

Game Programming Gems 1

par Matthieu Brucher (<http://matthieu-brucher.developpez.com/>) (Blog)

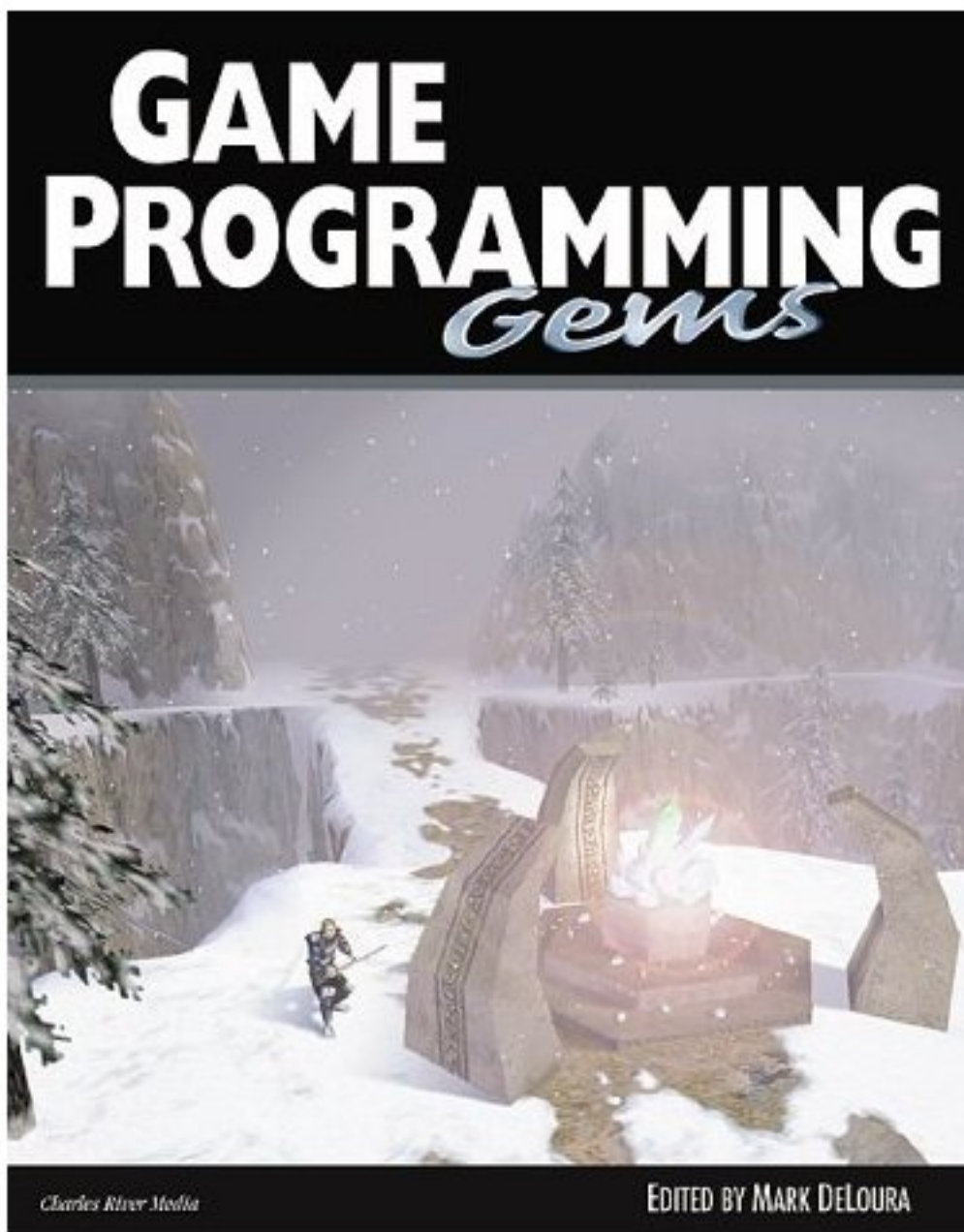
Date de publication : 10/05/2006

Dernière mise à jour : 02/12/2006

Critique de Game Programming Gems 1 édité par *Mark DeLoura*

- I - Description
- II - Table des matières
- III - Critique : Très utile
- IV - Liens annexes

I - Description



Pour chacune des nombreuses tâches impliquées dans la création d'un moteur de jeu, il y a un nombre égal de solutions possibles. Mais au lieu de passer des heures et des heures à développer ses propres réponses, vous pouvez maintenant savoir comment les pros font ! Game Programming Gems est une ressource compréhensible à portée de main incluant des algorithmes de jeu écrits par des experts de l'industrie du jeu et édité par Mark DeLoura, ancien chef du département logiciel pour Nintendo of America, Inc. et maintenant le nouvel éditeur en chef de Game Developer magazine.

Des animations et de l'intelligence artificiel au Z-buffering, calcul de lumières, effets climatiques, surfaces courbes, jeu sur Internet par multiples couches, musique et effets audio, toutes les techniques majeures pour développer un

moteur de jeu compétitif sont abordés. Game Programming Gems est écrit dans un style accessible aux individus d'un niveau expert. Tous les codes source de chaque algorithme sont inclus et peuvent être utilisés par des programmeurs avancés immédiatement. Pour les aspirants programmeurs, un tutoriel détaillé est fourni pour travailler avant d'utiliser le code, des suggestions pour de possibles modifications et optimisations sont aussi inclus.

II - Table des matières

- Section 1 Programming Techniques
 - 1.0 The Magic of Data-Driven Design
 - 1.1 Object-Oriented Programming and Design Techniques
 - 1.2 Fast Math Using Template Metaprogramming
 - 1.3 An Automatic Singleton Utility
 - 1.4 Using the STL in Game Programming
 - 1.5 A Generic Function-Binding Interface
 - 1.6 A Generic Handle-Based Resource Manager
 - 1.7 Resource and Memory Management
 - 1.8 Fast Data Load Trick
 - 1.9 Frame-Based Memory Allocation
 - 1.10 Simple, Fast Bit Arrays
 - 1.11 A Network Protocol for Online Games
 - 1.12 Squeezing More Out of Assert
 - 1.13 Stats: Real-Time Statistics and In-Game Debugging
 - 1.14 Real-Time In-Game Profiling
- Section 2 Mathematics
 - 2.0 Predictable Random Numbers
 - 2.1 Interpolation Methods
 - 2.2 Integrating the Equations of Rigid Body Motion
 - 2.3 Polynomial Approximations to Trigonometric Functions
 - 2.4 Using Implicit Euler Integration for Numerical Stability
 - 2.5 Wavelets: Theory and Compression
 - 2.6 Interactive Simulation of Water Surfaces
 - 2.7 Quaternions for Game Programming
 - 2.8 Matrix-Quaternion Conversions
 - 2.9 Interpolating Quaternions
 - 2.10 The Shortest Arc Quaternion
- Section 3 Artificial Intelligence
 - 3.0 Designing a General Robust AI Engine
 - 3.1 A Finite-State Machine Class
 - 3.2 Game Trees
 - 3.3 The Basics of A* for Path Planning
 - 3.4 A* Aesthetic Optimizations
 - 3.5 A* Speed Optimizations
 - 3.6 Simplified 3D Movement and Pathfinding Using Navigation Meshes
 - 3.7 Flocking: A Simple Technique for Simulating Group Behavior
 - 3.8 Fuzzy Logic for Video Games
 - 3.9 A Neural-Net Primer
- Section 4 Polygonol Techniques

- 4.0 Optimizing Vertex Submissions for OpenGL
- 4.1 Tweaking A Vertex's Projected Depth Value
- 4.2 The Vector Camera
- 4.3 Camera Control Techniques
- 4.4 A Fast Cylinder-Frustum Intersection Test
- 4.5 3D Collision Detection
- 4.6 Multi-Resolution Maps for Interaction Detection
- 4.7 Computing the Distance into a Sector
- 4.8 Object Occlusion Culling
- 4.9 Never Let 'Em See You Pop # Issues in Geometric Level of Detail Selection
- 4.10 Octree Construction
- 4.11 Loose Octrees
- 4.12 View-Independent Progressive Meshing
- 4.13 Interpolated 3D Keyframe Animation
- 4.14 A Fast and Simple Skinning Techniques
- 4.15 Filling the Gaps # Advanced Animation Using Stitching and Skinning
- 4.16 Real-Time Realistic Terrain Generation
- 4.17 Fractal Terrain Generation # Fault Formation
- 4.18 Fractal Terrain Generation # Midpoint Displacement
- 4.19 Fractal Terrain Generation # Particle Deposition
- Section 5 Pixel Effects
 - 5.0 2D Lens Flare
 - 5.1 Using 3D Hardware for 2D Sprite Effects
 - 5.2 Motif-Based Static Lighting
 - 5.3 Simulated Real-Time Lighting Using Vertex Color Interpolation
 - 5.4 Attenuation Maps
 - 5.5 Advanced Texturing Using Texture Coordinate Generation
 - 5.6 Hardware Bump Mapping
 - 5.7 Ground-Plane Shadows
 - 5.8 Real-Time Shadows on Complex Objects
 - 5.9 Improving Environment-Mapped Reflection Using Glossy Prefiltering and the Fresnel Term
 - 5.10 Convincing-Looking Glass for Games
 - 5.11 Refraction Mapping for Liquids in Containers
- Section 6 Appendix
 - 6.0 The Matrix Utility Library
 - 6.1 The Text Utility Library
 - 6.2 About the CD-ROM

III - Critique : Très utile

Avec ce seul livre, on ne construit pas un jeu. Ce n'est qu'un recueil bien incomplet de briques élémentaires, et encore, il en manque pas mal, il faut bien laisser un peu de travail !

C ou C++, le code fourni semble être standard. Malheureusement, chaque item n'est fourni que dans un seul langage, il faudra donc "traduire" certains bouts de code. Mais le principal est de comprendre ce qui est proposé, pas toujours très facile.

Une bonne moitié du livre est consacré au coeur d'un moteur de jeu, son intelligence et la gestion, que ce soit ressource, pathfinding, gestion des comportements, tandis que l'autre moitié est consacrée au graphisme. Ce qui se rapporte au son et au réseau est abordé dans d'autres GPG.

IV - Liens annexes

 [*Critique sur la page de livres Jeux*](#)

 [*Achat sur Amazon.fr*](#)

 [*Lien vers le site de l'éditeur*](#)

