Game Development for Computer Scientists

Alejandro Piad Morffis
David Darias Torres

Version 0.1

Prologue

The idea for writting this book came out from a course in game development we've being teaching for a few years now, at the University of Havana. After a few iterations and modifications to the course syllabus, we (boldly) decided that it would be nice to have the basic content of the course in a single written material. In this prologue we explain why we think this book is worth reading, and, if we convince you, then how it is best read.

Why reading this book?

The existence of this book is based on two main hypothesis. The first one is concerned with the content. Most of the books about game development we've seen (and used in our courses) are either very shallow or too specific. There are many books that don't require almost any background knowledge and there are others specifically devoted to game AI, physics, rendering, or some other topic. While we think many of these books are great in their own area, we wanted for our course a book that was both broad in the topics covered, and deep in the content. So this book is specially designed for students (or graduates) of a standard Computer Science major. We think that if you know CS, we can explain many of the concepts in game development in much greater detail, and than that makes you a better game developer.

The second hypothesis is concerned with the format. Again, many of the books we've seen take one of two approaches: they either try to be technology-agnostic or they are recipe collections for a specific technology. In the first group there are many game design books, books on design patterns, books about psychology and sociology of games, and many other gems. In the second group there are many great cookbooks, that can give a quick introduction to a specific technology and make you proficient at it. However, we wanted a hybrid approach for our course. On one hand, we want to tackle design issues, even psychological and sociological issues, that help game developers craft more compelling and engaging games. On the other hand, we want to tackle technological issues, algorithms, design patterns, languages, platforms, devices, which involve many practical decisions that can make a beatifully designed game either succeed or fail.

For these reasons, the following book is unique (or at least rare) in some senses. First, it presents theory (both theory of game design as well as theory on algorithms, techniques, etc.) with the depth and complexity than can be expected in academic rather than comercial books. Second, it presents plenty of code examples, practical advices, and technical content, oriented towards a specific game-making technology, which is the Unity game engine, in the extent that can be expected in comercial rather than academic books. We choose this particular technology because we believe it has a very well designed API, such that almost all concepts and patterns fit neatly within its architecture, it is also very efficient, and it has a thriving community of developers and users that produce plenty of documentation, examples, plugins, and other resources.

So, in order to answer the original, why should you read this book? Because once finished you will have a deep theoretical knowledge about how video games are designed and developed, and you will also have a pretty strong practical experience with the de-facto standard tool for making videogames in today's industry.

Who is this book for?

This book is for people with interest in video games design and development from a theoretical and practical point of view, with a competent knowledge in Computer Science topics. Particularly, the book has been designed for undergrad students of a Computer Science major, and we expect you to be comfortable around the following topics:

- Analysis of algorithms, order of growth, computacional complexity.
- Computer geometry and related algorithms and data structures.
- Numerical analysis and algebra, including numeric integration.
- Artificial intelligence, including problem-solving and machine learning.
- Formal languages, grammars, regular expressions, automata theory.
- Design patterns, basic concepts in software engineering.
- Comfortable programming in C#, or similar languages (Java, C++, . . .).

For all of the above topics, we expect a minimum understanding, at the level that can be expected in a Computer Science undergrad student that has taken at least the introductory course on each topic. That said, there are some chapters in the book that do not require any prior knowledge, concerned with game mechanics, design, and publishing and marketting.

Contributors

Every contribution is very much appreciated, anything from fixing a comma or mispelling to correcting source code or writing new demos, and even whole new chapters. We prefer to use the Github fork model for contributions, so please fork the project, do your changes, and submit a pull request. Issues are also appreciated, so if you discover an error, even if you cannot fix it, please submit the corresponding issue with as much info as possible of how to reproduce the error.

While we appreciate and encourage all forms of contribution, if you intend to submit a large modification (such as writing a new demo, or a new section or chapter), we would appreciate if you submit an issue first. The reason for this is that we want to maintain as much as possible a unified language and style (both in the book and the sample source code), and we would like to discuss first with you just to make sure we agree on the basic approach. This is just to ensure that your contribution helps as much as possible.

If you make **any** contribution, as simple as it is, don't forget to add your name (aphabetically sorted) to the following list of contributors:

List of Contributors

Alejandro Piad Morffis, David Darias Torres.

To all of them, thank you very much ;).

Contents