

# KINGDOM: Adaptive Non-Player Characters & Dynamic Quest Generation

*Project Designers:* Spencer Miller, Adam Porroni, Sriraman Subbaraman

*Faculty Advisor:* Dr. Norman Badler

## Abstract:

the project breaks the game AI paradigm of non-emergent systems and rote quest generation by **creating a dynamic AI system** that adapts to player and non-player changes; we call this environment of these dynamic non-player characters, DNPCs, **Kingdom**.

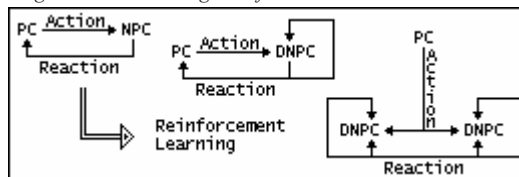
## Previous Systems:

- strategy specific – **RTS** – limited goal states and decision schemes: NPC-PC interaction determined by strategic hunter-seeker combat.
- tactics specific – **FPS** – limited goal states: NPC-PC interaction determined by PC initiated hunter-seeker combat.
- quest specific – **RPG** – all change incumbent upon PC or hardcore; system is predictable and results in rote tasks/quests.

## Major Objective:

- **Fundamentally change** the age-old paradigm of game AI, systems where the input wholly determines NPC state and consequent actions.
- Accomplish this goal in an inherently deterministic system using **emergent behavior** schemes, where **interactions** affect DNPC goals as well as subsequent DNPC-to-DNPC or DNPC-to-PC interactions.

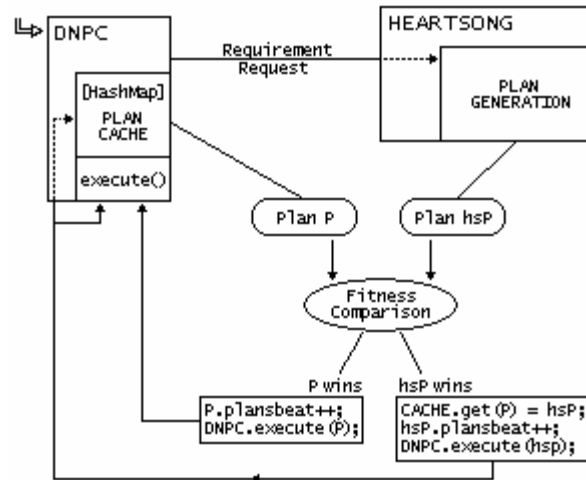
Figure 1: The Paradigm Shift:



## Project Thesis:

- Adopt an AI standard **belief-desire-intent (BDI)** model, but augment it with a **plan-action-resolution** scheme of our own making. The resultant system will ensure behavior rooted in DNPC-to-DNPC or DNPC-to-PC interactions.

Figure 2: A Part of the Plan-Action-Resolution system:



## Early Methods:

- Breaking the project into phases, we began with a basic needs simulation. This involved **low-level requirement-response design** where DNPCs would have a set of living requirements that they must keep within a certain range of their neutral state. If they failed, they would be penalized with a health reduction.
- From there we added extra levels of DNPC interaction, namely an ability to “call out” to other DNPCs; this action-set formed the **beginning of our quest generation system**.

## Evolved Methods:

- The next phases began with a **desire/driver scheme**, which allowed us to **model human-like goals and choices** for DNPC actions.
- This rubric subsequently **tied weights to actions** themselves, which allowed us to combine principles learned from a **Rule-Ordering with Bonus algorithm** proposed by Pieter Spronck (2007); thereby adding the element of proven dynamic ordering of NPC actions.
- These smaller implementations **have since simplified** what used to be a rather complex AI decision-scheme.

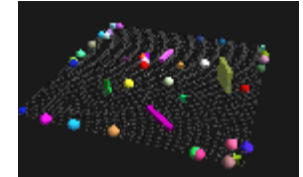
## Visualized Product:

- Initially designed within the *Neverwinter Nights 2* toolset framework, our project **encountered many obstacles**, namely in data structure implementation and manipulation in the toolset.
- We abandoned *NWN 2* and reprogrammed the entire project. For the final environment, we used a **Java-based graphical environment designed by Spencer Miller**. Less flash, but more effective.

*Neverwinter Nights 2*



*New Environment*



## Results:

- Our basic needs algorithm produced excellent results; DNPCs interact to successfully sate their needs, and they did so **with human-like rhythm**.
- The final system has since proven effective, but no individual-DNPC-level emergent behaviors have been observed.

## Conclusions:

- Pre-existing systems do not provide the modularity necessary to attach artificial intelligence modules. Design the AI system separately, but link it to basic systems, such as a graphics environment, in order to **preserve control over the essentials**.
- Design, design, design, and then code. Even simple emergent systems are hard to keep in check.
- It is relatively simple to get an artificial intelligence to obey a simple model of human behavior. But a more complex model makes for an exponentially more complex artificial intelligence module, especially when aiming for an emergent, non-probabilistic-predictable system. Adding a **desire/driver scheme** to our DNPCs **supplemented by basic needs** added more than a single layer of complexity. With this layer, higher level actions were simulated.

