3.4 Game Programming

Aspects of traditional computer science and software engineering – modified to address the technical aspects of gaming.This Core Topic includes physics, mathematics, programming techniques, algorithm design, game-specific programming and the technical aspects of game testing. Much of the material in this area could be taught under the auspices of a traditional computer science or software engineering curriculum. However, games do present a very specific set of programming challenges, such as optimization of mainstream algorithms such as path-finding and sorting, and real-time 3D rendering, that are addressed here.
+ = well covered; o = satisfactorily covered; - = not well covered.

1. Math and Science techniques +
   (a) Basic Newtonian physics
   (b) Computational mechanics
   (c) Probability and statistics
   (d) Geometry, discrete math and linear algebra
   (e) Vectors and Matrices
   (f) Coordinate spaces and transformations
   (g) Collision Detection
   (h) Computational geometry
   (i) Basic calculus and differential equations
2. Style & design principles +
   (a) Coherency
   (b) Object oriented programming paradigms
   (c) Design patterns
   (d) Game design patterns
3. Information design o
   (a) Data structures – data architecture, file formats, data organization, data compression
   (b) Asset pipelining
   (c) Computational geometry
   (d) Environmental models, spatial data structures
   (e) Database
4. Machine Architecture +
   (a) Optimization (CPU and GPU)
   (b) Embedded System Development
   (c) Configuration Control and Source Control Systems
   (d) Software Architecture
   (e) Software Engineering
5. Game Engine Design +
   (a) Purpose and importance
   (b) Architecture and design
   (c) Data Pipelines
   (d) Methodologies and practices to create stand-alone gaming applications,
   (e) Limitations of implementing cross-platform technology
   (f) Generic and universal issues in programming for 3D engines
   (g) Graphics libraries and 3D hardware issues
   (h) Programming object and camera motions
   (i) Collision detection and collision response
   (j) Performance analysis
   (k) Special effects
6. Prototyping +
   (a) Tools and skills for fast, iterative development
   (b) Building flexible systems, configurable by others

7. Programming teams -- structure and working relationships +
    (a) Working in interdisciplinary teams
    (b) Talking with programmers/artists/designers/producers/etc.
    (c) Team programming processes and methodologies
8. Design/Technology synthesis -
    (a) Supporting player goals and actions
    (b) Building intelligent, coherent, consistent, reactive game environments
    (c) Platform issues
9. System architecture for real time game environments and simulations +
    (a) Concurrent programming techniques
    (b) Integration of sub systems (Physics, Collision detection, AI, Input, Render, Scripting)
    (c) Incorporating and extending third party systems in a game engine.
    (d) Resource budgeting (CPU, GPU, memory)
10. Computer Architecture +
    (a) Structure of a CPU with implications to program design (eg, avoiding branching)
    (b) The memory hierarchy with implications to program design (eg, alignment of data structures in memory, locality of reference)
    (c) Algorithm design considerations for CPU versus GPU implementation
11. Tools construction -
    (a) "Tool Development"
    (b) GUI creation
    (c) Tools for multimedia content creation, modification and management
    (d) Custom design tools
    (e) Building flexible systems for non-programmers to use
12. Graphics Programming +
    (a) Rendering
        i. Transforms, lighting, texturing
        ii. Clipping, occlusions, transparency
        iii. Level of detail considerations
        iv. Using data structures to optimise rendering time
    (b) Animation
        i. Forward and inverse kinematics
        ii. Transform representations
        iii. Interpolation techniques
        iv. Camera animation
    (c) Graphics System Design
    (d) Procedural content generation (Textures, Models, etc.)
13. Sound / Audio Programming -
    (a) Physics of sound and human hearing
    (b) Programming 3D positional sound
    (c) Utilizing Audio Channels
    (d) Audio Prioritization
14. Artificial intelligence o
    (a) Difference in goals between Game AI and traditional AI
    (b) Path planning, search algorithms
    (c) Agent architectures
    (d) Decision-making systems

   (e) State machine design
   (f) Statistical machine learning
15. Networks +
   (a) Networking and Server design
   (b) Performance metrics
   (c) Topologies
   (d) Protocols – TCP/IP, UDP, …
   (e) Security
   (f) Game Servers
   (g) Game Protocol Development
   (h) Available Network Libraries
   (i) Open Source Network Game Case Studies
16. Game logic +
   (a) Compilers
   (b) Scripting languages
17. Play analysis -
   (a) Play testing to monitor player frustration, progress and enjoyment
   (b) Monitoring player state -- gameplay data logging
   (c) Player metrics