Preface

The title of this book is "Algorithmic and Architectural Gaming Design: Implementation and Development" and it does an admirable job of conveying both its theme and its focus. The theme is the investigation of the basic components of the computer programs that are created to implement video games. The general term for these systems is “game engine”. As is the case with virtually all computer applications, algorithms of one sort or another form the heart and soul of a game engine’s functionality. Most game engines are complex systems because they incorporate all of the components necessary to develop complete video games. To the casual user video games may seem relatively simple, or at least their preoccupation with the game itself leaves them no time to ponder the astonishing amount of computation required (graphics, game play updating for every animate object in the game, sound, physics, artificial intelligence, data storage and retrieval, etc.) to provide realistic game play at speeds of around thirty frames a second. The only way to manage this combination of complexity and speed is through the use of algorithms to provide the functionality and game engine architecture to lend some kind of order to the design of the game engine used to develop the games.

An algorithm is defined in literature as a “step-by-step procedure for calculations. More precisely, it is an effective method expressed as a finite list of well-defined instructions for calculating a function.” In a practical sense, algorithms are ways of accomplishing a task that have been found to be effective and refined over many uses to improve efficiency and adapt to variations of a specific task. The tricky part is deciding which algorithm to use (there can be several established algorithms applicable to the same kind of task) and deciding exactly how they need to be adapted.

Within the context of this book, the term architecture refers to the software architecture of the game engine used to develop video games. A game engine’s architecture is a description of the conceptual model upon which it is based, its components along with their function and interaction, and usually some connection to lower level software systems (i.e. graphics libraries) or even hardware components (i.e. video cards).

While video game algorithms and software architecture are the theme of the book, their implementation and development are its focus. It has been our experience that many game development issues are discussed in general in the literature but seldom are sufficient details included to allow the interested reader to actually build the component in software. It is the goal of this book to, whenever possible, provide the reader with implementation details of the topics covered. To avoid problems caused by different languages, the implementations presented throughout the book are written in pseudo-code. The book also contains chapters dedicated to the architecture of game engines for those readers interested in understanding or perhaps even implementing a game engine of their own.

With the commercialization of video games in the late 1970s and early 1980s the race was on to provide users with the most realistic and satisfying gaming experience possible. In the relatively short
time span since then the design and development of video games and game development engines has become a major entertainment industry worldwide with revenues expected to exceed $112 billion dollars in 2015. Algorithms and game engines to speed development quickly became a hot topic and the potential source of a competitive edge in a growing video game market. Recently, there has been a resurgence of individuals and small programming teams developing casual games with shorter production times and less investment in both time and money. This, coupled with the availability of open source game engines has led to a newly revived interest in both game algorithms and game engine architectures. A book that focuses on the implementation details of developing video games is especially useful to a wide spectrum of people with an interest in this rapidly expanding field.

The target audience for this book consists of undergraduate and graduate students, researchers, and game professionals working in the area of video game design and implementation. Readers will benefit by learning about many core concepts and how to implement them. Some of the latest research and implementation work is covered in the book and will provide both researchers and practitioners an opportunity to learn and apply the new-found knowledge. The book also provides a rich source of bibliographical information for each chapter for further study which graduate students and researchers will find very useful.

The book is organized into five major sections labeled as Artificial Intelligence in Games, Game Physics, Collision Detection in Games, Game Models and Implementation, and Serious Games. The topics covered under the Artificial Intelligence in Games section include group formation and steering behavior for navigation, pathfinding, behavior trees, collective decision making, intelligent and dynamic adaptation of difficulty, application of AI algorithms, and evolving bots. The topics covered under the Game Physics section include a rigid body linear complementary (LCP) solver and implementation of rocket jump mechanics for side scrolling platform games. The topics covered under the Collision Detection section includes two detailed chapters explaining how algorithms can be implemented efficiently for detecting collisions in a game. The topics covered under Game Models and Implementation covers development of a software framework for designing multiplayer online games, illustration of modular design of a game engine, an algorithmic perspective on game play model, and an example of a game’s development. The topics covered under the Serious Games section include development of a music tutor game using tower defense strategy, and the development of a game for training navy personnel about movement of ships.

Chapter 1, entitled “Co-ordinating Formations: A Comparison of Methods”, focuses on group behavior in computer games, which is directly applicable to real-time strategy games as well as any situations that involve control and coordination of a group of characters or entities. The task of coordinating formations is divided into defining a control structure for the formation, finding a path for the formation, and steering individual entities within the formation. Modeling of the formation is done with six different control structures. Several strategies are discussed for moving the characters into a formation, and three decision making methods for coordinating a formation are discussed and implemented. For a rigorous testing of the approach, obstacles of different types and sizes are considered for pathfinding while maintaining a formation. Tests are run on the three proposed methods, called behavior-based steering, fuzzy logic control, and mass-spring systems and results are reported. It is found that the behavior based steering is easy to implement, efficient, and yields the best results.

Chapter 2, entitled “Adapting Pathfinding with Potential Energy”, extends traditional pathfinding by allowing the non-player characters (NPCs) to learn from the experience of traversing generated paths. This is accomplished by allowing waypoints to have either positive or negative energy based on events
that have happened to NPCs around that node. As a result, NPCs can adapt to the environment and can avoid or be attracted to parts of the game world as appropriate.

Chapter 3, entitled “Behavior Trees: Introduction and Memory-Compact Implementation”, addresses the decision making and control of NPCs through behavior trees. Behavior trees are a method of organizing behaviors in a tree structure such that appropriate actions are taken in a cost-effective way and possibly without incurring huge memory footprints. The chapter presents, in detail, the concept of behavior trees including behavior execution states, actions, conditions, deciders, priority and probability selectors, decorators, and concurrency and speed up issues. After introducing the concept, the authors also give a sample implementation that uses a small memory footprint.

In Chapter 4, entitled “Nonmanipulable Collective Decision-Making for Games”, the authors introduce a novel method for allowing human and/or non-player characters to find a group consensus. The power of their approach is that no participant can manipulate the results in his or her favor by voting insincerely. The chapter gives an algorithm and example for several types of scenarios. The approach is applicable to situations that require cooperation and consensus among players or NPCs. The issue of asynchronous decision making has also been addressed, which makes the approach useful for a variety of game scenarios.

In Chapter 5, entitled “Understanding and Implementing Adaptive Difficulty Adjustment in Video Games”, the authors address the issue of providing the game an opportunity to dynamically adjust the difficulty of the game as it is played. The work proposes a way to model, build, and validate a system for dynamic decision adjustment, and it provides a deep insight into how to create a video game with adaptive difficulty.

In Chapter 6, entitled “Application and Evaluation of Artificial Intelligence Algorithms for StarCraft”, the author shows how to build intelligent NPCs for StarCraft. First, the installation and setup of the Brood War API (BWAPI) is discussed. Then, algorithms to automate essential game tasks, such as producing workers and assigning them to harvest resources, are covered. Also included are gathering algorithms, applications of swarm intelligence, expert systems, and hill climbing AI, with examples of situations where each method can be useful.

In Chapter 7, entitled “Evolving Bots’ AI in Unreal™”, the authors propose and implement a system to control the behavior of bots by changing the default AI of the bot in Unreal™. A genetic algorithm based approach is used to change the default, hardcoded values that determine bots’ parameters, and genetic programming is applied to obtain new sets of rules that govern bots’ behavior.

Chapter 8, entitled “Practical Introduction to Rigid Body Linear Complementary Problem (LCP) Constraint Solvers”, is useful for implementing real-time physics in games. It explains the principles for LCP constraint solvers with practical examples and implementation details. It provides code for modeling a rigid body class, an iterative LCP solver, and adaptation details. Examples of constraints, how they are handled, and a variety of implementation methods are discussed. Also, stability and reliability issues are addressed.

Chapter 9, entitled “Rocket Jump Mechanics for Side Scrolling Platform Games”, takes a popular concept from first person shooter games and incorporates it into a two dimensional platform game. It discusses the mechanics of rocket launching, static and timed obstacles, and rhythm based implementation. Next, it compares the results of their rocket jump implementation with that of conventional jump, and it also notes the importance of player perception, experience, and rhythm.

Chapter 10, entitled “Collision Detection in Video Games”, explains collision detection in 3D environments. After a brief review of vector math, the chapter goes into the details of three collision detection
methods: axially-aligned bounding boxes (AABBs), Object-Oriented Bounding Boxes (OOBs), and the Gilbert-Johnson-Keerthi distance algorithm (GJK). Each section gives a sample implementation of the algorithm, as well as methods for optimization by removing unnecessary intersection tests.

Chapter 11, entitled “Collision Detection Using the GJK Algorithm”, provides an in-depth tutorial of the famous algorithm. The chapter begins with an introduction to the basic principles needed to understand GJK, determining if a given shape is convex and calculating the Minowski sum and difference of two shapes. Each section of the algorithm is further explained with easy-to-follow examples. After the explanation of GJK, the author provides examples where the algorithm is useful. Finally, an extension to GJK, known as the Expanding Polytope Algorithm, that provides additional information used to resolve collisions is explained in detail.

Chapter 12, entitled “Designing Multiplayer Online Games Using the Real-Time Framework”, describes a framework for designing and executing online computer games using a new software platform developed by the authors. Major design issues including scalability and single server vs. multi-server applications are considered. The usefulness of the approach is shown with the design of a new multiplayer online game and the extension of the single server commercial game Quake 3 to multiple servers.

Chapter 13, entitled “Modular Game Engine Design”, provides a tutorial on using software engineering principles in creating a game engine. Specifically, the use of managers as classes that oversee objects with similar functions is emphasized. The authors use a simple tower defense game to demonstrate the implementation of managers to handle game states, game objects, and other game components.

Chapter 14, entitled “A Gameplay Model for Understanding and Designing Games”, takes a scientific view of gameplay as a combination of challenges and methods. Good gameplay is viewed as the creation of enjoyable tension between pressure from a challenge and understanding how to deal with it using the methods at a player’s disposal. A collection of challenges and methods are presented for synthesizing as well as analyzing gameplay in video games. Using the proposed model, two sample games are designed for illustration.

In Chapter 15, entitled “From a Game Story to a Real 2D Game”, the authors create a model that can then be used to guide the game’s implementation. The authors use the game Othello as an example. The simple story of Othello, where two players alternate placing either black or white pieces on an 8x8 grid, is used to develop a finite state machine. This machine is used to guide the step-by-step implementation of the game code. From there, the authors make two alterations to the game to include both single player and networked versions.

Chapter 16 is entitled “Music Tutor Using Tower Defense Strategy”. Vivace is a version of the classic tower defense game, where the player must protect a growing tree sapling by using flowers. Each type of flower represents a different musical note, and this note is played when the flower attacks an enemy. Chords containing multiple notes can also be created. Learning the correct chords to defeat the enemy attack makes the game significantly easier. Algorithms discussed include procedural generation of the tree, game balancing, and the design of a finite state machine to control the boss enemy.

In Chapter 17, entitled “Low Cost Immersive VR Solutions for Serious Gaming”, the authors discuss the implementation of a real-time ship training simulator. The life cycle and architecture of the project are described followed by physics based modeling of the ship motion and waves. Implementation of real time motion detection algorithm is discussed followed by rendering details and the validation process. By using off the shelf components and free and open source software, the cost of the simulator was kept to a minimum.

Few books focus on a practical approach to game development concepts, focusing instead on theoretical concepts. It has been shown many times that practical, hands-on learning contributes to a greater
understanding of the material, and it has been our experience that when students know how an aspect of a popular game works, they are much more excited about the topic. This book gives readers of all levels insight into practical implementation details for a variety of game development topics. Instructional chapters, such as the physics chapter using LCP solvers, collision detection using AABBs, OBBs, and the GJK algorithm, and the modular game engine chapter give readers at all levels an opportunity to use hands-on learning to achieve a greater understanding of the concept. With this knowledge, the reader can then go on to extend the given implementation or create one of their own.

Two chapters focus on the importance of designing a game before starting implementation. The defining aspect of a game is its gameplay, or how a player interacts with the game. This knowledge is important for students and anyone starting to create video games. By breaking games down into gameplay elements, the implementation of the game is separated from aspects that are largely interchangeable, such as the art style. The developer can then focus on the implementation details.

Several chapters feature new contributions to the field. For example, the Real-Time Framework provides a high-level method of developing online games such as MMOG (Massively Multiplayer Online Games) and first-person shooters. Scalability, distribution of game data, and server communication are just some of the important issues in the field of online games. The Real-Time Framework addresses these issues and more. Another chapter discusses the implementation of a model for adaptive difficulty in video games. Currently, some players are under-challenged and others are over-challenged, with neither scenario being enjoyable for the player. The goal is to have the difficulty of a game change based on the capabilities of the player to provide an appropriate level of challenge. The authors have implemented a new type of adaptive difficulty model that determines the skill level of the player and changes the difficulty accordingly.

Finally, games can be used for more than just entertainment purposes. The use of simulators is important in many fields for training purposes since using actual equipment for training can be extremely costly and dangerous. Using simulators allows for training scenarios that are expensive, difficult, or impossible to reproduce in any other way. The immersive training simulator shows that effective simulators can be made cheaply.

While not every game is developed using a commercial, open source, or user developed game engine, the material in this book can offer insight into some of the issues critical to the successful development of video games and assist the reader in understanding and implementing game engine components common to virtually all game development projects. It is our sincere hope that readers will take away a deeper understanding of some topic of particular interest to them or a clearer understanding of how to go about the task of actually implementing a common video game algorithm or building a game engine.

Sincerely yours,

Ashok Kumar  
University of Louisiana at Lafayette, USA

Jim Etheredge  
University of Louisiana at Lafayette, USA

Aaron Boudreaux  
University of Louisiana at Lafayette, USA