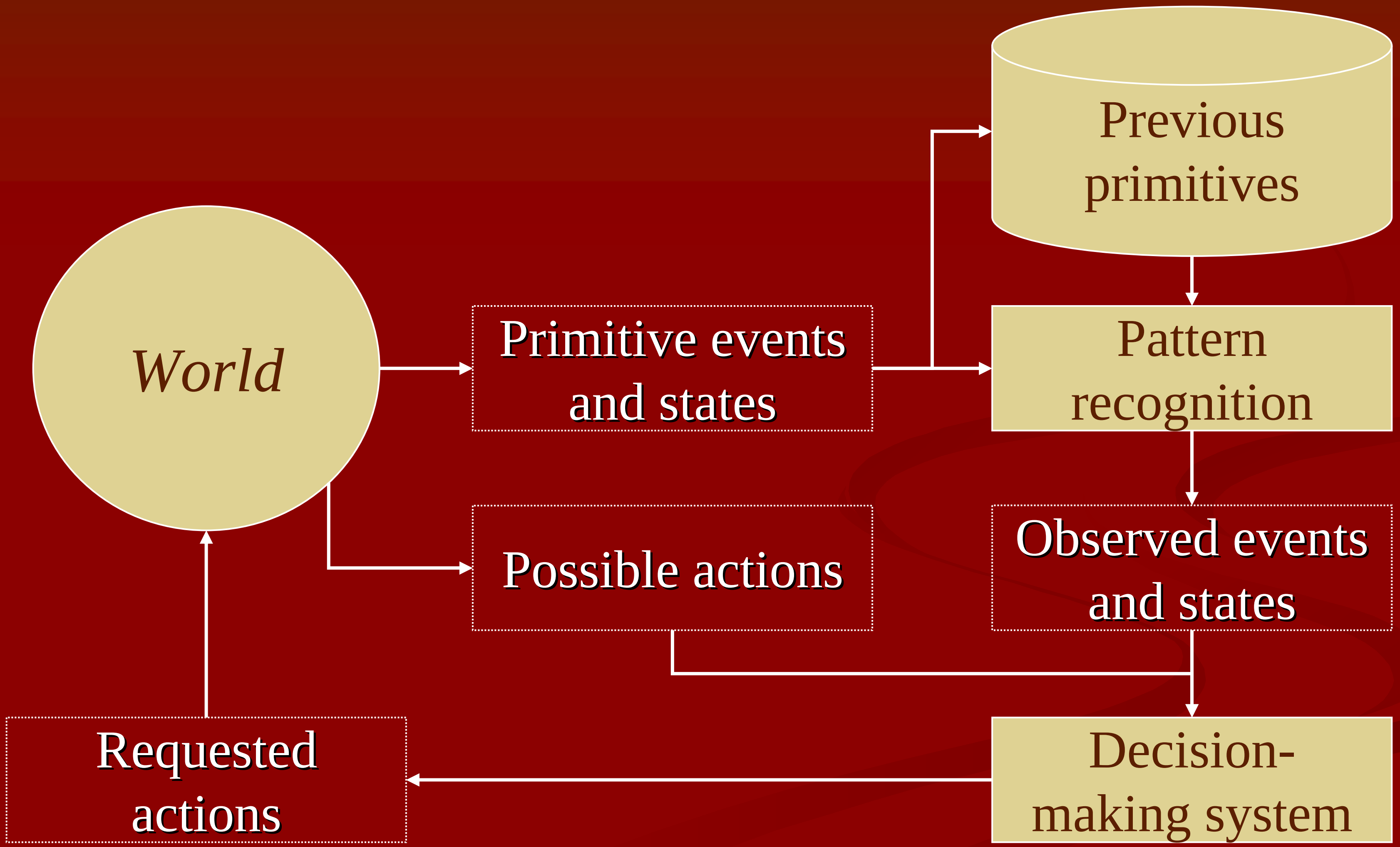
Algorithms and Networking for Computer Games

Chapter 6: Decision-making

AI system



Previous primitives
World
Primitive events and states
Pattern recognition
Possible actions
Observed events and states
Requested actions
Decision-making system

Three perspectives for decision-making in computer games

- level of decision-making
  - strategic, tactical, operational
- use of the modelled knowledge
  - prediction, production
- methods
  - optimization, adaptation

Level of decision-making

- strategic
  - what should be done
- tactical
  - how to actuate it
- operational
  - how to carry it out

Strategic level

- long-term decisions
  - infrequent → can be computed offline or in the background
- large amount of data, which is filtered to bring forth the essentials
  - quantization problem?
- speculative (what-if scenarios)
- the cost of a wrong decision is high

Tactical level

- medium-term decisions
- intermediary between strategic and operational levels
  - follow the plan made on the strategic level
  - convey the feedback from the operational level
- considers a group of entities
  - a selected set of data to be scrutinized
  - co-operation within the group

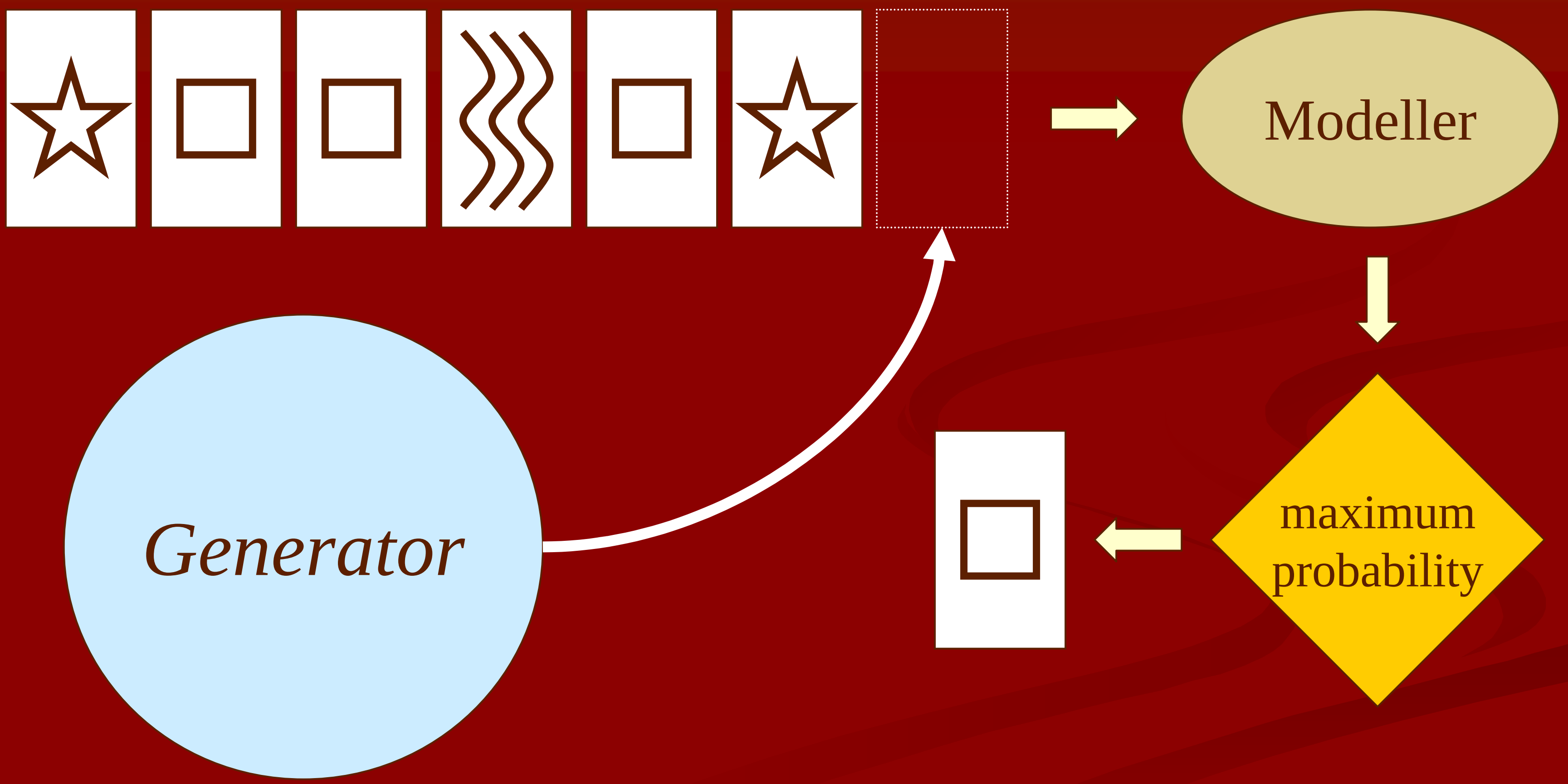
Operational level

- short-term decisions
  - reactive, real-time response
- concrete and closely connected to the game world
- considers individual entities
- the cost of a wrong decision is relatively low
  - of course not to the entity itself

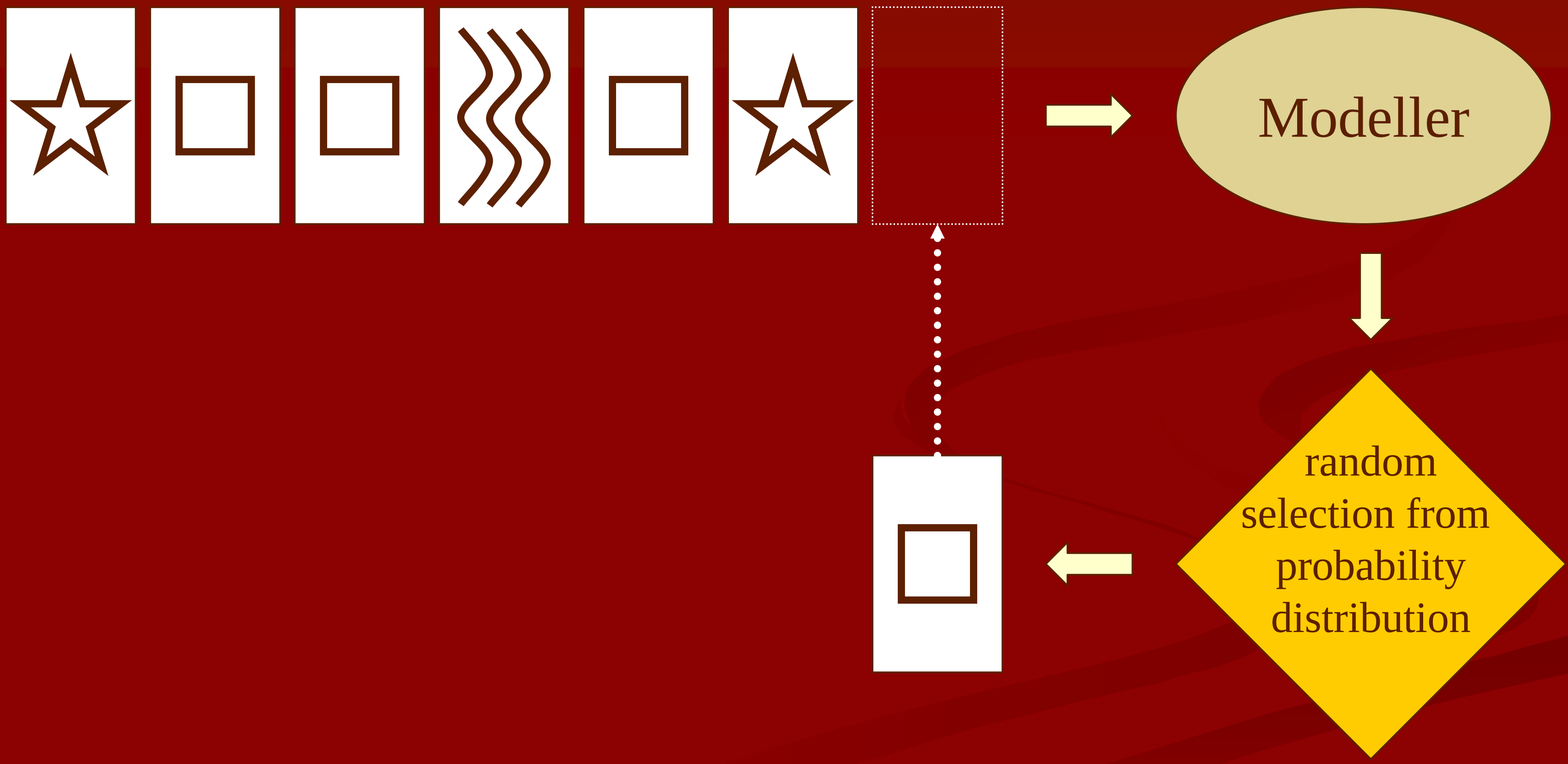
Use of the modelled knowledge

- time series data
- world = a generator of events and states, which can be labelled with symbols
- prediction
  - what the generator will produce next?
- production
  - simulating the output of the generator
- how to cope with uncertainty?

Prediction



Production



Modeller
random selection from probability distribution

Methods: Optimization

- elements:
  - objective function to be maximized/minimized
  - variables affecting the value of the objective function
  - constraints limiting feasible variable values
- goal: find among the feasible solutions the one that gives an optimum value for the objective function
- time consuming?
  → heuristic rules to guide the search

Methods: Optimization (cont'd)

- hill-climbing
  - how to escape local optima?
- tabu search
- simulated annealing
- genetic algorithms
  - multiple search traces
- swarm algorithms

Methods: Adaptation

- ability to make appropriate responses to changed circumstances → learning
- searches for a function behind given solutions
  - optimization: solution for a given function
  - affecting factors are unknown or dynamic
- pattern recognition

Methods: Adaptation (cont'd)

- neural networks
  - training
    - supervised learning
    - unsupervised learning (e.g., self-organizing maps)
  - execution
- hidden Markov model
  - recurring structures

Soft computing

- L. Zadeh: methodologies that try to solve problems arising from the complexity of the natural world
  - approximation
  - partial truth
  - imprecision
  - uncertainty
- computer games have used 'hard' computing
- as the game worlds get more complex, perhaps soft computing methods would suit better

Soft computing methods

- probabilistic reasoning
  - genetic algorithms
  - Bayesian networks
- artificial neural networks
  - back-propagation networks
  - self-organizing maps
- fuzzy logic
  - fuzzy sets
  - approximate reasoning

Finite state machine (FSM)

- components:
  - states
  - transitions
  - events
  - actions
- state chart: fully connected directed graph
  - vertices = states
  - edges = transitions

Properties of FSM

1. acceptor
   - does the input sequence fulfil given criteria?
2. transducer
   - what is the corresponding output sequence for a given input sequence?
3. computator
   - what is the sequence of actions for a given input sequence?

- these properties are independent!

Mealy and Moore machines

- theoretical cathegories for FSMs
- Mealy machine
  - actions are in transitions
  - the next action is determined by the current state and the occurring event
  - more compact but harder to comprehend
- Moore machine
  - actions are in states
  - the next action is determined by the next state
- helps to understand and use state machines in UML

Implementation

- design by contract
  - two parties: the supplier and the client
  - formal agreement using interfaces
- FSM software components
  - environment: view to the FSM (client)
  - context: handles the dynamic aspects of the FSM (supplier)
  - structure: maintains the representation of the FSM (supplier)

Noteworthy

- structure is static
  - hard to modify
- reactivity
  - memoryless representation of all possible walks from the initial state
- states are mutually exclusive: one state at a time
  - not for continuous or multivalued values
- combinatorial explosion
  - if the states and events are independent
- risk of total rewriting
  - high cohesion of actions

Flocking

- C. W. Reynolds: "Flocks, herds, and schools: A distributed behavioral model" (1987)
- a flock seems to react as autonomous entity although it is a collection of individual beings
- flocking algorithm emulates this phenomenon
- results resemble various natural group movements
- boid = an autonomous agent in a flock

Rules of flocking

1. **Separation**: Do not crowd flockmates.
2. **Alignment**: Move in the same direction as flockmates.
3. **Cohesion**: Stay close to flockmates.
4. **Avoidance**: Avoid obstacles and enemies.

→ boid’s behavioural urges

Observations

- stateless algorithm
  - no information needs to be maintained
  - boid re-evaluates the environment on each update cycle
- no centralized control
  - emergent behaviour

Other uses for flocking

- swarm algorithms
  - solution candidate = boid
  - solution space = flying space
  - separation prevents crowding the local optima
- obstacle avoidance in path finding
  - steer away from obstacles along the path

Influence maps

- discrete representation of the synthetic player's knowledge of the world
- strategic and tactical information
  - frontiers, control points, weaknesses
- influence
  - type
  - repulsiveness/alluringness
- recall path finding and terrain generation

Assumptions

- a regular grid over the game world
- each tile holds numeric information of the corresponding area
  - positive values: alluringness
  - negative values: repulsiveness

Construction

1. initialization
   - assign values to the tiles where the influence exists
2. propagation
   - spread the effect to the neighbouring tiles
   - linear or exponential fall-off
   - cut-off point

Aggregation

- influence map can be combined
  - the same (or compatible) granularity
- example
  - map 1 = my troops
  - map 2 = enemy’s troops
  - map 3 = map 1 + map2 = battlefield
- aggregation
  - operator: sum, product
  - weights: to balance the effects

Evaluation

- static features: compute beforehand
- periodical updates
  - categorize the maps based on the rate of change
  - lazy evaluation

Key questions for synthetic players

- how to achieve real-time response?
- how to distribute the synthetic players in a network?
- how autonomous the synthetic players should be?
- how to communicate with other synthetic players?