Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents

Jakub Gemrot, Rudolf Kadlec, Michal Bída, Ondřej Burkert, Radek Píbil, Jan Havlíček, Lukáš Zemčák, Juraj Šimlovič, Radim Vansa, Michal Štolba, Tomáš Plch, and Cyril Brom

Charles University in Prague, Faculty of Mathematics and Physics
Malostranské nám. 2/25, Prague, Czech Republic

Abstract. Many research projects oriented on control mechanisms of virtual agents in videogames have emerged in recent years. However, this boost has not been accompanied with the emergence of toolkits supporting development of these projects, slowing down the progress in the field. Here, we present Pogamut 3, an open source platform for rapid development of behaviour for virtual agents embodied in a 3D environment of the Unreal Tournament 2004 videogame. Pogamut 3 is designed to support research as well as educational projects. The paper also briefly touches extensions of Pogamut 3; the ACT-R integration, the emotional model ALMA integration, support for control of avatars at the level of gestures, and a toolkit for developing educational scenarios concerning orientation in urban areas. These extensions make Pogamut 3 applicable beyond the domain of computer games.

Keywords: Virtual agents, behaviour, control mechanisms, Unreal Tournament, 3D environment, agent development toolkit, emotions, ACT-R, gestures, education, ALMA.

1 Introduction

The development of control mechanisms for videogame agents (that is, their "artificial intelligence") is hard by itself. The developer's work is typically further complicated by the fact that many tedious technical issues, out of the main scope of their work, have to be solved. For instance, they have to integrate their agent within a particular 3D environment, develop at least simple debugging tools or write a code for low-level movement of their agents. This list may continue for pages and developers address these issues over and over, in most cases a waste of energy and time.

For last four years, we have been developing the Pogamut toolkit, which provides general solutions for many of these issues, allowing developers to focus on their main goals. A new major version, Pogamut 3, has been released. Importantly, Pogamut 3 is designed not only for advanced researchers, but also for newcomers. Pogamut 3 can help them to build their first virtual agents, a feature which makes it applicable as a toolkit for training in university and high-school courses, as demonstrated in [9].

F. Dignum et al. (Eds.): Agents for Games and Simulations, LNAI 5920, pp. 1–15, 2009.

The purpose of this paper is to review the main features of Pogamut 3, its architecture, extensions, and work in progress. Section 2 introduces the main features. Section 3 briefly discusses applications of Pogamut 2 and Pogamut 3. Section 4 introduces new extensions added to Pogamut 3 recently. Section 5 discusses related work. Section 6 discusses work in progress and concludes the paper.

This paper is an extension of a short paper presented in [21]. More information about Pogamut 3 can be found at the project homepage [3], where an installer is available for download. The project's page features tutorial videos, learning material about the platform and forums, which provide the always-needed support and help to build the community around Pogamut 3.

2 Features of Pogamut 3

The agent development cycle can be conceptualized into five stages: (1) inventing an agent, (2) implementing the agent, (3) debugging the implemented agent, (4) tuning the agent's parameters, and (5) validating the agent through series of experiments. Individual features of Pogamut 3 were purposely designed to provide the support during the last four stages.

The features of Pogamut 3 include:

1) a binding to the virtual world of the Unreal Tournament 2004 videogame (UT2004) [17],
2) an integrated development environment (IDE) with a support for debugging (Fig. 2),
3) a library with sensory-motor primitives, path-finding algorithms, and a support for shooting behaviour and weapon handling,
4) a connection to the POSH [11], which is a reactive planner for controlling behaviour of agents, and a visual editor for POSH plans (Fig. 4),
5) a support for running experiments, including distributed experiments running on a GRID.

Technically, the Pogamut 3 toolkit integrates five main components: *UT2004*, *GameBots2004*, the *GaviaLib library*, the *Pogamut agent*, and the *IDE* (Fig. 1). These components implement the features mentioned above, enabling a user to create, debug, and evaluate his or her agents conveniently.

In general, the components of the Pogamut 3 toolkit can be used *per partes*. For instance, one can use GameBots2004 independently as well as a part of the GaviaLib library or the IDE. We will return to this point later.

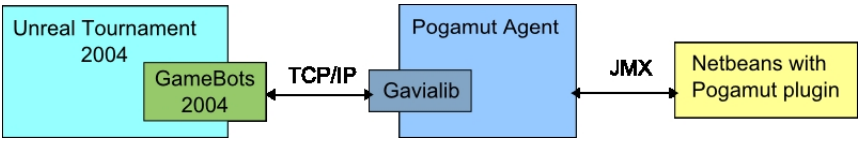

Fig. 1. Pogamut architecture

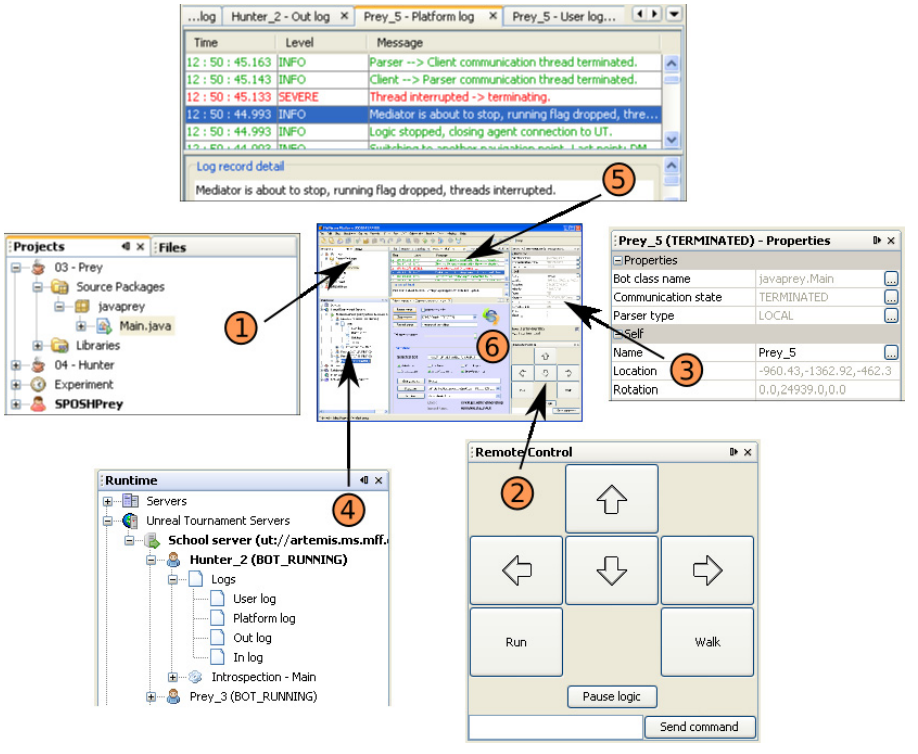

Fig. 2. Pogamut 3 NetBeans plugin (6) and its parts (1-5). The parts are described in the text.

2.1 Virtual Environment of Pogamut 3

UT2004 is a well-known action videogame with so-called "partly opened" code. In particular, the game features UnrealScript, a native scripting language developed by the authors of UT2004, in which a substantial part of the game is programmed; almost everything except of the graphical and the physical engine. Users can modify the code in UnrealScript. Additionally, the game features a lot of pre-built objects, maps, and a level editor. These two points allow users to create new game extensions and blend them with the original game content. In the context of the Pogamut 3 toolkit, the game provides the virtual environment for running agents.

Actually, there are more games with partly or fully opened codes, for instance Halo 2 [12] or Quake 3 [25]. Each of these games has its advantages and disadvantages – there is no single winner. We chose UT2004 since its user community is large and it is relatively new offering better graphical experience comparing e.g. to Quake 3. Concerning UT2004's disadvantages, it should be mentioned, for instance, that the whole code of Quake 3's engine is available, which is not the case of UT2004. Halo2 would likely be a better platform than UT2004 for developing tactic AI.

2.2 Interface to the Virtual World

One can implement agents directly in UnrealScript; however, it is often advantageous, both for educational as well as experimental reasons, to develop the agents using a different, external mechanism. This means that one has to create a binding to the UT2004 game. Several years ago, Adobbati et al. developed a generic binding for this purpose called GameBots [1]. It was swiftly adopted by the UT community. The original GameBots worked with UT99. GameBots2004 is our customized extension of the original GameBots, which connects the rest of the Pogamut toolkit to the UT2004.

GameBots2004 exports information about the game environment through a TCP/IP text based protocol, allowing users to connect to UT employing the client-server architecture. This means that the user implements an agent's control mechanisms at the client and uses GameBots2004 as the server.

Importantly, GameBots2004 can be used separately without other components of Pogamut 3. Thus, one can connect his/her own development environment to UT2004. On the other hand, if someone aims at connecting Pogamut 3 to other virtual environment, he/she has to develop his/her own world-interface replacing GameBots2004.

2.3 GaviaLib Library

The GaviaLib library has been developed as a general purpose Java library for connecting agents to almost any virtual environment, i.e. it is a generic interface for accessing a wide range of possible worlds. Only mild assumptions have been imposed upon virtual environments; in a nutshell, an environment has to work with objects and be able to provide information in an event-based manner.

GaviaLib provides:

- Abstract agent implementation, which handles the agent's life cycle, allows for the remote control of the agent through JMX [40], a standard Java protocol for remote instrumentation of programs, and defines a common set of logs,
- Agent's interface to the world, which manages object and event identities and notifies the agent about important object events (e.g. an object has appeared/disappeared/been changed),
- XML/XSLT framework for the definition of the world's objects and events.

What deserves most attention is the agent's interface. This interface provides API for listening for events and for querying object instances. Both of these use cases utilize Java class hierarchy allowing for different levels of abstraction, an analogy to semantic ontologies. This feature can be illustrated on the following example. Let us assume a simulator of a virtual world that can be populated by items such as fruits and vehicles, but currently only apples and cars are supported. Concerning agents that are to populate this world, this means that designers must provide a way for the agents to recognize items of these two kinds. When GaviaLib is used, designers have to create following classes representing the objects' types and their categories: "item", "fruit", "vehicle", "apple", and "car". Utilizing Java class inheritance, designers will further

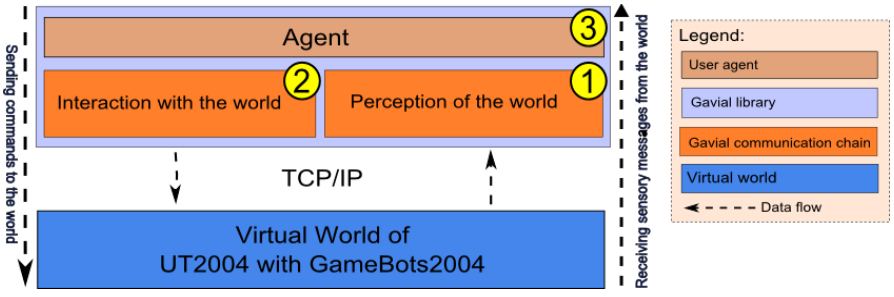

Fig. 3. High-level GaviaLib architecture overview. (1) Perceptual part servers to process information messages from the virtual world that Pogamut 3 is connected to. (2) Interaction part sends commands to the virtual world. (3) Agent abstractions comprise stubs of agents, defining high-level interface for agents' minds.

define that classes "vehicle" extends "item", "fruit" extends "item", "apple" extends "fruit", and "car" extends "vehicle". GaviaLib then propagates events according to this ontology; for instance, when an event happens on the object of class "apple", it is propagated also to "fruit" and "item". This makes the event model of GaviaLib flexible as the user may listen on all "fruit" events by attaching a listener on the class "fruit" instead of attaching the listener on every instance of fruit. This feature is not available when using GameBots2004 alone. This implementation also allows for custom extensions to existing code. When another designer creates a new class "banana" that extends "fruit", all existing listeners on "fruit" will automatically be able to report events that have happened on all bananas.

The GaviaLib's layered architecture (Fig. 3) makes it possible to use different means of communication with the virtual world. A GaviaLib agent can be connected through plain text TCP/IP protocols (like GameBots2004), CORBA or, in the case of Java environments, it can reside in the same Java Virtual Machine as the world simulator and communicate directly through method calls.

GaviaLib has also other features. These are fully detailed in [22].

The possibility of general usage of the library will be known only after rigorous testing with various virtual environments. Presently, the only stable binding featured by GaviaLib is the UT2004 binding, which employs Gamebots2004, as detailed in Sec. 2.4. However, we also have a work-in-progress binding with the Defcon game [27]. Defcon is simulation of a global thermonuclear conflict played in real-time. The player takes a role of a commander in charge of military forces under the flag of one nuclear nation. The successful binding of GaviaLib to Defcon would provide evidence that the library could be used as a general middleware for connecting to virtual environments. The investigation of GaviaLib possibilities on a rigorous basis presents future work.

2.4 UT2004 Agent

The UT2004 agent is a set of Java classes and interfaces derived from the basic classes of the GaviaLib library. The UT2004 agent is derived from all Parts (1) – (3) from Fig. 3, adding UT2004 specific features, such as UT2004's sensory-motor

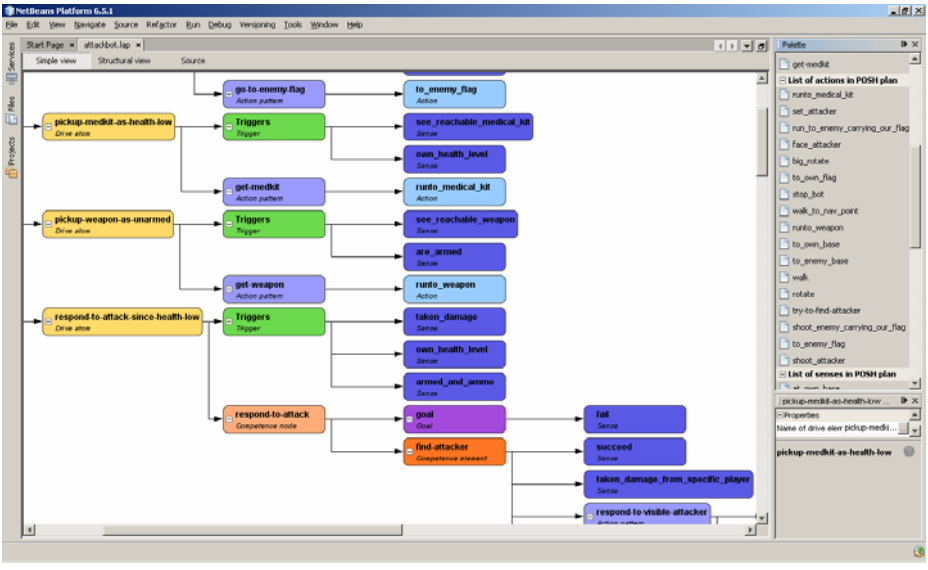

Fig. 4. Screenshot of POSH visual editor. A part of the plan of Hunter agent, which is a standard Pogamut example agent, is depicted.

primitives, the navigation utilizing the internal UT2004 way-point system and auxiliary classes providing information about the game rules. Thus, if a user wants to connect Pogamut to a different environment, he/she has to do both of these things: a) create a new interface to the virtual world (i.e. replace GameBots2004), b) create domain specific mechanisms at the side of the GaviaLib library (i.e. replace UT2004 agent).

Developers can program UT2004 agents using the classes of the Pogamut agent; they can implement an agent's control mechanism directly in Java or use the reactive planner POSH. POSH has been developed as an independent action selection planner by Joanna Bryson [11] and Pogamut 3 exploits it as a plug-in. From our perspective, the main advantage of this planner is that it is easy to understand for beginners allowing them to program their first agents quickly and easily comparing e.g. to Soar [44], Drools rules [29] or Jadex [4]. Similarly to some but not all of the other planners and reasoners, it enforces the design that clearly separates low-level actions and sensory primitives (coded in Java or Python) from a high level hierarchical plan of an agent's behaviour, simplifying the design process. POSH also lacks some features of more advanced planners and rule-based engines such as variable binding at the level of plans. This is often an advantage for beginners; some of these features may be unintelligible to them. However, at the same time, these features may be vital for advanced users. For those who already advance, it may be an option to use plain Java or another planner instead of POSH.

The editor for POSH plans is available (Fig. 4).

2.5 Integrated Development Environment

The IDE is developed as a NetBeans plugin [41] (Fig. 2). It can communicate with running agents via JMX. The IDE offers multiple features for a designer, such as empty agent project templates, example agents (e.g. hunter and prey agents or a khepera-like agent) (Fig. 2, Window 1), management of UT2004 servers, list of running agents (Fig. 2, Window 4), introspection of agent's variables and a quick access to general agent properties (Fig. 2, Window 3), the step-by-step debugging, log viewers (Fig. 2, Window 5), and the manual controller of agent's position (Fig. 2, Window 2). However, the Pogamut agents can also be developed without this plug-in in any other Java IDE using only GaviaLib with the UT2004 extension.

3 Applications and Scalability

Pogamut 2 (i.e. the previous version) has been used in many research projects [e.g. 42, 38], including student projects at our university ranging from utilizing evolutionary techniques for agent development [31], trough fuzzy rule control of behaviour of agents [39], up to episodic memory modelling [13]. Pogamut 2 has been employed for training in university and high-school courses [9, see also 7 for supplementary materials]. It was also picked as a base for 2K BotPrize competition[1], where participants were challenged to develop an agent that can fool the panel of judges that it is a human.

Some projects have been completed using Pogamut 3 (i.e. the present version), including a prototype of an educational game for teaching students the basics of developing virtual agents and virtual storytelling [8]. However, Pogamut 3 is presently more buggy than the previous version (although this may change soon).

The number of UT2004 agents a user can control via Pogamut 3 depends on the agents' complexity. Generally, Pogamut 3 cannot be expected to handle more than 10 agents at the same time. However, Pogamut 3, as well as Pogamut 2, allows for running both Pogamut agents as well as UT agents (i.e. agents programmed directly in the Unreal Script) simultaneously and the number of UT agents is limited only by the UT2004 server ability to handle the load.

The synchronous perception-action cycle for Pogamut 3 agents lasts 250 ms. This constant can be decreased to about 100 ms depending on the computer and the number and complexity of agents within the simulation. Decreasing the constant below this number can render the application unstable mainly due to GameBots2004 and UT2004 limitations. Asynchronous messages are processed instantaneously; the time lag caused by the propagation of the event via GameBots2004 and GaviaLib is in the order of milliseconds depending on the computer hardware used.

4 Pogamut's Extensions

Recently, several extensions to Pogamut 3 have been completed. The purpose of these extensions is to make Pogamut 3 useful also beyond the domain of gaming AI. The extensions have been developed independently: they are not integrated in a single

[1] http://botprize.org (27.9.2009)

package. Moreover, they are not included into the Pogamut 3 release version at the time of writing of this paper, but they may be available for download in coming months. In this section, we will review these extensions in further detail: ACT-R Integration, Gestures module, ALMA Integration, and Educational Scenarios.

4.1 ACT-R Integration

Recently, it has been proposed that computational cognitive science modelling can benefit from implementing models using virtual agents [10, 30]. This means, a neuro-/psychologically plausible model can be integrated within a virtual agent's "mind" and then tested trough scenarios in virtual worlds. Advantages of this approach have been discussed in [10]. One potential way how to do this is to adopt a general cognitive architecture, such as Soar [33, 44], ACT-R [2] or LIDA [19], as the main architecture of a virtual agent's mind. These architectures are somewhat plausible and tend to be modular; hence one can embed his or her model in one of the modules without loosing much of its original plausibility. In fact, Jilk and colleagues [30] did exactly this for a model of the visual cortex.

Pogamut 3 newly offers a binding to ACT-R cognitive architecture, so called PojACT-R [50] (Fig. 5). Because Pogamut is written in Java, we used jACT-R [24], the Java implementation of ACT-R. We run jACT-R in a separate thread. The important limitation of PojACT-R is that the cognitive cycle of ACT-R lasts 50 ms, while the Pogamut's cycle is longer (about 100-250 ms). This issue is partly ameliorated by a special type of perception buffers implemented by the default PojACT-R agent. These buffers hold perceptual information from the UT2004 temporarily and pass it to the jACT-R in a step-wise manner while giving high priority to asynchronous messages.

To demonstrate that PojACT-R works, we implemented a simple Hunter agent featuring the ACT-R architecture. Preliminary results suggest that the efficiency of this agent is acceptable for research purposes (e.g. the cognitive cycle of this agent featuring 160 rules lasted about 20 ms on average on Intel Core 2.66 GHz Duo with UT2004 server and client running [50]). Java principles and the modular architecture enable various kinds of experiments (e.g. spatial orientation, episodic memory modelling, multithreaded AI).

Different reasoning engines, such as Jadex [4], can be connected to Pogamut similarly to jACT-R. In fact, Educational scenarios described in Sec. 4.3 uses Drools rules [29] in this manner. We decided to connect jACT-R to make Pogamut 3 available for the community of computational cognitive psychology. BDI reasoners such as Jadex, which can make the platform available for traditional software agents community, can present another future goal.

4.2 Gestures Module

It is widely accepted that in order to achieve believable virtual agents, some level of gesturing and facial expressions is needed. Even though UT2004 features quite powerful animation system, gestures and facial expressions are typically expressed using animations prepared in advance (i.e. made in specialized applications). Our aim was to allow users to implement mechanisms that can compose animations of the agents' avatars as much on-the-fly as possible while depending on UT2004 as little as

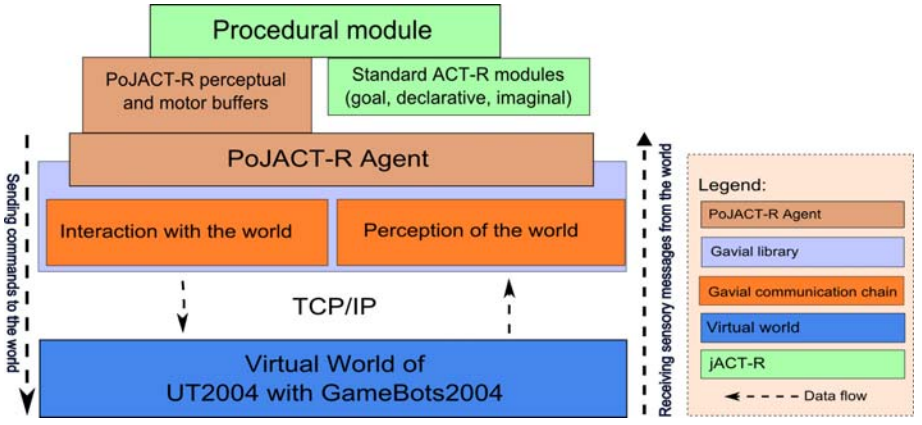

Fig. 5. Architecture of PojACT-R. Compare this figure with Fig. 3.

possible; in other words, to allow users to control UT2004 avatars directly from Pogamut 3. To this end, we had to find a way of defining, planning, decomposing, and transporting of animations.

Concerning planning, we decided to use Behaviour Markup Language (BML) [32]. BML is intended to become "a framework and an XML description language for controlling the communicative human behaviour of embodied conversational agents" [32]. It provides a convenient way to create and parse sequences of behaviours, including gestures, facial animations and speech, though in our case, only lexicalized gestures and gaze have been implemented so far.

For definition of lexicalized gestures, we have devised our own simplistic XML-based language, which has been designed to operate easily with BML. If reasonable, BML can be bypassed and the user can supply his own implementation of a planner through a simple interface.

For decomposition of animations the MPEG-4 with its "atomic animations" was chosen. Examples of such animations are "arm flexing," "arm twisting," or "arm abduction." These can be decomposed furthermore into lower level skeletal animations, which are then handled by Unreal Engine and composed into the final result.

Communication with the virtual world is carried using a slightly modified FBA bit-stream format specified in MPEG4 [14]. This bit-stream is quite efficient and lowers the amount of bits transferred to a bare minimum. In order to achieve even less traffic, we have decided to use a variable frame rate (instead of a constant one). On the other hand, concerning efficiency of the transfer, the limitation is the TCP protocol, which increases the traffic. Had we used RTP, the traffic would have been lower. We used TCP for prototyping purposes; in future, we plan to switch to RTP.

The work is detailed in [35] where also an example avatar is presented.

4.3 ALMA Integration

Emotion modelling is a multi-faced theme with many possible benefits for virtual agents. Emotions can affect decision making, expressive part of agents' behaviour or

just serve the agent as another source of information when interacting with humans. All of these points can increase virtual agents' believability.

Therefore we decided to incorporate the ALMA emotion model [20] into Pogamut 3. ALMA is based on OCC theory of emotions [34] and provides our agents with three distinct types of affects: *emotions* for short term affects, *mood* for medium term affects and *personality* for long-term affects. ALMA is coded in Java and offers a GUI where the model can be configured easily.

ALMA integration works as follows. First, a user defines a set of events that should cause emotional responses. These events will be detected in the environment by Pogamut's methods. Second, whenever one of these events occurs in the environment, it will be appraised by a set of ALMA emotion variables and sent as an input to ALMA. Third, ALMA will process the appraised event and generate a change of affective variables.

The ALMA-to-Pogamut 3 binding was already used in Project Emohawk [5] to provide virtual agents with affects. Project Emohawk features a simple scenario, taking place in small town called UnrealVille. The virtual agents (a boy and two girls) are able to invite each other to the cinema, escort each other home or go for a walk to the park. The agents can like or dislike other agents, get angry with them or even fall in love. The analysis confirmed that the ALMA-to-Pogamut 3 binding is stable and can be used for virtual agents emotion modelling.

4.4 Educational Scenarios

Educational scenarios (ES) present a framework aimed at easy authoring of short didactic scenes. The current prototype implements a scenario for exercising orientation in urban areas. The target audience of this prototype is primary school students. The students are challenged in a game-like world with tasks ranging from following a path presented on a 2D map in a 3D visualization of a city, trough marking the player's 3D location on a 2D map, to estimating direction to the player's home. However, the ES framework can be used in a more general way.

In the content creation phase, ES provides an Eclipse IDE plug-in for specifying areas and points of interest in a 2D map of a city (e.g. street crossings, the player's home etc.). These landmarks can be then referenced from a script describing behaviour of the whole scenario. Drools rules engine [29] is used as the underlying scripting language for this purpose. A simple domain specific language resembling restricted natural English has been defined on top of the Drools rules, which makes the process of scenario creation accessible even for non-expert programmers.

ES also extends the core functionality of Pogamut 3 with a set of dialog widgets visualized in the UT2004, making it possible to run a dialog system within the game in run time. This dialog system can be used independently of the plug-in for specifying landmarks and areas on the map.

The work is detailed in [47].

5 Related Work

With respect to the Pogamut's objective, we will break down related work into three categories; a) tools for teaching students basics of development of virtual agents and virtual storytelling applications, b) tools supporting research on gaming agents' AI, c)

general multi-agent platforms. Concerning Point (a), many tools for authoring 3D graphics exist, as opposed to tools for teaching programming of behaviour for virtual agents. Several authoring toolkits emerged from the cauldron of the virtual storytelling community in recent years [reviewed in 36], but they tend to provide very limited or no support for teachers and learners, many of them are poorly documented and some of them are not downloadable. To our knowledge, with respect to learning programming behaviour for virtual agents, there are only two tools (besides Pogamut) that support education explicitly and whose dissemination to schools has been documented: Netlogo [49] and Alice [15]. Netlogo is an excellent entry-level tool for building simple agents and running social simulations, but it does not allow for the creation of agents with complex behaviour and possesses no complex 3D presentation environment. Alice's main objective is the introduction of basic programming concepts like loop statements or object-oriented programming. The complexity of the built-in 3D environment is not comparable to current game engines and the environment is only a mean to achieve Alice's educational goal.

Concerning Point (b), there are several tools for the development of behaviour of human-like agents inside commercial game engines. Table 1 lists these tools along with their game engines and programming languages.

All of these tools except the Pogamut 3, UT3 .Net Bots and RIT's GameBots are using rather old game engines as their simulation environment. Quake 2 was released in 1998 and Unreal Tournament 99 in 1999. In comparison, Unreal Tournament 2004 used by Pogamut 3 gives better graphical experience. However, Unreal Tournament 3 (released in 2007) used by UT3.Net Bots is superior to all other game engines used by the listed tools. However, there is a trade-off: UT3 requires PCs with better performing hardware (i.e. graphic cards and CPUs), which may not be available, e.g. if the tool is used at high-schools for workshops.

Table 1. A list of open source tools for development of computer game agents. The year of the last release/update is based on web information, we have not contacted the authors. Thus, this information is not guaranteed.

Name	Game	Language	Last release	GUI	Note
F.E.A.R [16]	Quake 2	C++	2005	None	
Quagents [45]	Quake 2	C++	2005	None	Prolog and Matlab integration
QASE [23]	Quake 2	Java	2009	None	Matlab integration
Original Gamebots [1]	UT99	--	2001	None	Just a network protocol
RIT's Gamebots [37]	UT03, UT04	--	2007	Simple visualizer	Just a network protocol
JavaBots [28]	UT99	Java	2002	Simple	
UT3 .Net Bots [46]	UT3	C#	2008	Simple level map	
Pogamut 3 [3]	UT04	Java	2009	Netbeans plugin	POSH, BML, ACT-R

All of these frameworks are extensible but none of them besides Pogamut and Quagents comes with any 3rd party decision making system already connected. On the other hand, several tools offer a Matlab integration.

Finally, all implementations of pure Gamebots are merely communication protocols. They provide neither any feature of GaviaLib, nor a complex integrated development environment offered by Pogamut 3.

Concerning (c), the question is what is the relation of general multi-agent (MAS) platforms, such as Jade [43], to Pogamut 3 and GaviaLib. The point is that MAS platforms are typically not used in "agents in videogames" research; why is this? Our answer on this question should not be understood as a definite claim but rather as a statement aiming to motivate a further discussion on this topic. Our opinion is that MAS platforms are basically something different than toolkits like Pogamut; they have been built for different purpose. If one wants to use Jade in "agents in videogames" research, one would need to create a wrapper-agent that would encapsulate the whole virtual environment (e.g. UT2004) and that would provide communication interface to agents that would represent reasoners for in-game agents.

In other words, GameBots would be replaced by the wrapper-agent and the client-server architecture of a Pogamut-like toolkit by the MAS architecture. But why should anyone do this? On the one hand, all the work that we and others have done to implement GameBots would need to be done again in implementing the wrapper-agent, and all the work behind GaviaLib would need to be done again at the side of reasoning-agents. Actually, this reimplementation in a MAS platform may even bring a disadvantage. As GameBots communicate via a text-based protocol, the parsing is fast. If one wants to exploit FIPA-like messages [18], the parsing overhead would increase.

On the other hand, there are many features of MAS platforms without obvious use for game agents, for instance yellow pages. The whole "service-oriented" view of MAS seems to be needless for game agents. We see one possible advantage of a MAS platform in that the agents can communicate each other with directly. Each Pogamut agent can communicate with other agents in game only via the virtual environment. MAS platforms offer direct agent-agent communication channels, by-passing the environment. The trouble with this advantage is that game designers would often want to have all the communication directed via the environment for the reasons of plausibility. Still, direct agent-agent channel may be useful in more complicated games with many agents, e.g. in tactical games where the agents are required by designers to communicate each other with (e.g. solders via a walkie-talkie).

To summarize, Pogamut's main contributions are: full integration with Netbeans IDE, connection to the relatively new game engine, binding for ACT-R, POSH, BML, and ALMA, and continuing support and development of the platform.

6 Conclusion and Future Prospect

The Pogamut 3 toolkit provides many general solutions for developers of videogame agents. The important facet of Pogamut 3 is that it has been designed not only for advanced researchers, but also for newcomers. Its applicability for beginners has been proved by using it as a toolkit for teaching in a university and a high-school course on programming virtual agents [7, 9].

The current release [3] of Pogamut has also several limitations. Most notably, its integration with UT2004 and the support provided by GaviaLib classes make it most suitable for first-person shooter AI projects; this suits some, but is unsuitable for others. Although the ACT-R integration [50] is now available for cognitive psychologists and Educational scenarios [47] make Pogamut 3 partly applicable for the domain of serious games, features that would help with general virtual agents and general gaming AI development and education are limited.

To partly address this issue, we are now extending Pogamut 3 by a generic educational storytelling scenario with non-violent graphics [8], which utilizes the ALMA model [20]. Pogamut has been being connected to the Defcon game [27] and we are considering a Virtual Battle Space 2 (VBS2) [5] connection, which would feature the industry standard HLA interface (IEEE 1516) [26]. The creation of a general GaviaLib–HLA bridge would make it possible to connect even more environments with minimal effort.

Acknowledgments. Extension of several parts of this work was partially supported by the project CZ.2.17/3.1.00/31162 that is financed by the European Social Fund, the Budget of the Czech Republic and the Municipality of Prague. Other parts were partially supported by the Program "Information Society" under project 1ET100300517, by the grant 201/09/H057, and by the research project MSM0021620838 of the Ministry of Education of the Czech Republic.

References

1. Adobbati, R., Marshall, A.N., Scholer, A., Tejada, S.: Gamebots: A 3d virtual world test-bed for multi-agent research. In: Proc. of the 2nd Int. Workshop on Infrastructure for Agents, MAS, and Scalable MAS, Montreal, Canada (2001)
2. Anderson, J.R.: How can the human mind occur in the physical universe? Oxford University Press, Oxford (2007)
3. Artificial Minds for Intelligent Systems, Research Group: The Pogamut platform. Charles University in Prague, `http://artemis.ms.mff.cuni.cz/pogamut` (27.9.2009)
4. Braubach, L., Pokahr, A.: Jadex: BDI Agent System, `http://jadex.informatik.uni-hamburg.de` (27.9.2009)
5. Bída, M.: Artificial emotions in computer games. Diploma thesis. Charles University in Prague, Czech Republic (2009), `http://artemis.ms.mff.cuni.cz/pogamut/tiki-index.php?page=Emotional+AI+in+UT` (27.9.2009)
6. Bohemia Interactive: Virtual Battle Space 2, `http://virtualbattlespace.vbs2.com/` (27.9.2009)
7. Brom, C.: Curricula of the Course on Modelling Behaviour of Human and Animal-like Agents. In: Proceedings of the Frontiers in Science Education Research Conference, Famagusta, North Cyprus, pp. 71–79 (2009)
8. Brom, C., Bída, M., Gemrot, J., Kadlec, R., Plch, T.: Emohawk: Searching for a "Good" Emergent Narrative. In: Iurgel, I.A., Zagalo, N., Petta, P. (eds.) ICIDS 2009. LNCS, vol. 5915, pp. 86–91. Springer, Heidelberg (2009)
9. Brom, C., Gemrot, J., Burkert, O., Kadlec, R., Bída, M.: 3D Immersion in Virtual Agents Education. In: Spierling, U., Szilas, N. (eds.) ICIDS 2008. LNCS, vol. 5334, pp. 59–70. Springer, Heidelberg (2008)

10. Brom, C., Lukavský, J.: Episodic Memory for Human-like Agents and Human-like Agents for Episodic Memory. Technical Reports FS-08-04, Papers from the AAAI Fall Symposium Biologically Inspired Cognitive Architectures, Westin Arlington, VA, USA, pp. 42–47. AAAI Press, Menlo Park (2008)
11. Bryson, J.J.: Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents. PhD thesis, Massachusetts Institute of Technology (2001)
12. Bungie Studios: Halo 2 (2004), http://www.bungie.net/Projects/Halo2/default.aspx (27.9.2009)
13. Burkert, O.: Connectionist Model of Episodic Memory for Virtual Humans. Master thesis. Charles University in Prague. Czech Republic, http://artemis.ms.mff.cuni.cz/pogamut/tiki-index.php?page=Episodic+memory+for+virtual+agent (27.9.2009)
14. Capin, T.K., Petajan, E., Ostermann, J.: Very low bit rate coding of virtual human animation in MPEG-4. In: IEEE International Conference on Multimedia and Expo., vol. 2, pp. 1107–1110 (2000)
15. Carnegie Mellon University: Alice (1999-2009), http://www.alice.org/ (27.9.2009)
16. Champandard, A.: F.E.A.R. – Foundations for Genuine Game AI, http://fear.sourceforge.net/ (27.9.2009)
17. Epic Games: Unreal Tournament 2004 (2004), http://www.unreal.com (27.9.2009)
18. The Foundation for Intelligent Physical Agents: Agent Communication Language Specifications, http://www.fipa.org/repository/aclspecs.html (27.9.2009)
19. Franklin, S., Baars, B.J., Ramamurthy, U., Ventura, M.: The role of consciousness in memory. Brains, Mind, Media 1, 1–38 (2005)
20. Gebhard, P.: ALMA – A Layered Model of Affect. In: Proc. of the 4th Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, pp. 29–36 (2005)
21. Gemrot, J., Kadlec, R., Bida, M., Burkert, O., Pibil, R., Havlicek, J., Zemcak, L., Simlovic, J., Vansa, R., Stolba, M., Plch, T., Brom, C.: Pogamut 3 Can Assist Developers in Building AI for Their Videogame Agents. In: Dignum, F., et al. (eds.) Agents for Games and Simulations. LNCS (LNAI), vol. 5920, pp. 1–15. Springer, Heidelberg (2009)
22. Gemrot, J., Brom, C., Kadlec, R., Bída, M., Burkert, O., Zemčák, M., Píbil, R., Plch, T.: Pogamut 3 – Virtual Humans Made Simple. In: Gray, J., Nefti-Meziani, S. (eds.) Advances in Cognitive Systems. IET Publisher (in press)
23. Gorman, B., Fredriksson, M., Humphrys, M.: The QASE API - An Integrated Platform for AI Research and Education Through First-Person Computer Games. International Journal of Intelligent Games and Simulations 4(2) (2007), http://ut3bots.codeplex.com/ (27.9.2009)
24. Harrison, A.: jACT-R project, http://www.jactr.org/ (27.9.2009)
25. idSoftware: Quake III Arena (1999), http://www.idsoftware.com/games/quake/quake3-arena/ (27.9.2009)
26. IEEE: IEEE 1516, High Level Architecture (HLA) (2001), http://www.ieee.org
27. Introversion: Defcon, http://www.introversion.co.uk/defcon/ (4.3.2009)
28. JavaBots, http://utbot.sourceforge.net/ (27.9.2009)
29. JBoss community: Drools project, http://www.jboss.org/drools/ (27.9.2009)
30. Jilk, D.J., Lebiere, C., O'Reilly, R.C., Anderson, J.R.: SAL: An explicitly pluralistic cognitive architecture. Journal of Experimental and Theoretical Artificial Intelligence 20(3), 197–218 (2008)
31. Kadlec, R.: Evolution of intelligent agent behaviour in computer games. Masters thesis. Charles University in Prague, Czech Republic (2008), http://artemis.ms.mff.cuni.cz/main/papers/GeneticBots_MSc_Kadlec.pdf (27.9.2009)

32. Mindmakers: Behavior Markup Language (BML), Draft 1.0 (2009), http://wiki.mindmakers.org/projects:bml:draft1.0 (27.9.2009)
33. Newell, A.: Unified Theories of Cognition. Harward University Press, Cambridge (1990)
34. Ortony, A., Clore, G.L., Collins, A.: The cognitive structure of emotions. Cambridge University Press, Cambridge (1988)
35. Píbil, R.: Gestures and Facial Expressions of Virtual Humans in 3D Worlds. Bachelor thesis. Charles University in Prague, Czech Republic (in Czech) (2009), http://artemis.ms.mff.cuni.cz/pogamut/tiki-index.php?page=Gestures+and+Facial+expressions+-+BML (27.9.2009)
36. Pizzi, D., Cavazza, M.: From Debugging to Authoring: Adapting Productivity Tools to Narrative Content Description. In: Spierling, U., Szilas, N. (eds.) ICIDS 2008. LNCS, vol. 5334, pp. 285–296. Springer, Heidelberg (2008)
37. Rochester Institute of Technology: GameBots branch (2005), http://www.cs.rit.edu/~jdb/gamebots/ (26.9.2009)
38. Small, R.K.: Agent Smith: a real-time game-playing agent for interactive dynamic games. In: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation, Atlanta, GA, USA, pp. 1839–1842 (2008)
39. Štolba, M.: Decision rules for project Pogamut 2. Bachelor thesis. Charles University in Prague, Czech Republic (in Czech) (2008)
40. Sun Microsystems: JMX, http://java.sun.com/javase/6/docs/technotes/guides/jmx/ (27.9.2009)
41. Sun Microsystems: Netbeans IDE, http://www.netbeans.org (27.9.2009)
42. Tence, F., Buche, C.: Automatable evaluation method oriented toward behaviour believability for video games. In: International Conference on Intelligent Games and Simulation (GAME-ON 2008), pp. 39–43 (2008)
43. Telecom Italia Lab: Jade – Java Agent DEvelopment Framework, http://jade.tilab.com/ (27.9.2009)
44. University of Michigan: Soar, http://sitemaker.umich.edu/soar/home (27.9.2009)
45. University of Rochester: Quagents: Quake Agents, http://www.cs.rochester.edu/research/quagents/ (27.9.2009)
46. UT3 .Net Bots project, http://ut3bots.codeplex.com (27.9.2009)
47. Vansa, R.: Educational scenarios in the Pogamut project. Bachelor thesis. Charles University in Prague, Czech Republic (2009), http://artemis.ms.mff.cuni.cz/pogamut/tiki-index.php?page=Educational+scenarios (27.9.2009)
48. Vilhjalmsson, H., Cantelmo, N., Cassell, J., Chafai, N., Kipp, M., Kopp, S., et al.: The Behavior Markup Language: Recent Developments and Challenges. In: Pelachaud, C., Martin, J.-C., André, E., Chollet, G., Karpouzis, K., Pelé, D. (eds.) IVA 2007. LNCS (LNAI), vol. 4722, pp. 99–111. Springer, Heidelberg (2007)
49. Wilensky, U.: NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University (1999), http://ccl.northwestern.edu/netlogo/ (27.9.2009)
50. Zemčák, L.: ACT-R in Pogamut, Bachelor thesis. Charles University in Prague, Czech Republic (in Czech) (2009), http://artemis.ms.mff.cuni.cz/pogamut/tiki-index.php?page=PojACT-R (27.9.2009)