An Introduction to Verification of Visualization Techniques

Synthesis Lectures on Visual Computing

Computer Graphics, Animation, Computational Photography, and Imaging

Editor
Brian A. Barsky, ***University of California, Berkeley***

This series presents lectures on research and development in visual computing for an audience of professional developers, researchers and advanced students. Topics of interest include computational photography, animation, visualization, special effects, game design, image techniques, computational geometry, modeling, rendering, and others of interest to the visual computing system developer or researcher.

An Introduction to Verification of Visualization Techniques
Tiago Etiene, Robert M. Kirby, and Cláudio T. Silva
2015

Finite Element Method Simulation of 3D Deformable Solids
Eftychios Sifakis and Jernej Barbič
2015

Virtual Crowds: Steps Toward Behavioral Realism
Mubbasir Kapadia, Nuria Pelechano, Jan Allbeck, and Norm Badler
2015

Efficient Quadrature Rules for Illumination Integrals: From Quasi Monte Carlo to Bayesian Monte Carlo
Ricardo Marques, Christian Bouville, Luís Paulo Santos, and Kadi Bouatouch
2015

Numerical Methods for Linear Complementarity Problems in Physics-Based Animation
Sarah Niebe and Kenny Erleben
2015

Mathematical Basics of Motion and Deformation in Computer Graphics
Ken Anjyo and Hiroyuki Ochiai
2014

Mathematical Tools for Shape Analysis and Description
Silvia Biasotti, Bianca Falcidieno, Daniela Giorgi, and Michela Spagnuolo
2014

Information Theory Tools for Image Processing
Miquel Feixas, Anton Bardera, Jaume Rigau, Qing Xu, and Mateu Sbert
2014

Gazing at Games: An Introduction to Eye Tracking Control
Veronica Sundstedt
2012

Rethinking Quaternions
Ron Goldman
2010

Information Theory Tools for Computer Graphics
Mateu Sbert, Miquel Feixas, Jaume Rigau, Miguel Chover, and Ivan Viola
2009

Introductory Tiling Theory for Computer Graphics
Craig S.Kaplan
2009

Practical Global Illumination with Irradiance Caching
Jaroslav Krivanek and Pascal Gautron
2009

Wang Tiles in Computer Graphics
Ares Lagae
2009

Virtual Crowds: Methods, Simulation, and Control
Nuria Pelechano, Jan M. Allbeck, and Norman I. Badler
2008

Interactive Shape Design
Marie-Paule Cani, Takeo Igarashi, and Geoff Wyvill
2008

Real-Time Massive Model Rendering
Sung-eui Yoon, Enrico Gobbetti, David Kasik, and Dinesh Manocha
2008

High Dynamic Range Video
Karol Myszkowski, Rafal Mantiuk, and Grzegorz Krawczyk
2008

GPU-Based Techniques for Global Illumination Effects
László Szirmay-Kalos, László Szécsi, and Mateu Sbert
2008

High Dynamic Range Image Reconstruction
Asla M. Sá, Paulo Cezar Carvalho, and Luiz Velho
2008

High Fidelity Haptic Rendering
Miguel A. Otaduy and Ming C. Lin
2006

A Blossoming Development of Splines
Stephen Mann
2006

An Introduction to Verification of Visualization Techniques

Tiago Etiene, Robert M. Kirby, and Cláudio T. Silva

ISBN: 978-3-031-01459-8 paperback
ISBN: 978-3-031-02587-7 ebook

DOI 10.1007/S00679ED1V01Y201511CGR022

A Publication in the Springer series
SYNTHESIS LECTURES ON VISUAL COMPUTING: COMPUTER GRAPHICS, ANIMATION, COMPUTATIONAL PHOTOGRAPHY, AND IMAGING

Lecture #22
Series Editor: Brian A. Barsky, *University of California, Berkeley*
Series ISSN
Print 2469-4215 Electronic 2469-4223

An Introduction to Verification of Visualization Techniques

Tiago Etiene
Modelo Inc.

Robert M. Kirby
University of Utah

Cláudio T. Silva
New York University

SYNTHESIS LECTURES ON VISUAL COMPUTING: COMPUTER GRAPHICS, ANIMATION, COMPUTATIONAL PHOTOGRAPHY, AND IMAGING #22

ABSTRACT

As we increase our reliance on computer-generated information, often using it as part of our decision-making process, we must devise tools to assess the correctness of that information. Consider, for example, software embedded on vehicles, used for simulating aircraft performance, or used in medical imaging. In those cases, software correctness is of paramount importance as there's little room for error. Software verification is one of the tools available to attain such goals. Verification is a well known and widely studied subfield of computer science and computational science and the goal is to help us increase confidence in the software implementation by verifying that the software does what it is supposed to do.

The goal of this book is to introduce the reader to software verification in the context of visualization. In the same way we became more dependent on commercial software, we have also increased our reliance on visualization software. The reason is simple: visualization is the lens through which users can understand complex data, and as such it must be verified. The explosion in our ability to amass data requires tools not only to store and analyze data, but also to visualize it.

This book is comprised of six chapters. After an introduction to the goals of the book, we present a brief description of both worlds of visualization (Chapter 2) and verification (Chapter 3). We then proceed to illustrate the main steps of the verification pipeline for visualization algorithms. We focus on two classic volume visualization techniques, namely, Isosurface Extraction (Chapter 4) and Direct Volume Rendering (Chapter 5). We explain how to verify implementations of those techniques and report the latest results in the field of verification of visualization techniques. The last chapter concludes the book and highlights new research topics for the future.

KEYWORDS

visualization, verification, isosurfaces, volume rendering, geometry processing, verifiable visualization

Contents

Preface

The term *verification* has become ubiquitous in both the computer science and engineering communities as denoting a process that somehow convinces the user that verified tools, whether those be circuits, algorithms, implementations, etc. are more safe, accurate, or complete than other tools that have not been verified. Although the term verification has a common root usage within both communities, it has evolved to mean something specific to each subarea of computer science and of engineering. For instance, within computer science, the verification of a circuit denotes either the exhaustive testing or proof that under all possible inputs, the circuit will produce the correct (specified) outputs. Similarly, for software, verification relates how well an implementation represents the behavior of its specification under all possible inputs. Within the engineering world, verification takes on a different, more nuanced meaning. One assumes that there exists an "exact solution" or "exact representation" resulting from the solution of a mathematical system of equations. In all but the most trivial circumstances, this exact solution is not attainable, and approximate solutions must be formed. The process of quantifying how well a numerical scheme or representation approximates the exact solution is referred to as verification. Verification may involve looking at how well (or quickly) an approximate solution converges (in an appropriate norm) to the exact solution, or may involve identifying features or invariants of the solution that should be maintained regardless of the approximate representation. As visualization models, algorithms and implementations lie at the interface of these two branches, what does it mean to produce *verifiable visualizations*?

This question motivated the research work that has become the foundation of this book. To answer such a broad question, we started as most researchers would: by examining a concrete example in which our ideas could be refined. We started with isosurface extraction. Many tests and a few software bugs (which our process found) later, we realized that not only were our results worth communicating to the community, but that there was much work still to do. We moved to verifying different techniques used within the visualization community—in turn learning new things along the way. We began to appreciate that verification is a process, and that articulating the guiding principles of that process was itself a contribution to our community. The various papers we reference outline the specific contributions of our work. This book is meant to make that work accessible to the general reader in a pedagogical way. We hope the reader will take away not just a particular technique, but a way of approaching and testing visualization algorithms and their implementations. In the end, we hope that all successful visualization techniques will produce verifiable visualizations.

Any work of this size and scope has benefitted by many people both indirectly and directly. We wish to thank our collaborators that helped to shape this work, in particular Luis

Gustavo Nonato, Carlos Eduardo Scheidegger, Julien Tierny, Thomas J. Peters, Valerio Pascucci, Daniel Jönsson, Timo Ropinski, João Luiz Dihl Comba, Anders Ynnerman, Lis Custódio, and Sinésio Pesco. We also thank the various faculty and students at the SCI Institute (University of Utah) with whom we sharpened our ideas. In addition, we would like to thank the various Federal Funding Agencies that have supported our research efforts over the years. The papers we reference which are co-authored by us detail those acknowledgements. Lastly, we would like to thank our spouses, without whose patience and encouragement we would probably not have made it this far.

Tiago Etiene, Robert M. Kirby, and Cláudio T. Silva
December 2015