12 tips on AI

Colin Bruneau
CREAJEUX

From Jeff Orkin
Monolith Productions

1. Do your homework

- The emphasis of the AI in the game
  - What are the needs in AI? the focus? steps?
- The schedule and budget for engineering AI
  - How much time?
  - How much money?
- The team make-up
  - How many programmers and designers?
  - What experience level?

2. K.I.S.S plan

- Keep It Simple Stupid!
- Avoid complex and out of control
- Ideal: do complex AI with simple parts
- Easy to comprehend, reuse, debug and maintain
- Simplify game design evolution
- Ex: Instead of Idle and Attack, Idle, GoTo, FireWeapon, Retreat, CallForHelp, etc.

3. Try it out on paper first

- Do not jump into code right away!
- Outline of the code
- Rough draft of sample data files
- Sketches of scenario (prog art)
- Review with designers
- Keep scenarios as documentation and tutos
- Update them frequently

4. Precompute navigation

- 3D navigation is expensive!
- Precompute pathfinding data
- Auto-generate or place hint information:
  - Paint floor geometry for blocked or prefered areas
  - Place preset paths for agents to follow
  - Place boxes with clear line of sight to another
  - Define areas with no collisions
  - Use a navigational-mesh generator on polygons

5. Put the smarts in the World, not in the AI

- Agent code can grow big!
- Write simple AI system that
  - chooses a destination
  - navigate to destination
  - follows instructions (messages or script)
- Ex: Agent is hungry
  - go to nearest edible object (refrigerator)
  - refrigerator tells the agent to play open door
- Makes the AI infinitely extensible (The Sims)

6. Give every action a Timeout and a Fallback

- An agent can be wrong but not repeat the same mistake forever
- Check success conditions within time t
- If conditions not met, give up and:
  - Fallback to Idle animation (confusion)
  - Revaluate situation if not too expensive

7. Use a hierarchy of states

- Putting states into a hierarchy facilitates the reuse of simple lower-level states
  - Lower-level states: specifics (animations, sound)
  - Higher-level states: decision and planning
- Ex:
  - Sub-states: Move and Attack
  - Parent states: Attack in level 1 or boss level

8. No agent interference with Story Events

- The player must not miss the story
  - Conversing
  - Listening to a dialog
  - Solving a puzzle
- Agents should back-off to get out of the way
- Solutions:
  - Non-interactive story sequences
  - Agents should be aware of story events

9. Keeps agents aware of global world state

- Agents should remember what has happened to them and to others
- Change their behavior and dialog accordingly
- Solutions:
  - Global flags and player's progress data
  - Reputation system (agent communication)
  - Decision-tree learning

10. Create variety through the data, not through the code

- Variety requires many behaviors
- Programming a behavior takes time
- Code only a few behavior types that are in-finitely customizable through data
- Expose as many variables as possible
  - velocity, awareness, fov, states, inventory, etc.
- Provide good defaults values
- Good documentation and tutorials

11. Make the data easily accessible to designers

- Interesting AI requires experimentation
- Designers should tweak values:
  - statistics
  - formulas
- A user interface is a better tool than text files for designers!

12. Factor stat formulas into AI

- Agents abilities are defined by statistics
- Ex: in RPG, Strength, Agility, Int., Magic, etc.
- Extend the concept to every aspect
  - how fast it travels
  - how fast it animates
  - how intelligently it navigates
  - what attack and defense it chooses
  - the size of spells it casts