Middleware for Computer games and external bot AI

Pogamut

Jakub Gemrot
AMIS research group (MPF-CU)
Outline

- Motivation
- Problem detailed
- “Proof-of-concept” live show case
- Architecture detailed
- Recent / Related / Future work
- Discussion
Interest in using rich and complex virtual worlds
- Military simulations and training
- Therapeutic applications
- Serious “games”
- Virtual storytelling
- Computer games (having the best virtual world)
We have a Virtual world …
  … managed (simulated) by progr. Game engine
  … inhabited by virtual beings – Bots
  … which have their own Logic (behaviour)
  … which is performed by some Reasoner
There is an interest to use various reasoners to model bot behaviors

- JACK reasoner used for military training scenarios

A lot of reasoners exists, but they are not designed to control embodied agents thus must be evaluated first

- AFAWK there is no platform that “easily allows” that
Situation
Interfacing problem

- Game engine on one side
  - With no explicit world state interface
- Reasoner that we want to use on the other side
  - Expectation how the world state is represented and accessed
- We need some sort of „extractor/translator“ of the world state
  - Neither side is explicitly working with ontologies
- Both are already (to a certain degree) of production quality
- Following slides are based on our 5 years of experiences in interfacing computer games and reasoners
What must be solved?
Game engine has world state spread in various data structures.

VS

Reasoner needs explicit world state representation. It may require specific knowledge representation (e.g., Prolog facts).
Game engine has world state spread in various data structures.

\[ VS \]

Reasoner needs explicit world state representation. It may require specific knowledge representation (e.g., Prolog facts).

\[ => \]

The world state must be extracted “manually” and translated into messages/objects of required knowledge representation.
Game engine employs mixed world-state retrieval strategies.

VS

Reasoner may rely on PUSH or PULL strategies exclusively.
Game engine employs mixed world-state retrieval strategies.

VS

Reasoner may rely on PUSH or PULL strategies exclusively.

=>

It must be compensated for.
Game engine represent implicitly only information needed by its game mechanics.

VS

Different scenarios requires different information and reasoner may require implicit representation of information of various complexity.
Game engine represent implicitly only information needed by its game mechanics.

\[ \forall S \]

Different scenarios requires different information and reasoner may require implicit representation of information of various complexity.

=>

Hooks for computation of implicit information must be provided. There must be a layer that automatically infers facts for the reasoner.
Computer game is a real-time system maintaining its own clock.

\[VS\]

Reasoner execution may span a few ticks or it may need to work asynchronously.
Computer game is a real-time system maintaining its own clock.

\[ V/S \]

Reasoner execution may span a few ticks or it may need to work asynchronously.

\[ => \]

World state synchronization / locking.
There are many game engines.

AND

There are many reasoners.
There are many game engines. 

AND

There are many reasoners. 

=>

M:N interfacing problem, waste of time to reinvent the shared part of the wheel.
There are many game engines.

\textit{AND}

There are many reasoners.

\Rightarrow

M:N interfacing problem, waste of time to reinvent the shared part of the wheel.

Shared part to be identified.
Middleware proposal

We need a “magical box” that provides solutions for mentioned problems, some sort of “adapter/translator”.

Diagram:
- Reasoner
- Middleware Adapter
- Game Engine
  - Bot_7
  - Bot_2
  - Chair_25
  - Table_23
  - GirlBot:
    - name
    - location
    - logic
“Proof-of-concept” impl.

- Pogamut-UT2004 middleware
  - Java
  - Interfaces UT2004 with POSH (Python)
  - Plugin for NetBeans
  - Picked by others (GOAL, Jadex, BotPrize)
Pogamut-UT2004
Architecture detailed

Unreal Tournament 2004

GameBots2004

TCP/IP

Middleware

Agent

POSH

Sheena

Tom

Hello Sheena!

Unreal Tournament 2004

GameBots2004

TCP/IP

PogamutCore

Commands

Events

IDE

I see Sheena, let's say hello to her!

SayPrivate Hello Sheena!

Say

(Message Hello Sheena!)

... SEE {Id DM-Flux2.Sheena} ...

Code, Controls, Observes

Tom

senses

2

3

4

5

6
Live showcase

- EmoHawk / EmoDoc scenario from Artificial Beings course exam
Pogamut-UT2004
Architecture zooming in
Pogamut-UT2004

How does this map to mentioned problems?
Our solutions (1/5)

Game engine has world state spread in various data structures.

=>

No luck for general solution here, we need a module inside the game itself that gathers required world state together and export.
Our solutions (2/5)

Game engine employs mixed information retrieval strategies.

=>

1) Observe and push all important world-state changes.

2) Provide a way for both push/pull strategies.
Game engine represent implicitly only information needed by the game mechanics.

=>

We’re providing a way how to observe changes in the world state and produce new WorldEvents.
Our solutions (4/5)

Computer game is a real-time system maintaining its own clock.

=>

Lockable “database” maintaining the world-state. Synchronized with bot logic.
There are many game engines.

AND

here are many reasoners.

=>

Generalization of the concept (next slide).
Concept generalization
Concept generalization

May we apply this concept elsewhere?
Defcon
Simulation of thermonuclear war
Defcon
Simulation of thermonuclear war

- Simulation of thermonuclear war
  - Bot lacks a body
  - Different bot actions
  - Different world representation
  - JNI instead of TCP/IP

- Concept applied, WorldView reused completely, other support middleware parts as well
Defcon architecture

Defcon

Reasoner

DefconWorldView

Defcon Message Producer

JBotSupport

JBot

Commands

Infos, Events

JNI

Defcon

General concept

Reasoner

World-state "DB"

Adapter

Translator

Commands

Executor

Exporter

Computer game
**WorldView**

**Fully reusable part**

- Minimalistic virtual world model
  - WorldEvent(s)
  - WorldObject(s)
    - Unique IDs / ObjectUpdates
- Reasoner able to lock the world view
- Utilizing class inheritance hierarchy as a simple ontology (push/pull)

Person implements ILocated, IVisible, IPlayer
Leaving untouched

- Embedding reasoner into the game engine
- Discussion of world-state objects complexity
- Reactive layer for bot’s complex actions
- Action result uncertainty
Recent work
Extending virtual world model

- Further information categorization
  - Static / Dynamic information
  - Subjective / Objective information
- Data structure that allows share dynamic-objective information by multiple bots
  - Shares traits with distributed databases (chronicles for shared „atomic“ pieces of info)
Related work

- High Level Architecture – HLA
  - Military standard for distributed simulations
  - Based on specification of information that the simulation consist of + its simulators
  - Simulation consists of list the list of federates, every federates may listen for changes and alter world-state as well
  - Hard to use, standard is still evolving
Future work

- A lot of thoughts, some of them are more academic than others…
- Extending “portfolio” of connected reasoners and game engines
- Investigating impact of having information explicitly (procedurally) vs. implicitly (precomputed) available
- Creating tournament server

Suggestions are welcomed!
Thank you for your attention!