DM810

Computer Game Programming II: AI

Rolf Fagerberg

Fall 2008

Goals for Todays Lecture

Introduction to course:

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

Why Computer Game Programming?

- Fun, attraction, curiosity
- Career goal (in US, game industry twice as large as movie industry in sales)
- Great display of use of many Computer Science subjects and courses:
  - Programming (DM502, DM503, DM528)
  - Algorithms and data structures (DM507)
  - Linear algebra and other math (DM527, MM501, MM502, MM505)
  - Numerical analysis (MM518)
  - AI (eg. finite automatons from DM517)
  - Computer architecture (DM506)

Computer Game Development

- Large game company (100 persons, 20 M$/year turnover):
  - Game programmers: 30-40
  - Game artists, model designers: 30-40
  - Game level designers, testers: 10-30
  - Game designers: 2
  - Game producers: 3
  - Business and management persons: 5
- Casual game company (1-3 persons, ?? $/year turnover):
  - Each person has many roles.

Computer games in Computer Science: the study of

Methods and principles of game programming

Course Sequence at Imada

Fall 2008, 3rd quarter:

Computer Game Programming I: Graphics

Fall 2008, 4rd quarter:

Computer Game Programming II: AI

Spring 2009, 1st quarter:

Computer Game Programming III: Physics

Spring 2009, 2nd quarter:

Computer Game Programming IV: Project

(Other possibilities: DADIU, bachelorproject.)

Subjects covered in course sequence

- GPU pipeline [Course I]
- 3D geometry (transformation, projection) [Course I]
- Rendering (color, textures, lighting and shading) [Course I]
- Acceleration techniques (culling, level of detail, spatial data structures) [Course I]
- Image based techniques (skyboxes, billboards,. . .) [Course I]
- Game AI (path finding, chasing and evading, fighting, flocking,. . .) [Course II]
- Collision detection [Course III]
- Physics modeling [Course III]

ubjects not covered in course sequenc

- Graphics APIs (self-study)
- Software engineering, testing (known)
- Game engines (a bit in AI course)
- Level editors, scripting (a bit in AI course)
- Modeling
- Animation
- Sound, music
- Gameplay, narrative, study of genres

Course II Textbook

Artificial Intelligence for Games
By Ian Millington
Published by Morgan Kaufmann, 2006
ISBN 978-0-12-497782-2

On the Textbook

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 contents 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.

Course Plan

Course plan for Computer Game Programming II:

Subject	Lectures	Chapters
Introduction	1	1,2
Movement	3	3
Pathfinding	2	4
Decision making	2	5
Tactical and strategic AI	1	6
Learning	2	7
Board game AI	1	8
Supporting technologies	2	9–11

If time allows, AI techniques by game genre (Ch. 12 and 13) could be included.

Formal Course Description

Prerequisites:	Programming (DM502+DM503), algorithms and data structures (DM507), 3D computer game rendering is an asset (DM809)
Literature:	Textbook
Evaluation:	Implementation projects (pass/fail), oral exam (7-scale)
Credits:	5 ECTS
Course language:	Danish or English

Time and Place

- Mondays 12.15-14.00
- Fridays 12.15-14.00

All lectures in Imadas seminar room.

No examinatorier (programming projects take up the time).

Project

Small project (in groups of 2–3) must be passed to attend the oral exam:

Try out at least two AI techniques (in different areas).

May be example programs, or a continuation of previous project from DM809 (or combination, if only one AI technique is relevant for previous project).

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

On the Course

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

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 behaviour generation in computer games.

Academic AI Eras

- Prehistoric era (-1950): Philosophic questions, mechanical novelty gadgets.
- Symbolic era (1950-1985): symbolic representations of knowledge, reasoning 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 succesful 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.

Model of Game AI

Three main areas:

- Strategic AI (long term, group behaviour)
- Decision making (short term, single character behaviour)
- Movement (single character)

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 behaviour.

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 NPC (non-player characters) 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 behaviour itself), and should correspond to relevant events (like being seen).

Hacks and Heuristics

Besides general techniques and algorithms (as focused on by the book), real Game AI often imploys ad hoc hacks and heuristics.

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

In particular, we do not study the principles behind human behaviour (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.

This course (and the book), however, of course has a focus on general techniques for generating behaviour.

Efficiency

AI is done on CPU.

More tasks taken over by GPU $\Rightarrow$ more time for AI. Could be 10-50% of cycles.

Heavy AI calculations can be scheduled over many frames.

Parallelism: easiest when there are several NPCs.

PC development: should run on varied hardware. Often hard to implement AI that scales with changing hardware (without impacting gameplay). So minimum hardware requirements of game set the AI standard.

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.

For AI (as for graphics), generic (in-house) engines are now common. For this, AI knowledge representation forms needs to be decided upon and put into level editing tools.

(See Figure 2.2. More details in Ch. 11.)