AI Diagnostic tools

Colin Bruneau
CREAJEUX

"Building an AI Diagnostic Toolset"
Paul Tozour, Ion Storm Austin

Introduction

"Countless times when developing AI,
you will find yourself staring at the screen,
shaking your head, wondering how on earth your
character ended up doing the insanely stupid thing
it just did..."

Cracking open the cranium

- Behavioral flaws are inevitable
- Programmers often try to guess what went wrong...
- There is a need to diagnose and correct as quickly and easily as possible
- Not just spend time staring at the system!

AI tools/debugger

- AI tools are NOT a substitute for a debugger
- A debugger cannot replace good diagnostics!
- A debugger gives depth but not breadth
  - view the entire state of the system at a time
  - not good at showing changes over time
- Watching a debugger stack and variables is not intuitive
- Need AI tools for Testers and Level Designers

Building flexible diagnostics

- A diagnostic apply to a set of AI entities:
  - all AIs in the game
  - all AIs of an alignment
  - specially tagged AIs
  - selected AIs
  - nearest AI
- Need a way to specify the set of AI to diagnose
  - debug user interface

User interface

- The best user interface is an extensible in-game menu system
  - more intuitive than console commands
  - more interactive than log files
- 2 categories of AI tools:
  - **Commands** to change the state of the system
  - **Diagnostics** to view any part of the system without changing it

AI commands

- Destroy: destroy AI-controlled units
- Invulnerable: Make units invulnerable
- Stop movement: Make units unable to move
- Freeze: Halt the units AI
- Blind: Disable all visual input
- Deaf: Disable all audio input
- Insensate: Disable all sensory input

AI commands

- Duplicate: Clone the units
- Forget: make units lose all knowledge
- Reset: reset units and return to starting state
- Set state: set the current state of units
- Set target: set the units target to an entity
- Change game speed: up, down or pause
- Teleport to location: move the camera to a location

AI commands

- Teleport to AI: teleport to a specific unit
- Follow AI: set the camera to follow an unit
- Switch player control: take control of another player or AI
- Spawn objects: dynamically pop objects to use or to give to an unit
- other commands specific to your game...

AI diagnostic tools

- implementation of diagnostic tools saves precious time on testing, tweaking and debugging
- Simple tools required:
  - 3D line-drawing primitive
  - display of arbitrary text on-screen
- All diagnostic tool should be independent
  - Make a list of activated diagnostic tools
- Should be available in-game

AI diagnostics

- Unit identification: display the text name and/or ID of units
- Unit statistics: display the type or species of units, hit points, armor level, coordinates, inventory, weapons, skills, etc.
- Unit AI state: display the state of units.
- View search space: display the grid, nodes, navmesh, etc.

AI diagnostics

- View pathfinding search: display open and closed list, obstacles, cost values, and provide a "step by step" option
- View computed movement path: display a set of connected lines the units follows
- View tactical info: waypoints and cover points
- View past locations: display a set of connected lines the units have visited in the past
- View current target: draws an arrow to the target

AI diagnostics

- View patrol paths: display lines for patrol paths
- View formation leader: draw an arrow from each unit to its leader
- View sensory knowledge: draw a colored line to any stimulus (visual, audio, tactile)
- View animation and audio commands: display "play animation" and "play sound" commands
- View current animations
- View player commands: display on top of the units the commands issued by player

AI diagnostics

- Show strategic and tactical data structures: display influence maps, functional asset trees, dependency graphs
- Show fire arcs considered: display lines of shooting considered and lines blocked
- Show tracers: display the actual path of fast-moving projectiles

Conclusion

- "When a sink breaks, a plumber would open the cabinets and look at the pipes"
- "Game programmers prefer to stand around and hypothesize about why the sink might have broken, tinker endlessly with the faucet, and occasionally convince themselves that the sink is hopeless and install a new one"
- Don't do that!