Towards fast prototyping of IVA's behavior: Pogamut

Ondřej Burkert, Rudolf Kadlec, Jakub Gemrot, Michal Bída, Jan Havlíček, Martin Dörfler, Cyril Brom

Charles University in Prague, Faculty of Mathematics and Physics
Dept. of Software and Computer Science Education, Prague, Czech Republic
ondra@atrey.karlin.mff.cuni.cz
http://artemis.ms.mff.cuni.cz/pogamut

Main contribution

Pogamut provides an IDE and an auxiliary library for fast creation of IVA's behavior in virtual world of Unreal Tournament^{TM} 2004[1]. Pogamut uses new version of popular GameBots[2] interface and integrates a reactive planner POSH[3].

Introduction

We have created a platform for easy prototyping of virtual human **behavior** in complex virtual world of Unreal Tournament 2004.

Fig. 1 An example of a bot from Unreal Tournament^{TM}

Our goal

To fill the gap on the field of educational tools for virtual humans development.

Who is it for?

Students and researchers who are interested in modeling of behavior of virtual humans.

Pogamut features and development

Pogamut supports three main stages of IVA's creation. Each stage is supported by following features:

Implementation – core Java library of sensomotoric primitives

- Memory that stores sensory information
- Functional primitives for the control of IVA's body
- Inventory to manage items picked up by the agent
- Methods for moving around the map that solve navigation issues, including A*

Pogamut currently supports development in:

- Java, POSH[3], Python
- and possibly more languages with Java Script API binding

Debugging and tuning – Plugin for Netbeans^{TM} IDE

- List of Unreal Tournament servers
- List of running IVAs
- Introspection of IVA's variables
- Log viewers – logs allow filtering
- Bot remote control – arrows allow you to move the IVA
- Server control – change map, save replay of the game etc.

Validation – binding with JBoss Rules^{TM} [4] rule engine

- Experiments defined by declarative rules are suitable for testing IVA's behavior in different scenarios

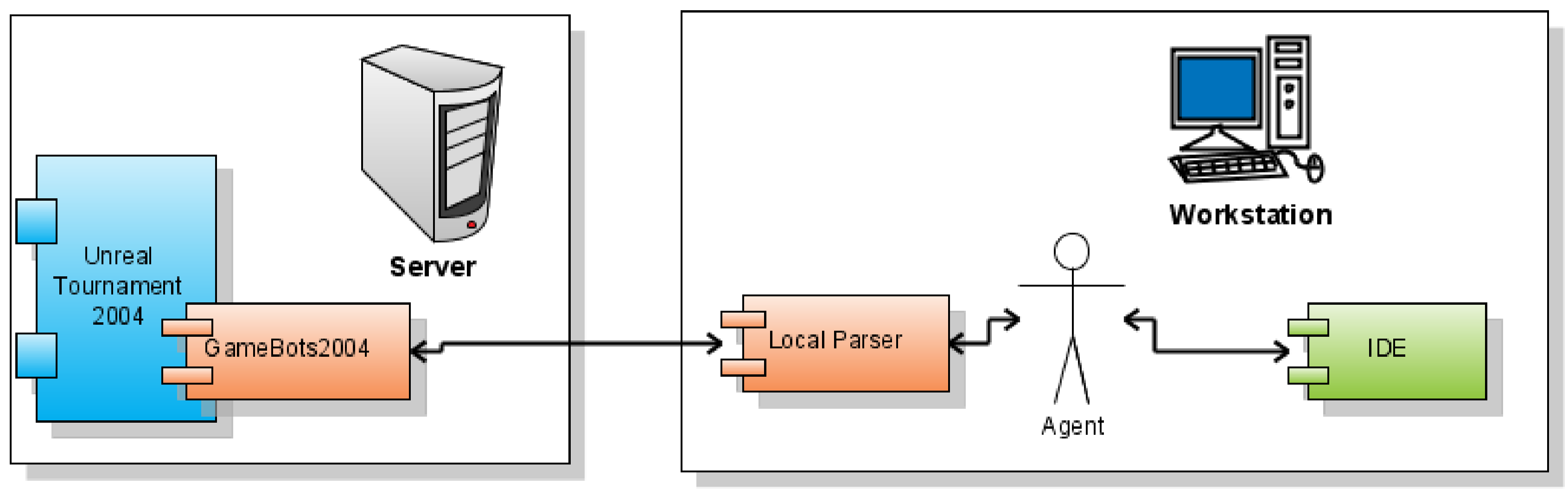


Fig. 2 Pogamut architecture

Architecture

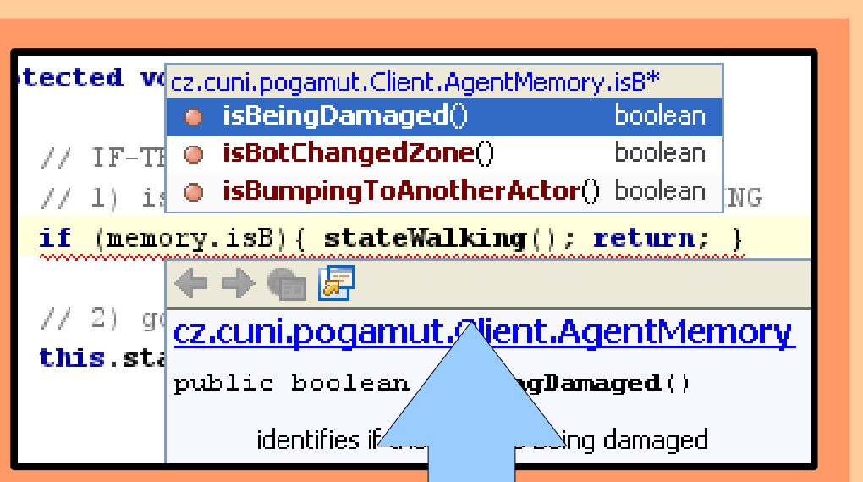
- **Unreal Tournament^{TM} 2004** – commercial game used as a virtual world. It is extensible and contains environmental editor.
- **Gamebots 2004** – built-in server in the UT04, which export information from UT04 for the Agent.
- **Parser** – translates text messages from the GB04 to Java objects.
- **Agent** – here goes the user logic.
- **IDE** – plug-in for Netbeans^{TM} development environment. Provides support for coding and debugging the agent.

Working with Pogamut

1. Implementation

Example use of sensomotoric primitives

```
// 1) do you see enemy? -> start shooting / hunt the enemy
if (memory.getSeeAnyEnemy() &&
    memory.hasAnyLoadedWeapon()) { statePursue(); return; }
// 2) are you shooting? -> stop shooting, you've lost your target
if (memory.getIsShooting()) { body.stopShoot(); return; }
// 3) are you being shot? -> turn around - try to find your enemy
if (memory.getIsBeingDamaged()) { body.turnHorizontal(355); return; }
```



```
tected vo
// IF-TH
// 1) is                                   NG
if (memory.isB){ stateWalking(); return; }
// 2) go
this.sta
```

cz.cuni.pogamut.Client.AgentMemory.isB*

isBeingDamaged()	boolean
isBotChangedZone()	boolean
isBumpingToAnotherActor()	boolean

cz.cuni.pogamut.Client.AgentMemory
public boolean ...ngDamaged()
identifies if ...ing damaged

Code completion for Java provided by Netbeans^{TM} IDE

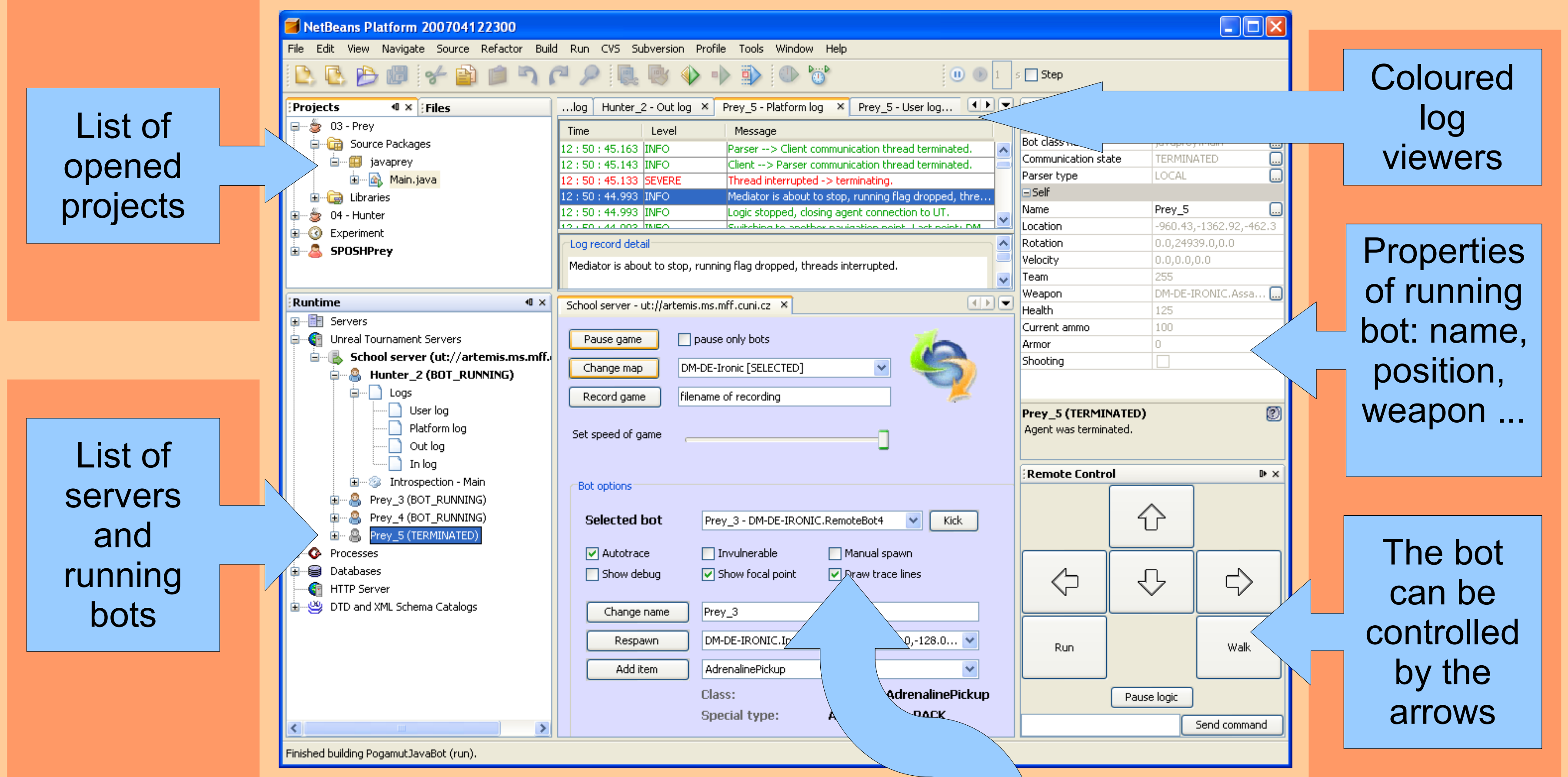
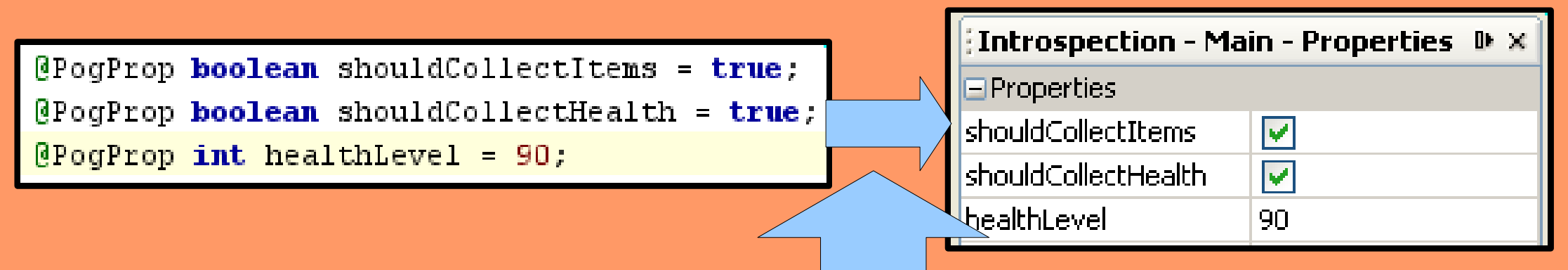

List of opened projects
Coloured log viewers
Properties of running bot: name, position, weapon ...
List of servers and running bots
The bot can be controlled by the arrows

Fig. 3 IDE in work

2. Debugging and Tuning

The map can be changed, a game replay can be saved, items can be added to the bot etc.



```
@PogProp boolean shouldCollectItems = true;
@PogProp boolean shouldCollectHealth = true;
@PogProp int healthLevel = 90;
```

Introspection - Main - Properties

Properties	
shouldCollectItems	✓
shouldCollectHealth	✓
healthLevel	90

Introspection of annotated variables

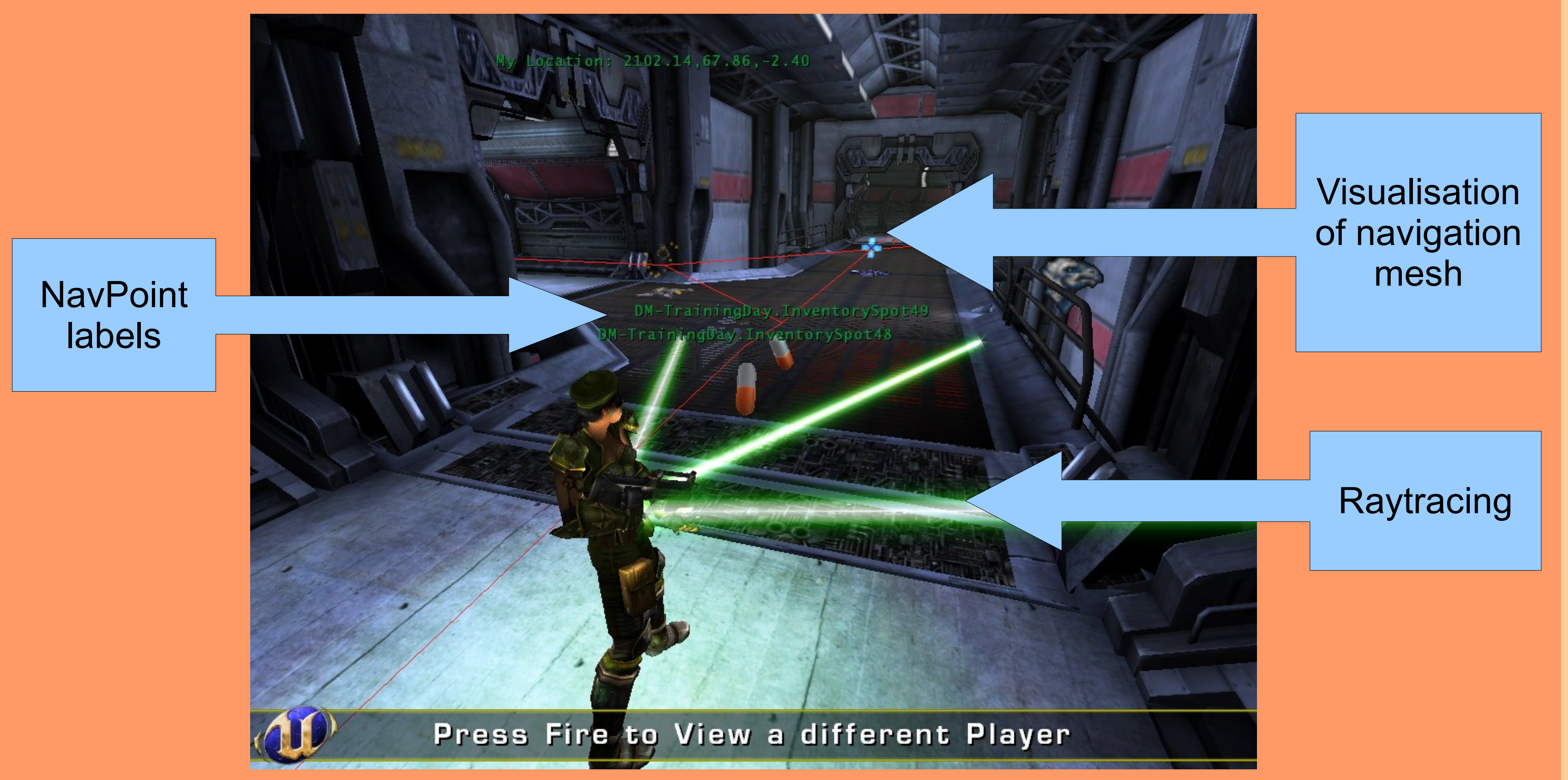

My Location: 2102.14,67.86,-2.40
Press Fire to View a different Player
NavPoint labels
Visualisation of navigation mesh
Raytracing

Fig. 4 Screenshot of the Unreal Tournament 2004 showing some GameBots 2004 debugging features

3. Validation

Validate the implementation of the IVA in a different conditions, log the course of tests and evaluate the results with a third-party statistical software.

```
rule "Hunter sees White Rabbit"
    when
        hunterMemory : AgentMemory( name == "Hunter" )
        eval ( hunterMemory.seeEnemy(globals.get("whiteRabbit").getMemory().getUnrealID) )

    then
        if (hunterMemory.isShooting())
            log.info("Hunter sees White Rabbit and is shooting.")
        else
            log.severe("Hunter sees White Rabbit and is NOT shooting.")
end
```

Number of example bots (eg. Khepera like bot from Fig. 4) is included in the Pogamut installer.

Acknowledgment:

This work was supported by grant GA UK 1053/2007/A-INF/MFF, Research Project of the Ministry of Education of the Czech Republic No. MSM0021620838 and by grant "Information Society" 1ET100300517.

References:

[1] Unreal Tournament.Epic Games, Inc.Product homepage http://www.unrealtournament.com [4th Sep 2007]
[2] Adobbati, R., Marshall, A. N., Scholer, A., and Tejada, S.:Gamebots: A 3d virtual world test-bed for multi-agent research. In: Proceedings of the 2nd Int. Workshop on Infrastructure for Agents, MAS, and Scalable MAS, Montreal, Canada (2001)
[3] Bryson, J.:The Behavior-Oriented Design of Modular Agent Intelligence. In: Mueller, J. P. (eds.): Proceedings of Agent Technologies, Infrastructures, Tools, and Applications for E-Services, Springer LNCS 2592 (2003) 61—76
[4] Jboss Rules, http://www.jboss.com/products/rules [4th Sep 2007]