Practical AI modules for Non-Player Characters in Video Games

Stavros Vassos
(Ph.D. student of Hector Levesque)

Department of Computer Science
University of Toronto

May 13th, 2008

Overview

- Cognitive robotics
- Applying cognitive robotics to video games:
  - the AI researcher perspective;
  - the game developer perspective.
- The need for practical AI modules for video games
- The temporal projection module
- Future work / discussion

Cognitive robotics

Cognitive Robotics is concerned with building robotic (or software) agents with higher-level cognitive abilities that involve reasoning.

Cognitive robotics

Cognitive Robotics is concerned with building robotic (or software) agents with higher-level cognitive abilities that involve reasoning.

Cognitive abilities such as being able to:

- learn from past experience,
- think ahead and plan before acting,
- communicate and coordinate with other agents, ...

Cognitive robotics

Cognitive Robotics is concerned with building robotic (or software) agents with higher-level cognitive abilities that involve reasoning.

Cognitive abilities such as being able to:

- learn from past experience,
- think ahead and plan before acting,
- communicate and coordinate with other agents, ...

These are based on reasoning or “taking into account”:

- the current state of the world and sensory data,
- the goals that need to be achieved,
- the actions that can be performed in the world,
- the mental states of other agents, ...

Cognitive robotics and video games

Modern video games are ideal for applying cognitive robotics research:

- detailed software worlds with advanced 3D game engines,
- the world consists of objects that follow realistic physics,
- many autonomous Non-Player-Characters (NPCs) interact with the objects in the world, the player, and with each other,
- rich sensor data for each NPC.

Some of the NPCs in a video game could be implemented as cognitive robots! Let's see how that would go..

Cognitive NPC: the AI researcher perspective

Pick a cognitive robotics system, implement the NPC..

..and connect it to the game world (3D game engine).

Cognitive NPC: the AI researcher perspective

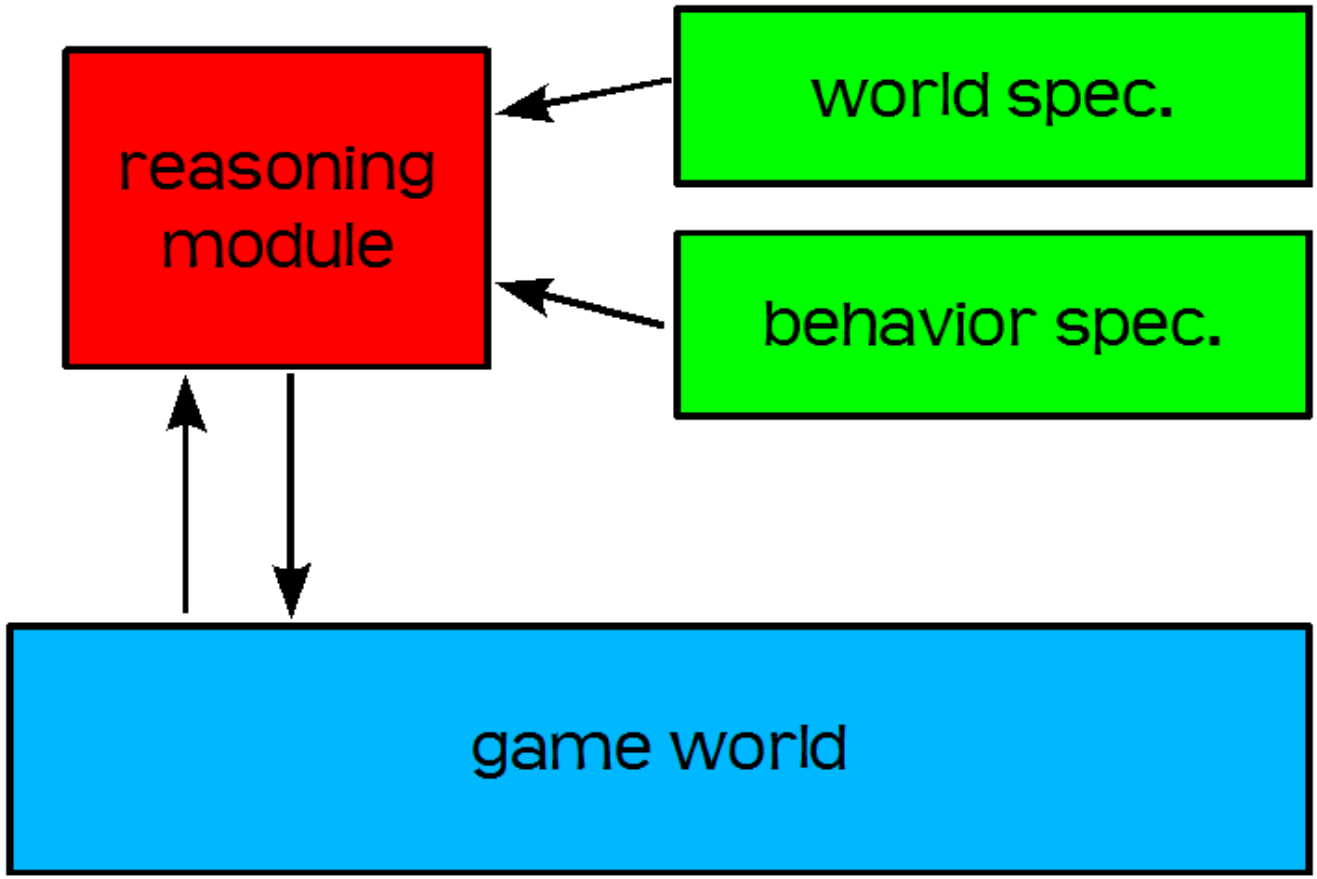
Pick a cognitive robotics system, implement the NPC..

1. Specify the world that the NPC is situated:
   - the *changing properties* of the world and their current value;
     `door1=locked, availWeapons=[w1,w2]`
   - the *actions* that can take place in the world;
     `kick(x), fireWeaponAt(x), move(d,s)`
   - the *effects* of the actions in the world;
     `kick(door)` $\rightarrow$ `open(door) provided ...`
2. Specify the intended behavior of the robot using features such as *goals*, *intentions*, *preferences*, *utility functions*, as well as predefined *reactive rules*, *plans*, etc;

The reasoning core of the system mixes ingredients 1 and 2 appropriately to produce intelligent behavior.

..and connect it to the game world (3D game engine).

Cognitive NPC: the AI researcher perspective



Cognitive NPC: the game developer perspective

Practical difficulties:

- It requires a lot of memory and computing time.
- The NPCs would not be responsive in real-time.

Cognitive NPC: the game developer perspective

Practical difficulties:

- It requires a lot of memory and computing time.
- The NPCs would not be responsive in real-time.

Moreover, it would be awkward to implement cognitive NPCs like this because:

- Typically the AI system assumes full control over the agent:

  All aspects of the NPC's behavior need to be encoded in the AI system using the language for specifying parts 1 and 2.
  For instance, how about path-finding?
- In order to use the AI system for only some reasoning task you need to know the inner workings of the system..

Let's see some more details on the implementation of an NPC in a 3D game engine..

Cognitive NPC: the game developer perspective

- 3D game engine in C++, NPC is an instance of a C++ class.
- In every frame a “thinking” function is called that specifies what the NPC should do.

Cognitive NPC: the game developer perspective

- 3D game engine in C++, NPC is an instance of a C++ class.
- In every frame a “thinking” function is called that specifies what the NPC should do.

  The body of this function usually implements a finite-state-automaton (FSA) as a big if-then-else block. The conditionals depend on:
  - the state of the NPC, i.e. variables of the form `state=ATTACKING`, `energy=80`;
  - sensory data, i.e. C++ functions of the form `bool enemyInSight();`
  - other implemented helping functions such as `bool pathfind(from,to,options).`

Cognitive NPC: the game developer perspective

- 3D game engine in C++, NPC is an instance of a C++ class.
- In every frame a “thinking” function is called that specifies what the NPC should do.

Game developer:

```
int NPC::think(){
   if (state==IDLE &&
       energy>70 &&
       enemyInSight())
     state=ATTACKING;
   else if ...
   ...
}
```

Cognitive NPC: the game developer perspective

- 3D game engine in C++, NPC is an instance of a C++ class.
- In every frame a “thinking” function is called that specifies what the NPC should do.

Game developer:

```
int NPC::think(){
  if (state==IDLE &&
      energy>70 &&
      enemyInSight())
    state=ATTACKING;
  else if ...
  ...
}
```

AI researcher:

```
int NPC::think(){
  get sensory data;
  push data to the AI
  system;
  wait for the system to
  compute the next move;
  perform the next move;
}
```

Two extremes: either all in C++ or all in the AI system..

Practical AI modules for NPCs

- Two extremes: either all in C++ or all in the AI system.
- Middle ground:

  Identify practical AI modules that can replace C++ code that the game developer needs to implement for the behavior of the NPC.
- Historically that's the most common way that AI techniques have been used in commercial video games.
- Most widely used AI technique:

  A* heuristic search, used for path-finding:
  - solves a particular sub-problem with clear input and output;
  - can be embedded easily to any control structure for NPCs.
- That's the kind of AI modules we should be aiming for!

Practical AI modules for NPCs: Temporal Projection

A temporal projection module: keeps track of the current state of the world and predicts how it changes when actions take place.

You can think of it as a special database or data structure that:

A stores:
- the *changing properties* of the world and their current value;
- the *actions* that can take place in the world;
- the *effects* of the actions in the world;

B features a language for expressing queries and getting answers about
- the *current state* of the world;
- the *future states* of the world after actions have been performed.

Practical AI modules for NPCs: Temporal Projection

A temporal projection module: keeps track of the current state of the world and predicts how it changes when actions take place.

You can think of it as a special database or data structure that:

A stores:
- the *changing properties* of the world and their current value;
- the *actions* that can take place in the world;
- the *effects* of the actions in the world;

B features a language for expressing queries and getting answers about
- the *current state* of the world;
- the *future states* of the world after actions have been performed.

Big question: What interface should we use? SQL? something similar to C++ code? a subset of natural language?

Practical AI modules for NPCs: Temporal Projection

A temporal projection module: keeps track of the current state of the world and predicts how it changes when actions take place.

You can think of it as a special database or data structure that:

A stores:
- the *changing properties* of the world and their current value;
- the *actions* that can take place in the world;
- the *effects* of the actions in the world;

B features a language for expressing queries and getting answers about
- the *current state* of the world;
- the *future states* of the world after actions have been performed.

Big question: What interface should we use? SQL? something similar to C++ code? a subset of natural language?

Bigger question: Would this module be useful?

Video games and AI: Discussion

One last point about the use of AI techniques in video games.

- A* dates back to 1968.
- The only reasoning module ever used in a commercial game (F.E.A.R., 2005, Monolith Productions, Vivendi) is a simplified version of a STRIPS planner that dates back to 1971.
- AI techniques need feedback from the video game industry so that they evolve as needed for the video games.
- What kind of modules should AI provide? What are the things that are difficult to deal with right now?

http://www.cs.toronto.edu/cogrobo
stavros@cs.toronto.edu