Creating game bots in a few easy steps

Rudolf Kadlec Ondřej Burkert Cyril Brom
Charles University in Prague
Malostranské nám. 2/25
Prague, Czech Republic
rudolf.kadlec@gmail.com ondrej.burkert@gmail.com brom@ksvi.mff.cuni.cz

ABSTRACT
This paper describes the content of our tutorial concerned with the development of artificial intelligence for virtual characters in computer games, so-called bots. The Pogamut platform was chosen for this tutorial.

Categories and Subject Descriptors
D.2.11 [**Software Architectures**]: Domain-specific architectures.

General Terms
Design, Experimentation.

Keywords
Virtual agents, Decision making, Java, Unreal Tournament 2004.

1. INTRODUCTION
The creation of computer game bots acting in complex and life-like virtual worlds has been the domain for expert programmers for a long time. However, in recent years, software tools designed for use by the non-expert programmers have emerged [2]. The Pogamut software platform [1] was designed with the novice programmer in mind.

Thanks to Pogamut, the creation of bots can be made easier and therefore more accessible for a beginner programmer. But still there are some key programming concepts that a beginning programmer should be familiar with before making his or her first bot. In this tutorial, we will explain these concepts, creating one example bot using the Pogamut platform.

2. BASICS OF BOT PROGRAMMING
Typically, bots have a modular control architectures. While one module can be responsible for high level reasoning, another one can deal with issues like long term planning of the bot's movement or obstacle avoidance. Many modules can coexist depending on the bot's purpose, but the three mentioned above are common in almost all life-like bots. Thus, these modules will be discussed during the tutorial a bit closer.

The *high level reasoning module*, often called *decision making system* (DMS) or *action selection mechanism* (ASM), is the central arbiter coordinating other modules. The main task of this module is to decide what to do next. Typically, a bot has a possible set of actions it can execute and the DMS basically selects the most appropriate for a particular situation. Two of the most widespread techniques used for creating DMS are *finite state machines* (FSM) and *reactive rules* (also known as IF-THEN rules). Both of these methods will be explained during the tutorial.

Every bot acts in a at least 2D space., where it has to move from one location to another from time to time. The *path planning module* is responsible for finding a path between the bot's current location and its intended destination. The A* algorithm, a known heuristic graph search algorithm, is often used for this task, optimizing the path based upon various factors – distance, elevation difference etc.

Once the path is computed, the bot starts following it. However, in a dynamic environment, the mindless following of a precomputed path may not always work. For example, other bots or movable objects can block the path or the conditions may change making the path unsuitable. Fortunately, several simple and computationally efficient algorithms dealing with the issue of intelligent avoiding of obstacles can be used to overcome the issue. In general, these algorithms are derived from the original *steering mechanisms* described by Reynolds [6]. Both the A* and steering mechanisms will be introduced during the tutorial, including how they are merged with selection of other actions by a DMS.

When dealing specifically with human-like bots, other issues arise. Simulating and generating emotions becomes important. Emotional responses to the objects and events the bot encounters in the virtual world can be either scripted by hand or generated by a general emotion appraisal model, such as ALMA [3]. General emotion models make it possible to generate responses the designer didn't intend, thus leading to fully emergent behaviour of the bot. The issue of emotional modeling will be introduced during the tutorial.

3. POGAMUT PLATFORM
The Pogamut platform is an open source software tool for easier developing and testing of a bot's DMS. Solutions for the low level problems like path finding, obstacle avoidance etc., are provided by the platform, thus the user can concentrate on creating the bot's behaviour.

The Pogamut platform consists of these main components:

- Virtual environment – the commercial game Unreal Tournament 2004 extended by a GNU licensed communication interface GameBots2004 is used as the virtual reality simulator for programmed bots; Figure 1 shows a screenshot of the environment.
- Class library – a set of Java classes providing API for programming the behavior.
- Integrated development environment – Pogamut provides a plug-in for Netbeans IDE, assisting with debugging and tuning of the bot.

- Emotional generator – a general model of emotion appraisal can be parameterised for each bot to give him an unique personality

Figure 1: Interaction of two bots in modified environment of game Unreal Tournament 2004 (Copyright 2009 Epic Games)

The overview of the Pogamut's architecture is shown in Figure 2. Bot's bodies are embodied in the Unreal Tournament 2004 and are controlled like puppets via the GameBots2004 communication interface. Thus, their "minds" are programmed out of the Unreal Tournament 2004, making it possible to use any arbitrary technique for their controll. During the tutorial, the interface provided by GameBots2004 will be used by the Java library ("Pogamut agent" on Figure 2). Thanks to the Gavialib component, the underlying TCP/IP communication is handled transparently with respect to the bot's code eliminating the need to take care of the low-level communication by the user . Bot's logs and internal parameters can be observed from the user-friendly Netbeans IDE with the installed Pogamut plug-in.

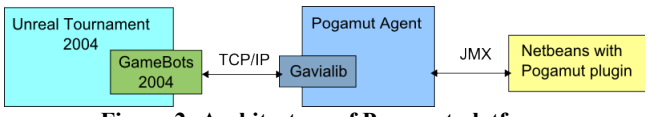

Figure 2: Architecture of Pogamut platform

The plug-in contains about 10 well commented example bots that can be used as a template for creating one's own bots. Documentation and online support on forums are provided via the homepage of the Pogamut project [5]. Besides introducing the theoretical concepts for controlling bots, an example bot – a hunter bot – will be programmed during the tutorial. The hunter bot's abilities include walking, object avoidance, following, objects' collecting and weapon selection behaviour. All these behaviours are encoded in a collection of IFTHEN rules.

Pogamut is still being actively developed. We are working both on extensing the core functionality as well as refining applications built on top of the Pogamut platform. New core extensions include the binding for the ALMA emotional model [3] and the programming language StorySpeak [4] designed for coordination of multiple agents when programming simple storytelling applications. The ALMA model and how it can be used for modifying bot's behaviour will be introduced in the presented tutorial.

4. CONCLUSION

The purpose of this paper was to give a short overview of the problems concearned with bots' development and to present the Pogamut platform - a software tool designed for easy bot prototyping. Both the theoretical background and the Pogamut platform will be discussed more thoroughly in the tutorial session. The attendees will also have the opportunity to install Pogamut on their notebooks and try to create their own bots during the session.

5. ACKNOWLEDGMENTS

This work was supported by the grant GAUK no. 21809. It was also 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.

6. REFERENCES

[1] Burkert, O., Kadlec, R., Gemrot, J., Bída, M., Havlíček, J., Dörfler, M., Brom, C. 2007. Towards fast prototyping of IVAs behavior: Pogamut 2 In: Proceedings of 7th International Conference on Inteligent Virtual Humans, LNCS Vol. 4722. Paris, France. Springer-Verlag, Berlin.

[2] Cooper, S., Dann, W., Pausch, R. 2003. Teaching Objects-first In Introductory Computer Science. In Proceedings of the SIGCSE technical symposium on Computer science education (Reno, Nevada, USA). ACM New York, NY, USA. 191-195. DOI=http://doi.acm.org/10.1145/611892.611966

[3] Gebhard, P. 2005. ALMA - A Layered Model of Affect. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05), 29-36, Utrecht. URL: http://www.dfki.de/~gebhard/alma/index.html [5.6.2009]

[4] Gemrot. J. 2009. Joint behaviour for virtual humans. Master thesis. Charles University in Prague

[5] Pogamut project homepage, URL: http://artemis.ms.mff.cuni.cz/pogamut [20.6.2009]

[6] Reynolds, C. 1999. Steering Behaviors for Autonomous Characters. In Game Developers Conference 1999