

REAL-TIME RENDERING

COMPUTER GRAPHICS
with CONTROL ENGINEERING

Gabriyel Wong
Jianliang Wang



CRC Press
Taylor & Francis Group

About the pagination of this eBook

Due to the unique page numbering of this book, the electronic pagination of the eBook does not match the pagination of the printed version. To navigate the text, please use the electronic Table of Contents or the Search function.

REAL-TIME RENDERING

COMPUTER GRAPHICS
with CONTROL ENGINEERING

AUTOMATION AND CONTROL ENGINEERING

A Series of Reference Books and Textbooks

Series Editors

**FRANK L. LEWIS, Ph.D.,
Fellow IEEE, Fellow IFAC**

Professor
The University of Texas Research Institute
The University of Texas at Arlington

**SHUZHONG SAM GE, Ph.D.,
Fellow IEEE**

Professor
Interactive Digital Media Institute
The National University of Singapore

PUBLISHED TITLES

Real-Time Rendering: Computer Graphics with Control Engineering,
Gabriyel Wong; Jianliang Wang

Anti-Disturbance Control for Systems with Multiple Disturbances,
Lei Guo; Songyin Cao

Tensor Product Model Transformation in Polytopic Model-Based Control,
Péter Baranyi; Yeung Yam; Péter Várlaki

Fundamentals in Modeling and Control of Mobile Manipulators, *Zhijun Li;
Shuzhi Sam Ge*

Optimal and Robust Scheduling for Networked Control Systems, *Stefano Longo;
Tingli Su; Guido Herrmann; Phil Barber*

Advances in Missile Guidance, Control, and Estimation, *S.N. Balakrishna;
Antonios Tsourdos; B.A. White*

End to End Adaptive Congestion Control in TCP/IP Networks,
Christos N. Houmkozis; George A Rovithakis

Robot Manipulator Control: Theory and Practice, *Frank L. Lewis;
Darren M Dawson; Chaouki T. Abdallah*

Quantitative Process Control Theory, *Weidong Zhang*

Classical Feedback Control: With MATLAB® and Simulink®, Second Edition,
Boris Lurie; Paul Enright

Intelligent Diagnosis and Prognosis of Industrial Networked Systems,
Chee Khiang Pang; Frank L. Lewis; Tong Heng Lee; Zhao Yang Dong

Synchronization and Control of Multiagent Systems, *Dong Sun*

Subspace Learning of Neural Networks, *Jian Cheng; Zhang Yi; Jiliu Zhou*

Reliable Control and Filtering of Linear Systems with Adaptive Mechanisms,
Guang-Hong Yang; Dan Ye

**Reinforcement Learning and Dynamic Programming Using Function
Approximators,** *Lucian Busoniu; Robert Babuska; Bart De Schutter; Damien Ernst*

Modeling and Control of Vibration in Mechanical Systems, *Chunling Du;
Lihua Xie*

Analysis and Synthesis of Fuzzy Control Systems: A Model-Based Approach,
Gang Feng

Lyapunov-Based Control of Robotic Systems, *Aman Behal; Warren Dixon;
Darren M. Dawson; Bin Xian*

System Modeling and Control with Resource-Oriented Petri Nets,
MengChu Zhou; Naiqi Wu

Sliding Mode Control in Electro-Mechanical Systems, Second Edition,
Vadim Utkin; Juergen Guldner; Jingxin Shi

Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications,
Shuzhi Sam Ge; Frank L. Lewis

Linear Control Theory: Structure, Robustness, and Optimization,
Shankar P. Bhattacharyya; Aniruddha Datta; Lee H. Keel

Optimal Control: Weakly Coupled Systems and Applications,
Zoran Gajic

Deterministic Learning Theory for Identification, Recognition, and Control,
Cong Wang; David J. Hill

Intelligent Systems: Modeling, Optimization, and Control,
Yung C. Shin; Myo-Taeg Lim; Dobrila Skataric; Wu-Chung Su; Vojislav Kecman

FORTHCOMING TITLES

Linear Control System Analysis and Design with MATLAB®, Sixth Edition,
Constantine H. Houppis; Stuart N. Sheldon

Modeling and Control for Micro/Nano Devices and Systems,
Ning Xi; Mingjun Zhang; Guangyong Li

REAL-TIME RENDERING

COMPUTER GRAPHICS
with CONTROL ENGINEERING

Gabriyel Wong
Jianliang Wang



CRC Press

Taylor & Francis Group

Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

MATLAB® and Simulink® are trademarks of The MathWorks, Inc. and are used with permission. The MathWorks does not warrant the accuracy of the text or exercises in this book. This book's use or discussion of MATLAB® and Simulink® software or related products does not constitute endorsement or sponsorship by The MathWorks of a particular pedagogical approach or particular use of the MATLAB® and Simulink® software.

Cover design by Gabriyel Wong

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2014 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
Version Date: 20130822

International Standard Book Number-13: 978-1-4665-8360-3 (eBook - PDF)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Visit the Taylor & Francis Web site at
<http://www.taylorandfrancis.com>

and the CRC Press Web site at
<http://www.crcpress.com>

Especially for God, Crystal, Xavier, Xana, and Xaron, the love of my life.

G.W.

Could we with ink the ocean fill,
And were the skies of parchment made,
Were every stalk on earth a quill,
And every man a scribe by trade;
To write the love of God above
Would drain the ocean dry;
Nor could the scroll contain the whole,
Though stretched from sky to sky.

Frederick M. Lehman (1868–1953)

Contents

List of Figures	xiii
List of Tables	xvii
List of Abbreviations.....	xix
Preface.....	xxi
Acknowledgements	xxiii
Summary.....	xxv
Authors.....	xxvii
Chapter 1 Introduction	1
1.1 Background and Motivation	1
1.2 Objectives and Contributions	2
1.3 Scope of Work	3
1.4 Book Outline	3
Chapter 2 Preliminaries	5
2.1 Fundamentals of Real-Time 3D Rendering.....	5
2.1.1 Polygon-Based Rendering	5
2.1.2 Volumetric Rendering	8
2.1.3 Image-Based Rendering	9
2.2 System Identification	10
2.2.1 Data Collection.....	11
2.2.2 Model Selection.....	12
2.2.3 Computing Model Parameters.....	13
2.2.4 Evaluating Quality of Derived Model.....	13
2.3 Literature Review	14
2.3.1 Comparative Study on Existing Research.....	14
2.3.2 Control-Theoretic Approaches to Computer Systems...	16
2.3.3 Control Principles in Computer Graphics Software	17
Chapter 3 Linear Model Analysis of Real-Time Rendering	19
3.1 Introduction	19
3.2 Background.....	19
3.2.1 Control-Centric Definition for Rendering Time Control.....	21
3.2.2 Challenges in Using Heuristics	21
3.2.3 Purpose of Workload Characterisation and Analysis ...	22
3.3 Case for Data-Driven Modelling	23
3.3.1 Basis for Selection of System Variables	23

3.4	Linear System Model Representation for Real-Time Rendering	25
3.5	Experiments.....	27
3.5.1	Experiment 1: Single-Input–Single-Output (SISO) System	27
3.5.2	Experiment 2: Multiple-Input–Single-Output (MISO) System.....	28
3.5.3	Experiment 3: Control Framework Using System Model.....	29
3.6	Results	30
3.6.1	Experiment 1	30
3.6.2	Experiment 2	33
3.6.3	Experiment 3	38
3.7	Discussion.....	40
3.7.1	Comparison with Other Estimation Techniques	41
3.8	Superposition in 3D Rendering System Model	43
3.8.1	Principle of Superposition.....	43
3.8.2	Experiment	44
3.8.3	Simulation	46
3.8.4	Summary	48
3.8.5	Additional Notes.....	49
3.9	Conclusion	49
Chapter 4	Modelling Non-Linear Rendering Processes	51
4.1	Introduction	51
4.2	Background.....	51
4.2.1	System Modelling with Neural Networks	51
4.2.2	Systems Modelling with Fuzzy Logic.....	53
4.3	Experiments.....	56
4.3.1	Time Delay Neural Network	56
4.3.2	Adaptive Neuro-Fuzzy Inference System	56
4.4	Experiment Results.....	60
4.4.1	Time Delay Neural Networks.....	60
4.4.2	ANFIS Model.....	61
4.5	Discussion.....	63
4.6	Linearised Approximation from Non-Linear Models	64
4.7	Conclusion	66
Chapter 5	Model-Based Control	67
5.1	Introduction	67
5.2	Control System Perspective of Computer Graphics Rendering Process	67
5.2.1	Control System Architectures for Real-Time Rendering	68

- 5.2.2 Control System Performance Concepts
Applicable to Real-Time Rendering 70
- 5.3 PID Control and Tuning 71
 - 5.3.1 Implementing PID Control for Rendering Process 72
 - 5.3.2 Data Preprocessing in PID Control System 75
 - 5.3.3 Gain Scheduling for Non-Linear Rendering
Process Models..... 76
 - 5.3.4 Neural PID Control 79
- 5.4 Experiments..... 81
- 5.5 Results 83
 - 5.5.1 Simulation Environment 83
 - 5.5.2 Control System with Actual Rendering Process 83
 - 5.5.3 Gain Scheduling Control System 85
- 5.6 Conclusion 86

- Chapter 6** Model-Less Control..... 89
 - 6.1 Introduction 89
 - 6.2 Fuzzy Control System 89
 - 6.3 Adaptive Neural Fuzzy Control..... 90
 - 6.4 Experiment 92
 - 6.5 Results 95
 - 6.5.1 Simulation 95
 - 6.5.2 Fuzzy Control System with Rendering Process 95
 - 6.6 Discussion..... 97
 - 6.7 Conclusion 98

- Chapter 7** Applications, Challenges, and Possibilities..... 99
 - 7.1 System Architectures..... 99
 - 7.1.1 Software Design 101
 - 7.2 Software and Hardware Performance Considerations..... 103
 - 7.2.1 Data Integrity 103
 - 7.2.2 Plant–Controller Communication Latency 103
 - 7.2.3 Data Structures and Handling..... 103
 - 7.2.4 Complexity of Control Algorithm..... 104
 - 7.3 Applications of Rendering Control Systems 104
 - 7.3.1 Extension of Control System Framework..... 105
 - 7.4 Convergence with Future Technology 105
 - 7.4.1 Greater Computing Parallelism..... 105
 - 7.4.2 Increased Use of Mobile Devices..... 105
 - 7.4.3 Vast Improvements in Internet Infrastructure..... 106
 - 7.5 Economic and Productivity Impacts 106
 - 7.5.1 Enhanced Product Lifespan 106
 - 7.5.2 Increased Productivity..... 106
 - 7.5.3 New Products and Markets 107

Chapter 8	Conclusion	109
8.1	Performance Analysis.....	109
8.1.1	Frame Rate Stability.....	109
8.1.2	Transient Response.....	110
8.1.3	Adaptive Tracking Capability	112
8.2	Summary	117
8.3	Future Work.....	118
Annex A: Sample Applications	121
A.1	Overview	121
A.2	ProgressiveMesh Sample.....	121
A.3	How Sample Works	121
A.4	Tessellation Sample	122
A.5	How Sample Works	122
A.6	Samples.....	122
Annex B: Patent for Application Method and System for Adaptive Control of Real-Time Computer Graphics Rendering	153
	Title of Invention	153
	Field of Invention	153
	Background of Invention.....	153
	Summary of Invention.....	154
	Brief Descriptions of Figures	155
	Detailed Descriptions of Figures.....	155
	Control Design and Mechanism.....	156
	I. PID Gain Scheduling.....	156
	II. Fuzzy Control (Model-Less Control)	159
	Claims (Preliminary).....	161
Annex C: Neural PID Control System Code	167
References	171
Publications and Achievements	177
	Patent Application	177
	Book	177
	Book Chapters	177
	Conference Papers.....	177
	Achievements	178

List of Figures

FIGURE 2.1	Real-time 3D rendering pipeline.....	6
FIGURE 2.2	Camera view frustum in 3D space.....	7
FIGURE 2.3	Programmable rendering pipeline (DirectX 11).	9
FIGURE 2.4	Samples of surface shading effects that can be achieved with pixel programs.	10
FIGURE 2.5	Process flow in system identification methodology.	11
FIGURE 2.6	Comparison of two model outputs with measured system response.....	14
FIGURE 2.7	The comparative literature review workflow.	15
FIGURE 3.1	Visual effect of varying vertex count for 3D object in discrete steps.....	25
FIGURE 3.2	ARX model structure.....	26
FIGURE 3.3	Screenshot of hardware tessellation sample application from DirectX SDK adapted with Stanford Dragon model in Experiments 1 and 2.	28
FIGURE 3.4	Screenshot of application in Experiment 1.	29
FIGURE 3.5	Screenshot of application in Experiment 3.	30
FIGURE 3.6	Input and output profiles of application in Experiment 1.	31
FIGURE 3.7	Steady-state frame time and vertex count relationship in Experiment 1.....	31
FIGURE 3.8	Measured and simulated output of rendering application in Experiment 1.....	32
FIGURE 3.9	Error between measured and simulated output of application in Experiment 1.....	33
FIGURE 3.10	Steady-state outputs of the system based on selected combinations of two input variables.	35
FIGURE 3.11	Profiles of two inputs and output of rendering system in Experiment 2.....	36
FIGURE 3.12	Measured and simulated outputs of MISO rendering system in Experiment 2.....	37

FIGURE 3.13	Profiles of input and output of rendering system in Experiment 3.	38
FIGURE 3.14	Measured and simulated rendering system output in Experiment 3.....	39
FIGURE 3.15	SISO control system in Experiment 3.....	40
FIGURE 3.16	Simulated reference tracking with PID controller.	40
FIGURE 3.17	Reference tracking with actual rendering application.	41
FIGURE 3.18	Screenshot of test application in superposition experiment.....	45
FIGURE 3.19	Measured output and predicted output from Model A.	46
FIGURE 3.20	Measured output and predicted output from Model B.....	46
FIGURE 3.21	Measured output and predicted output from Model C.	47
FIGURE 3.22	Comparison of outputs from Model C and summed outputs of Models A and B.	47
FIGURE 4.1	(a) Perceptron neuron. (b) Multi-layer perceptron network (MLP).....	52
FIGURE 4.2	Two-layer distributed time delay neural network with time delays at inputs of each layer.	53
FIGURE 4.3	Fuzzy inference system.....	54
FIGURE 4.4	Screenshot of application in Experiment 1.	57
FIGURE 4.5	Screenshot of application in Experiment 2.....	57
FIGURE 4.6	Adaptive network.	58
FIGURE 4.7	Neural network in Experiment 1.....	60
FIGURE 4.8	Data collected from Experiment 1.....	61
FIGURE 4.9	Data collected from Experiment 2.....	62
FIGURE 4.10	Screenshot of rendering application in Experiment 3.....	62
FIGURE 4.11	Measured and reference output from ANFIS in Experiment 3.	63
FIGURE 5.1	Rendering process from system perspective.....	68
FIGURE 5.2	Closed-loop feedback control system.	68
FIGURE 5.3	Rendering system with adaptive controller and quality of service feedback.	69
FIGURE 5.4	Modular adaptive control system for real-time rendering.	70
FIGURE 5.5	PID control system in MATLAB.....	73

FIGURE 5.6	(a) Setting PID controller gain values in MATLAB. (b) Interactive graphical user interface in MATLAB/Simulink for tuning PID controller.....	74
FIGURE 5.7	Steady-state frame time and vertex count relationship shown in Experiment 1.....	76
FIGURE 5.8	Gain scheduling PID control system.	78
FIGURE 5.9	Single neuron PID control system.	79
FIGURE 5.10	Comparison of system outputs using SNPID and PID controllers.	81
FIGURE 5.11	Control input from SNPID and PID controller.....	82
FIGURE 5.12	Screenshot of application with PID control.....	82
FIGURE 5.13	Reference tracking using PID controller (low to high).....	83
FIGURE 5.14	Reference tracking using PID controller (high to low).....	84
FIGURE 5.15	Reference tracking using PID controller (to higher FPS).	84
FIGURE 5.16	Reference tracking using PID controller (to lower FPS).....	85
FIGURE 5.17	Simulated output with gain scheduling PID controller.....	86
FIGURE 6.1	Fuzzy control system in Simulink/MATLAB.....	90
FIGURE 6.2	Configuring fuzzy controller in Simulink/MATLAB.....	91
FIGURE 6.3	ANFIS editor graphical user interface in Simulink/MATLAB.....	92
FIGURE 6.4	Neural network model structure in ANFIS.	93
FIGURE 6.5	Using ANFIS for controlling real-time rendering process.....	93
FIGURE 6.6	Input and output membership functions.	94
FIGURE 6.7	Fuzzy logic control system.	95
FIGURE 6.8	Reference tracking using fuzzy controller (high to low).	95
FIGURE 6.9	Reference tracking using fuzzy controller (low to high).	96
FIGURE 6.10	Reference tracking using fuzzy controller (to lower FPS).	96
FIGURE 6.11	Reference tracking using fuzzy controller (to higher FPS).....	97
FIGURE 6.12	Continuous reference tracking using ANFIS controller.....	98
FIGURE 7.1	The timing diagram of the rendering application used in the control system.....	101
FIGURE 7.2	The high-level design of the rendering application.....	102
FIGURE 8.1	Experiment results from Poudroux and Marvie's research.	110

FIGURE 8.2	Experiment results from Gobbetti and Bouvier’s multi-resolution technique.....	111
FIGURE 8.3	Experiment results from Jeschke et al.’s approach with usage of imposters.....	112
FIGURE 8.4	Experiment results from Paravati et al’s adaptive control technique.	113
FIGURE 8.5	Screenshot of application in our experiment.	114
FIGURE 8.6	Reference tracking using PID controller (low to high).....	114
FIGURE 8.7	Experiment results from Zheng et al’s work on rendering large 3D models online.	115
FIGURE 8.8	Experiment results from Li and Shen’s research on time-critical multi-resolution volume rendering using 3D texture mapping hardware.....	116
FIGURE 8.9	Quick-VDR: Interactive view-dependent rendering of massive models, Yoon et al.....	117
FIGURE 8.10	Experiment results from Scherzer, Yang, and Mattausch’s research on exploiting temporal coherence in real-time rendering. ...	118
FIGURE B.1	System model of open-loop rendering process.	162
FIGURE B.2	Closed-loop control system with feedback.	162
FIGURE B.3	Deployment in single computer device.....	163
FIGURE B.4	Deployment in distributed computer environment.	163
FIGURE B.5	Plot of steady-state values of input and output data.....	164
FIGURE B.6	Control system.	164
FIGURE B.7	Relationship of input and output of rendering system.	165

List of Tables

TABLE 2.1 Results from the Research Review Classification 16

TABLE 3.1 Performance Counters in DirectX.....24

TABLE 3.2 Parameters of ARX Model in Experiment 1 33

TABLE 3.3 Parameters of State Space Model in Experiment 1 34

TABLE 3.4 Parameters of ARX Model in Experiment 2 38

TABLE 3.5 Parameters of ARX Model in Experiment 3 39

TABLE 3.6 Parameters of ARX Model A in Superposition Experiment48

TABLE 3.7 Parameters of ARX Model B in Superposition Experiment48

TABLE 3.8 Parameters of ARX Model C in Superposition Experiment48

TABLE 5.1 Linear Operating Ranges..... 85

TABLE 6.1 Fuzzy Inference Rule Set.....90

List of Abbreviations

3D	Three-dimensional (computer graphics)
ANFIS	Adaptive neuro-fuzzy inference system
ANN	Artificial neural network
ARX	Auto-regressive with exogenous (input or term)
BIBO	Bounded-input bounded-output
CAD	Computer-aided design
CAM	Computer-aided manufacturing
DTDNN	Distributed time-delay neural network
FPS	Frames per second (also known as frame rate)
GPU	Graphical processing unit
GUI	Graphical user interface
LAN	Local area network
LoD	Level of detail
MISO	Multiple-input–single-output
MLP	Multi-layer perception
N4SID	N4 subspace identification method
PID	Proportional, integral, derivative (control)
QoS	Quality of service
SISO	Single-input–single-output
SNPID	Single neuron PID
TCP	Transmission control protocol

Preface

Interactive computer graphics is a mature field of study. In fewer than 15 years, the improvements in speed and realism of computer-generated graphics from even consumer grade computers have been phenomenal. There is no lack of evidence to substantiate this statement as we observe the ever-increasing number of cutting-edge interactive applications such as computer games, virtual prototyping, and visualisation software. However, real-time computer graphics applications are often oriented toward meeting a particular set of goals without consideration of some form of global optimisation. A number of years ago, through real-life encounters in large-scale system implementation, the idea of convoluting computer graphics rendering with control theory was born.

From a larger perspective, computer graphics rendering is akin to any other process that runs on a computer. In recent years, researchers found that the increasing inclination to employ control engineering techniques in computer-related processes is not so much a matter of computer control (using a computer as a controller) as controlling the processes within a computer. Examples of such implementation are discussed in the vast array of research literature about server performance, network traffic control, and adaptive software with defined quality-of-service metrics. We believe the trend is no coincidence; it represents wide acceptance of benefits from integrating control theory with computer processes.

Our motivation for this work is simple. First, we want to provide a fundamental analysis of interactive computer graphics rendering from a systems perspective. Second, we want to establish a framework that facilitates interactive computer graphics rendering in an environment providing optimal utilisation of resources and good responses to rendering load changes. These goals can be accomplished through the adoption of digital signal processing, system identification and control engineering techniques that we believe will draw the interest of researchers and practitioners in the computer graphics-related fields.

While classical control demands meticulous evaluation of numerous criteria, the goals of our control system described in this book focus on tracking user-defined performance objectives while providing good transient responses so that changes arising from rendering load control will not lead to abrupt changes in visual displays. Furthermore, unlike physical systems utilised in aircraft, motors, and chemical mixers in which a failure of a control mechanism may lead to a catastrophic outcome, interactive computer graphics rendering is generally fail-safe.

In the course of this work, the computer graphics rendering process is modelled from a data-driven and black-box approach. We have shown the possibilities of various input–output configurations in a system model setting. While some may argue that the rendering process is too complex to be modelled by a few variables, we hope the reader can appreciate that the modelling technique in this book is in fact not congruent to this argument, but rather a systematic approach because the derived system models are substantiated with measured data.

Finally, it is our sincere hope that this work can further stimulate cross-disciplinary research and provide a premise upon which more interesting modelling and control techniques for real-time computer graphics may be developed.

MATLAB® is a registered trademark of The MathWorks, Inc. For product information, please contact

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098 USA
Tel: 508-647-7000
Fax: 508-647-7001
E-mail: info@mathworks.com
Web: www.mathworks.com

Acknowledgements

Words are just inadequate to express my gratitude toward Professor Wang Jianliang, who is more than just my supervisor, he is a mentor and friend for many years. Through him, I have learnt to appreciate the beauty of control theory. More importantly, his enduring encouragement and support have left me with a deep appreciation of him as a true educator. If there is one conversation I would choose to remember for life, it would be when he distilled the spirit of academic research as a pursuit of excellence and challenge.

This book would not have been possible without the support of many, especially my family. My heartfelt appreciation goes to my parents who did not have an extensive education yet believed wholeheartedly in the value of continual education, to the extent of making sacrifices for me in so many ways. To me, there is no closer personification of selfless love than this. My wife Crystal, the gem of my life, has been most instrumental in this endeavor. I wish to thank her for carrying the burden on the home front and being such a dedicated partner. She is a godsend whom I can never do without in every season of my life. The credit and fruit of this labor belong to all of them.

Last but certainly not least, I thank God for this journey of molding and growth. I thank Him for all the people who have made a difference in my life through this work and every step which He has hand-held me. To complete writing this book is a task that requires unimaginable perseverance and strength which He has so graciously given to me.

Gabriyel Wong

Summary

The value of interactive computer graphics is underscored by myriad applications in many domains of our lives. Consumers today can expect extremely realistic imagery generated in real time from commodity graphics hardware in applications such as virtual prototyping, computer games, and scientific visualisation. However, the constant and increasing demands for fidelity coupled with hardware architecture advancement pose many challenges to researchers and developers as they endeavour to find optimal solutions to accommodate speed of rendering and quality in interactive applications with real-time computer graphics rendering. The qualitative requirement of such applications, apart from the subjective perception of the displayed imagery, is the response time of a system based on user input. In other words, the requirement translates to the speed at which the machine can produce a rendered image according to the input provided by the person in the loop of the feedback system.

Earlier research attempted to address the frame latency problem by providing mathematical models of the rendering process. The models were often primitive because they were derived from coarse approximation or depended on specific application level data structures. Most approaches are based on heuristics and algorithms and are largely dependent on a specific type of application corresponding to the research. A major shortcoming of such techniques lies in the non-guarantee of performance.

From a systems perspective, the rendering process is modelled from an open-loop approach underpinned by constraints and estimations of the constituents of the rendering process. As a result, the output often fluctuates within an acceptable performance range. Furthermore, many such techniques rely on specific hardware or they may require unfriendly implementation on current computer graphics hardware. The advent of more sophisticated consumer graphics hardware in recent years has caused the rendering pipeline to be used in a far more complex manner to achieve ultra-realistic visual effects. Consequently, adapting models into applications becomes progressively more challenging as hardware and software technologies continue to evolve.

We can see from this background the exciting opportunities for the introduction of modelling and control principles into existing computer graphics systems. Our research focused on a systematic approach to realising a framework for modelling and control of real-time computer rendering in two stages:

1. Investigation, analysis, and implementation of a data-driven system identification process for real time rendering
2. Structured analysis of the derived model for the selection and design of a suitable control strategy

The first part of this book focuses on the modelling aspects of real-time rendering. Based on the dynamic natures of the possible and myriad variations of render states, polygon streams, and the non-linearity of the rendering process, we propose

a data-driven modelling approach that accurately represents the system behaviour of this process from two angles: (1) the larger operating range where non-linearity exists and (2) the piecewise linear operating range. We propose two techniques for tackling the modelling challenge: (1) using a feed-forward time delay neural network derived from experimental data and (2) fuzzy modelling.

We demonstrate that both techniques can yield very accurate results in comparison with actual measured data. In addition, we compare the estimated outputs of our models with other mathematical estimation methods to show that the models derived from our approach yield better results than mathematical estimations. Starting with single-input–single-output (SISO) system models, we extend our work to investigate the validity of multiple-input–single output (MISO) systems as well.

The second part of this book focuses on the design of a control strategy based on the process nature investigated in the earlier chapters. The benefits of applying control theory in the context of a computer graphics system are explained and the relative advantages of the theory over the performances of existing heuristics and algorithms (open-loop estimations of rendering) are highlighted.

Our research proposed two controller designs to achieve stable output with accurate tracking: (1) proportional, integral, derivative (PID) control and (2) neural and fuzzy control. We investigated control system implementation in both local and distributed configurations.

In the local configuration, the rendering process (“plant”) and controller reside in the same computer. In the distributed configuration, the controller runs on a computer different from the one used for rendering. The control activities and plant feedback are communicated between the computers via a network link. Despite network latency, this configuration allows flexible usage of system-wide resources in an integrated environment. The approach will be especially useful if elaborate controller designs adopted in the future result in the introduction of heavy computational loads into overall systems.

Authors

Gabriel Wong is an entrepreneur, innovator, and author with extensive experience spanning leadership, managerial, and consulting roles in technology businesses. He currently works for one of Europe's largest private equity businesses in e-commerce and heads product performance and strategy activities in Southeast Asia.

Wong was the co-founder of XPEGIA, a Singapore-based start-up specialising in interactive media solutions for the advertisement and education markets. Before that, he was the R&D Director at EON Reality, a global leader in virtual reality technology based in the United States; he spearheaded the company's research and development. Before joining EON Reality, Wong was the founding director of gameLAB, the first research laboratory in Singapore to focus on computer game design and technology. He was a faculty member at Singapore's Nanyang Technological University (NTU) and lectured in both undergraduate and post-graduate programs.

He started his career as a technical lead at Singapore Technologies, one of Asia's largest engineering conglomerates and led the pioneering work on advanced computer graphics technology for defense applications.

Wong has published papers and spoken at conferences around the world and secured public and commercial funding for patenting his inventions. He earned B Eng and M Eng degrees in 2000 and 2012 from NTU and will be earning his PhD in 2013.

Jianliang Wang, PhD, earned a BE in electrical engineering from Beijing Institute of Technology in China in 1982 and pursued MSE (1985) and PhD (1988) degrees in electrical engineering from The Johns Hopkins University in the United States.

From 1988 to 1990, Dr Wang was a lecturer in the Department of Automatic Control at Beijing University of Aeronautics and Astronautics. In 1990, he joined the School of Electrical and Electronic Engineering at Nanyang Technological University, Singapore, where he is currently a tenured associate professor.

Dr Wang's current research interests include modelling and control of computer graphics rendering systems and also robust and reliable controls, nonlinear controls, and their applications to flight control systems. He has published 4 book chapters, about 70 journal papers, and more than 130 conference papers.

Dr Wang currently serves as an associate editor of *Transactions of the Institute of Measurement* and the *Asian Journal of Control*. He was a guest editor for a special issue of *Control and Intelligent Systems*. The special issue of this international journal published in January 2012 was dedicated to networked control and unmanned systems.

Dr Wang also served as the general chair of IEEE's 2007 International Conference on Control and Automation and program chair for the 2010 conference. He also chaired various aspects of several conferences including the International Conference on Control, Automation, Robotics, and Vision; the Asian Control Conference; the Chinese Control Conference; and others. He was named chairman of IEEE's Singapore Control Systems Chapter for 2008–2009. He is a senior member of IEEE.

1 Introduction

1.1 BACKGROUND AND MOTIVATION

While modern rendering software claims to have controlling mechanisms that enhance runtime performance, the mechanisms are often very primitive and inadequate. The results of this deficiency are indeterminate drops in the visual quality of generated imagery and frame rates that can severely affect usage experience. By applying control theories in real-time rendering, it is possible to rectify these shortcomings altogether.

The vision is to create an intelligent rendering system that can systematically adapt to its operating environment to produce optimum runtime performance at all times. To our best knowledge, no commercial product exists as this work is written and no active research is in progress in this cross-disciplinary application field.

The application of control concepts in the computer graphics software provides new opportunities for better performance derivable from graphics hardware. Until today, typical rendering applications struggled to utilise hardware efficiently. Much of the burden of optimisation falls on the software programmer who must be extremely conversant with the graphics pipeline.

The predominance of interactive computer graphics is underscored by a burgeoning variety of applications in various aspects of daily life. For example, it is easy to observe various types of interactive systems in an urban environment such as a shopping mall or an office building. These systems range from digital signage to projection-based displays and touch panels. At the industrial level, interactive computer graphics technology powers important processes such as computer-aided design and manufacturing, virtual prototyping, and scientific visualisation and simulation.

While customers constantly demand high quality computer-generated graphics, the cost associated with their demands may not be within reach. To illustrate, the price of a performance workstation is typically many times more than the cost of a desktop PC for home use. Furthermore, mobile devices such as PDAs and cell phones lack sufficient computing power to render high quality graphics for productivity at work.

Our research concerns a fully automated technology that circumvents the aforementioned problems and allows users to enjoy high quality interactive computer graphics on both desktop and mobile devices. The objective of this project is to leverage earlier research on this subject and extend the work to allow a product-ready toolkit to be developed for commercialisation opportunities.

Over the past few years, we developed a framework that realises the concept of delivering adaptive interactive rendering through laboratory experiments, theoretical modelling, and simulation. Our technology employs control theory and the system identification methodology, both of which are mature fields, proven by their use in

aeronautical, mechanical and electrical engineering, and electronics industries. The concept is based on feedback control that can provide consistent performance monitoring and regulation with no requirement for human intervention. From a systems perspective, the technical challenge translates into the form of a “plant” (process to be controlled) and a “controller” component that ensures the process performs optimally according to predefined objectives.

This technology clearly has numerous applications and commercialisation possibilities. We conceived the possibilities listed below.

Computer-aided design (CAD) and -manufacturing (CAM)—Three-dimensional (3D) datasets used widely in many industrial applications. Our technology will allow a user to view such datasets even on a mobile device. This brings productivity out of the office and makes it available to people on the move.

Virtual communication—The market for 3D virtual communication is growing, particularly in the education and corporate services segments. As a viral social networking medium or mode of communication in professional exchanges, 3D interactive applications will remain key factors in online virtual communication. We see our technology as an enabling factor for linking more people to such networks.

Marketing and sales—More companies are moving toward high quality interactive content intended for consumers. This provides an opportunity for us to introduce our technology so that more people can utilise it without the limitations imposed by hardware. As a result, commercial entities can expect greater market reach and corresponding increases in revenue.

Training and education—Our technology can be deployed in various training and education products, enabling them to be delivered to audiences utilising hardware with different capabilities. The benefit offered by our technology is the easy ability to visualise 3D information even in a collaborative environment, therefore enhancing the value of training and knowledge dissemination.

1.2 OBJECTIVES AND CONTRIBUTIONS

Based on the shortcomings of current real-time rendering software, our research entailed the investigation and development of a feasible solution that would allow accurate and sustainable control of the real-time rendering process on different hardware platforms. The two key objectives affecting implementation of the technology are:

1. Despite the complexities involved in real-time rendering, it is imperative to devise a systematic method to describe this process in a form that relates its inputs and outputs consistently.
2. Based on the derivable form and the known characteristics of the rendering process, it is critical to find applicable control principles and frameworks that will ensure control of the process over a variety of scenarios.

Our research spans knowledge of the computer science (computer graphics rendering) and control engineering disciplines. Both fields imposed challenges that made our research both exciting and fulfilling. Our key research contributions are listed below.

1. We describe a novel framework by which the real-time rendering process may be modelled accurately. This framework involves the adoption of data-driven system identification methodology. Previous attempts to characterise the rendering process via only observable variables and case-specific formulations led to inaccurate models. Our model addresses these shortcomings.
2. Apart from linear models, our data-driven framework is extended to non-linear models using soft computing techniques such as neural networks and fuzzy models.
3. We developed control system frameworks for both linear and non-linear models in real-time rendering using (a) PID control with and without gain scheduling and (b) fuzzy control with and without adaptive neural networks.

The application of our control frameworks has shown much better resource utilisation in the real-time rendering process than earlier work that generally demonstrated coarse performance tracking.

1.3 SCOPE OF WORK

Real-time rendering is a vast topic in the field of computer graphics. Although the modelling techniques and control framework may be applicable to areas such as volume- and image-based rendering, our study deals with polygonal-based rendering pipelines found in commodity graphics hardware and it leverages geometry subdivision technique as a basis for controlling the input to the rendering system.

At this juncture, our work is based largely on the rendering of a single large 3D mesh that is used as a pseudo-representation of more complex 3D scenes with numerous objects. From a different perspective, this system is useful for applications involving a single large object of interest, for example, massive model rendering and computer-aided design.

Since the focus of this research is on real-time rendering relating to the response time of a system in an interactive environment, we consider the time required to render an image (frame) as the critical performance metric. While computer graphics activity is essentially visual, the quality of the generated image is frequently taken as the next most important metric for assessment. However, due to the subjectivity and complexity involved in processing image comparisons, the image quality component is omitted as a performance object in this work. From the system perspective, the real-time rendering framework proposed in this research is flexible to accommodate a multiple-input–multiple-output (MIMO) configuration. This means the user has the full freedom to implement additional output variables, which may include image quality related performance variables.

1.4 BOOK OUTLINE

Chapter 1 provides the background and motivation that led to this research. Chapter 2 discusses the fundamental knowledge in two key disciplines related to this research—real-time computer graphics rendering and system identification

methodology. We then provide a systems perspective of the rendering process and explain the impacts of variables surrounding the system inputs and outputs. After that, a survey of previous research in the areas of rendering load control and characterisation is discussed.

Chapter 3 delves into the details of our data-driven modelling approach to real-time rendering with a focus on linear system structures and their derivation. Through experiments, we provide rendering models for single-input–single-output (SISO) systems and show how they may be extended to more complex and practical systems involving multiple inputs.

In Chapter 4, we explore the use of soft computing techniques for modelling the real-time rendering process. The application of such techniques is performed at the operating range of the rendering system where non-linear characteristics are exhibited. Following that, we provide the basis for linearisation from the derived non-linear rendering system model.

Chapter 5 begins with the introduction of model-based control and deals with the control system framework for the linear rendering system model obtained in Chapter 3. The key control mechanism discussed in this chapter is the closed-loop feedback design with PID controller. We demonstrate how systems with single and multiple inputs may be controlled as well.

The focus of Chapter 6 is on advanced control techniques and considers our proposed framework from a model-less perspective. This chapter illustrates the establishment of a control system framework without the need for an explicit system model as described in Chapters 3 and 4. By using a variety of fuzzy control techniques, we demonstrate that a control system can perform very well when tracking the performance of a real-time rendering process.

Chapter 7 discusses applications, challenges, and possibilities, including system architectures, software and hardware performance and future technology.

The conclusions of our research and suggestions for future work are discussed in Chapter 8.

Annex A contains sample applications.

Annex B discusses the authors' patent for Method and System for Adaptive Control of Real-Time Computer Graphics Rendering.

Annex C delineates Neural PID Control System Code.

2 Preliminaries

2.1 FUNDAMENTALS OF REAL-TIME 3D RENDERING

In real-time computer graphics, 3D rendering refers to the process of generating a sequence of images that produces not just the animated effect of motion and change but the visual cue of depth for objects in the imagery given an external input or stimulus to the system. In typical applications, the goal is to provide visual feedback to the user when there is interaction via the human-computer interface. The speed at which each image, known as a frame, of the animation sequence is generated defines the performance of the system.

Because speed of rendering every image is crucial in real-time rendering, both the computer hardware and software have to work together in the most optimal way so that the best possible image quality can be achieved in tandem with an acceptable frame rate (a metric that measures the number of frames that can be generated in one second). Over many years of research and development, the real-time 3D rendering process has taken leaps and bounds in terms of the image quality that is produced in various real-world applications such as computer games, training simulators and 3D product demonstrations. This involves an intricate process that spans the preparation of 3D content in elaborate modelling tools to processing combinations of rendering algorithms with myriad configurations of parameters for the final output which is the image to be shown eventually on the display device. Modern computers have dedicated hardware to handle computer graphics rendering. This hardware provides acceleration to computer graphics rendering routines so that the computer's central processor unit (CPU) can focus on other non-computer-graphics-related and auxiliary tasks. In general, real-time or interactive 3D rendering applications are supported by an abstraction layer that communicates with the hardware. This layer is commonly known as the 3D rendering Application Programming Interface (API) and it is fully responsible for pushing rendering commands to the hardware and managing the render state machine.

2.1.1 POLYGON-BASED RENDERING

Figure 2.1 shows the multi-stage 3D real-time rendering pipeline. The transformation of inputs to the final visible pixels on a display device may be described systematically via the following steps.

- **Creation in Local 3D Model Coordinate System**
 - Each object is created individually in its own 3D coordinate system.
 - Objects may be represented in a variety of geometry formats (triangles, rectangles, strips of polygons, etc.). Essentially, every polygon in a 3D space consists of points known as vertices.

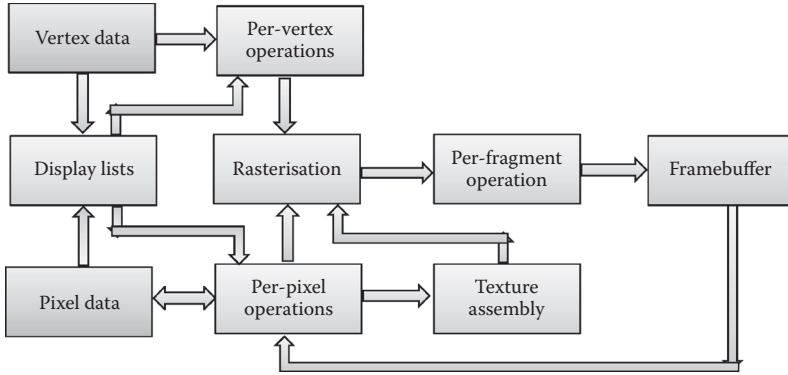


FIGURE 2.1 Real-time 3D rendering pipeline.

- For polygons to be rendered with visually correct features, each vertex is associated with a set of attributes such as position (coordinates in 3D space), colour, normal (perpendicular) vector from a surface, texture coordinates (user-defined mapping onto the surface), and other factors.
- **Transformation to Global World Coordinates**
 - To compose a scene in 3D space consisting of different objects, all created 3D objects must be transformed into the same coordinate system.
 - These transformations modify only the relative positions of the vertices and the normal. Visual attributes such as colour and texture coordinates are not modified.
- **Transformation to 3D View Coordinate System**
 - A viewpoint in 3D space is commonly cited as the “camera” location.
 - The geometry (vertex arrangement) from the 3D space is transformed into the camera view coordinate system. Depending on the rendering software, the common definition for this space is based on a right-handed coordinate system with the camera at the origin pointing down the negative z axis. The x axis is to the right and the y axis up. The projection from 3D to 2D space is performed at this stage.
 - The depth information of any object can be obtained from the z coordinate value at this stage.
 - The effect of virtual “lights” that create illumination properties in the 3D scene is computed at this stage. For example, a surface colour shading algorithm known as Gouraud shading will be computed at each vertex of a 3D object using the light parameters, light position, normal vectors, and the 3D object’s texture or material properties.
 - The removal of polygonal surfaces not shown in the view due to occlusion is known as “culling” and is performed at this stage as well.
 - Culling is related to the attributes of the camera view defined by a virtual trapezoid volume known as the “view frustum” using six planes (left, right, up, down, front, and back) as shown in Figure 2.2.

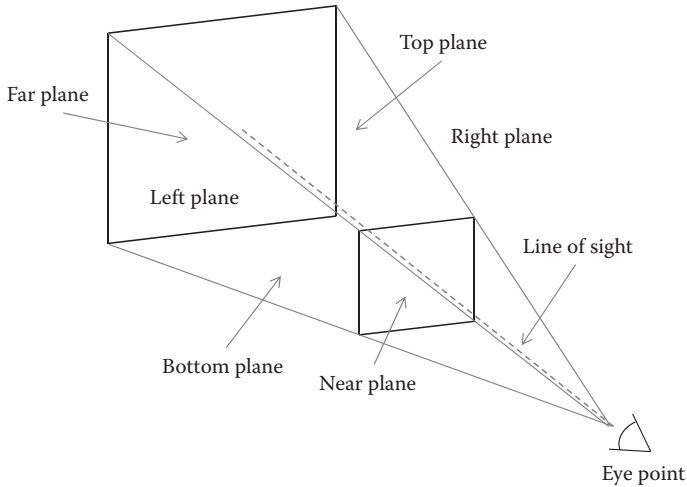


FIGURE 2.2 Camera view frustum in 3D space.

- **Transformation to 3D Clip Coordinate System**
 - The geometry data in this stage are prepared for a post-processing step known as “clipping.”
 - The transformation of the geometry depends on the type of view projection used. Certain non-linear transformation may take place, for example, when perspective projection creates a tapering-off view of objects at a distant horizon in contrast to orthographic projection that consistently preserves the dimensions of a 3D object.
- **Transformation to Normalised Device Coordinates**
 - The geometry is normalised for display in a 2D window on a physical display device.
 - Further clipping is done to remove geometry outside the user-defined window boundaries.
- **Transformation to Display Window Coordinates**
 - All vertices are converted to units of the display (pixels) window.
 - Typically, the origin of reference is at the lower left corner of the display window.
- **Transformation to 2D Screen Coordinate System**
 - The conversion to screen pixels (rasterisation) is performed. Pixels are visible colour dots that can be displayed on a screen.
 - To generate shaded pixels, attributes such as texture coordinates, colour, and normal vectors are used in the computation and interpolated across the vertices and polygon surfaces.
 - Algorithms may be used to perform further hidden surface removal by using depth information obtained from the geometry.

- The final colour of the pixel is determined by combining all other effect state settings (e.g., blending and stencil operations) in the rendering pipeline.
- The output of this stage is the final colour of every pixel placed in the memory of the display hardware (the frame buffer).

In the course of rendering a 3D scene, many inputs and settings such as the geometries of 3D objects and their material “look” parameters are sent to the graphics hardware for processing. About a decade ago, outdated graphics hardware relied solely on a few hard-wired algorithms to process such data via a method known as the fixed function rendering pipeline. As a result, real-time rendering application developers had little space to control the look of a 3D object based on a limited set of functions that computed the rendering output. The impact of such limitations is the lower quality of imagery generated from computer graphics hardware.

This problem was circumvented by the advances represented by a new generation of computer graphics hardware that allows rendering routines known as *shaders* to be injected into the hardware before or during the runtime of an application. This capability now gives application developers full control over the quality of the generated output by varying shader routines. Figure 2.3 depicts this new-generation fully programmable rendering pipeline.

Shaders come in two formats: vertex and pixel types. A vertex shader is a graphics processing function used to add special effects to objects in a 3D environment. It is executed once for each vertex sent to the graphics processor. The purpose is to transform each vertex’s 3D position in virtual space to the 2D coordinate at which it appears on the screen and the as a depth value in the graphics hardware. A pixel shader is a computation kernel function that computes colour and other attributes of each pixel. Pixel shader functions range from always outputting the same colour to applying a lighting value to adding visual effects such as bump mapping, shadows, specular highlights, and translucency properties. They can alter pixel depth or output more than one colour if multiple render targets are active. Figure 2.4 illustrates an example of the effects of pixel shaders on a 3D object. Apart from vertex and pixel shaders, an important feature of state-of-the-art graphics rendering architectures is the functionality of geometry shaders. Geometry shaders are added to the rendering pipeline to enable generation of graphics primitives, such as points, lines and different types of triangles after the execution of vertex shaders. With this capability, it is then possible to perform operations such as mesh resolution manipulation and procedural geometry generation.

Computer hardware technology and new rendering algorithms continue to advance quickly. The evolution of the real-time rendering pipeline also continues as this book is written.

2.1.2 VOLUMETRIC RENDERING

In Section 2.1.1, we described how animation can be produced using 3D data and physics-based principles for surface shading effects. Another technique for producing 3D visualization is through the usage of volume data that consists of not

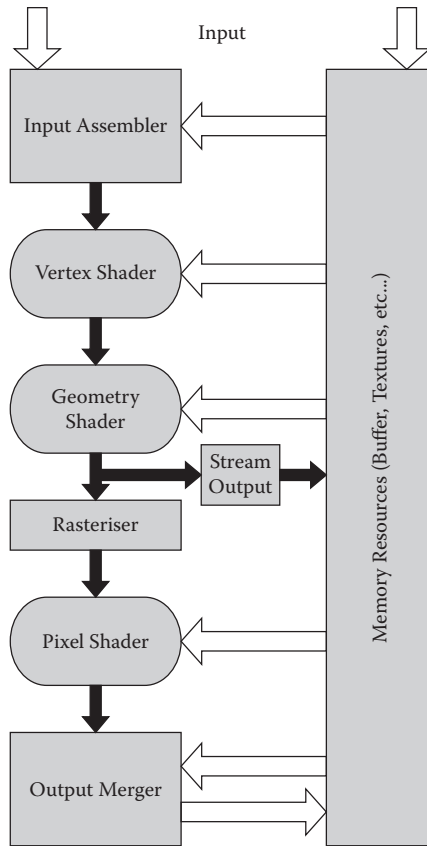


FIGURE 2.3 Programmable rendering pipeline (DirectX 11).

just positional information in 3D space but continuous depth data with additional dimensions and possibly its materials information as well. This type of spatial data is commonly used in scientific and medical work where cross-sectional information is important for evaluation and study. Volume rendering produces the exterior and the interior look of an object, usually with visual cues such as transparency and color differentiation. The image generation process considers the absorption of light along the ray path to the eye and volume rendering algorithms can be designed to avoid visual artifacts caused by aliasing and quantisation.

2.1.3 IMAGE-BASED RENDERING

In contrast to polygon-based rendering in which 3D geometry is provided for constructing the 3D hull of an object, image-based rendering techniques render novel 3D views by using a set of input images. This avoids the need for a stage where 3D data has to be explicitly provided by manual labour or some data acquisition means. These techniques focus on computer vision algorithms in feature detection and extraction from a set of basis images and thereafter reconstruct a 3D object or scene.

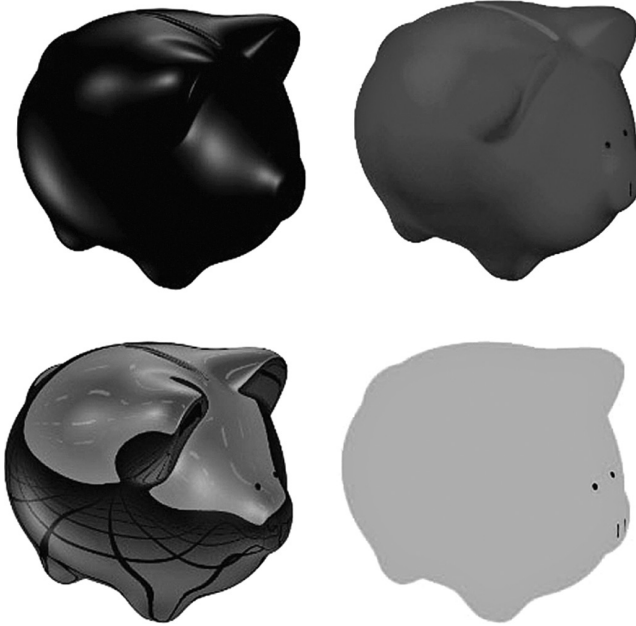


FIGURE 2.4 (See colour insert.) Samples of surface shading effects that can be achieved with pixel programs.

Image-based rendering techniques are often classified according to the degree by which geometry information is used. More importantly and in recent years, there has been a confluence of image-based techniques with polygon-based rendering in many applications due to the close continuum in 3D and 2D space in computer graphics.

As volume and image-based rendering are topics beyond the scope of this research, they are introduced here as auxiliary information on alternative 3D rendering techniques and more information can be found on the Internet and major research publication portals.

2.2 SYSTEM IDENTIFICATION

The goal of system identification is to derive a mathematical model of a dynamic system based on observed input and output data. Usually *a priori* information pertaining to a system will be useful for postulating the preliminary model structure. The system may then be modelled according to empirical data (black-box modelling) or conceivable mathematical functions such as physical laws (white-box modelling). Often, real world systems are non-linear and operate with reliance on state memory. The systems are dynamic and thus their outputs may depend on a combination of previous inputs, outputs, and states. The combination provides the basis for time series and regression mathematical expressions (models) for different reproducible systems.

System identification is an iterative procedure that can be summarised briefly by the flowchart in Figure 2.5. A model structure is chosen in advance based on

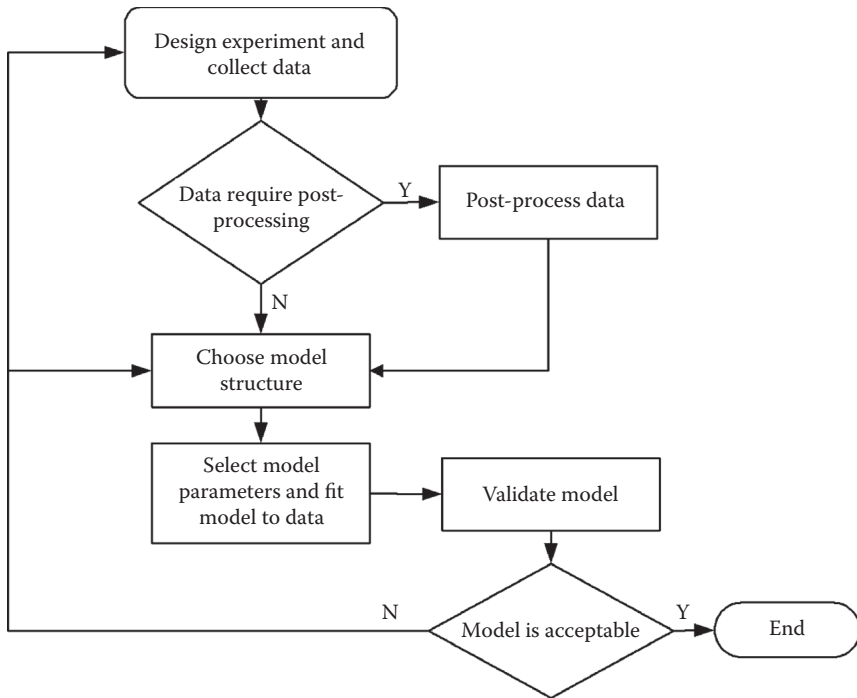


FIGURE 2.5 Process flow in system identification methodology.

preliminary information obtained from the system. The parameters of this model structure are then computed based on the set of experimental data collected previously. A portion of this data is allocated for model validation and the entire process from choosing a model structure to validation is repeated until the user-defined simulation performance criteria are met.

From a system identification perspective, we treat the real-time rendering process as the subject to be modelled. Since the rendering process cannot be described intuitively by physical laws such as mass, velocity, and temperature, black-box modelling is adopted. The system is first tested with a set of predefined inputs and the outputs are collected. This input–output dataset that captures a certain dynamic range of the behaviour of the system is then used with mathematical regression techniques to derive the estimated model.

Due to the scope of this book, we briefly summarise the steps in the system identification process below. A detailed and authoritative coverage of this topic can be found in Ljung’s book [1].

2.2.1 DATA COLLECTION

To obtain an effective model of a system, it is necessary for the measured data to capture and show the behaviour of the system adequately. An appropriate experimental design can ensure that the correct variables and dynamics of the system are

measured at sufficiently good resolution. In general, the following principles should be observed:

1. Select inputs that can excite the system dynamics adequately.
2. Minimise the effects of noise and disturbance to obtain a good signal-to-noise ratio.
3. Choose appropriate sampling intervals for measuring data.
4. Set a sufficient long duration of data collection to ensure capture of important time constants.

2.2.2 MODEL SELECTION

In system identification, we begin by determining the model structure best expressed by a mathematical relationship between input and output variables. This model structure typically provides the flexibility to describe a system based on certain parameters. Some examples of model structures include parameterised functions and state space equations. To illustrate, a linear parametric model is provided in the equation below.

$$y(k) = ay(k-1) + bu(k) \quad (2.1)$$

where u is the input, y , the output, k , the discrete time step and a and b are model structure variables.

Essentially, system identification is a systematic approach that begins with the selection of a model structure and then using approximation techniques to estimate the numerical values of the model parameters. While it may seem arbitrary to start with the selection of a model structure, it is not an entirely ad hoc process. The following approaches may be adopted in deciding on an appropriate model structure.

1. Start with the simplest system model structures to avoid unnecessary complexity in cases where the data can be modelled by a simple structure. Alternatively, a user can try various mathematical structures in a technique known as black-box modelling.
2. Designate a specific model structure for the data to be modelled by establishing certain predetermined principles; this technique is known as grey-box modelling.

Some well known system model structures from established research include the:

- Auto-regressive exogenous (ARX) model
- Auto-regressive moving average (ARMA) model
- Box–Jenkins model
- Output error model
- State space model

2.2.3 COMPUTING MODEL PARAMETERS

In system identification, the model parameters are estimated by minimising the function that describes errors between the derived system model output and the measured response. Assuming a system is linear and time-invariant, the output of the linear model y_{model} can be expressed as

$$y_{model}(t) = G(s)u(t) \quad (2.2)$$

where $G(s)$ is the transfer function, y the model output and u , the input to the model. To determine $G(s)$, we can minimise the difference between the model output $y_{model}(t)$ and the measured output $y_{meas}(t)$. We can use the minimisation criterion which is a weighted norm of the error $v(t)$:

$$v(t) = y_{meas}(t) - y_{model}(t) = y_{meas}(t) - G(s)u(t) \quad (2.3)$$

where $y_{model}(t)$ is either the model's simulated response given an input $u(t)$ or its predicted response given a finite series of past output measurements, i.e., $(y_{meas}(t-1), y_{meas}(t-2), \dots)$.

From the above, $v(t)$ is otherwise known as the simulation error or prediction error. The objective of the estimation algorithm is to generate a set of parameters in the model structure G such that eventually this error is minimised.

2.2.4 EVALUATING QUALITY OF DERIVED MODEL

The steps taken to evaluate the quality of a derived system model generally include the comparison of the model response to the measured response and the analysis of model residuals. Figure 2.6 compares the outputs of two different models with a measured output.

Residuals are differences between a model's one-step-predicted output and the measured data. In other words, residuals may be understood as portions of validation data that are not well described by the model. In residual analysis, the whiteness and independence tests are key performance indicators.

The whiteness test examines whether a model includes a residual auto-correlation function inside the confidence interval of the estimates. If it does, the model passes the test and the outcome indicates that the residuals are not correlated.

In addition, a model is qualified when it passes the independence test (no correlation between its residuals and past inputs). If evidence indicates such a correlation, the information revealing how the output relates to the input is incomplete. A simple example is an output $y(t)$ beyond the confidence interval during a lag k that originates from the input $u(t-k)$. A good model should perform both tests relatively well.

The system identification methodology accommodates an iterative process in the determination of the final model structure and parameters. A real world system may not be represented by only a single model structure. Whenever a derived model is found inadequate, it is necessary to revisit the model selection process, reconsider

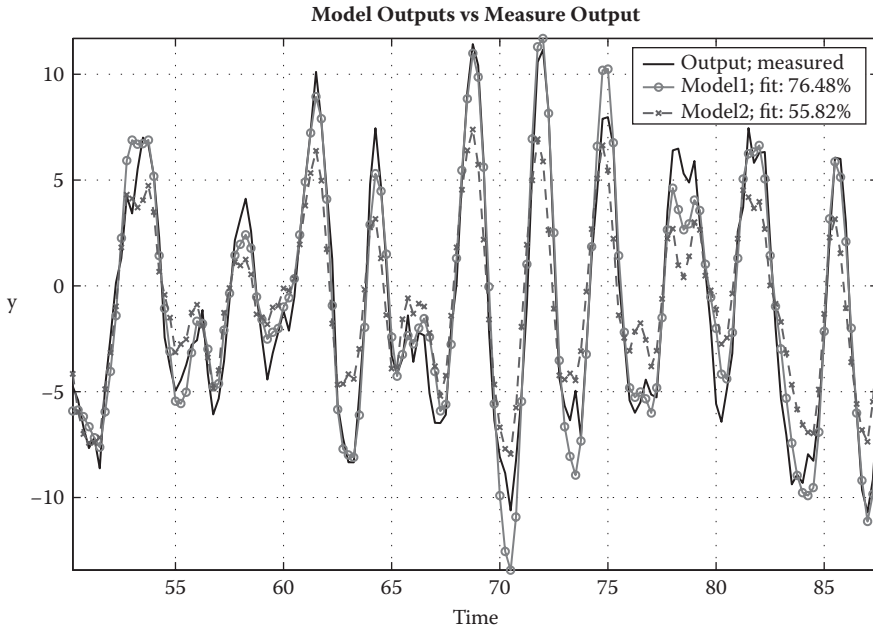


FIGURE 2.6 Comparison of two model outputs with measured system response.

the model parameter determination algorithm, and perhaps re-evaluate the data collection procedure.

2.3 LITERATURE REVIEW

While control theory is a mature field of study developed after the industrial revolution, the adoption of the techniques in the domain of computer software, particularly real-time computer graphics systems, remains nascent. This literature review provides a survey of research in these areas as background for our research.

2.3.1 COMPARATIVE STUDY ON EXISTING RESEARCH

The premise of the novelty in our research is founded upon close examination of previous work done in the fields of both real-time computer graphics and control theory, particularly those that have been successful in fusing the two disciplines and a careful thought process in terms of innovation in this area. A broad-stroke but systematic and progressive approach was taken to consider research publications within two decades to ensure that relevant techniques are not missed out regardless of their age and how they might contribute to further knowledge development.

Figure 2.7 shows the research comparative study flow conducted in this work which consists of the Classification Stage and the Qualitative Comparison Stage. In the Classification Stage, we begin with the most relevant keywords in the literature search terms. We consider the following words as the “lowest denomination”

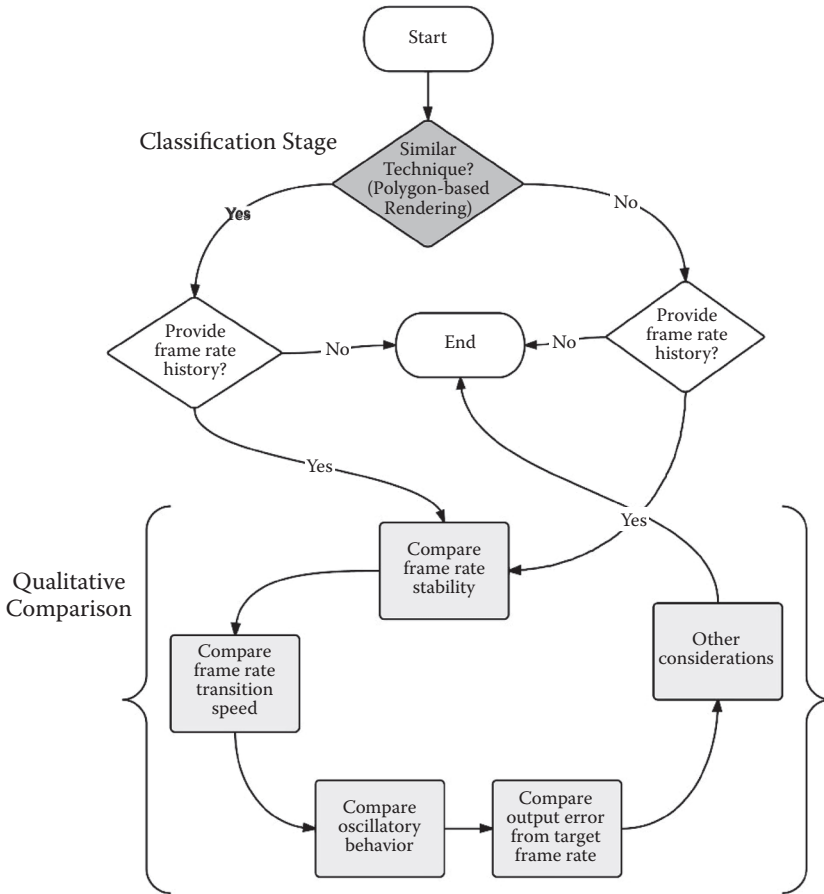


FIGURE 2.7 The comparative literature review workflow.

because of their relative importance in a subject matter. For example, omitting words such as “real-time”, “graphics” and “3D” since they are either rhetorical in computer graphics research or they may be replaced by stronger keywords such as “interactive”, “rendering” and directly meaningful candidates such as “frame rate” and “control”. These keywords are used in search fields in major research publication online portals which indexes the world’s largest collection of research literature. The gleaned process covers more than 500 research papers in a combined cohort of 4,000 search results from the publication portals.

As described in Section 1.3 in Chapter 1, the research in this thesis is primarily focused on polygon-based rendering technique which is predominant in common consumer and industrial applications such as computer games, virtual reality software and computer-aided design and prototyping. Hence, the Classification stage ends with segregating research literature that shares the same technique and is related to the topic of interactive 3D rendering. Table 2.1 shows the results from this classification stage from the initial pool of publications.

TABLE 2.1
Results from the Research Review Classification

	Polygon-based Rendering	Non-Polygon-based
With Frame Rate Data	[62] [64] [65] [67] [72] [74] [77] [81] [84] [85] [92]	[69] [73] [80] [83] [86] [87]
Without Frame Rate Data	[61] [63] [66] [68] [75] [78] [79] [82] [89] [90] [91]	[70] [71] [76] [88]

The next step in this comparative study is to select literature which provides experimental results on frame rate control since we need to conduct a qualitative analysis on them. For this purpose, these results should contain a history of data points in the time domain that demonstrate certain desirable qualities such as stability, offset errors and smooth frame rate transitions. These features would be compared to the results we obtain from the experiments conducted using the techniques proposed in this research with both qualitative and quantitative perspectives.

While research papers could be found relating to the topic of interactive rendering, however many of them alluded to concrete experiment results on sustainable performance as shown in the references from the bottom row of Table 2.1. In some cases [61] [71] [76], only static frame rates are given as an approximation to the interactive requirement. Furthermore, other researchers have chosen to work on volumetric [71] [76] [80] and image-based rendering [68] [70] [83] techniques which are prevalent in medical and large-scale visualization research but they differ from polygon-based rendering vastly. As a result, it is not straightforward to establish a direct comparison on the benefits offered by our research with these techniques. Despite these differences, we strive to provide a detailed qualitative and quantitative analysis on the aforementioned rendering architectures and their respective performance with our rendering framework in Chapter 8, Section 8.1.

2.3.2 CONTROL-THEORETIC APPROACHES TO COMPUTER SYSTEMS

As computer systems become increasingly complex through advances in hardware and software technology, traditional approaches to providing performance guarantees have become inefficient. In recent years, control engineering principles used successfully in real-world applications such as mechanical and electrical systems and process control have emerged as promising solutions to meet performance control challenges such as real-time scheduling, network bandwidth control, and power management in complex computer systems.

The comprehensive framework presented by Abdelzaher et al. [2] introduces feedback performance control in software services. The authors emphasised the importance of guaranteed quality of service (QoS) in modern computer software and systems that indicates the need for robust frameworks to achieve certain performance objectives. They further defined and explained the attributes of a QoS-aware service consisting of performance metrics such as queuing delays, execution latencies, and service response times. They also demonstrated that a software

system can be approximated by a linearised model with corresponding conceptual software representations of actuators and sensors. Although the feedback control architecture was provided for generic software systems, the entire work focuses on web server applications.

Abdelwahed et al. [3] proposed a generic online control framework to design self-managing computer systems. The control actions governing system operations were obtained by optimising system behaviour as forecasted by a mathematical model over a specified time horizon. The case studies cited deal with power management under time-varying workloads and signal detection accuracy and latency levels.

Since computer systems in networked environments are gaining importance due to increasing Internet usage, Li and Nahrstedt [4] proposed a task control model to illustrate the dynamics of QoS adaptations using digital control theory. The objective was to provide optimum resource allocation to tasks in a distributed environment where multiple applications compete for and share limited system resources, thus ensuring the best user experience and efficiency. A proportional, integral, derivative (PID) controller was used to achieve the desired performance objectives relating to the QoS metrics.

Hellerstein et al. [5] provided a comprehensive overview of the challenges in control engineering of computer systems. Similar studies were reported by Abdelzaher et al. [7], Lu et al. [8], and Karamanolis et al. [9].

2.3.3 CONTROL PRINCIPLES IN COMPUTER GRAPHICS SOFTWARE

In Li and Shen's work [10], a fuzzy logic controller serves as an automatic mechanism for controlling error tolerance in hierarchical volume rendering. Volume rendering is a technique for directly displaying a sampled 3D scalar field without first fitting geometric primitives to the 3D discrete sampled data set. The performance criterion is a user-defined frame rate that the control system will strive to achieve based on adjusting the error tolerance.

Sort-last rendering is a computer graphics applications technique for rendering extremely large datasets in clusters of computers, usually in a distributed environment. Kirihata et al. [11] showed that it is possible to use feedback control to harness large data transfer processes in sort-last rendering.

Another example of the adoption of control principles in computer graphics software is the work by Dayal et al. [97]. They proposed an adaptive form of frameless rendering with the potential to increase rendering speed dramatically over conventional interactive rendering. This is done without the rigid sampling patterns of framed renderers and by allowing sampling and reconstruction to adapt with very fine granularity over spatial-temporal colour changes. A sampler uses closed-loop feedback to guide sampling toward edges or motion in the image to maximise rendering efficiency.

To date, little research has focused on the adoption of control principles in computer graphics applications related to rendering. While the potential benefits are immense based on a broader perspective in which control techniques have been used successfully in generic software, the challenges usually lie in specific applications that require in-depth understanding and appropriate modelling before the control concepts may be introduced.

3 Linear Model Analysis of Real-Time Rendering

3.1 INTRODUCTION

The real-time computer graphics rendering process embodies complex state transitions and fast dynamics amidst observable steady-state behaviour. To yield realistic or visually useful graphical information, the rendering process may be loaded with myriad combinations of the input variables and states to the point where it is important to describe this function in simple terms.

In this chapter, we describe the application of system identification methodology to real-time rendering. The basis for such an approach is that the rendering process may be treated from a system perspective as a data processing function. This allows us to analyse the process input and output data to establish some formal relationship between them.

3.2 BACKGROUND

The perennial and increasing demands for fidelity, coupled with hardware architecture advancements, pose many challenges to researchers and developers as they endeavour to find the optimal solution to accommodate both speed and quality of rendering. To this end, key techniques developed since the evolution of computer graphics three decades ago revolve around their ability to reduce the rendering load at application runtime. They are largely based on the principles of visibility reduction, geometry decimation, image-based methods, and more recently, techniques such as programmable shading.

As space does not permit an exhaustive review of these research efforts, we refer the reader to the comprehensive surveys by Cohen-Or et al. [12], Haines [13], and Akenine-Moller et al. [14]. Despite the ability of each approach to reduce rendering loads during runtime, their common weakness lies in the inability to guarantee stable frame rates.

In this chapter, we introduce a novel framework for obtaining an accurate model of an interactive rendering process. This framework is based upon the system identification methodology [1] that is a mature field of study associated with systems and control theory.

In addition to expanding understanding of the dynamics relating to the rendering process, the objective of modelling this process is to establish the groundwork for a control framework. Only with an accurate model can we design this control framework around the rendering process to yield the sustainable performance we desire.

In this research, we focus on exploiting a current trend in hardware technology that provides fine resolution in geometry control, known as *tessellation*. Since geometry is the primitive construct of any object in 3D space, it becomes a natural choice as one of the modelling variables in our framework. In brief, tessellation is the process of sub-dividing surfaces into smaller shapes with the objective of generating higher resolution information of the 3D model. Tessellation, also known as a subdivision technique, is a well researched field in computer graphics and had been adopted widely in many interactive rendering applications because of the visual acuity it provides. However, only recently has graphics hardware provided sufficient support for tessellation-based techniques in applications [30].

We introduce our modelling framework via experiments in two interactive rendering applications that use subdivision techniques in rendering load control. We aim to establish the fundamental validity of a system-based approach to modelling the rendering processes in applications similar to those selected in these experiments. Since rendering tasks are inherently complex in real-world applications, we provide a systematic extension from a single-input–single-output (SISO) model of the rendering process to a multiple-input–single-output (MISO) model that more closely resembles a broader class of applications. We hope that this progression along with the system modelling principles and fundamental considerations related to 3D rendering will enable readers to appreciate the value of this framework and acquire the necessary knowledge for its implementation.

Current research in rendering workload characterisation [16,17] and rendering time estimation [18,19] strives to profile the attributes of rendering without providing a systematic way to control the process. Often, the user is expected to arrive at some form of a primitive control strategy based on profile information. This often requires several attempts to re-evaluate control strategy and ad hoc refinement steps are often needed to remove major rendering bottlenecks.

This motivated us to attempt to utilise a systems perspective to model the rendering process. In this chapter, we demonstrate that accurate models can be obtained via our data-driven framework and extend this framework by introducing the use of a controller that can track and regulate frame rates with guaranteed performance. In comparison with other work, our research offers the following benefits:

- Our framework does not require the pre-processing of the 3D content utilised in other research [20,21,22]. Its performance is not limited to static pre-processed geometry and scenes.
- Our approach leverages hardware-accelerated technology (tessellation) that provides smooth transitions in geometry scaling unlike techniques that may generate visual hysteresis [21,22,23].
- The outputs of the derived rendering models exhibit very high accuracy when compared to actual rendering process outputs.
- When the derived rendering model is used in conjunction with a suitable controller, the closed-loop system can produce guaranteed frame rates. The self-correction process occurs entirely online during runtimes unlike current techniques that may require repetitive and labour-intensive offline evaluation.

3.2.1 CONTROL-CENTRIC DEFINITION FOR RENDERING TIME CONTROL

In contrast to previous research on interactive and time-critical rendering [20,22,24], we define rendering time control as a mechanism that should produce stable frame rates very close to the user-defined target instead of fluctuating below it. To date, much research on rendering time control has focused loosely on keeping the time required to render each frame within a certain budget and ignoring the quality of the control or the fluctuations resulting from the technique. This leads to two consequences.

The first implies that the times allocated to perform other tasks in an interactive application such as logic computation, collision detection, and animation will not be consistent. In some cases, “starvation” of other processes that require CPU or GPU resources may occur. This is detrimental to the effectiveness of visual simulation applications in which external devices that require CPU cycles are tightly coupled to the rendering process.

Second, weak frame rate control leads to suboptimal resource use. For example, an object rendered at 15 FPS that achieves acceptable visual quality should not be rendered at 25 FPS unless allowed by the user for valid reasons. This requirement is especially critical in interactive applications and systems with tight resource control policies such as in game consoles [25,26] and portable devices where sustainable and guaranteed performance is vital because processor time must be allocated for related non-graphics computations. In contrast, applying control engineering leads to analysis of system attributes such as output overshoot, settling time, and steady-state errors that constitute a better qualitative framework for performance monitoring. We feel that this is a more powerful outcome than the current research focus on frame rate control.

3.2.2 CHALLENGES IN USING HEURISTICS

Heuristics usually refers to an experience-based speculative formulation of a solution to a problem. Much research in the area of rendering performance control has been based on heuristics and analytical models [22,23,24,27]. As Gobbetti and Bouvier noted [24]:

“...Static heuristics are not adaptive and are therefore inherently unable to produce uniform frame rates....”

Leukbe describes the difficulty in modelling the rendering process in his book on level of detail (LoD) for 3D graphics [28]:

“...a predictive scheduler estimates the complexity of the frame about to be rendered... this approach is substantially more complicated to implement...because it requires an accurate way to model how long the specific hardware will take to render a given set of polygons.”

The challenge in establishing reliable heuristics is straightforward. Driven by commercial demand and innovation, computer graphics hardware and software continue to change at unprecedented rates. In confirmation of this fact, Dumont et al. [29]

stated that given the complexity of real-rendering applications today, heuristics may fail in controlling rendering time. Haines [13] also describes this trend:

“Perhaps one of the most exciting elements of computer graphics is that some of the ground rules change over time. Physical and perceptual laws are fixed, but what was once an optimal algorithm might fall by the wayside due to changes in hardware, both on the CPU and GPU. This gives the field vibrancy: we must constantly re-examine assumptions, exploit new capabilities, and discard old ways.”

Based on these findings, dissecting the rendering process into distinct components that contribute to rendering cycle time is no trivial task. Tack et al. [18] did not consider overhead time in their performance model because of the complexity and additional costs it represented. The heuristics proposed in Wimmer and Wonker’s work [19] varied in performance for different applications. This implies that unless an application is specially built to fit into their proposed framework it may not be easy to achieve stable frame rates across a broader range of applications.

Heuristics ignore non-linearity in their formulation, that is, they assume that functional relationships are always linear. This is unrealistic in practical applications because of the underlying hardware. Our experiments have shown that the time taken to render a vertex varies at different total processed vertex counts. The work of Lakhia et al. on interactive rendering [22] demonstrated that texture size has a non-linear relationship with the time taken to render a 3D object. Finally, heuristics face the same challenges as other frame rate control mechanisms in terms of balancing qualitative requirements such as visual hysteresis [23] and rendering performance.

3.2.3 PURPOSE OF WORKLOAD CHARACTERISATION AND ANALYSIS

Apart from heuristics in the quest to limit rendering time, researchers also analysed rendering workloads with the goal of identifying and eradicating bottlenecks at runtime. Kyöstilä et al. [16] created a debugger and system analyser for graphics applications running on mobile hardware. Monfort and Grossman [17] attempted to characterise the rendering workloads of 3D computer games via a specially developed tool. In recent years, major graphics hardware vendors have provided software toolkits that allow low level access to their hardware for debugging and in-depth analysis of graphics workload with the goal of optimising performance of interactive applications during runtime.

However, workload characterisation and analysis are not adaptive mechanisms that will bring about stable frame rates. They are helpful only for tracing bottlenecks and manifesting an application’s rendering workload profile. To utilise these mechanisms for runtime performance, the process usually involves (1) identification of the problem (such as the cause of a bottleneck) during runtime followed by (2) manual effort to eradicate the bottleneck offline and then re-run the same scenario. This approach does not guarantee performance when the application use or 3D scene content changes. Since interactive rendering usually causes dynamic changes to visual content, the approach of using workload characterisation and tuning is not generally robust.

3.3 CASE FOR DATA-DRIVEN MODELLING

In system identification, we approach the problem of modelling a dynamic system from the observable data generated by its input and output. The case for using data-driven modelling is especially compelling for real-time rendering because the process is inherently complex. Rendering is a computer system process that thus raises considerations at both the hardware and software levels. Furthermore, unlike mechanical systems or chemical processes, no physical laws or intuitive functional relationships can be applied easily to achieve high accuracy.

Considering the real-time rendering process as a *black box* does not necessarily imply high risk of inappropriate modelling of the system as long as reasonable assumptions are based on a priori understanding of the system and can be reinforced from experiment results. In this book, we approach the challenge of modelling a rendering system by considering the expanded scopes of both single and multiple inputs. We also consider the output of the rendering process in terms of measurable quantities and the benefit of registering them as system outputs. This chapter discusses the inputs and outputs considered in system modelling and their eventual roles in system model representations.

To proceed with the modelling process, we first establish the relationship between the input and output of a system. This means that we must define and qualify the set of inputs and outputs before proceeding to identify their relationship. In the context of a real-time rendering application, it is reasonable to associate the geometry used for construction of 3D objects with the input to the rendering system and the output with the frame rate since empirical data indicate that they have an inverse relationship. Furthermore, in system identification, the input variables must be modifiable by the user in a straightforward manner. This is different from research in workload characterisation and heuristics where the defined variables are quantities such as hardware level parameters and processing time that cannot be changed by a user during runtime.

3.3.1 BASIS FOR SELECTION OF SYSTEM VARIABLES

With reference to the data flow in the computer graphics rendering pipeline shown in Figure 2.3 in Chapter 2, the inputs to the rendering process are obtained from memory resources (rectangle at far right) of the computer system. These inputs consist of various types of data ranging from geometry information to textures (image-related information) and rendering routines such as shader programs.

In order to define a set of variables to describe a rendering system, the input and output variables must be easily measurable. Furthermore, it is imperative that the input variables are controllable so that control actions can be implemented properly. Based on these criteria, we investigated the available performance counters with common low level graphics rendering profiler toolkits that included Microsoft's PIX. Table 3.1 shows a set of performance counters commonly used in many computer graphics applications.

Since many performance counters fall into the same category and are derivatives of one another, we chose the lowest denomination or most primitive variable in each

TABLE 3.1
Performance Counters in DirectX

Direct3D Counter Description	Official Name
FPS (#)	D3D FPS
Frame time in milliseconds	D3D frame time
Driver time in milliseconds	D3D time in driver
Triangle count (#)	D3D triangle count
Triangle count instanced (#)	D3D triangle count instanced
Batch count (#)	D3D batch count
Locked render targets count (#)	D3D locked render targets count
AGP/PCIE memory used in integer MB (#)	D3D agpmem MB
AGP/PCIE memory used in bytes (#)	D3D agpmem bytes
Video memory used in integer MB (#)	D3D vidmem MB
Video memory used in bytes (#)	D3D vidmem bytes
Total video memory available in bytes (#)	D3D vidmem total bytes
Total video memory available in integer MB (#)	D3D vidmem total MB

Source: NVPerfKit documentation from www.nvidia.com

selected category. To illustrate, the input geometry to the rendering pipeline may include lines, triangle fans, strips, and polygons. These are different input formats that share the same basis—3D geometry data. Hence the natural choice as the input variable of a rendering system should be the vertex count.

In addition to finding the appropriate variable by using its simplified form, another very important characteristic that determines suitability is whether a variable can be changed easily. For example, the batch counts and batch sizes of indexed buffers can impact the performance of a rendering system. However, little can be done to control these variables during an application runtime because these batches of vertices are predefined.

Finally, the resolution at which the selected variable may be adjusted affects the quality of the system model as well. The ideal case would involve a variable that allows fine resolution changes. For example, since the number of vertices is used as an input variable of a rendering system, it may be difficult to obtain an accurate model when this number can be varied only in limited steps.

One reason for this limitation is the underlying geometry LoD mechanism that controls the resolution of a 3D object with a certain topological objective and algorithm. The discrete LoD technique is an example of such a mechanism. Figure 3.1 illustrates the progressive variation (in steps) in the number of vertices that describe a 3D object. Conversely, other techniques such as progressive meshes and geometry tessellation allow 3D geometry variation at fine resolution levels. These techniques are preferred in comparison to the approaches cited earlier.

So far we have discussed guidelines for inputs to the rendering system. As for the output of the rendering system, the performance metric of primary concern to a user of real-time computer graphics is widely accepted as the frame rate (inverse of the time required to render one frame or image in a sequence) and quality of the generated imagery. The frame rate has a significant impact on the quality of the visual

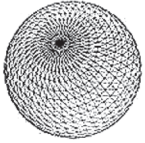
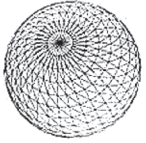
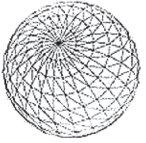
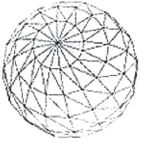

Image					
Vertices	~5500	~2880	~1580	~670	140
Notes	Maximum detail, for closeups.				Minimum detail, very far objects.

FIGURE 3.1 Visual effect of varying vertex count for 3D object in discrete steps. (Source: http://en.wikipedia.org/wiki/Level_of_detail#A_discrete_LOD_example)

experience offered by a real-time rendering application. While the quality of the generated imagery may be important to the user, the interactive experience is usually dominated by the application response rather than the quality of the generated imagery. Furthermore, quality is a subjective notion that complicates the adequacy of any useful metric.

3.4 LINEAR SYSTEM MODEL REPRESENTATION FOR REAL-TIME RENDERING

This section describes the modelling process applied to the real-time rendering system and the derivation of the mathematical models for various types of rendering applications. Using the system identification methodology, we demonstrate that linear time-invariant models can be obtained from the input and output data collected from experiments conducted using sample rendering applications.

A basic relationship between the input and output of a system may be expressed as a linear difference equation as follows.

$$y(t) + a_1y(t-1) + \dots + a_{n_a}y(t-n_a) = b_1u(t-n_k) + \dots + b_{n_b}u(t-n_k-n_b+1) + e(t) \quad (3.1)$$

where:

$a_1 \dots a_{n_a}$ and $b_1 \dots b_{n_b}$ are parameters to be estimated.

$y(t)$ is the output of the system at time t .

$y(t-1)$ and $y(t-n_a)$ are the previous outputs on which the current output depends.

$u(t-n_k)$ and $u(t-n_k-n_b+1)$ are the previous inputs on which the current output depends.

n_a is the number of poles of the system or the order of the system.

n_b represents the number of zeroes plus one.

n_k denotes delay in the system.

$e(t)$ equals noise.

An alternative way to represent Equation (3.1) in a more compact manner is the ARX model described below:

$$A(q)y(t) = B(q)u(t-n_k) + e(t) \quad (3.2)$$