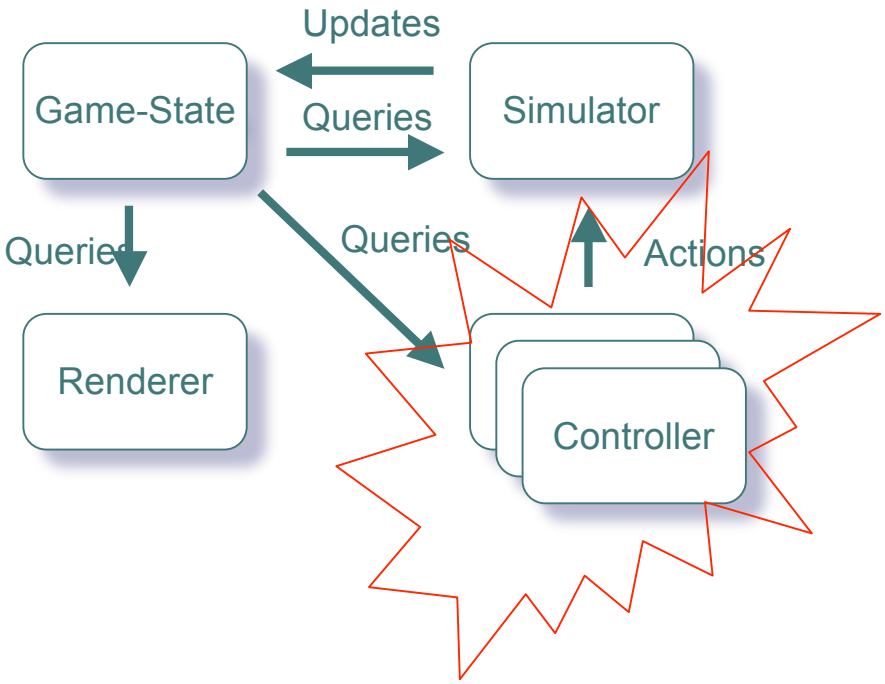
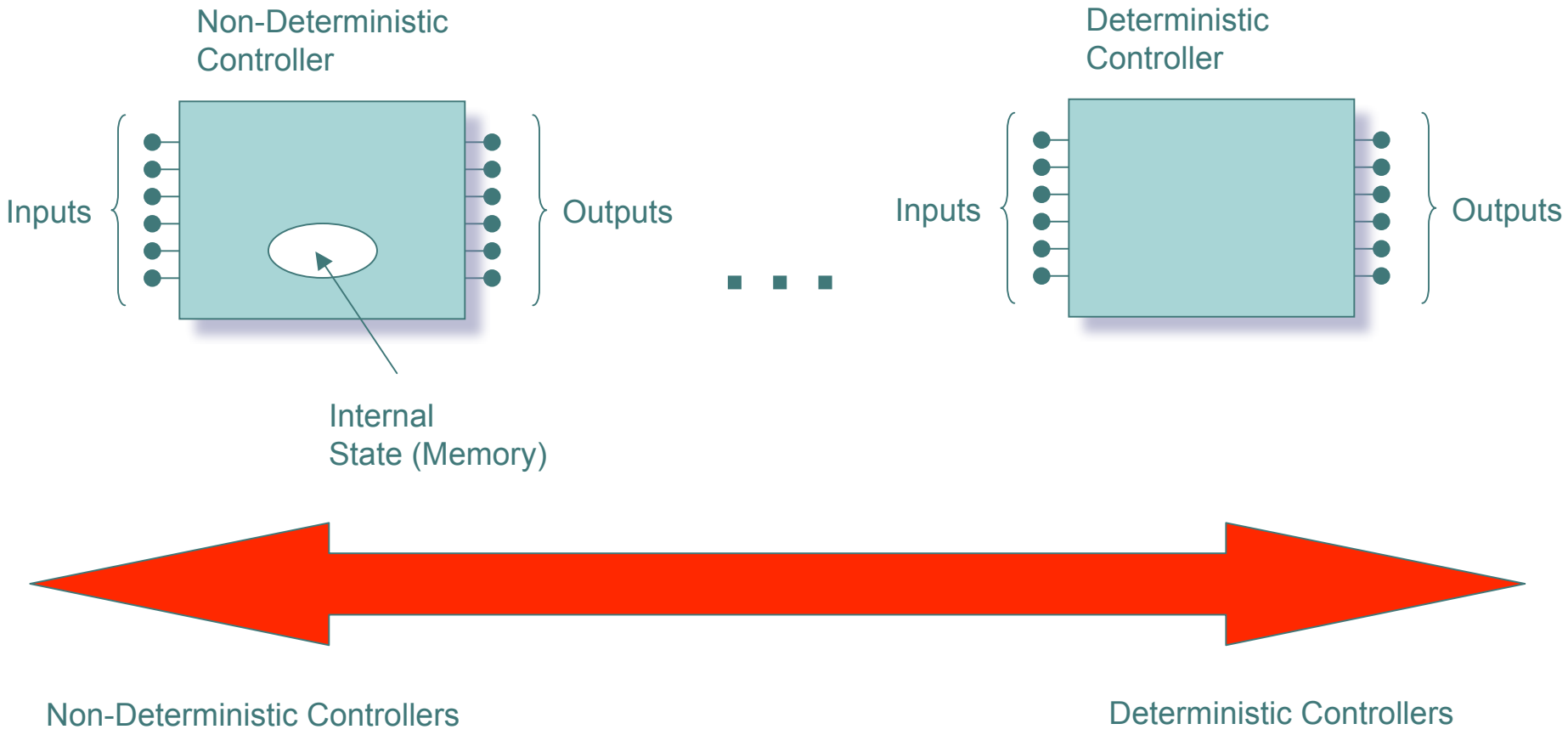
Game Engine Architecture



Source: AI for Computer Games, John D. Funge, AK Peters, 2004.

- Game-State – the game-state represents the current state of the world. It knows about all the objects in the world and provides access to them so that they can be queried by all the other components for information about their current state.
- Simulator – the simulator encodes the rules of how the game-state changes based on the "game physics." Together with a set of animations the simulator is responsible for generating a character's moves in response to the actions chosen by the associated controller.
- Renderer – together with the game's geometry and texture maps, the renderer is responsible for rendering a depiction of the game-state; usually with images and sound.
- Controller – each character in the game has at least one controller ("brain") associated with it. The controller is responsible for selecting actions. For player characters the controller interprets joystick interactions. For NPCs the controller consists of the AI and low-level control.

Controller (Brain) Architectures



Pros & Cons

- Non-Deterministic Controllers:
  - Can easily accommodate very complex behaviors/actions.
  - Can be computationally very complex.
  - Different reaction (outputs) to the same situations (inputs) – non-deterministic behavior.

  ☞ difficult to debug.

- Deterministic Controllers:
  - Can only implement reflexive behavior, behavior that only depends on the current set of inputs (no memory/history/internal state).
  - Computationally usually very simple (lookup table).
  - Same reaction (outputs) to the same situations (inputs) – deterministic behavior

  ☞ easy to debug.

Planning vs. Reactive Behavior

- Planning usually involves thinking ahead.
  - Often this manifests itself as searching; as in searching for the best path or searching for the best chess move, etc.
  - In order to accomplish this the controller needs an internal state:
    - Goal-Subgoal lists, keeping track when goals are fullfilled or need to be adjusted.
- Reactive Behavior is well...reactive
  - Simply maps the inputs into the outputs.
  - Is often rule or table based.

Planning vs. Reactive Behavior

- Planning can be very complex
- It turns out that full planning needs to be done in very few cases, reactive behavior can be substituted in many cases without loss of intelligent behavior
- Reactive behavior maps particularly well onto animats:
  - Collect information from the senses
  - Interpret this information
  - Perform an action

What have we learned so far?

- Intelligence is the *computational* part of attaining goals.
- AI is the science and engineering of constructing intelligent machines.
- Agents are different from programs (autonomous, long lived, communicate, *etc*).
- Animats are agents with a “body” implying limited capabilities and senses.

What have we learned so far?

- From an AI perspective, the key part in a game engine is the controller
  - Non-deterministic (stateful, planning)
  - Deterministic (stateless, reactive behavior)
- From a software development perspective we tend to prefer deterministic controllers
  - Easy to implement
  - Easy to debug
  - Mimics reflexive behavior found in real creatures in nature

Assignments

- Read: 3,5-8 (Alex' Book)
- Programming Assignment 1 due Wednesday 2/3 @ 10pm