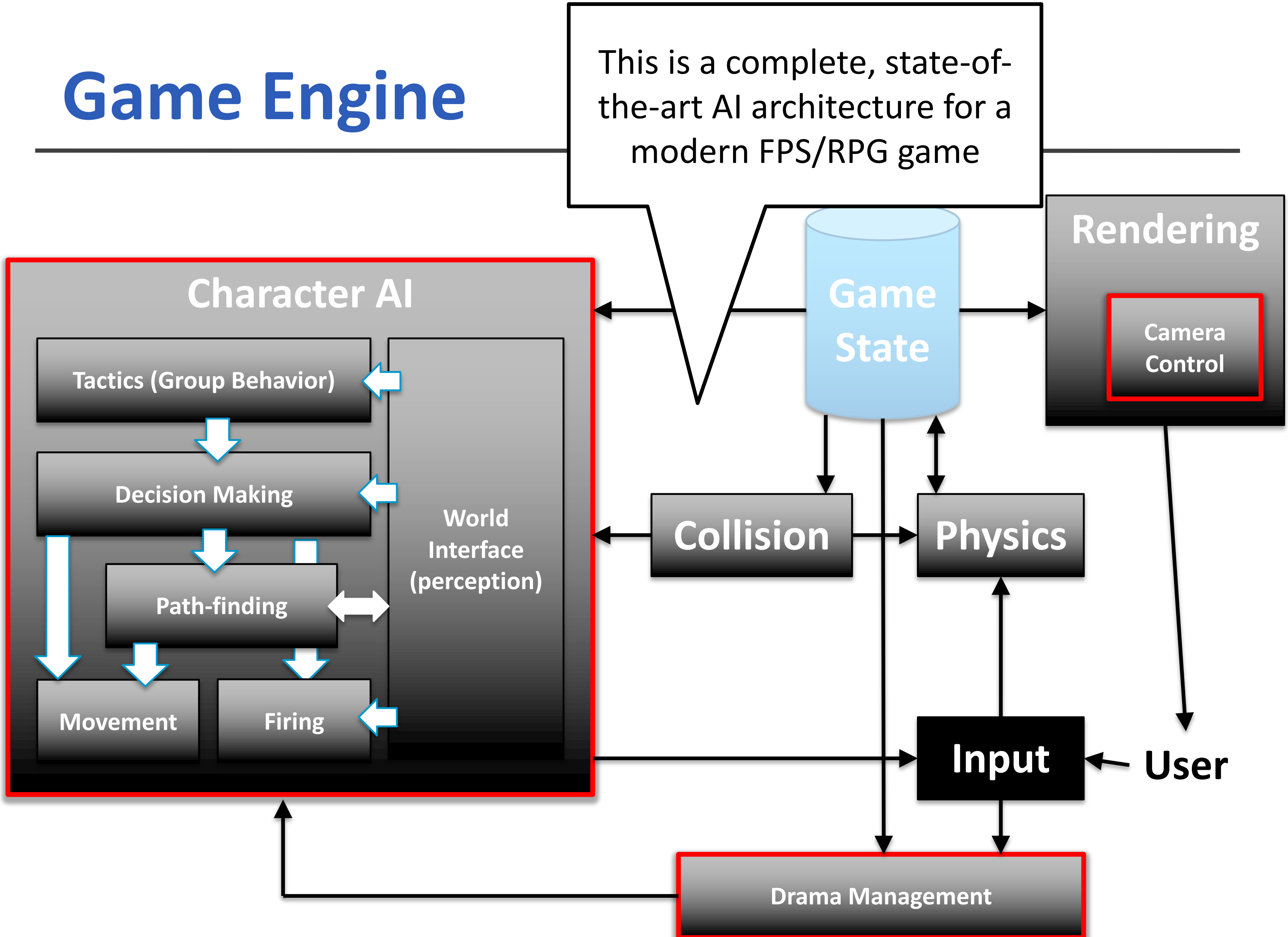


- The game predicts the path the player needs to take to get to the next “goal”.
- Then, the game makes Elizabeth be in that path in order to make sure she is on the camera.
- Otherwise, if she was just following, she would always be behind, and the player would need to turn around to see the NPC.

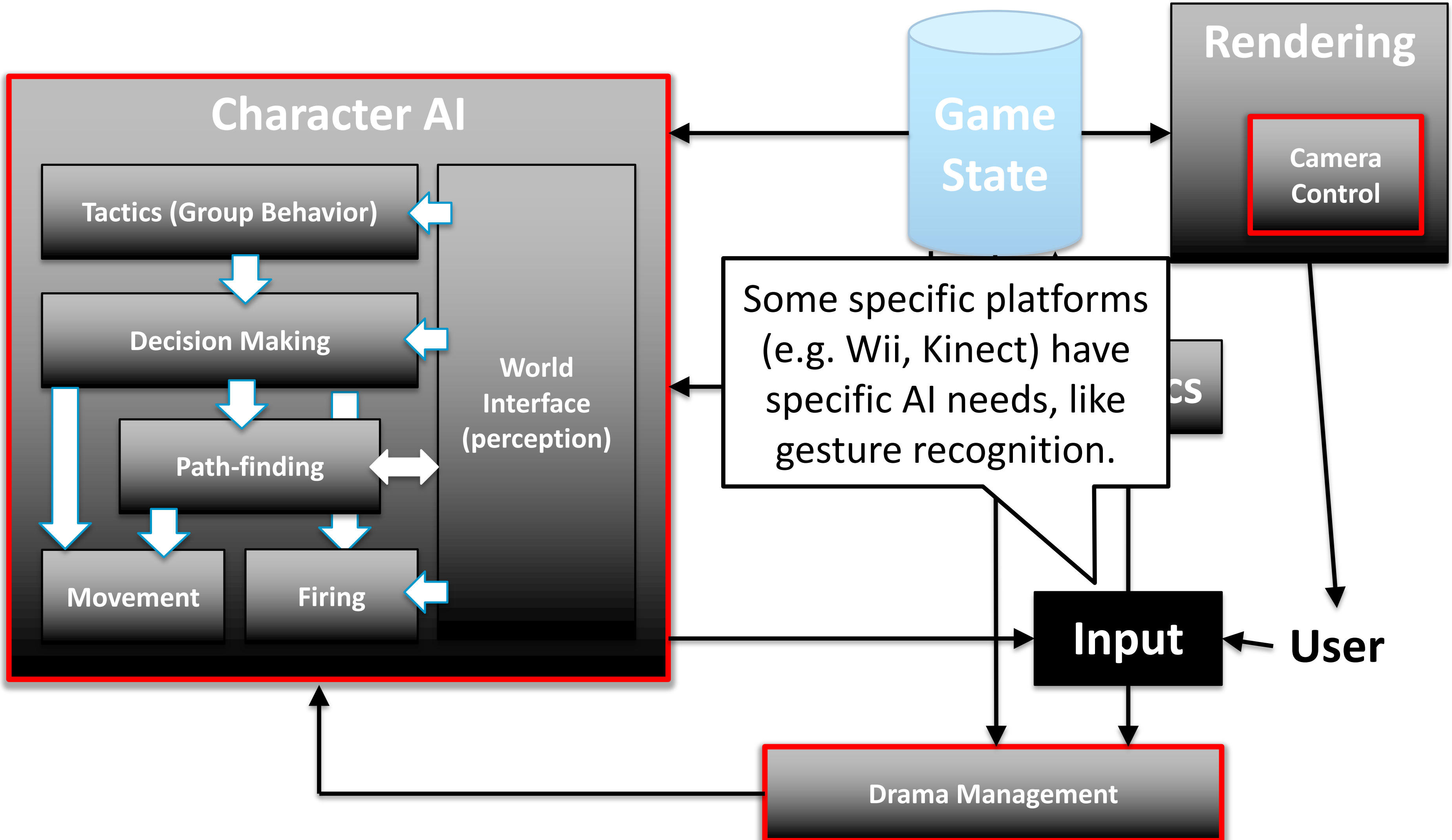
Game Engine


Character AI
Tactics (Group Behavior)
Decision Making
Path-finding
Movement
Firing
World Interface (perception)
Game State
Rendering
Camera Control
Collision
Physics
Input
User
Drama Management

Game Engine


This is a complete, state-of-the-art AI architecture for a modern FPS/RPG game
Character AI
Tactics (Group Behavior)
Decision Making
Path-finding
Movement
Firing
World Interface (perception)
Game State
Rendering
Camera Control
Collision
Physics
Input
User
Drama Management

Game Engine



Character AI
Tactics (Group Behavior)
Decision Making
Path-finding
Movement
Firing
World Interface (perception)
Game State
Rendering
Camera Control
cs
Some specific platforms (e.g. Wii, Kinect) have specific AI needs, like gesture recognition.
Input
User
Drama Management

Computational Narrative

- Algorithmically analyze, structure and generate stories.
- Generating stories is an “AI-complete” problem. Involves many yet-unsolved problems:
  - believable characters / emotional modeling
  - natural language generation
  - common sense reasoning
  - narrative aesthetics
- Let’s talk about story generation, and then we can discuss how would this be adapted to games.

The First Book Ever Written by a Computer

THE POLICEMAN'S BEARD IS HALF CONSTRUCTED

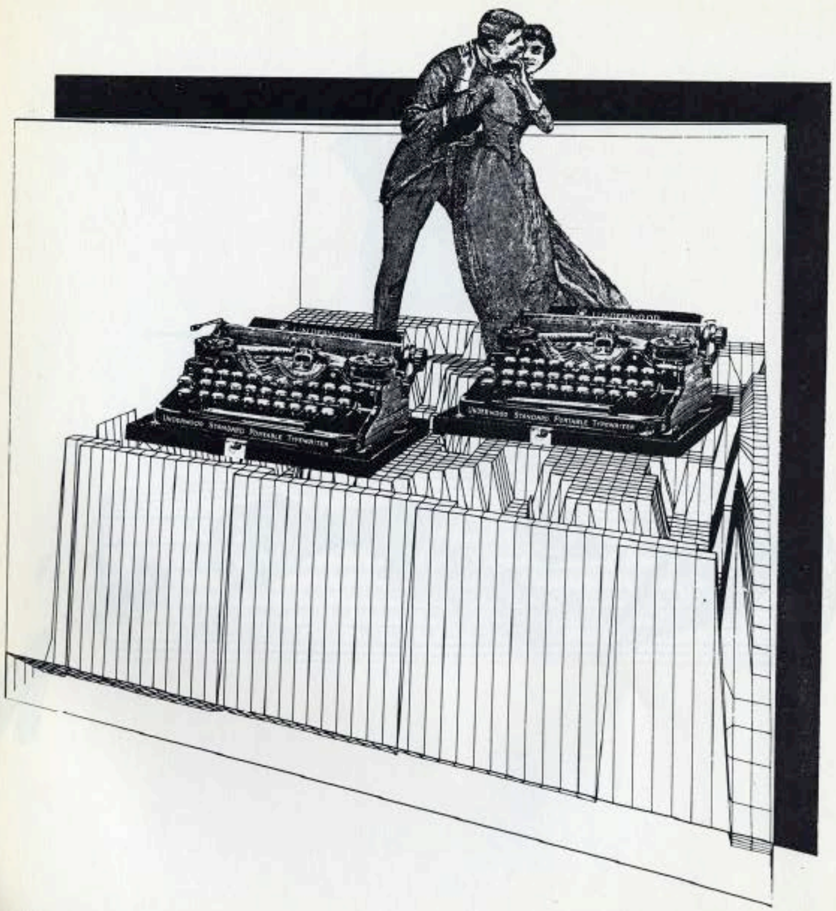
Computer prose and poetry
by
Racter

Illustrations by Joan Hall

Introduction by William Chamberlain

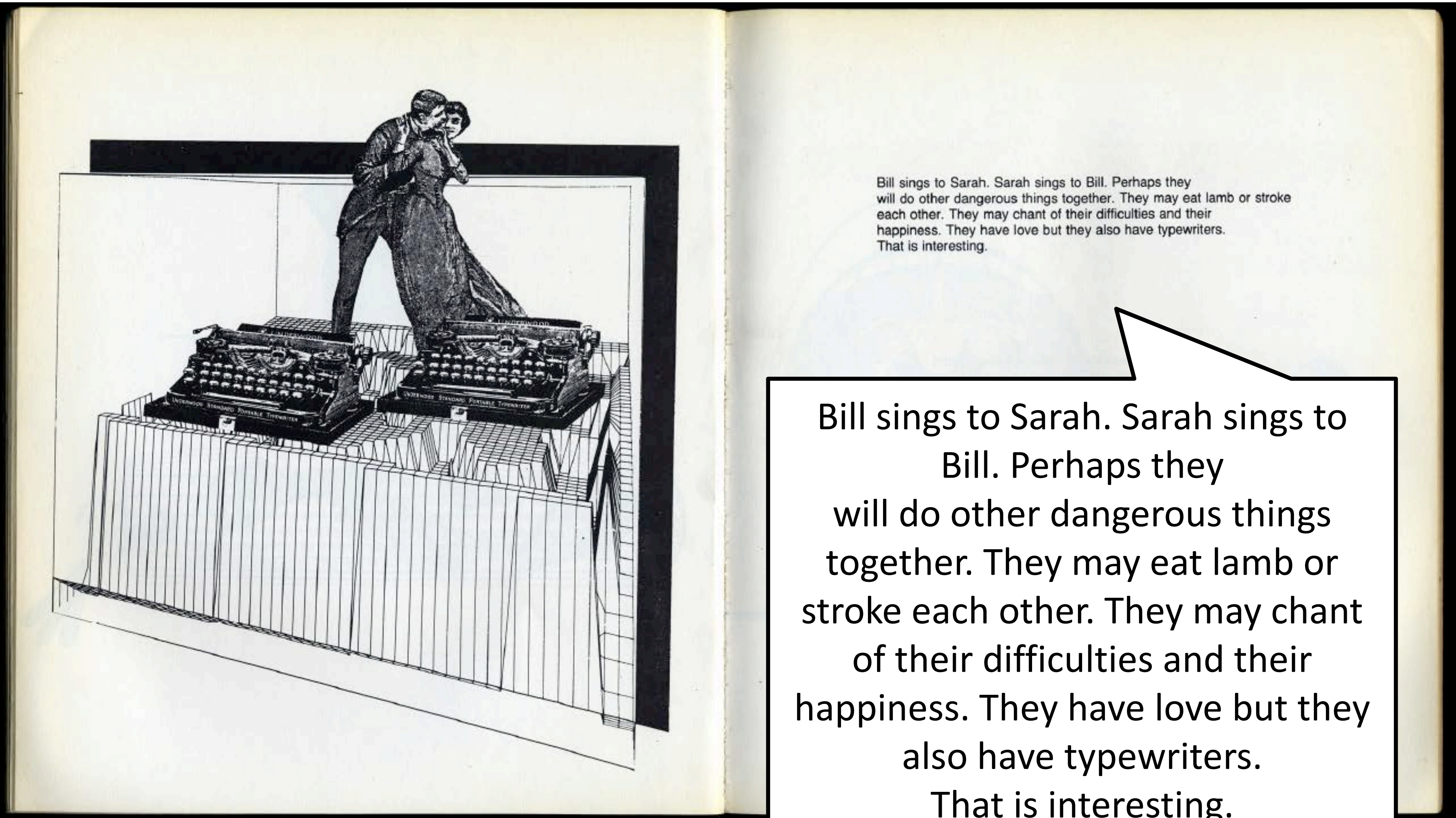
A Bizarre and Fantastic Journey into the Mind of a Machine

The Policeman's Beard is Half Constructed



Bill sings to Sarah. Sarah sings to Bill. Perhaps they will do other dangerous things together. They may eat lamb or stroke each other. They may chant of their difficulties and their happiness. They have love but they also have typewriters. That is interesting.

The Policeman’s Beard is Half Constructed

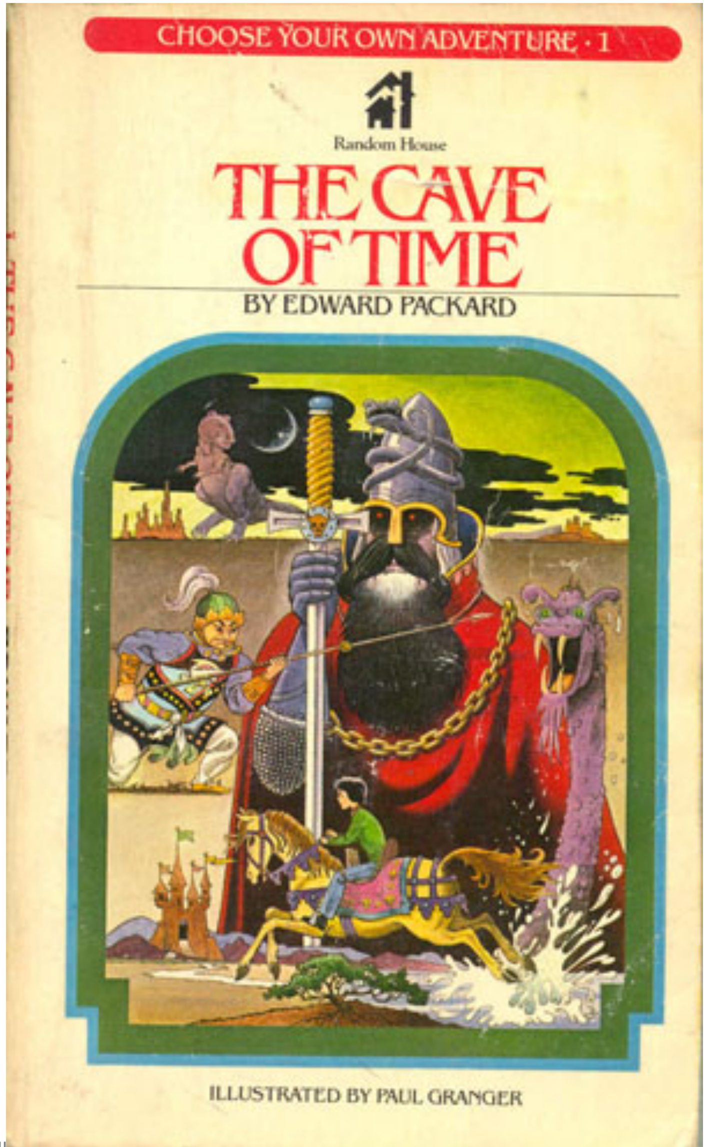


Bill sings to Sarah. Sarah sings to Bill. Perhaps they
will do other dangerous things together. They may eat lamb or stroke each other. They may chant of their difficulties and their happiness. They have love but they also have typewriters.
That is interesting.

Story Generation

- Computers are far from generating novels or game plots at the level of human authors.
- However, many useful techniques have emerged from the story generation research community.
- This is especially applicable to video games.

Choose Your Own Adventure



"But what I would most like to do, if you're interested, is go to my library and talk, and get to know each other better. I could tell you tales about the fall of the Great Underground Empire, and about the early campaigns against the evil Krill. I used to be considered a spellbinding storyteller."

"That sounds interesting," says Bivotar.

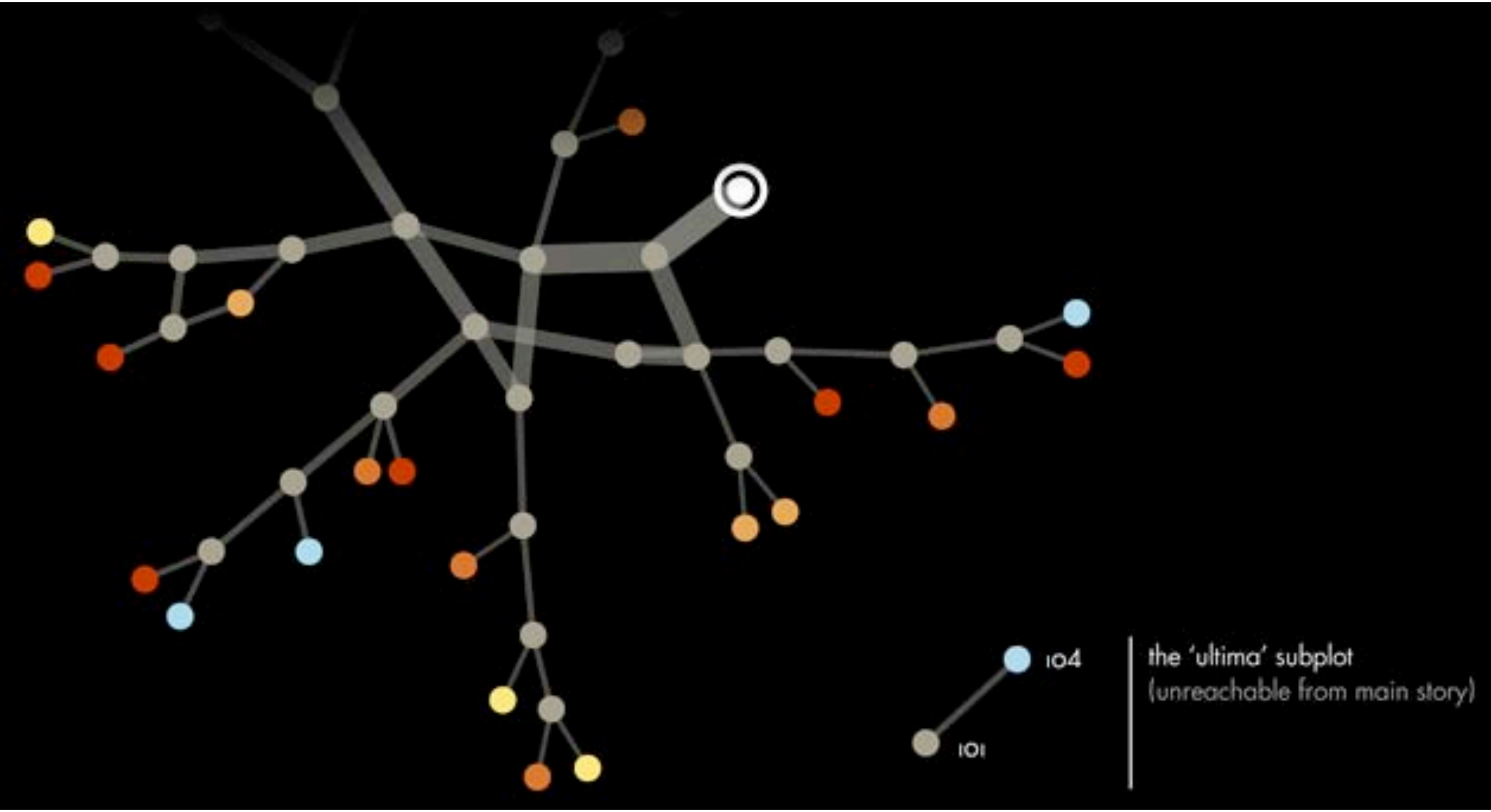
Visit the crafts fair?
Go to page 19.

Stay and listen to Syovar's stories?
Go to page 21.

Go to the banquet and play?
Go to page 13.

Attend the jousting finals?
Go to page 15.

Choose Your Own Adventure

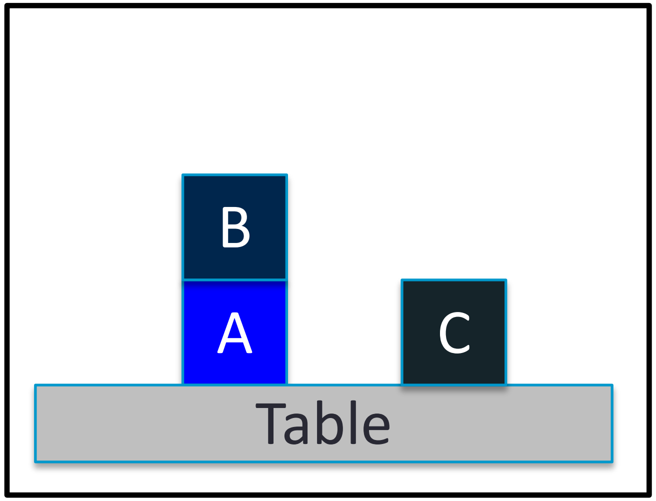


Automated Planning

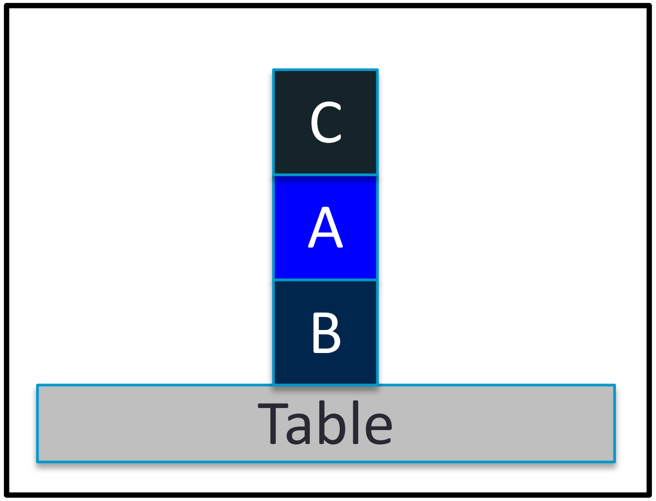
- Planning:
  - Find the sequence of **actions** that will take us from an **initial state** to a **target state**
- Automated Planning:
  - Typically solved with specialized search algorithms

Automated Planning: Example

- Blocks world



Initial State



Target State

Possible actions:

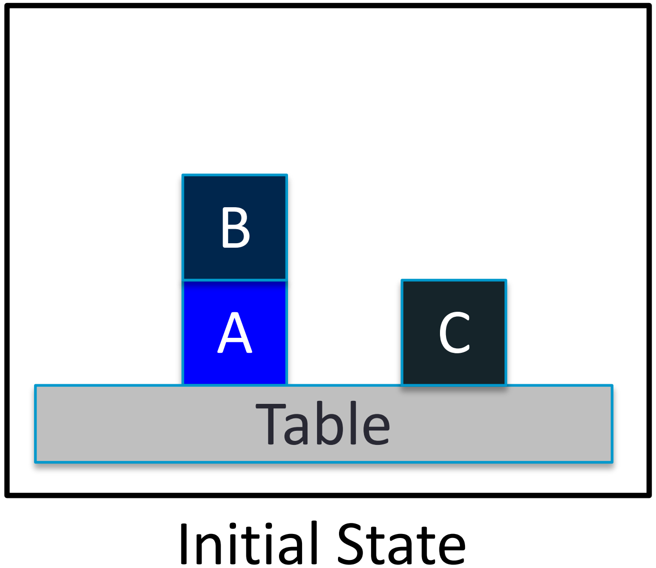
Take(X)

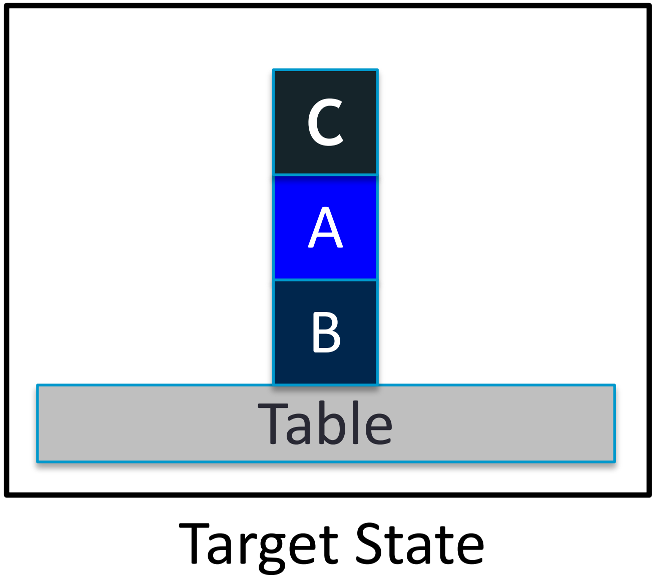
Put(X,Y)

Automated Planning: Example

- Action definition:
  - Take(X)
    - Preconditions:
      - We have nothing in our hands
      - X is a block
      - Nothing on top of X
    - Postconditions:
      - X is not on top of anything
      - X is in our hands
  - Put(X,Y)
    - Preconditions:
      - X is in our hands
      - Y is the table or Y is a block with nothing on top
    - Postconditions:
      - X is not in our hands
      - X is on top of Y

Automated Planning: Example





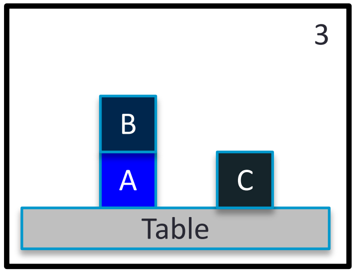
Solution:
- Take(B)
- Put(B,Table)
- Take(A)
- Put(A,B)
- Take(C)
- Put(C,A)

Automated Planning

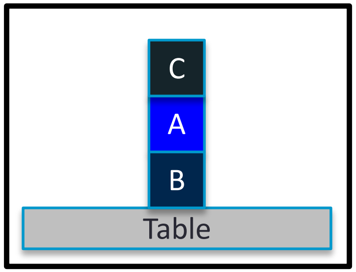
- Many approaches to solve the problem exist:
  - Simplest is known as "Forward Search", and it means using A*
- Forward Search:
  - Each possible configuration of the world is a state in the A* search
  - Heuristic measures how many of the conditions in the target state are not satisfied, for example:

Current State:		Target State:		
on(A,Table)	→	on(B,Table)	→	h(s) = 3
on(B,A)		on(A,B)		
on(C,Table)		on(C,A)		

Planning with A*:



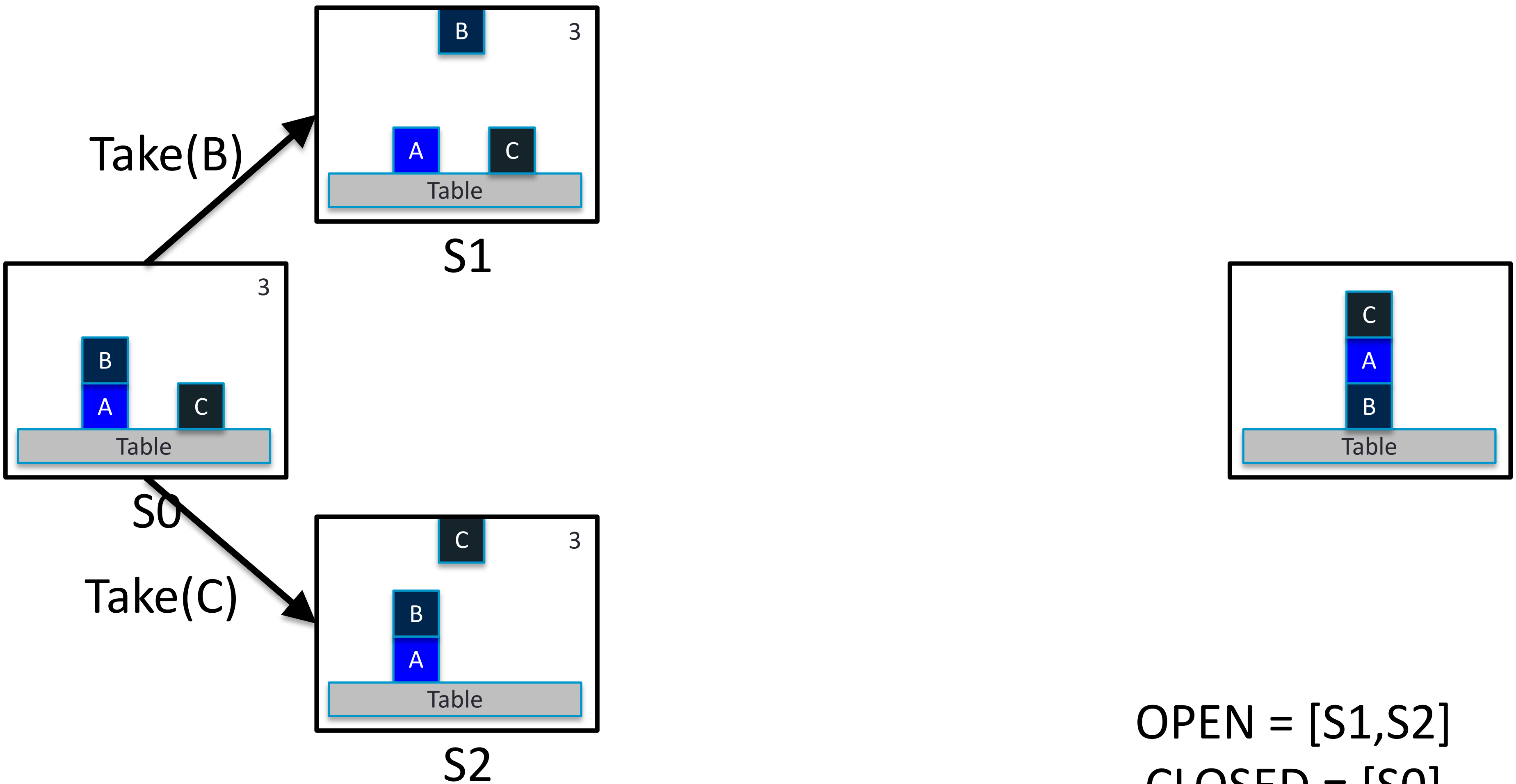
S0



OPEN = [S0]

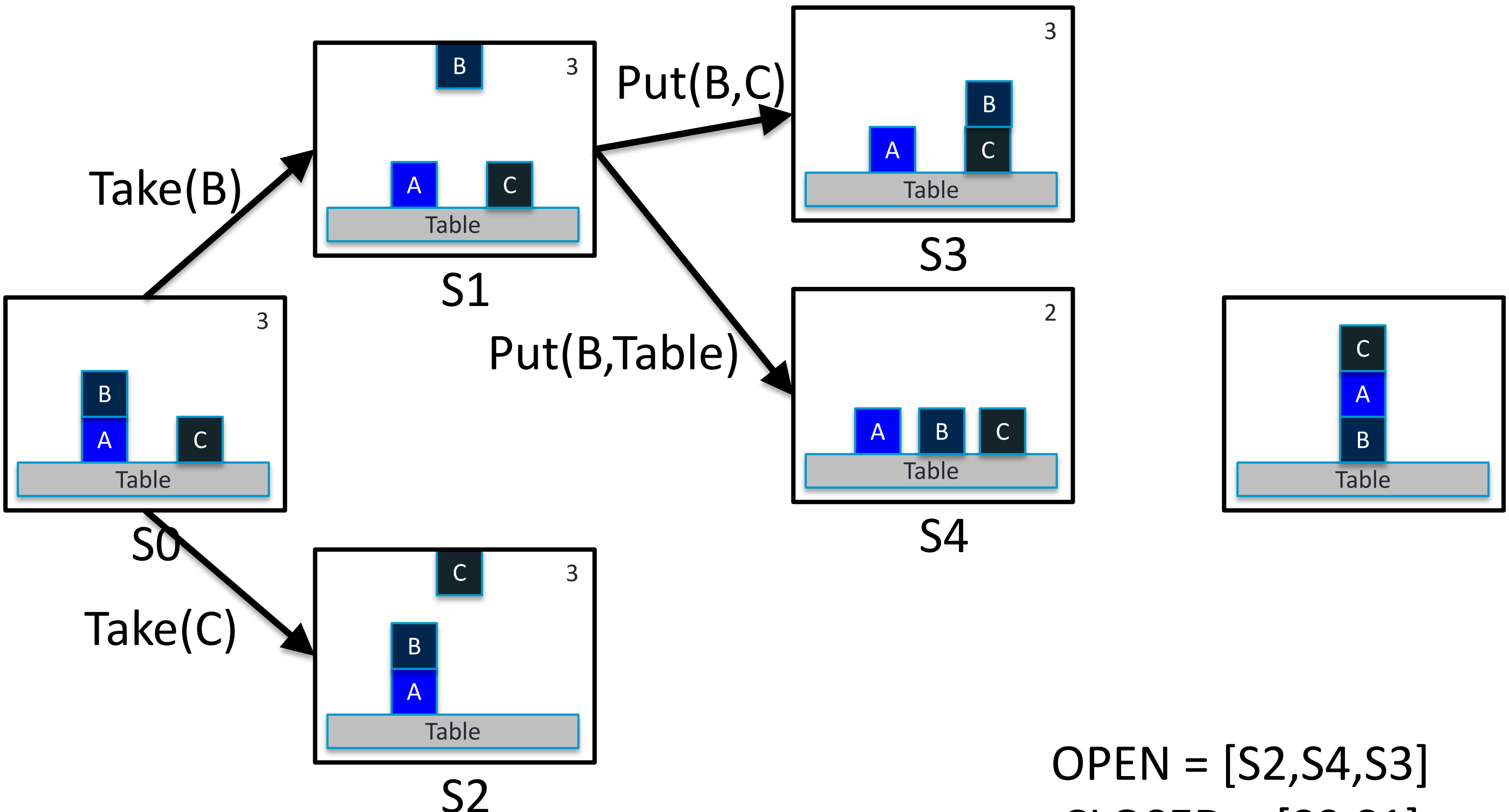
CLOSED = []

Planning with A*:



OPEN = [S1,S2]
CLOSED = [S0]

Planning with A*:



OPEN = [S2,S4,S3]
CLOSED = [S0,S1]

Automated Planning

- The previous slide is an example of “classic planning”
- There are other variants of the planning problem:
  - Temporal planning (actions take time)
  - Probabilistic planning (actions have probabilistic effects)
  - Nonlinear planning (plans might have parallel actions)
- Many other algorithms:
  - Means-ends-analysis
  - Graph-plan
  - FF
  - HTN
- Many heuristics:
  - Relaxation

Tale-spin

- "The program, simply described, simulates a small world of characters who are motivated to act by having problems to solve. When an event occurs, it is expressed in English, thus forming the text of the story. Central to the simulation, therefore, are the techniques for solving problems."
(Meehan, 1976)

Tale-spin

- A story is generated in the following way…
- Given:
  - An initial state (entered by the user):
    - Characters
    - Setting (relationships between the characters, locations, etc.)
    - One character with a goal (hungry, thirsty, etc.)
  - A set of possible actions to perform (defined in the system)
- Find a plan that makes the problem disappear:
  - Both the initial state and each event (action or goal) will be translated to natural language, thus forming the story

Tale-spin: Example

- Once upon a time George ant lived near a patch of ground. There was a nest in an ash tree. Wilma bird lived in the nest. There was some water in a river. Wilma knew that the water was in the river. George knew that the water was in the river. One day Wilma was very thirsty. Wilma wanted to get near some water. Wilma flew from her nest across a meadow through a valley to the river. Wilma drank the water. Wilma was not thirsty any more.

- George was very thirsty. George wanted to get near some water. George walked from his patch of ground across the meadow through the valley to a river bank. George fell into the water. George wanted to get near the valley. George couldn't get near the valley. George wanted to get near the meadow. George couldn't get near the meadow. Wilma wanted George to get near the meadow. Wilma wanted to get near George. Wilma grabbed George with her claw. Wilma took George from the river through the valley to the meadow. George was devoted to Wilma. George owed everything to Wilma. Wilma let go of George. George fell to the meadow. The end.

Tale-spin: Example

Initial state

- Once upon a time George ant lived near a patch of ground. There was a nest in an ash tree. Wilma bird lived in the nest. There was some water in a river. Wilma knew that the water was in the river. George knew that the water was in the river. One day Wilma was very thirsty. Wilma wanted to get near some water. Wilma flew from her nest across a meadow through a valley to the river. Wilma drank the water. Wilma was not thirsty any more.

goal

Story 1

- George was very thirsty. George wanted to get near some water. George walked from his patch of ground across the meadow through the valley to a river bank. George fell into the water. George wanted to get near the valley. George couldn't get near the valley. George wanted to get near the meadow. George couldn't get near the meadow. Wilma wanted George to get near the meadow. Wilma wanted to get near George. Wilma grabbed George with her claw. Wilma took George from the river through the valley to the meadow. George was devoted to Wilma. George owed everything to Wilma. Wilma let go of George. George fell to the meadow. The end.

goal

Story 2

Tale-spin

- Tale-spin uses a form of means-ends analysis planning
- Means-ends analysis starts with the conditions in the target state not yet satisfied in the current state, and tries to find an action to satisfy them.
- For example: “John bear is not hungry” can be satisfied with the action “John bear ate X”
- But for John bear to eat something, he must have it
- Difference with A* planning (explained before) is:
  - A* searches forwards (from initial state to target state)
  - Means-ends analysis searches backwards (from target state to initial state)

Tale-spin: Example

- Initial state (user defined):
  - Once upon a time Sam bear lived in a cave. Sam knew that Sam was in his cave. There was a beehive in an apple tree. Betty bee knew that the beehive was in the apple tree. Betty was in her beehive. Betty knew that Betty was in her beehive. There was some honey in Betty's beehive. Betty knew that the honey was in Betty's beehive. Betty had the honey. Betty knew that Betty had the honey. Sam knew that Betty was in her beehive. Sam knew that Betty had the honey. There was a rose flower in a flowerbed. Sam knew that the rose flower was in the flowerbed.
- Problem:
  - Sam bear is hungry.

Tale-spin: Example

Once upon a time Sam bear lived in a cave. Sam knew that Sam was in his cave. There was a beehive in an apple tree. Betty bee knew that the beehive was in the apple tree. Betty was in her beehive. Betty knew that Betty was in her beehive. There was some honey in Betty's beehive. Betty knew that the honey was in Betty's beehive. Betty had the honey. Betty knew that Betty had the honey. Sam knew that Betty was in her beehive. Sam knew that Betty had the honey. There was a rose flower in a flowerbed. Sam knew that the rose flower was in the flowerbed.

Goal: Sam bear not hungry

Tale-spin: Example

Once upon a time Sam bear lived in a cave. Sam knew that Sam was in his cave. There was a beehive in an apple tree. Betty bee knew that the beehive was in the apple tree. Betty was in her beehive. Betty knew that Betty was in her beehive. There was some honey in Betty's beehive. Betty knew that the honey was in Betty's beehive. Betty had the honey. Betty knew that Betty had the honey. Sam knew that Betty was in her beehive. Sam knew that Betty had the honey. There was a rose flower in a flowerbed. Sam knew that the rose flower was in the flowerbed.

Tale-spin knows (for each type of goal) the set of actions that can satisfy it. In this case:

- "go towards something edible"
- "have something edible"

One is picked at random

Goal: Sam bear not hungry

Tale-spin: Example

Once upon a time Sam bear lived in a cave. Sam knew that Sam was in his cave. There was a beehive in an apple tree. Betty bee knew that the beehive was in the apple tree. Betty was in her beehive. Betty knew that Betty was in her beehive. There was some honey in Betty's beehive. Betty knew that the honey was in Betty's beehive. Betty had the honey. Betty knew that Betty had the honey. Sam knew that Betty was in her beehive. Sam knew that Betty had the honey. There was a rose flower in a flowerbed. Sam knew that the rose flower was in the flowerbed.

Goal: have <u>something edible</u>

↓

Goal: Sam bear not hungry

Tale-spin: Example

Once upon a time Sam bear lived in a cave. Sam knew that Sam was in his cave. There was a beehive in an apple tree. Betty bee knew that the beehive was in the apple tree. Betty was in her beehive. Betty knew that Betty was in her beehive. There was some honey in Betty's beehive. Betty knew that the honey was in Betty's beehive. Betty had the honey. Betty knew that Betty had the honey. Sam knew that Betty was in her beehive. Sa
flower in a flowerbed. Sam

Tale-spin knows (for each type of character) the set of thins they eat. In this case, a bear:

- Honey
- Salmon
- etc.

One is picked at random

Goal: have <u>something edible</u>

↓

Goal: Sam bear not hungry

Tale-spin: Example

Once upon a time Sam bear lived in a cave. Sam knew that Sam was in his cave. There was a beehive in an apple tree. Betty bee knew that the beehive was in the apple tree. Betty was in her beehive. Betty knew that Betty was in her beehive. There was some honey in Betty's beehive. Betty knew that the honey was in Betty's beehive. Betty had the honey. Betty knew that Betty had the honey. Sam knew that Betty was in her beehive. Sam I
flower in a flowerbed. Sam kr

Every time a goal is posted, Tale-spin generates text, to motivate the actions of the characters. In this case:

"Sam bear wanted to get some honey"

Goal: have honey

↓

Goal: Sam bear not hungry

Tale-spin: Example

Once upon a time Sam bear lived in a cave. Sam knew that Sam was in his cave. There was a beehive in an apple tree. Betty bee knew that the beehive was in the apple tree. Betty was in her beehive. Betty knew that Betty was in her beehive. There was some honey in Betty's beehive. Betty knew that the **honey** was in Betty's beehive. Betty had the honey. Betty knew that Betty had the honey. Sam knew that Betty was in her beehive. Sam knew that Betty had the honey. There was a rose flower in a flowerbed. Sam knew that the rose flower was in the flowerbed.

Goal: have honey

Goal: Sam bear not hungry

Tale-spin: Example

Once upon a time Sam bear
There was a beehive in an ap
apple tree. Betty was in her
There was some honey in Bett
beehive. Betty had the honey.
Betty was in her beehive. Sa
flower in a flowerbed. Sam

Tale-spin searches among the possible actions that can satisfy "have honey":

- Take(honey)
- Persuade-to-give(Betty,honey)
- Persuade-to-abandon(Betty,honey)
- etc.

Take(honey) cannot be used, since its precondition is that "honey" is not owned by someone else. So, Tale-spin selects one of the others at random.

Goal: have honey

↓

Goal: Sam bear not hungry

Tale-spin: Example

Once upon a time Sam bear lived in a cave. Sam knew that Sam was in his cave. There was a beehive in an apple tree. **Betty bee** knew that the beehive was in the apple tree. Betty was in her beehive. Betty knew that Betty was in her beehive. There was some honey in Betty's beehive. Betty knew that the **honey** was in Betty's beehive. Betty had the honey. Betty knew that Betty had the honey. Sam knew that Betty was in her beehive. Sam knew that Betty had the honey. There was a rose flower in a flowerbed. Sam knew that the rose flower was in the flowerbed.

Goal: persuade Betty to abandon honey

↓

Goal: have honey

↓

Goal: Sam bear not hungry

Tale-spin

- Goals: hungry, have something, persuade someone, go somewhere, etc.
- For each goal: a collection of strategies + preconditions
- Domain knowledge:
  - Which actions can certain characters execute
  - Which food do different animals like
  - Inter-character relations and how to they affect the actions the can execute
  - etc.
- Using those 3 things: stories are generated at random, from the set of possible ways to solve goals.

Tale-spin

- Planning-based story generation
- Generated stories are about solving problems
- Each individual character plans on its own (no joint behaviors between characters)
  - Tale-spin is "character-centric"
  - Stories are always "coherent" (actions of characters are motivated)

Story Generation in Games

- Plot generation:
  - Story generation can be used to automatically generate plot
  - Given an initial state (generated, for example, using PCG), and a goal (selected, for example, at random from a set of goals)…
  - Generate a story from initial state to goal: that is the path that the player will have to follow.
- Quest generation (analogous):
  - Given a goal for the player (generated, for example, at random)…
  - Generate a story in which the player achieves the goal from the current state.
  - The actions in that story are the things the player needs to do to accomplish the quest.

Game AI Redux

- Game AI has become very sophisticated
  - Now, many components and many layers of development within each component
- Can these state-of-the-art players *think*?
  - Well, though they're much more complex, they "think" in much the same ways as old-school "AI" like Dr. NIM!

Game AI Redux

DR. NIM AND COMPUTERS

By now you have played against DR. NIM enough to respect and appreciate his ability. Does he really think? You certainly had to do a lot of thinking to beat him. Did he have to? You will probably say that DR. NIM does not "think" despite the fact that he plays a clever game of NIM. If this is your answer, you would also be convinced by more study that a large electronic computer does not "think" either. The large computer is more like DR. NIM in its capability than like a human. By the way, you *"PROGRAMMED"* DR. NIM each time that you positioned or set his elements at the beginning of each game.

So, let us leave this subject of "Can Machines Think" for the moment, and consider DR. NIM first from the computer machine point of view and then the computer programming point of view. Then we will come back to this question of thinking machines.

Next time…

- We will explore a different perspective on thinking and intelligence in Cognitive Systems…