FlexBot, Groo, Patton and Hamlet : Research using Computer Games as a Platform

Aaron Khoo, Robin Hunicke, Greg Dunham, Nick Trienens and Muon Van

Computer Science Department, Northwestern University
1890 Maple Avenue
Evanston, IL 60201
{khoo, hunicke}@cs.northwestern.edu, {gdunham, n-trienens, van}@northwestern.edu
Fax : (847) 491-5258

Abstract

This paper describes four systems we intend to demonstrate at the AAAI-02 Conference. The first system is FlexBot – a software agent research platform built using the Half-Life game engine. The remaining three systems are research applications that were developed on top of the FlexBot architecture:

- Groo – an efficient bot constructed using behavior-based techniques.
- Patton – a system for monitoring and controlling bots through remote, possibly mobile, devices.
- Hamlet – the first part of a system for monitoring players and dynamically adjusting gameplay to promote dramatic/narrative immersion.

This demonstration is designed to show FlexBot in action and to exhibit the flexibility, efficiency and overall ease with which the FlexBot architecture supports a variety of AI research tasks.

FlexBot

In the last few years, there has been a surge of interest in using games as platforms for software agent research (Laird and van Lent 00). This is not surprising since computer games provide a rich, dynamic environment for AI research and experimentation. We were similarly interested in using computer games as a test-bed for our work on cooperative robotics and software agents. To this end, we constructed an architecture, codenamed FlexBot, that was designed to be flexible and efficient. We chose Half-Life (a popular first-person shooter game developed by Valve Studios) as a foundation for the work. The engine's open-source status provided an accessibility of code and a broad online community of veteran programmers/mentors, making it an obvious choice.

Our first step was to develop an NPC or 'bot' SDK for the Half-Life game. This SDK, written in C++, provides a 'fake client' interface for creating NPCs. The interface includes a set of sensors and actuators for programming bots, which reside in a DLL that talks to the Half-Life engine. FlexBot coders are responsible for writing control programs – separate DLLs that talk to the FlexBot interface DLL. This SDK was used to create Groo, our resident NPC, which is described below.

After the completion of Groo, we decided to add a set of development tools to FlexBot. Again, each tool is stored in a separate DLL:

- FlexDebug – Outputs debugging messages
- FlexStat – Provides statistical in-game information
- FlexMonitor – Provides player/NPC information
- FlexControl – Sends remote commands to NPCs

These development tools are being used extensively in both the Patton and Hamlet projects, described below.

In addition to its role as a development platform for independent research, FlexBot was used as a simulator in the Behavior-Based Robotics class at Northwestern University. About thirty students used the system to successfully construct their own bots, which then competed in a head-to-head tournament run on a single CPU. Not once during the tournament did the system crash.

Groo

Groo is a recent bot implementation that uses behavior-based techniques (Khoo et al 02). In their purest form, behavior-based systems divide sensing, modeling, and control between many parallel task-achieving modules called behaviors. Each behavior contains its own task-specific sensing, modeling, and control processes. Because behaviors are usually simple enough to implement as feed-forward circuits or simple finite-state machines, they can completely recompute sensor, model, and control decisions from moment to moment, responding immediately to changes in the environment. Thus, behavior-based bots are particularly suited to dealing with dynamic, complex, real-time environments.

In our system, Groo bots run concurrently with the Half-Life game server on the same physical machine. They are both efficient and stable. We have successfully run up to 31 bots on a server; the game engine itself placed a limited of 32 players in a game total, and the final spot had to be reserved for the sole human player.

While we have yet to run empirical studies similar to (Laird and Duchi 00), but anecdotal evidence indicates that our system has succeeded in presenting bots with realistic adversarial behavior. The Ledgewalker and Groo bots have been playtested at Northwestern by people familiar with first-person shooters, and during demonstrations at IJCAI-2001. In general, the reaction was positive, and most players felt that the bots exhibited behaviors associated with human death-match players.

Patton

Patton is an effort to develop a system for controlling and monitoring FlexBot from a remote device. The idea here is that the user is now a commander in charge of dispensing team- into the system at appropriate times.

At present, Patton augments the development tools provided by FlexBot with four new HTML-driven components:

- StatServer –Displays a set of in-game statistics, such as number of enemy kills, friendly kills, suicides, etc. Statistics can be sorted by individual bots, team or behavior type.
- BigBrother – Provides the user with an overhead 2D view of the current map, with player positions updated in real-time. Information is available by simply clicking on a particular player.
- RemoteConsole – Allows the commander to create/remove bots in Half-Life, set their skill levels, and switch their behaviors dynamically.
- VirtualJoystick – Allows users to directly control the movements of a bot in the game. Although primitive, this proof of concept shows that we can command the bots from a remote device.

Because Patton is entirely HTML-driven, any browser-enabled device will be able to utilize Patton's functions. We have successfully used Patton on a number of mobile devices as well as desktops, and continue to develop and refine the system. Our goal is an integrated demonstration of high-level control, in which a commander can make tactical-level decisions for bots to carry out autonomously.

Hamlet

Hamlet is a decision-theoretic system designed to manage the flow of gameplay experiences by supplying "just-in-time" aid to a player. Using techniques drawn from probability and utility theory, the Hamlet system examines the player's inventory and strategically places aid (health, weapons, armor and ammo) within reach at critical points during the action.

Our aim is to create an environment that is both responsive and responsible: a game should never be so difficult that it becomes impossible, nor should it be so easy that it becomes boring. This means lowering player frustration (needless hunting for ammo or health, repeated death at the same point) while maintaining the dramatic tension of the game (the player's experience of challenge, struggle and triumph at the controls).

Many commercial games already have some type of difficulty adjustment – most often the traditional "easy/medium/hard" setup options. Our system is designed to explore the advantages and disadvantages of different techniques for representing and reasoning about uncertainty in game environments, to see how these approaches can be extended and combined to create flexible, interactive experiences that adjust on the fly.

In its current form, the Hamlet system is responsible for altering only the physical interactions of a game environment. Future work will extend the system's capabilities for managing the strategic and narrative elements of gameplay, such as the placement of enemies, obstacles and information within the game world. Our ultimate goal is a system that combines local and global changes to create a responsive game environment where every action has a measured and dramatic consequence.

Reference

R.C. Arkin(1998) Behavior-based Robotics. MIT Press. Cambridge, MA.

A. Khoo, G. Dunham , N. Trienens , S. Sood (2002) *Efficient, Realistic NPC Control Systems using Behavior-Based Techniques*. To appear in 2002 AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment.

J. Bates(1994) The Role of Emotion in Believable Agents. Communications of the ACM, vol. 37, no. 7, pp. 122-125

J.E. Laird and J.C. Duchi(2000) Creating Human-Like Synthetic Characters with Multiple Skill Levels: A Case Study Using the Soar Quakebot AAAI 2000 Fall Symposium Series : Simulating Human Agents, November 2000 : AAAI Technical Report FS-00-03

J.E. Laird and M. van Lent(2000) Human-level AI's Killer Application : Interactive Computer Games. AAAI Fall Symposium Technical Report, North Falmouth, Massachusetts, 2000, 80-97.