DM810

Computer Game Programming II: AI

Lecture 1
Introduction

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Intro to Course

- Motivation
- Contents of course
- Formalities of course
- Textbook
- Tentative course plan
- Intro to Game AI

Outline

1. Course Introduction
2. Game AI

Prerequisites

- Programming (DM502+DM503)
- Algorithms and data structures (DM507)
- Software engineering, testing (known)
- flexibility to work with several programming languages (C++, java, python)
- skills in integrating code with pre-existing engines
- 3D computer game rendering is an asset (DM809)

Schedule

- Schedule (28 lecture hours):
  - Monday, 16:15-18:00
  - Tuesday, 16:15-18:00
  - Wednesday, 16.15-18:00
  - Last lecture: Wednesday, December 19, 2012

All lectures in U64.
No examinatorier (programming assignments take up the time).

- Communication tools
  - Course Public Webpage (WWW) $\Leftrightarrow$ BlackBoard (BB)
    (link from http://www.imada.sdu.dk/~marco/DM810/)
    (you can comment there)
  - Announcements in BlackBoard
  - Personal email
- Text book:
  Artificial Intelligence for Games by Ian Millington Published by Morgan Kaufmann, 2006 ISBN 978-0-12-497782-2

On the Book

Book structure:

- Ch. 1-2: Intro to Game AI
- Ch. 3-8: Game AI techniques
- Ch. 9-11: Surrounding issues (AI execution scheduling, gameworld interfacing, tools and content creation).
- Ch. 12-13: Game AI technology choices by game genre.

Total: 825 pages...

The textbook is comprehensive, structured, and oriented towards real-life Game AI practice. Author is both well educated in AI, and has 20+ years experience as AI programmer and designer in game industry.

On the Course

- Bunch of different techniques
- Often a bit limited in depth
- Quite a number of pages to read
- Includes programming

Course Plan

Course plan for Computer Game Programming II:

See web page

Evaluation

- 5 ECTS
- Course language: Danish and English
- Obligatory Assignments, pass/fail, evaluation by teacher (3 handins)
- Evaluation: Oral exam, 7-grade scale, external censor
    - not based on assignments
    - without preparation
    - portfolio of techniques approved aforehead
    - draw two topics from the portfolio
    - present each topic in 10 minutes
    - we may ask questions related to the topic
    - possible questions on the rest of the curriculum.
    - 20-25 minutes

Assignments

- Small projects (in groups of 2) must be passed to attend the oral exam:
- Try out at least two AI techniques (in different areas).
- Possibly two controlled assignments on AI competitions
- Final assignment is free. It may be:
  - example programs,
  - continuation of previous project from DM809
  - continuation of one of previous assignments

  Must contain two AI techniques.
  Must run without problems on either Imada machines (Linux), or on Windows XP or Vista.

To Decide Together

- Assignments
- Date of exam
- Schedule, twice per week, which ones?
- Who took DM828 Intro to Artificial Intelligence?
- Programming skills? Environment?
  Linux, MacOsX, Windows XP or Vista

Outline

Classic AI

AI = make machines behave like human beings when solving “fuzzy problems”.

Classic AI:

- Philosophical motivation: What is intelligence, what is thinking and decision making?
- Psychological motivation: how does the brain work?
- Engineering motivation: make machine carry out such tasks.

Game AI: Related to third point. Happy to draw on results from AI research, but goal is solely behavior generation in computer games.

Academic AI Eras

- Prehistoric era (-1950): Philosophic questions, mechanical novelty gadgets, what produces thought?
- Symbolic era (1950-1985): symbolic representations of knowledge + reasoning (search) algorithms working on symbols.
  Examples: expert systems, decision trees, state machines, path finding, steering.
- Natural era (1985-): techniques inspired by biological processes.
  Examples: neural networks, genetic algorithms, simulated annealing, ant colony optimization.

Author: natural techniques currently more fashionable, but not more successful than symbolic. Game AI techniques often are symbolic.

No AI technique works for everything ("knowledge vs. search trade-off"). In games, simple is often good.

AI approach

Bottom up approach:
Agent-based models with emergent behavior.
Movement and individual decision making are the two basic elements.
(eg: computational intelligence, swarm intelligence)

Top down approach:
non agent-based models in which everything is simulated and optimized and then single character's actions are decided.

In games more ad hoc techniques. Agents are called game characters

AI in Games, examples

- Pacman (1979) first game using AI
  - Opponent controllers, enemy characters, computer controlled char.
  - used a finite state machine
- Warcraft 1994
  - path finding, robust formation motion, emotional models, personality
- The Sims 2000, Creatures 1997
  - neural network brain for creatures
- Present:
  - Simple AI
  - Bots in first person games and simulators, advanced AI
  - Real-time strategy (RTS) games, advanced AI
  - Sport and driving games still pose challenges (dynamic path finding)
  - Role-playing games (RPG) conversation still challenging

- Actions:
  attacking, standing still, hiding, exploring, patrolling, ...
- Movement:
  - going to the player before the attack
  - avoiding obstacles on the way
  - a lot done by animation but still need to decide what to do. Eg: eating animation
- Decision making: deciding which action at each moment of the game. (then movement AI + animation technology)
- Strategy:
  coordinate a team while still leaving to each individual its own decision making and movement

Needs for AI in Games

Three main areas:

- Movement (single characters)
- Decision making (where? short term, single character behavior)
- Strategic AI (long term, group behavior)

Not all games have all areas (eg. chess vs. platform game).

Associated issues:

- Gameworld interface (input to AI)
- Execution/scheduling of AI
- Scripting, content creation
- Animation ($\neq$ movement) and Physics (not in book)

The complexity fallacy

Fallacy: More complex AI gives more convincing behavior.

Often, the right simple technique (or combination of simple techniques) looks good.

Complex, intricate techniques often look bad.

"The best AI programmer are those who can use a very simple technique to give the illusion of complexity."

The Perception Window

Most non-player characters (NPC) are met briefly. Adapt AI complexity to players exposure to the character. Advanced AI will look random (so simply use randomization).

Change of behavior is very noticeable (more than behavior itself), and should correspond to relevant events (like being seen).

Algorithms, Hacks and Heuristics

This course (and the book) focuses on general techniques and algorithms for generating behavior and representations for interfacing.

However, real Game AI often employs ad hoc hacks and heuristics.

In game, perceived behavior, not underlying technique, is what matters.

In particular, we do not study the principles behind human behavior (as academic AI does), but tries to emulate it sufficiently well. And if an ad hoc method or simple emotional animation will do it, then fine. Behaviorist approach.

Efficiency

AI is done on CPU.
Trend: more tasks taken over by GPU, hence more time left for AI in CPU.
Could be 10-50% of cycles.

Heavy AI calculations can be scheduled over many frames.

Parallelism: easiest when there are several NPCs. SIMD and Multi-core possibilities.

Branch-prediction and, above all, cache-efficiency may impact considerably performance

PC development: should run on varied hardware. Often hard to implement AI that scales with changing hardware (without impacting gameplay). So developers target AI to the minimum hardware requirements. Scaling may be done by reducing number of NPCs.

Console development: development platform is often PCs, so many small tweaks during development is harder.

The AI Engine

- Reuse of code saves programming time.
- Content creation takes up the bulk of a games development time. Tools for this is necessary.

$\rightsquigarrow$ For AI (as for graphics), generic (in-house) engines are now common.

$\rightsquigarrow$ interfacing

$\rightsquigarrow$ infrastructure, action, support technology

(Example, CD code, python game)

For this, AI knowledge representation forms needs to be decided upon and put into level editing tools.