ABSTRACT

HARMON, BRENDAN ALEXANDER. Tangible Landscape. (Under the direction of Eugene Bressler and Ross Meentemeyer.)

Tangible interfaces enable users to kinaesthetically interact with computations. Theoretically this should embody cognition in human-computer interaction so that users can cognitively grasp digital data with their bodies and offload challenging cognitive tasks onto their bodies. I have co-designed Tangible Landscape – a tangible interface powered by a geographic information system – and used it to study how tangible computing mediates spatial thinking.

Tangible Landscape couples a malleable, interactive physical model of a landscape with a digital model of the landscape so that users can 3D sketch ideas and immediately analyze them with scientific rigor. The physical and digital models are linked through a real-time cycle of physical interaction, 3D scanning, geospatial computation, and projection. As users interact with a physical model of a landscape – for example by sculpting the topography – the changes are scanned into a geographic information system (GIS), for geospatial analysis and simulation and the resulting analytics are projected onto the physical model. My colleagues and I have developed applications for Tangible Landscape including stormwater management, flood control, landscape management and erosion control, trail planning, viewshed analysis, solar analysis, wildfire management, disease spread management, invasive species management, urban growth, and sea level rise adaption.

In a series of terrain modeling experiments my colleagues and I studied how Tangible Landscape mediates spatial thinking and performance. We used quantitive and qualitative methods to assess participants' spatial performance using digital and tangible modeling. We used geospatial analytics such as geomorphometry and per-cell statistics supplemented by semi-structured interviews and direct observation to analyze multidimensional spatial performance. We found that 3D sketching with Tangible Landscape improves 3D spatial performance and enables a rapid iterative modeling process.

Copyright 2017 by Brendan Alexander Harmon
Creative Commons Attribution 4.0 International License

Tangible Landscape

by
Brendan Alexander Harmon

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Design

Raleigh, North Carolina

2017

APPROVED BY:

Art Rice	Helena Mitasova
Eugene Bressler Co-chair of Advisory Committee	Ross Meentemeyer Co-chair of Advisory Committee

BIOGRAPHY

Brendan is a PhD candidate at North Carolina State University with a Bachelor of Arts from Sewanee, the University of the South, a Master of Landscape Architecture from the Harvard Graduate School of Design, and a Master of Philosophy in Geography and the Environment from the University of Oxford.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **vi**

LIST OF FIGURES . . . . . **vii**

Chapter 1 Introduction **1**
1.1 Tangible interaction 1
1.2 Tangible geospatial interaction 1
1.2.1 Theory 2
1.2.2 Prototypes 2
1.3 Research 3
1.3.1 Methods 3
1.3.2 Results 3
1.4 Organization 4

Chapter 2 Embodied Spatial Thinking in Tangible Computing **8**
Introduction 9
Methodology 10
Preliminary Results 10
Future Work 11
Conclusions 11
Acknowledgements 11
References 12

Chapter 3 Tangible Landscape: Cognitively Grasping the Flow of Water **13**
Introduction 14
Methods 15
Results 17
Discussion 17
Conclusions 18
References 18

Chapter 4 Cognitively Grasping Topography with Tangible Landscape **21**
Introduction 22
Tangible Landscape 30
Coupling Experiment 37
Difference Experiment 51
Water Flow Experiment 55
Discussion 58
Future Work 63
Conclusions 65
Appendix A 66
Appendix B 68
Appendix C 70

Appendix D . . . 71
Appendix E . . . 72
Appendix F . . . 72
Appendix G . . . 72
References . . . 73

Chapter 5 Tangible Modeling with Open Source GIS . . . 78
Introduction . . . 89
System Configuration . . . 104
Building Physical 3D Models . . . 118
Basic Landscape Analysis . . . 139
Surface Water Flow and Soil Erosion Modeling . . . 151
Viewshed Analysis . . . 163
Trail Planning . . . 169
Solar Radiation Dynamics . . . 182
Wildfire Spread Simulation . . . 189
Coastal Modeling . . . 198
Appendix A . . . 203

APPENDICES . . . 218
Appendix A GIS-based environmental modeling with tangible interaction and dynamic visualization . . . 219
Introduction . . . 220
Methods . . . 221
Applications . . . 224
Discussion . . . 226
Conclusion . . . 227
Appendix B Integrating Free and Open Source Solutions into Geospatial Science Education . . . 228
Introduction . . . 230
Approach to Course Design and Implementation . . . 231
Course Implementation . . . 235
Future Directions . . . 238
Conclusions . . . 239
Appendix . . . 240
References . . . 242
Appendix C Open source approach to urban growth simulation . . . 244
Introduction . . . 245
FUTURES Model . . . 246
Integration in GRASS GIS . . . 246
Case Study . . . 248
Discussion . . . 250
Conclusion . . . 250
References . . . 250
Appendix D Immersive Tangible Geospatial Modeling . . . 252

Introduction 253
System Overview 253
Demonstration 255
Conclusion 256
References 256
Appendix E Conference talks 257

LIST OF TABLES

Table 3.1 Water depth statistics . . . 17
Table 3.2 Minimum distance statistics . . . 17
Table 3.3 Depression statistics . . . 17
Table 3.4 Depression percentages . . . 17

Table 4.1 Actuated pin table interfaces . . . 28
Table 4.2 Augmented architectural modeling interfaces . . . 28
Table 4.3 Augmented clay interfaces . . . 28
Table 4.4 Augmented sandbox interfaces . . . 29
Table 4.5 Bivariate scatterplots of elevation values . . . 43
Table 4.6 Coupling experiment: per-cell statistics and analyses . . . 45
Table 4.7 Students . . . 46
Table 4.8 Academics and professionals . . . 46
Table 4.9 Landscape architecture students . . . 47
Table 4.10 GIS students . . . 47
Table 4.11 3D novices . . . 48
Table 4.12 3D experts . . . 48
Table 4.13 Coupling experiment: percent cells . . . 49
Table 4.14 Mean minimum distance . . . 49
Table 4.15 Difference experiment . . . 53
Table 4.16 Difference experiment: per-cell statistics . . . 53
Table 4.17 Difference experiment: geospatial analyses . . . 53
Table 4.18 Difference experiment: percent cells . . . 53
Table 4.19 Difference experiment: comparison . . . 54
Table 4.20 Water flow experiment: per-cell statistics . . . 56
Table 4.21 Water flow experiment: geospatial analyses . . . 56
Table 4.22 Water flow experiment: percent cells . . . 56
Table 4.23 Water flow experiment: comparison . . . 57
Table 4.24 Interviews . . . 59
Table 4.25 How to build Tangible Landscape . . . 66

Table 5.1 Naming conventions for GRASS GIS modules . . . 112
Table 5.2 CNC settings: horizontal rough cut with MDF . . . 126
Table 5.3 CNC settings: parallel finish cut with MDF . . . 126
Table 5.4 Classes of the Anderson fuel model . . . 190

Table C.1 Predictors and estimated coefficients for the site suitability model . . . 249
Table C.2 Time and memory needed to run the simulations . . . 250

LIST OF FIGURES

Figure 2.1 3D sketching with Tangible Landscape 11
Figure 2.2 Standard deviation of differences 11
Figure 2.3 Reference model 12
Figure 2.4 Model made with projection augmented modeling 12
Figure 2.5 Model made with digital sculpting 12

Figure 3.1 Modeling the flow of water with Tangible Landscape 15
Figure 3.2 How Tangible Landscape works 15
Figure 3.3 Reference landscape with simulated water flow 15
Figure 3.4 Digitally sculpting and tangibly sculpting 16
Figure 3.5 Water depth anlaysis 19
Figure 3.6 Depression analysis 19
Figure 3.7 Difference analysis 19
Figure 3.8 Detail of difference analysis 19
Figure 3.9 Minimum distance for digital sculpting 20
Figure 3.10 Minimum distance for tangible sculpting 20

Figure 4.1 Modeling the flow of water with Tangible Landscape 23
Figure 4.2 How Tangible Landscape works 31
Figure 4.3 Modes of interaction with Tangible Landscape 31
Figure 4.4 Scanning arms and hands as topography 31
Figure 4.5 Naturally exploring subsurface soil moisture 32
Figure 4.6 Collaborative 3D sketching with VR 32
Figure 4.7 Software schema for Tangible Landscape 34
Figure 4.8 Accuracy assessment 35
Figure 4.9 Scanning accuracy 35
Figure 4.10 Casting polymeric sand models with 3D printed molds 36
Figure 4.11 Sculpting a terrain model using the difference analytic 37
Figure 4.12 Trail planning with Tangible Landscape 38
Figure 4.13 Testing coastal flood defences with Tangible Landscape 38
Figure 4.14 Managing the simulated spread of termites 38
Figure 4.15 Participants 39
Figure 4.16 Coupling experiment: digital modeling 41
Figure 4.17 Coupling experiment: analog modeling by hand 41
Figure 4.18 Coupling experiment: projection augmented modeling 41
Figure 4.19 Landforms identified by r.geomorphon 44
Figure 4.20 Pairwise comparison by category of participants 44
Figure 4.21 Water flow experiment 55
Figure 4.22 Tangible Landscape with robotic fabrication 65
Figure 4.23 Stickers 67
Figure 4.24 System setup 67

Figure 5.1 Simulated water flow projected over a physical model 91

Figure 5.2 Physical, digital, and hybrid modeling processes 91
Figure 5.3 inFORM 94
Figure 5.4 The Tangible Geospatial Modeling System 97
Figure 5.5 3D sketching a dam with Tangible Landscape 98
Figure 5.6 Collaboratively sculpting a pond 99
Figure 5.7 3D sketching a levee breach 100
Figure 5.8 System configuration 107
Figure 5.9 Mobile system setup 108
Figure 5.10 Laboratory system setup 109
Figure 5.11 Mesh generated by Kinect Fusion 114
Figure 5.12 Calibration 115
Figure 5.13 A CNC routed model of Sonoma Valley, CA 119
Figure 5.14 Sculpting a polymeric sand model 120
Figure 5.15 A laser cut contour model 123
Figure 5.16 A CNC routed model of Jockey's Ridge, NC 124
Figure 5.17 A CNC router carving a terrain model 125
Figure 5.18 How to prepare MDF for CNC routing 125
Figure 5.19 Magnetic markers 127
Figure 5.20 Modes of interaction using markers 127
Figure 5.21 Casting polymeric sand from 3D printed molds 128
Figure 5.22 Casting polymeric sand from CNC routed molds 129
Figure 5.23 A thermoformed polystyrene model 129
Figure 5.24 3D renderings of lidar data 130
Figure 5.25 3 x 3 neighborhood around an elevation value 142
Figure 5.26 Slope and aspect maps 143
Figure 5.27 Profile and tangential curvature 143
Figure 5.28 Landforms identified with geomorphons 144
Figure 5.29 Geomorphons 144
Figure 5.30 Lake Raleigh Woods 145
Figure 5.31 Basic workflow with DEM differencing 146
Figure 5.32 Slope and aspect of a scanned model 148
Figure 5.33 Profile and tangential curvature of a scanned model 148
Figure 5.34 Landform classification for a scanned model 150
Figure 5.35 Flow accumulation and watershed boundaries 156
Figure 5.36 Water depth computed with r.sim.water 157
Figure 5.37 Erosion and deposition for a scanned model 158
Figure 5.38 Lake Raleigh and it's surroundings 159
Figure 5.39 Lake Raleigh's current conditions and a new scenario 159
Figure 5.40 Flood simulation for current conditions 161
Figure 5.41 Flood simulation for new scenario 161
Figure 5.42 Line of sight analysis 164
Figure 5.43 Lake Raleigh Woods DSM 165
Figure 5.44 Placing a marker to identify a viewpoint 166
Figure 5.45 Viewsheds 167

Figure 5.46 Viewsheds from new buildings . . . 168
Figure 5.47 The TSP applied to a network of potential trails . . . 172
Figure 5.48 The slope and cross-slope of a trail . . . 172
Figure 5.49 Friction map for Lake Raleigh . . . 175
Figure 5.50 A trail around Lake Raleigh . . . 177
Figure 5.51 Trail analytics with Tangible Landscape . . . 179
Figure 5.52 Iteratively designing trails with Tangible Landscape . . . 180
Figure 5.53 Different configurations of buildings . . . 184
Figure 5.54 Shadows cast by different building configurations . . . 186
Figure 5.55 Direct solar irradiation . . . 187
Figure 5.56 Solar irradiation for different building configurations . . . 188
Figure 5.57 Orthophotograph and fuel map for Centennial Campus . . . 192
Figure 5.58 The simulated spread of fire without intervention . . . 194
Figure 5.59 Creating a firebreak with Tangible Landscape . . . 195
Figure 5.60 The spread of fire after creating a firebreak . . . 196
Figure 5.61 The spread of fire after creating another firebreak . . . 196
Figure 5.62 Jockey's Ridge, Nags Head, NC . . . 199
Figure 5.63 Simulated inundation for Jockey's Ridge . . . 200
Figure 5.64 Designs for pavilions on Roanoke Island, NC . . . 201
Figure 5.65 Siting pavilions with Tangible Landscape . . . 201
Figure 5.66 Simulated water flow and erosion for the pavilions . . . 202
Figure 5.67 Landforms in the High Tatras, Slovakia . . . 204
Figure 5.68 The evolution of Oregon Inlet . . . 204
Figure 5.69 Simulated storm surge for Rodanthe, NC . . . 205
Figure 5.70 Land change for a watershed in Charlotte, NC . . . 205
Figure 5.71 Reconstruction of a paleolake, Mongolia . . . 206
Figure 5.72 Sand models of beaches and streams . . . 207
Figure 5.73 Simulated breach of a coal ash pond . . . 208
Figure 5.74 Generating time-series data . . . 208
Figure 5.75 Interacting with disease and invasion models . . . 209
Figure 5.76 Exploring subsurface soil moisture . . . 210
Figure 5.77 Starting GRASS GIS . . . 213
Figure 5.78 Displaying raster maps . . . 213
Figure 5.79 Running GRASS GIS modules . . . 214

Figure A.1 Physical setup . . . 221
Figure A.2 Different methods of creating models . . . 222
Figure A.3 Real-time water flow modeling . . . 224
Figure A.4 The current condition of Lake Raleigh Woods . . . 225
Figure A.5 First development scenario . . . 225
Figure A.6 First modified development scenario . . . 225
Figure A.7 Second development scenario . . . 226
Figure A.8 Wildfire spread simulation . . . 226

Figure B.1 Instructions for the GRASS GIS part of an assignment . . . 232

Figure B.2 GIS GIS tutorial 234
Figure B.3 Geospatial Modeling and Analysis assignment reports 236
Figure B.4 Geospatial Modeling and Analysis exam and project paper 236
Figure B.5 Students visualizing a digital elevation model 237
Figure B.6 Students working on projects with Tangible Landscape 238
Figure B.7 Students presenting designs for a trail system 238

Figure C.1 Simplified schema of FUTURES conceptual model 246
Figure C.2 Diagram of FUTURES workflow 247
Figure C.3 An example of *r.future.demand* output plot 247
Figure C.4 2011 land cover and protected areas in the Asheville, North Carolina 248
Figure C.5 Three realizations of multiple stochastic runs with different scenarios 249
Figure C.6 Area in km2 of converted land for urban sprawl and infill scenarios 249

Figure D.1 Physical setup of Tangible Landscape and the proposed extension 254
Figure D.2 Application process diagram 254
Figure D.3 Interaction modes supported by Tangible Landscape 254
Figure D.4 Sculpting topography to form ponds 255
Figure D.5 Drawing patches of trees with a laser pointer 256
Figure D.6 Designing an optimized trail 256

1 Introduction

1.1 Tangible interaction

In a seminal paper Ishii and Ullmer envisioned tangible user interfaces that would 'bridge the gap between cyberspace and the physical environment by making digital information (bits) tangible' [13]. They described 'tangible bits' as 'the coupling of bits with graspable physical objects' [13]. Tangible interfaces – interfaces that couple physical and digital data [1] – are designed to physically manifest digital data so that users can cognitively grasp and absorb it, so that we can think with it rather than about it [14]. Since 'the physical state of tangibles embodies key aspects of the digital state of an underlying computation' [12] data is not just visualized, but can also be felt and directly, physically manipulated, leveraging highly developed motor skills. When interactions are embodied intention, action, and feedback should be seamlessly connected. Interaction should be automatic, subconscious, and intuitive.

1.2 Tangible geospatial interaction

Scientific computing can used to study and solve complex spatial problems. It can be challenging, however, to visualize and interact with complex spatial computations. Tangible interfaces for geospatial modeling give spatial computations an interactive, physical form that can be kinaesthetically explored and transformed. Users should be able to intuitively grasp, understand, and manipulate spatial computations. These tangible interfaces should computationally augment users' spatial cognition, while offloading the cognitive tasks of interaction and perception onto their bodies.

1.2.1 Theory

Theoretically tangible interfaces for geospatial modeling should enable intuitive digital sculpting, streamline design processes by seamlessly integrating analog and digital workflows, and improve collaboration and communication by enabling multiple users to simultaneously interact in a natural way [21]. GIS provide the extensive libraries for geospatial databasing, processing, analysis, modeling, simulation, and visualization needed to address real world problems [23]. Tangible interfaces for geospatial modeling have the potential to make spatial science more intuitive by enabling a rapid iterative process of observation, hypothesizing, testing, and inference. With these systems users can for example dynamically explore how topographic form influences landscape processes [15].

A review of tangible interfaces for geospatial modeling (Tables 4.1-4.4), however, reveals a dearth of critical research. While many prototypes have been developed there have been few case studies or qualitative user studies and no quantitative studies. The theories that have inspired tangible interfaces for geospatial modeling have not yet been critiqued or proven.

1.2.2 Prototypes

Tangible interfaces like Urp [24], Illuminating Clay [20], and SandScape [21] enriched physical models of urban spaces and landscapes with spatial analyses and simulations like wind direction, cast shadow, slope, aspect, curvature, and water direction in order to enhance and streamline design processes. Many of the analyses used in Illuminating Clay were adapted from GRASS GIS [20] and eventually Illuminating Clay was coupled with GRASS GIS to draw on its extensive libraries for spatial computation. The aim of coupling Illuminating Clay with GRASS GIS was to ‘explore relationships that occur between different terrains, the physical parameters of terrains, and the landscape processes that occur in these terrains’ [15]. The research to couple a physical landscape model with GRASS GIS led to the development of the Tangible Geospatial Modeling System [23] and Tangible Landscape [17, 18]. Tangible Landscape couples a malleable physical model and a digital geospatial model of a landscape through a real-time cycle of physical interaction, 3D sensing, geospatial computation, and projection. Tangible Landscape was designed for natural and collaborative interaction with GIS. Such natural interaction with geospatial models should enhance spatial cognition and encourage spatial learning.

1.3 Research

The aim of this research was to make scientific spatial computing natural, rapid, and collaborative in order to streamline spatial science and design. The objective was to design a tangible interface for geospatial modeling and study how it mediates spatial cognition in order to implement, study, critique, and refine theories about tangibles and spatial thinking. Through an iterative cycle of design and research my colleagues and I addressed the following research questions:

- How can we design an effective tangible interface for geospatial modeling?
- How can we use tangible geospatial analytics to solve spatial problems?
- How does coupling physical and digital models of topography mediate spatial thinking?
- How do different tangible geospatial analytics mediate spatial thinking?
- How do tangible geospatial analytics mediate how users relate form and process?

1.3.1 Methods

I co-designed Tangible Landscape and ran a series of terrain modeling experiments studying how tangible geospatial modeling mediates spatial cognition and performance. The experiments studied how coupling physical and digital models and providing real-time analytic feedback affected 3D spatial performance. Spatial performance was assessed using qualitative methods including semi-structured interviews and direct observation and quantitive methods including geospatial modeling, analysis, simulation, and statistics. This research has informed the continued design and development of Tangible Landscape.

1.3.2 Results

Through this research we developed a unique tangible interface for geospatial modeling – the only real-time system powered by a GIS and capable of a wide variety of tangible geospatial models, analyses, and simulations [5, 17, 18]. We implemented, demonstrated, and tested many novel applications for tangible geospatial modeling [5, 17–19]. We also conducted the first quantitative, spatial studies of tangible geospatial modeling. We found that coupled digital and physical models can improve 3D modeling performance [2, 5]. We also found that tangible

geospatial analytics like differencing and water flow can enable iterative modeling processes [5] and help users connect form to processes [5, 10].

1.4 Organization

This thesis is a collection of papers and a book about Tangible Landscape.

Chapter 2 describes a pilot study for the first terrain modeling experiment focused on coupling digital and physical models. This chapter is a reprint of the paper *Embodied Spatial Thinking in Tangible Computing* from the Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction [2].

Chapter 3 describes a pilot study for the third terrain modeling experiment using Tangible Landscape's water flow analytic. This chapter is a reprint of the paper *Tangible Landscape: Cognitively Grasping the Flow of Water* from the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences [10]

Chapter 4 describes the design of Tangible Landscape and then describes the series of terrain modeling experiments. This chapter is a draft of the paper *Cognitively Grasping Topography with Tangible Landscape*, which has been submitted for publication [5].

Chapter 5 describes the design of Tangible Landscape, its applications, how to set it up, and how use it. This chapter is a reprint of the book *Tangible Modeling with Open Source GIS* [18].

Appendix A describes the first generation of Tangible Landscape and demonstrates applications. This appendix is a reprint of the paper *GIS-based environmental modeling with tangible interaction and dynamic visualization* published in the Proceedings of the 7th International Congress on Environmental Modelling and Software [17].

Appendix B describes how free, open source software can be integrated into geospatial education to promote open and reproducible science. Tangible Landscape is used in a couple the case studies demonstrating this approach. This appendix is a reprint of the paper *Integrating Free and Open Source Solutions into Geospatial Science Education* published in the ISPRS International Journal of Geo-Information [16].

Appendix C describes the integration of the FUTURES (FUTure Urban – Regional Environment

Simulation) model into GRASS GIS and makes a case for GRASS GIS as a platform for free, open source geospatial software development. This appendix is a reprint of the paper an *Open source approach to urban growth simulation* published in The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences [19].

Appendix D describes the coupling of Tangible Landscape with an immersive virtual environment so that users can visualize and design landscapes at a human-scale. This appendix is a preprint of the paper *Immersive Tangible Geospatial Modeling* that has been accepted for publication in the Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems [22].

Appendix E links conference talks related to Tangible Landscape [3, 4, 6–9, 11].

BIBLIOGRAPHY

1. Dourish, P. Where the action is: the foundations of embodied interaction. Cambridge, MA: MIT Press, 2001.
2. Harmon, B. A. Embodied spatial thinking in tangible computing. *Proc. TEI '16*, 693–696.
3. Harmon, B. A. Embodied spatial thinking in tangible computing. *TEI '16: Tangible, Embedded, and Embodied Interaction 2016*. 2016.
4. Harmon, B. A. Tangible geospatial modeling for landscape architects. *2014 Geodesign Summit*. 2014.
5. Harmon, B. A. Cognitively Grasping Topography with Tangible Landscape. Submitted for publication. 2016.
6. Harmon, B. A. Creative spatial thinking with Tangible Landscape. *American Association of Geographers Annual Meeting 2016*. 2016.
7. Harmon, B. A. Serious Gaming with Tangible Landscape. Raleigh, NC, 2016.
8. Harmon, B. A. Tangible geographies. *Royal Geographical Society Annual International Conference 2016*. 2016.
9. Harmon, B. A. Tangible interaction for GIS. *FOSS4G NA 2016*. 2016.
10. Harmon, B. A. Tangible Landscape: cognitively grasping the flow of water. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 2016.
11. Harmon, B. A. Tangible Landscape: cognitively grasping the flow of water. *International Society of Photogrammetry and Remote Sensing 2016*. 2016.
12. Ishii, H. Tangible bits: beyond pixels. *Proc. TEI '08*, xv–xxv.
13. Ishii, H., & Ullmer, B. Tangible bits: towards seamless interfaces between people, bits and atoms. *Proc. CHI '97*, 234–241.
14. Kirsh, D. Embodied cognition and the magical future of interaction design. *ACM Trans. Comput.-Hum. Interact.* **20**(1), 2013: 3:1–3:30.
15. Mitasova, H. Real-time landscape model interaction using a tangible geospatial modeling environment. *IEEE Computer Graphics and Applications*, **26**(4), 2006: 55–63.
16. Petras, V. Integrating Free and Open Source Solutions into Geospatial Science Education. *ISPRS International Journal of Geo-Information*, **4**(2), 2015: 942–956.
17. Petrasova, A. GIS-based environmental modeling with tangible interaction and dynamic visualization. *Proceedings of the 7th International Congress on Environmental Modelling and Software*. 2014.

18. Petrasova, A. Tangible Modeling with Open Source GIS. Springer, 2015.
19. Petrasova, A. Open source approach to urban growth simulation. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 2016, 953–959.
20. Piper, B. Illuminating clay: a 3-D tangible interface for landscape analysis. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '02*. 2002, 355.
21. Ratti, C. Tangible User Interfaces (TUIs): A Novel Paradigm for GIS. *Transactions in GIS*, **8**(4), 2004: 407–421.
22. Tabrizian, P. Immersive Tangible Geospatial Modeling. *Proc. SIGSPATIAL'16*.
23. Tateosian, L. TanGeoMS: tangible geospatial modeling system. *IEEE transactions on visualization and computer graphics*, **16**(6), 2010: 1605–12.
24. Underkoffler, J., & Ishii, H. Urp: a luminous-tangible workbench for urban planning and design. *Proc. CHI '99*, 386–393.

2 Embodied Spatial Thinking in Tangible Computing

Reprint

Brendan A. Harmon. 2016. Embodied Spatial Thinking in Tangible Computing. In *TEI '16: Proceedings of the Tenth International Conference on Tangible, Embedded, and Embodied Interaction*. Eindhoven, Netherlands: ACM Press, 693–696. DOI:`http://dx.doi.org/10.1145/2839462.2854103`

Attribution

I co-designed Tangible Landscape, designed and ran this experiment, and wrote this paper. Anna Petrasova, Vaclav Petras, and Helena Mitasova contributed to this research by collaboratively designing Tangible Landscape, running the experiment, and editing the paper. Ross Meentemeyer contributed with advice, guidance, support, and editing.

Embodied Spatial Thinking in Tangible Computing

Brendan Alexander Harmon
North Carolina State University
Raleigh, USA
brendan.harmon@gmail.com

ABSTRACT
Tangible user interfaces are based on the premise that embodied cognition in computing can enhance cognitive processes. However, the ways in which embodied cognition in computing transform spatial thinking have not yet been rigorously studied. I have co-designed Tangible Landscape – a continuous shape display powered by a geographic information system – and used it to explore how technology mediates spatial cognition in a rigorous experiment.

In this terrain modeling experiment I use geospatial analytics to analyze how visual computing with a GUI and tangible computing with a shape display mediate multidimensional spatial performance.

My initial findings suggest that: **1.** digital sculpting via a GUI is unintuitive, **2.** shape displays like Tangible Landscape can be intuitive, enhance spatial performance, and enable rapid iteration and ideation, and **3.** different analytics encourage significantly different modes of spatial thinking and strategies for modeling.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces – Evaluation/methodology; Interaction styles; Prototyping; Theory and methods

Author Keywords
Human-computer interaction; tangible user interfaces; tangible interaction; embodied cognition; spatial thinking; 3D modeling

INTRODUCTION
In embodied cognition the mind is embedded in the body. Higher cognitive processes rely on lower level processes such as emotion and sensorimotor processes that link perception and action [2]. Thus body and action mediate thought. Cognitive processes can be physically simulated with cognition offloaded onto action. Objects such as tools can be cognitively grasped and temporarily, contingently incorporated into ones body schema [5]. This view of cognition considers feeling, action, and perception to be functionally integral to thought.

Based on the theory of embodied cognition researchers have theorized a physical-digital divide in human-computer interaction – positing that the high level of abstraction required to interact with a computer via a graphical user interface (GUI) in a visual computing paradigm constrains how we think – and designed innovative technologies for bridging this divide such as tangible user interfaces (TUIs) [3, 1]. TUIs are designed to physically manifest digital data as tangible bits that afford the ability to directly, physically feel and manipulate data [3]. TUIs make pragmatic representations of digital data. Pragmatic representations are conceptual models for rapidly generating action that are primarily based on tactile feedback and are processed automatically, immediately, and subconsciously [4]. By physically manifesting data so that we can pragmatically interact with computations, TUIs should enable rapid, intuitive action and expression in a way that was not possible in visual computing. By enabling embodied cognition in human-computer interaction TUIs should let us cognitively grip data as an extension of our bodies, intuitively manipulate data, and physically simulate processes. This should enhance spatial thinking.

Research on TUIs has been more focused on the design and development of the technology than empirically testing the use and effects of the technology. Rasmussen et al. for example discuss the focus on the design and development of shape displays rather than empirical research about their use. [7]. Research on tangible interaction has focused on user experience and to a lesser degree user performance using methods such as participant observation, semi-structured interviews, video analysis, and small-scale experiments.

While the theory of embodied cognition has inspired the design of novel technologies for human-computer interaction, its applied principles need to be empirically tested, critiqued, and refined in order to improve interaction design. The aim of this research is to study how tangible computing mediates spatial thinking in order to improve interaction design, enhance spatial performance, and improve spatial education.

Testable hypotheses
I hypothesize that:

1. limited feedback constrains spatial performance,
2. misinterpretations of 3D space in visual computing reduce spatial performance,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
TEI '16, February 14-17, 2016, Eindhoven, Netherlands.
Copyright © 2016 ACM ISBN/978-1-4503-3582-9/16/02$15.00.
DOI http://dx.doi.org/10.1145/2839462.2854103

3. computational analytics like differencing and flow simulations can enhance spatial performance,

4. computationally enriched analytical thinking can be coordinated with embodied thinking,

5. tangible computing that computationally enhances embodied spatial thinking can improve spatial performance, and

6. tangible computing can enable a rapid iterative process of exploratory, generative form-finding that can improve spatial performance.

METHDOLOGY

I have conducted a rigorous experiment using geospatial analytics to compare spatial performance in analog modeling by hand, in tangible computing using Tangible Landscape, and in visual computing using 3D modeling programs via a GUI. Tangible Landscape is a continuous shape display tightly integrated with GRASS GIS that was designed to intuitively 3D sketch landscapes (Figure 1). Conceptually, Tangible Landscape couples a physical model with a digital model in a real-time feedback cycle of 3D scanning, geospatial modeling and simulation, and projection (Figure 1a) [6].

I have begun to test my hypotheses using quantitative methods Rather than studying basic spatial cognitive abilities that can not usefully be disentangled in applied use, I am studying applied spatial cognitive processes using the metrics and parameters of the given application. I am studying how participants perform in a series of time constrained terrain modeling exercises using analog modeling, visual computing, and tangible computing. 6 landscape architects and 6 geospatial analysts have participated in this pilot study. I am assessing their performance with each technology using spatial statistics, topographic parameters, morphology, process-form interactions, and differencing.

Terrain modeling experiment

The 1st exercise tests spatial performance in analog modeling to establish the baseline spatial ability of each participant. In this exercise participants sculpt a given terrain in polymeric sand using their hands with a CNC routed model for reference.

The 2nd exercise tests spatial performance in projection augmented analog modeling to establish a baseline for spatial performance in tangible computing. In this exercise participants sculpt a given terrain in polymeric sand using a projected map of the given elevation and contours as a guide.

The 3rd-5th exercises test spatial performance in tangible computing augmented with different analytics. In the 3rd exercise participants use Tangible Landscape to sculpt a given terrain in polymeric sand using the given and scanned contours as a guide. In the 4th exercise participants use Tangible Landscape to sculpt a given terrain in polymeric sand using the difference between the given and the scanned landscape as a guide. In the 5th exercise participants use Tangible Landscape to sculpt a given terrain in polymeric sand using the simulated water flow over the scanned landscape as a guide in order to study 4D thinking.

The 6th and 7th exercises test spatial performance in visual computing. In the 6th exercise participants digitally sculpt a given terrain in Rhinoceros, a 3D modeling program, using the given 3D contours as a guide. In the 7th exercise participants digitally sculpt a given terrain in VUE, a 3D modeling program designed for intuitive terrain sculpting, with a CNC routed model for reference.

Data collection and analysis

In the 1st-5th exercises the resulting physical models are 3D scanned, imported into GRASS GIS as point clouds, and interpolated as DEMs for analysis. In the 6th and 7th exercises the resulting digital models are exported as point clouds, imported into GRASS GIS, and interpolated as DEMs for analysis. For each model I compute the elevation and its histogram and bivariate scatterplot, the slope, the difference and its histogram, the simulated water flow, and morphology. For each exercise I compute a map of the coefficient of variance of all the of DEMs, bivariate scatterplots of the covariance and correlation matrix of all the of DEMs, and the mean and the absolute value of the mean of the differences between the the reference terrain and the modeled terrain.

Code

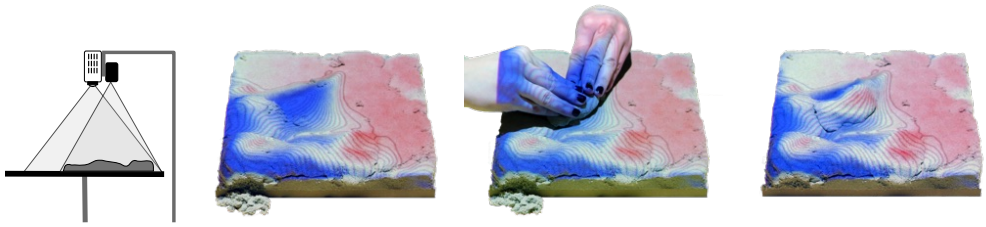
As a work of open science the data and the python scripts for data analysis are publicly available on GitHub at https://github.com/baharmon/tangible_topography released under the GNU General Public License version 2. Both GRASS GIS and Tangible Landscape are open source projects released under the GNU General Public License version 2. Tangible Landscape is available on GitHub at https://github.com/ncsu-osgeorel/grass-tangible-landscape.

PRELIMINARY RESULTS

Select results for one participant in the pilot study are shown in Figures 3, 4, and 5.

In the 1st exercise participants tended to model the overall form well, but exaggerated the slopes and misplaced the main ridge. In the 2nd exercise participants tended to model the landscape very well, accurately capturing the ridge and valley, but missing the saddle of the valley (Figure 2a). In the 3rd exercise participants tended to model the overall form well, but approximately with indistinct landforms and a lot of roughness. Many participants used a layered *pancake*-modeling strategy here. The 4th exercise tended to have the best results. Participants used a rapid iterative process to accurately model the landscape. In the 5th exercise participants tended to model the overall form well, but exaggerated the slopes of the stream channels. In the 6th exercise most participants modeled the form very poorly especially the interior space (Figure 2b) – either modeling only the edges or modeling very approximate massings of the main ridge and peak. Expert 3D modelers, however, modeled the form better, but still missed much of the interior space. In the 7th exercise participants tended to model the landscape suggestively in the x- and y-dimensions, but underestimated and grossly distorted the z-dimension.

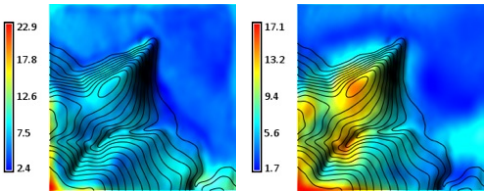
The pilot study suggests that **1.** analog sculpting by hand can be highly intuitive, but the lack of analytical feedback limits



(a) (b) (c) (d)

Figure 1. 3D sketching with Tangible Landscape: (a) a cycle of scanning and projection couples a physical and digital model, (b) the differencing analytic with blue where sand should be added and red where sand should be removed, (c) a participant adding sand, and (d) the updated differencing analytic.

ones ability to critique the model and refine its form, **2.** shape displays like Tangible Landscape can be intuitive, enhance spatial performance, and enable rapid iteration and thus creative spatial learning. **3.** digital sculpting via a graphical user interface is unintuitive and visually ambiguous leading to misinterpretations of depth, height, and interior space.



(a) (b)

Figure 2. Standard deviations of the differences for all participants: (a) in Ex. 2 using projection augmented modeling and (b) in Ex. 6 using Rhinoceros.

FUTURE WORK

The full study will have approximately 40 participants. Based on the results of the study I hope to develop a theory of how technology mediates spatial cognition. In the long-term I plan to expand this study of embodied spatial cognition in tangible computing as a collaborative research project with psychologists using spatial analytics, eye tracking, biometric sensors, and EEGs.

CONCLUSIONS

This study is unique in applying geospatial analyses to asses spatial performance in tangible interaction. My initial findings suggest that shape displays like Tangible Landscape can with the right computational analytics enhance performance in novel ways and enable rapid iterations of generative form-finding and critical analysis. Different analytics encourage significantly different modes of spatial thinking and strategies for modeling. The differencing analytic leads to significant improvements in spatial performance and enables an iterative process.

ACKNOWLEDGMENTS

My colleagues in the NCSU Open Source Geospatial Research and Education Lab (http://geospatial.ncsu.edu/osgeorel/) Anna Petrasova, Vaclav Petras, Helena Mitasova, and Ross Meentemeyer contributed to this research by collaboratively designing Tangible Landscape and helping to run the experiment.

REFERENCES

1. Paul Dourish. 2001. *Where the action is: the foundations of embodied interaction*. MIT Press, Cambridge, MA.
2. Benoit Hardy-Vallée and Nicolas Payette. 2008. *Beyond the Brain: Embodied, Situated and Distributed Cognition*. Cambridge Scholars Publishing, Newcastle, UK.
3. Hiroshi Ishii and Brygg Ullmer. 1997. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*. ACM Press, 234–241. DOI: http://dx.doi.org/10.1145/258549.258715
4. Marc Jeannerod. 1997. *The Cognitive Neuroscience of Action*. Blackwell, Cambridge, MA. 236 pages.
5. David Kirsh. 2013. Embodied cognition and the magical future of interaction design. *ACM Transactions on Computer-Human Interaction* 20, 1 (2013), 3:1–3:30. DOI:http://dx.doi.org/10.1145/2442106.2442109
6. Anna Petrasova, Brendan Harmon, Vaclav Petras, and Helena Mitasova. 2015. *Tangible Modeling with Open Source GIS*. Springer.
7. Majken K Rasmussen, Esben W Pedersen, Marianne G Petersen, and Kasper Hornbaek. 2012. Shape-Changing Interfaces : A Review of the Design Space and Open Research Questions. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems CHI 12* (2012), 735–744. DOI: http://dx.doi.org/10.1145/2207676.2207781

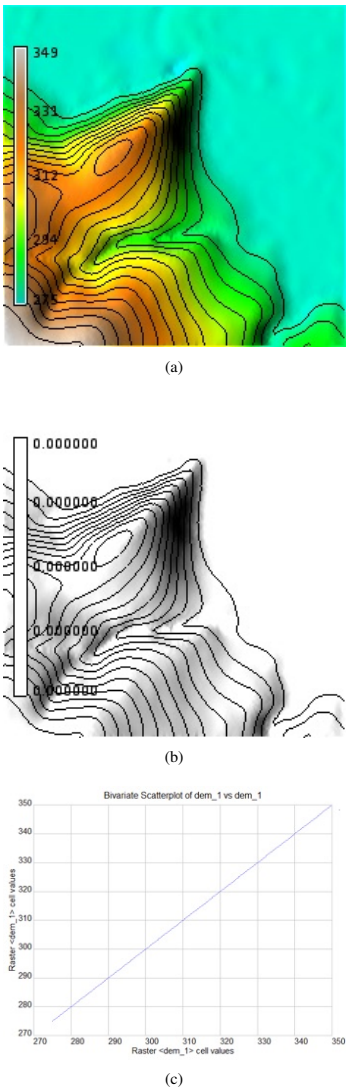


Figure 3. **Reference model:**
(a) the digital elevation model (DEM),
(b) the difference between the reference DEM and the reference DEM, and
(c) the bivariate scatterplot of the difference between the reference DEM and the reference DEM.

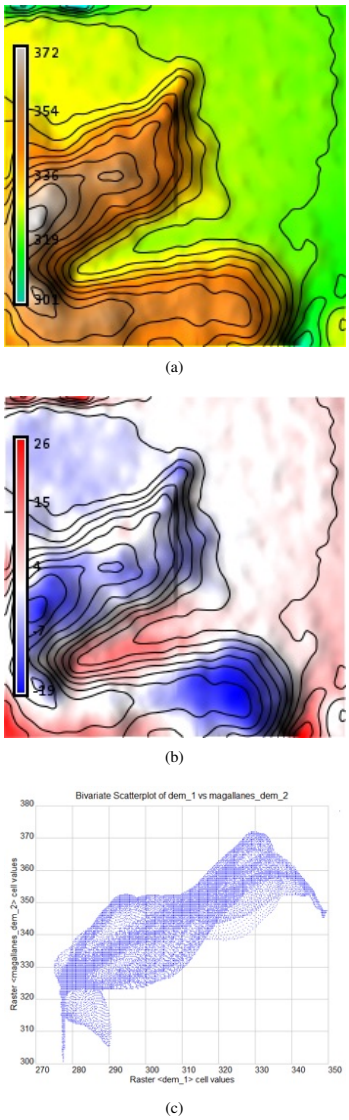


Figure 4. **Participant's model for Ex. 2 using projection augmented modeling:**
(a) the DEM,
(b) the difference between the participant's DEM and the reference DEM, and
(c) the bivariate scatterplot of the difference between the participant's DEM and the reference DEM.

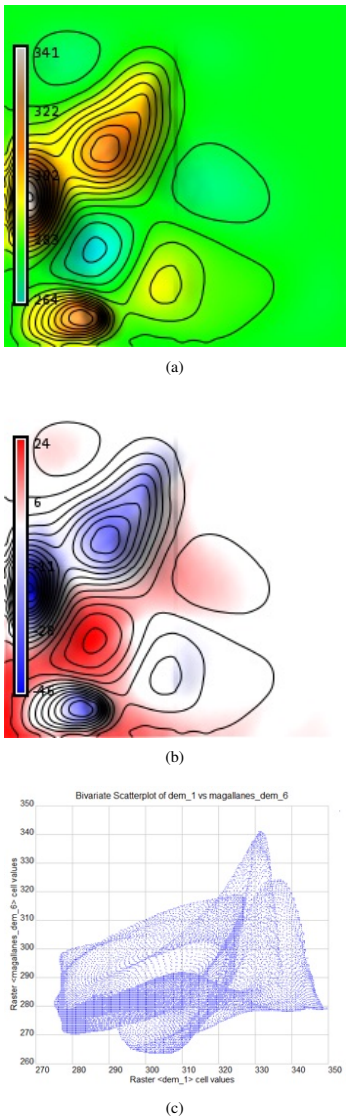


Figure 5. **Participant's model for Ex. 6 using digital sculpting with Rhinoceros:**
(a) the DEM,
(b) the difference between the participant's DEM and reference DEM, and
(c) the bivariate scatterplot of the difference between the participant's DEM and the reference DEM.

3 Tangible Landscape: Cognitively Grasping the Flow of Water

Reprint

Brendan A. Harmon, Anna Petrasova, Vaclav Petras, Helena Mitasova, and Ross K. Meentemeyer. 2016. Tangible Landscape: cognitively grasping the flow of water. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Prague: International Society of Photogrammetry and Remote Sensing. DOI:`http://dx.doi.org/10.5194/isprsarchives-XLI-B2-647-2016647`

Attribution

I co-designed Tangible Landscape, designed and ran this experiment, and wrote this paper. Anna Petrasova, Vaclav Petras, and Helena Mitasova contributed to this research by collaboratively designing Tangible Landscape, running the experiment, and editing the paper. Ross Meentemeyer contributed with advice, guidance, support, and editing.

The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLI-B2, 2016
XXIII ISPRS Congress, 12–19 July 2016, Prague, Czech Republic

TANGIBLE LANDSCAPE: COGNITIVELY GRASPING THE FLOW OF WATER

B. A. Harmon[a]*, A. Petrasova[a], V. Petras[a], H. Mitasova[a], R. K. Meentemeyer[a]

[a] Center for Geospatial Analytics, North Carolina State University - (baharmon, akratoc, vpetras, hmitaso, rkmeente)@ncsu.edu

KEY WORDS: embodied cognition, spatial thinking, physical processes, water flow, hydrology, tangible user interfaces, user experiment, 3D

ABSTRACT:

Complex spatial forms like topography can be challenging to understand, much less intentionally shape, given the heavy cognitive load of visualizing and manipulating 3D form. Spatiotemporal processes like the flow of water over a landscape are even more challenging to understand and intentionally direct as they are dependent upon their context and require the simulation of forces like gravity and momentum. This cognitive work can be offloaded onto computers through 3D geospatial modeling, analysis, and simulation. Interacting with computers, however, can also be challenging, often requiring training and highly abstract thinking. Tangible computing – an emerging paradigm of human-computer interaction in which data is physically manifested so that users can feel it and directly manipulate it – aims to offload this added cognitive work onto the body. We have designed Tangible Landscape, a tangible interface powered by an open source geographic information system (GRASS GIS), so that users can naturally shape topography and interact with simulated processes with their hands in order to make observations, generate and test hypotheses, and make inferences about scientific phenomena in a rapid, iterative process. Conceptually Tangible Landscape couples a malleable physical model with a digital model of a landscape through a continuous cycle of 3D scanning, geospatial modeling, and projection. We ran a flow modeling experiment to test whether tangible interfaces like this can effectively enhance spatial performance by offloading cognitive processes onto computers and our bodies. We used hydrological simulations and statistics to quantitatively assess spatial performance. We found that Tangible Landscape enhanced 3D spatial performance and helped users understand water flow.

INTRODUCTION

Understanding physical processes

Physical processes like the flow and dispersion of water are challenging to understand because they unfold in time and space, are controlled by their context, and are driven by forces like gravity and momentum. The flow of water across a landscape is controlled by the morphological shape and gradient of the topography. It is challenging to understand how water will flow across a landscape because one must not only understand how the shape and gradient of the terrain control the flow and dispersion of water locally, but also how water will flow between shapes and gradients – how the morphology is topologically connected. Understanding a physical process requires thinking at and across multiple spatial scales simultaneously.

This cognitive work can be offloaded onto computers through 3D geospatial modeling, analysis, and simulation. Interacting with computers, however, can also be challenging, requiring training and highly abstract thinking that adds a new cognitive burden.

Tangible interfaces for GIS

Tangible interfaces are designed to make interaction with computers easier and more natural by physically manifesting digital data so that users can cognitively grasp it as an extension of their bodies, offloading cognition onto their bodies. In embodied cognition higher cognitive processes are built upon and grounded in sensorimotor processes – upon bodily action, perception, and experience – so thinking can be performed subconsciously through action and extended with tools (Kirsh, 2013). Tangible interfaces have been developed for geographic information systems (GIS) to ease the cognitive burden of complex spatial reasoning, spatial visualization, and human-computer interaction by embodying these processes. Theoretically tangible interfaces for GIS should help users understand environmental processes by giving multidimensional geospatial data an interactive, physical form that they can cognitively grasp and kinaesthetically explore in space and time.

In a seminal paper Ishii and Ullmer envisioned tangible user interfaces that would 'bridge the gap between cyberspace and the physical environment by making digital information (bits) tangible' (Ishii and Ullmer, 1997). They described 'tangible bits' as 'the coupling of bits with graspable physical objects' (Ishii and Ullmer, 1997). Tangible interfaces like Urp (Underkoffler and Ishii, 1999), Illuminating Clay (Piper et al., 2002a), and SandScape (Ratti et al., 2004) enriched physical models of urban spaces and landscapes with spatial analyses and simulations like wind direction, cast shadow, slope, aspect, curvature, and water direction in order to enhance and streamline spatial thinking, design, and decision-making.

Many of the analyses used in Illuminating Clay were adapted from the open source GRASS GIS project (Piper et al., 2002b) and eventually Illuminating Clay was coupled with GRASS GIS to draw on its extensive libraries for spatial computation. The aim of coupling Illuminating Clay with GRASS GIS was to 'explore relationships that occur between different terrains, the physical parameters of terrains, and the landscape processes that occur in these terrains' (Mitasova et al., 2006). The effort to couple a physical landscape model with GRASS GIS led to the development of the Tangible Geospatial Modeling System (Tateosian et al., 2010) and Tangible Landscape (Fig. 1) (Petrasova et al., 2014, Petrasova et al., 2015).

Tangible Landscape Tangible Landscape – a tangible user interface powered by GRASS GIS – couples a physical and digital model of a landscape through a continuous cycle of 3D scanning, geospatial modeling, and projection so that users can intuitively interact with the modeled landscape in near real-time (Fig. 2). The physical model is often made of polymer-enriched sand so

*Corresponding author

Figure 1: Tangibly modeling the flow of water with Tangible Landscape

that users can easily sculpt forms in a medium that will hold its shape, has good plasticity, and has a familiar feel and aesthetic. As users sculpt the physical model the model is 3D scanned and interpolated in GIS as a digital elevation model. The digital elevation model is used to compute geospatial analyses, models, and simulations, which are then projected back onto the physical model – all in near real-time. Conceptually, this enables users to hold a GIS in their hands – feeling the shape of the earth, sculpting its topography, and directing the flow of water. This should enable users to naturally model topography and interact with simulated physical processes in a rapid, iterative process of observation, hypothesis generation and testing, and inference.

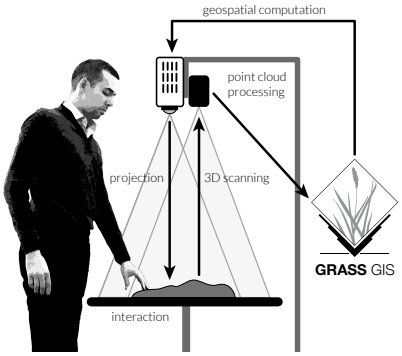


Figure 2: How Tangible Landscape works: a near real-time feedback cycle of interaction, 3D scanning, point cloud processing, geospatial computation, and projection

Cognitively grasping spatial processes

The aim of this research was to study how tangible interfaces for GIS mediate spatial thinking about landscape processes like water flow. Theoretically a tangible interface for a GIS that enables intuitive digital sculpting while providing analytical feedback should allow users to dynamically explore how topographic form influences landscape processes. With a physical model one can cognitively grasp topographic form, offloading the cognitive work of understanding and shaping topographic form onto the body. A GIS can offload the cognitive work of simulating complex physical processes like the flow of water through computation. With these physical and computational affordances combined in a tangible interface for GIS users should be able to more easily understand and shape spatial processes. We empirically tested how a tangible interface for GIS mediated spatial performance in a water flow modeling experiment.

METHODS

We conducted a terrain and water flow modeling experiment in which participants tried to recreate a given landscape. We quantitatively assessed their spatial performance using hydrological simulation, summary statistics, and cell-by-cell statistics.

Flow modeling experiment

In the experiment 11 participants were asked to sculpt a given landscape using different technologies – first using Vue, a triangulated irregular network based 3D modeling program designed for intuitive terrain sculpting, and then using Tangible Landscape. The participants were asked to model a real landscape – a region of Lake Raleigh Woods in Raleigh, North Carolina – using each technology. We selected a region of the landscape with distinctive, clearly defined landforms – a central ridge flanked by two stream valleys (see Fig. 3). The digital elevation model (DEM) for this region was derived from a 2013 airborne lidar survey using the regularized spline with tension interpolation method.

In the 1st exercise each participant had 10 minutes to digitally sculpt the topography of the study landscape in Vue's terrain editor using a physical model as a reference (see Fig. 4a). Participants were given a 2 minute introduction to terrain sculpting in Vue and then 10 minutes to experiment and become familiar with the interface before beginning the exercise. Vue's terrain editor was designed to emulate sculpting by hand – there are tool brushes that are analogous to basic actions in sculpture including pushing, pulling, and smoothing.

In the 2nd exercise each participant had ten minutes to sculpt the study landscape in polymer enriched sand using Tangible Landscape's water flow simulation as a guide (see Fig. 4b). As participants sculpted they could switch between projected maps of either the *a)* simulated water flow across the given landscape that they were trying to replicate *b)* or the simulated water flow across the scanned landscape.

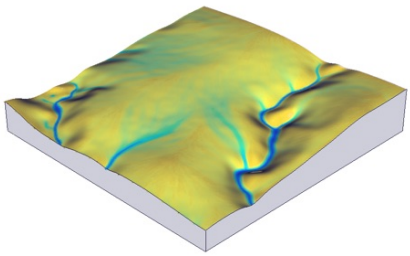
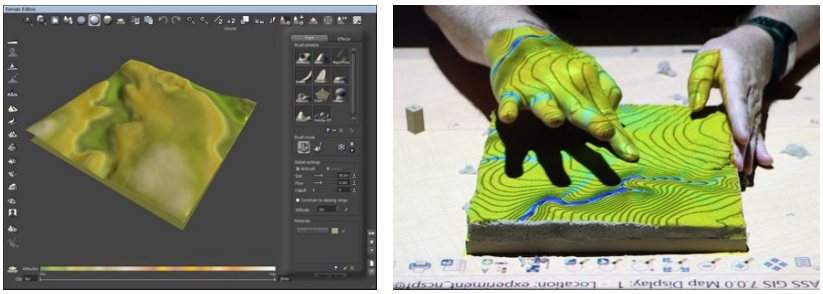


Figure 3: Our reference landscape with simulated water flow



(a) (b)

Figure 4: (a) Digitally sculpting with Vue in the 1st exercise and (b) tangibly sculpting with Tangible Landscape in the 2nd exercise

Implementation

We used GRASS GIS both for geospatial computation with Tangible Landscape and for the analysis of the results of the experiment. The algorithms used in GRASS GIS are based on peer-reviewed scientific publications and have been transparently implemented with free, publicly available, version controlled source code. Open, transparent algorithms are needed to fully reproduce computational science (Rocchini and Neteler, 2012). In this experiment we used the overland flow hydrologic simulation using a path sampling method (SIMWE) implemented in GRASS GIS as the module r.sim.water.

Shallow overland flow We simulated shallow overland water flow controlled by spatially variable topography, soil, landcover, and rainfall parameters using the SIMWE model to solve the continuity and momentum equations for steady state water flow with a path sampling method. Shallow water flow can be approximated by the bivariate form of the St Venant equation:

$$\frac{\partial h(\mathbf{r},t)}{\partial t} = i_e(\mathbf{r},t) - \nabla \cdot \mathbf{q}(\mathbf{r},t) \tag{1}$$

where:

$\mathbf{r}(x,y)$ is the position [m]

t is the time [s]

$h(\mathbf{r},t)$ is the depth of overland flow [m]

$i_e(\mathbf{r},t)$ is the rainfall excess [m/s]
(rainfall − infiltration − vegetation intercept)

$\mathbf{q}(\mathbf{r},t)$ is the water flow per unit width $[\mathrm{m}^2/\mathrm{s}]$.

By integrating a diffusion term $\propto \nabla^2[h^{5/3}(\mathbf{r})]$ into the solution of the continuity and momentum equations for steady state water flow diffusive wave effects can be approximated so that water can flow through depressions.

$$-\frac{\varepsilon(\mathbf{r})}{2}\nabla^2[h^{5/3}(\mathbf{r})] + \nabla \cdot [h(\mathbf{r})\mathbf{v}(\mathbf{r})] = i_e(\mathbf{r}) \tag{2}$$

where:

$\varepsilon(\mathbf{r})$ is a spatially variable diffusion coefficient.

This equation is solved using a Green's function Monte Carlo path sampling method (Mitasova et al., 2004).

Data collection and analysis

We used summary and cell-by-cell statistics to compare the results of each exercise using the reference landscape as a baseline. The digital models sculpted in the 1st exercise were manually georeferenced and imported into GRASS GIS as point clouds for modeling and analysis. The final state of the physical models sculpted in 2nd exercise were automatically 3D scanned, georeferenced, and imported into GRASS GIS as points clouds with Tangible Landscape for modeling and analysis. All of the point clouds were randomly resampled and interpolated as DEMs using the regularized spline with tension interpolation method (Mitasova et al., 2005). For each model we simulated shallow overland water flow, identified depressions, and computed the difference between the modeled and reference water flow depth. To compute the difference we subtracted the modeled values from the initial, reference values. We identified and computed the depth of depressions by generating a depressionless DEM and then calculating the difference between the DEM from the depressionless DEM. Then we computed the mean, sum, and maximum of the water depths, the depressions, and the difference in water depths for each exercise in the experiment. We visually assessed the spatial pattern of water flow and continuity using these maps. In order to quantitatively compare how well the modeled streams matched the reference stream we computed the distance between cells with concentrated water flow in the reference depth map and mean of the modeled depth maps. First we extracted points for each cell with high water depth values ($>= 0.05$ ft) in the reference water depth raster and the mean water depth raster for each exercise. Then we calculated the minimum distance between the reference points and the nearest points from each exercise. In order to quantitatively compare the continuity of flow we computed the percentage of cells with depressions for each exercise.

Code and data

As a work of open science we invite you to replicate or build upon this experiment by using or adapting our tangible interface, experimental methodology, code, and data. The python scripts for data processing and analysis used in this experiment are available on GitHub at `https://github.com/baharmon/tangible-water-flow` released under the GNU General Public License (GPL). See the documentation in the GitHub repository for detailed instructions for replicating this experiment. The data used in this experiment are available on the Open Science Framework at `osf.io/zv3t6` under the Creative Commons Zero license.

GRASS GIS is available at https://grass.osgeo.org/ under the GNU GPL. Tangible Landscape is available at https://github.com/ncsu-osgeorel/grass-tangible-landscape under the GNU GPL. To build your own Tangible Landscape see the documentation in the GitHub repository and refer to the book Tangible Modeling with Open Source GIS (Petrasova et al., 2015).

While the rest of the software used in this research is free and open source, Vue [1] is proprietary software. We used Vue's terrain editor for this study because it was designed specifically for intuitive terrain modeling, but open source 3D modeling programs like Blender [2] could be substituted for Vue.

RESULTS

The models sculpted with Tangible Landscape more accurately replicated the flow of water over the study landscape. The digitally sculpted models tended to have more diffuse water flow and more water pooling in depressions, whereas the tangibly sculpted models tended to have more concentrated flow in stream channels and less pooling in depressions. Furthermore, the spatial distribution of water flow on the tangibly sculpted models more closely fit the distribution of water flow on the reference landscape.

Exercise	Mean	Max	Sum
Reference	0.01	0.47	786.25
Digital	0.21	1.83	3.60
Tangible	0.27	2.31	4.60

Table 1: Highest mean, maximum, and summed water depths (ft)

Exercise	Mean	Sum
Digital	21.13	29244
Tangible	18.59	25724

Table 2: The minimum distances from reference stream cells to the nearest modeled cell with concentrated flow (ft)

Exercise	Max	Sum
Reference	0.10	0.62
Digital	24.5	53
Tangible	22.2	44

Table 3: Highest max and summed depth of depressions (ft)

Exercise	Cells
Reference	0%
Digital	44%
Tangible	17.66%

Table 4: Percent of cells with depressions (3 ft ^{2})

The models sculpted with Tangible Landscape had more concentrated flow in stream channels with higher mean, maximum, and summed water depths (see Table 1). Fig. 6 shows that water flow in the 2nd exercise was spatially concentrated in the upper stream channel, the lower stream channel, and the nascent swale to the right.

Furthermore the water flow across models sculpted with Tangible Landscape more closely fit the reference. Fig. 8 and the details in Fig. 9 show that water flow in the 1st exercise was more spatially diffuse and dispersed, whereas water flow in the 2nd exercise was a better fit. Fig. 10 - 11 show how well the flow of the digitally and tangibly sculpted models fit the reference flow. The digitally sculpted models tended to loosely fit the upper stream with fewer, more dispersed clusters of concentrated flow at further distances from the reference on average. They also tended to have very few clusters at considerable distance from the swale on the right and small, dispersed clusters that tightly fit the lower stream. The tangibly sculpted models, however, tended to more densely, tightly fit the upper stream and the swale on the right. The tangibly sculpted models tended to miss or misplace the lower stream channel (siting it too far southeast). The near-real time water flow analytic was not very useful for modeling the lower stream segment because there was not enough contributing area on the model to generate much concentrated flow. Water flow over the sculpted models was only computed over the model, whereas the reference water flow was computed over the entire watershed. Overall the tangibly sculpted models fit the reference flow 8.32% better than the digitally sculpted models (see Table 2).

The digitally sculpted models had higher max and summed water depth in depressions (see Table 1). Furthermore the depressions covered 26.34% more area of the digitally sculpted models than the tangibly sculpted models (see Table 4, Fig. 5, and Fig. 7). The presence of large depressions on the central ridge in the 1st exercise reveals concave topographic morphology where there should be convex morphology. The pervasive presence of depressions in the stream channels in both exercises reveals disrupted flows with broken continuity where water should be flowing.

DISCUSSION

The modeling results show that participants tended not to clearly understand how topography directs water flow, but began to learn about the importance of curvature and continuity with the aid of the tangible interface.

When digitally sculpting participants typically focused on making the streams lower than the surrounding topography, but not on draining water into the streams or directing a continuous flow downstream. In the 1st exercise participants made low points approximately where the streams should be, but tended not to grade smooth slopes with appropriate curvature. The participants use of concave rather than convex morphology on the central ridge reveals that some did not understand how topographic curvature controls water flow. The extensive depressions distributed widely across the terrain demonstrate that participants tended not understand the importance of topographic curvature and continuity for water flow.

With the tangible interface they typically focused on directing water into streams, but did not tend focus on directing a continuous flow downstream. In the 2nd exercise participants tended to grade smoother slopes with fewer depressions whose curvature directed water towards channels, concentrating it in streams. They, however, did not tend to grade the channels to continuously slope downstream. While participants performance improved substantially in the 2nd exercise, the pervasive presence of depressions in the stream channels in both exercises demonstrates that participants did not fully understand the morphology of streams by the end of the experiment.

We observed participants using an iterative modeling process with the tangible interface – they *a)* sculpted, *b)* studied the updated water flow simulation, *c)* critiqued the form of their model, *d)* and continued to sculpt. The substantial improvement in participants'

[1] http://www.e-onsoftware.com/products/vue/
[2] https://www.blender.org/

performance in the 2nd exercise shows that they successfully integrated the tangible interface's water flow analytic into their modeling process. To improve their performance they must have been able to successfully manage the added cognitive load of the analytical feedback. The tangibility of the interface may have let them offload some of the cognitive work of sensing and shaping 3D form onto their bodies so that they could focus on water flow.

While their performance did improve in the 2nd exercise, it may still have been adversely affected by emotions like frustration. If the cognitive load of simultaneously modeling 3D form and water flow became too great – if participants were unable to fluidly connect cause (topographic form) and effect (flow) – they might become frustrated and demotivated. Future research should investigate the role of affect, motivation, and metacognition in tangible interaction.

More analytics may enhance participants' understanding of water flow and stream morphology. Identifying and computing the depth of depressions highlighted the role of topographic curvature and continuity in water flow. While topographic parameters such as curvature and slope would highlight important aspects of the morphology, an additional step of reasoning and imagination is required to link these parameters to water flow. The ponding of water in depressions directly links topographic controls and water flow and thus should be a more intuitive analytic. We propose combining shallow overland water flow with ponding in depressions as an analytic to help users understand how water flows.

CONCLUSION

Tangible Landscape's water flow analytic enabled an iterative cycle of form-finding and critical assessment that helped participants learn how form controls process. Seeing the water flow simulation update in near real-time enabled participants to generate hypotheses, test hypotheses, and draw inferences about the way that water flows over topography. As one participant said, 'seeing the flow takes away the mystery of topography.' Physically manifesting topographic data enables users to cognitively grasp the terrain. Coupling tangible topography with the simulated flow of water lets users understand the simulation with their body.

REFERENCES

Ishii, H. and Ullmer, B., 1997. Tangible bits: towards seamless interfaces between people, bits and atoms. In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, ACM Press, New York, New York, USA, pp. 234–241.

Kirsh, D., 2013. Embodied cognition and the magical future of interaction design. *ACM Transactions on Computer-Human Interaction* 20(1), pp. 3:1–3:30.

Mitasova, H., Mitas, L. and Harmon, R., 2005. Simultaneous spline approximation and topographic analysis for lidar elevation data in open-source GIS. *IEEE Geoscience and Remote Sensing Letters* 2, pp. 375–379.

Mitasova, H., Mitas, L., Ratti, C., Ishii, H., Alonso, J. and Harmon, R. S., 2006. Real-time landscape model interaction using a tangible geospatial modeling environment. *IEEE Computer Graphics and Applications* 26(4), pp. 55–63.

Mitasova, H., Thaxton, C., Hofierka, J., McLaughlin, R., Moore, A. and Mitas, L., 2004. Path sampling method for modeling overland water flow, sediment transport, and short term terrain evolution in Open Source GIS. *Developments in Water Science* 55, pp. 1479–1490.

Petrasova, A., Harmon, B. A., Petras, V. and Mitasova, H., 2014. GIS-based environmental modeling with tangible interaction and dynamic visualization. In: D. Ames and N. Quinn (eds), Proceedings of the 7th International Congress on Environmental Modelling and Software, International Environmental Modelling and Software Society, San Diego, California, USA.

Petrasova, A., Harmon, B., Petras, V. and Mitasova, H., 2015. *Tangible Modeling with Open Source GIS*. Springer.

Piper, B., Ratti, C. and Ishii, H., 2002a. Illuminating clay: a 3-D tangible interface for landscape analysis. In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '02*, ACM Press, Minneapolis, p. 355.

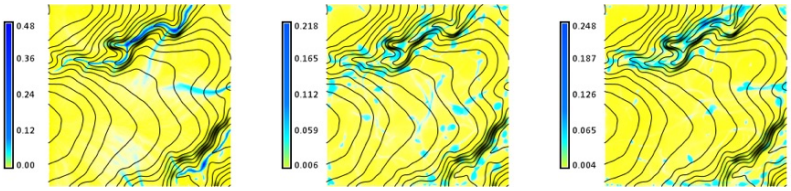
Piper, B., Ratti, C. and Ishii, H., 2002b. Illuminating Clay: A Tangible Interface with potential GRASS applications. In: *Proceedings of the Open Source GIS - GRASS users conference 2002*, Trento, Italy.

Ratti, C., Wang, Y., Ishii, H., Piper, B., Frenchman, D., Wilson, J. P., Fotheringham, A. S. and Hunter, G. J., 2004. Tangible User Interfaces (TUIs): A Novel Paradigm for GIS. *Transactions in GIS* 8(4), pp. 407–421.

Rocchini, D. and Neteler, M., 2012. Let the four freedoms paradigm apply to ecology. *Trends in Ecology and Evolution* 27(6), pp. 310–311.

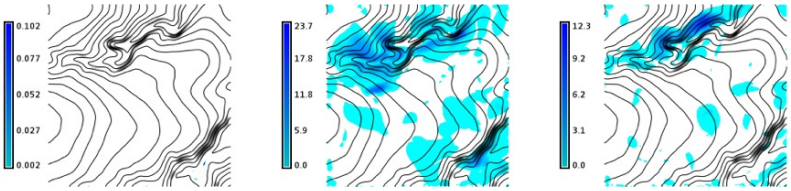
Tateosian, L., Mitasova, H., Harmon, B. A., Fogleman, B., Weaver, K. and Harmon, R. S., 2010. TanGeoMS: tangible geospatial modeling system. *IEEE transactions on visualization and computer graphics* 16(6), pp. 1605–12.

Underkoffler, J. and Ishii, H., 1999. Urp: a luminous-tangible workbench for urban planning and design. In: *CHI '99 Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM Press, New York, New York, USA, pp. 386–393.



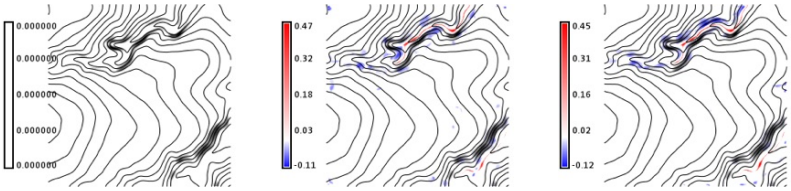
(a) (b) (c)

Figure 5: (a) The depth of simulated water flow across the reference landscape, (b) the mean depth using digital modeling, (c) and the mean depth using tangible modeling



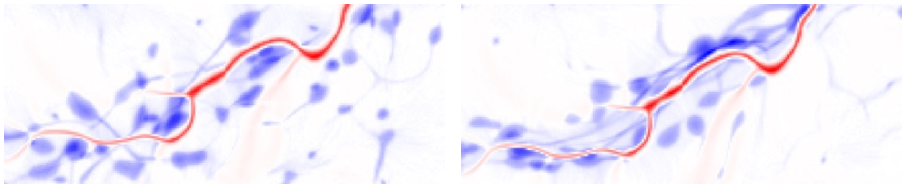
(a) (b) (c)

Figure 6: (a) The depth of depressions in the reference landscape, (b) the maximum depth of depressions using digital modeling, (c) and the maximum depth of depressions using tangible modeling



(a) (b) (c)

Figure 7: (a) The difference of the reference landscape, (b) the mean of digitally sculpted models, (c) and the mean of tangibly sculpted models *from the reference landscape*



(a) (b)

Figure 8: Detail of the difference of (a) the mean of digitally sculpted models and (b) the mean of tangibly sculpted models *from the reference landscape*

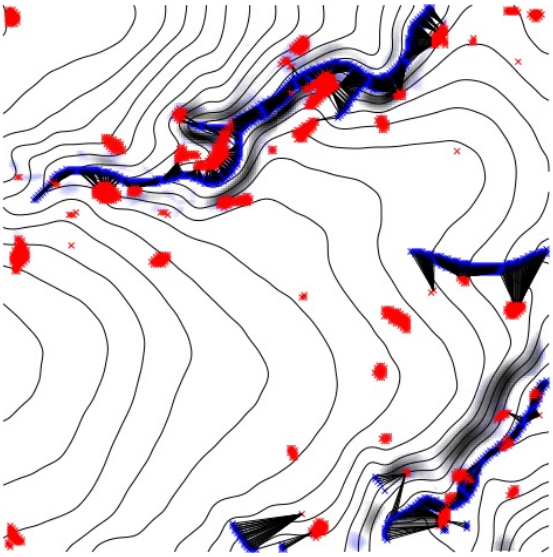


Figure 9: The lines represent the minimum distance between blue points with high concentrated flow from the reference and red points with high concentrated flow from the mean depth of digitally sculpted models

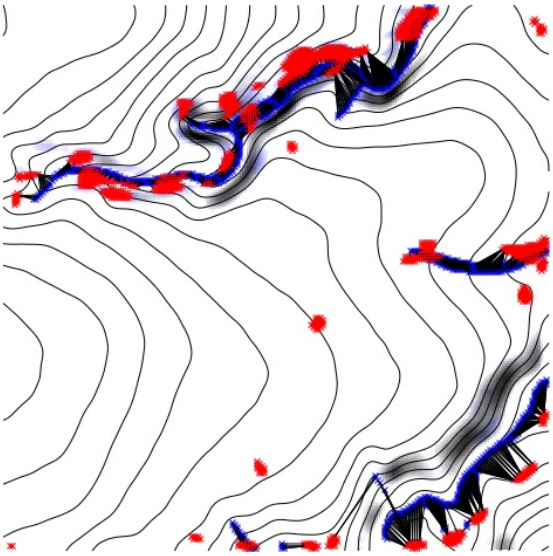


Figure 10: The lines represent the minimum distance between blue points with high concentrated flow from the reference and red points with high concentrated flow from the mean depth of tangibly sculpted models

4 Cognitively Grasping Topography with Tangible Landscape

Attribution

I co-designed Tangible Landscape, designed and ran the experiments, and wrote the paper. Anna Petrasova, Vaclav Petras, and Helena Mitasova contributed to this research by collaboratively designing Tangible Landscape, running the experiment, and editing the paper. Ross Meentemeyer, Gene Bressler, and Art Rice contributed with advice, guidance, support, and editing.

Cognitively Grasping Topography with Tangible Landscape

BRENDAN ALEXANDER HARMON, North Carolina State University
ANNA PETRASOVA, North Carolina State University
VACLAV PETRAS, North Carolina State University
ROSS KENDALL MEENTEMEYER, North Carolina State University
EUGENE BRESSLER, North Carolina State University
ART RICE, North Carolina State University
HELENA MITASOVA, North Carolina State University

Tangible interfaces for spatial modeling combine embodied, kinaesthetic interaction with spatial computations. Theoretically this should enable users to intuitively interact with multidimensional digital models of space, offloading challenging cognitive tasks onto the body and computationally enhancing how they think about space. We have designed Tangible Landscape – a tangible interface powered by a geographic information system (GIS) that gives 3D spatial data an interactive, physical form so that users can naturally sense and shape it. Tangible Landscape couples a physical and a digital model of a landscape through real-time cycles of physical manipulation, 3D scanning, spatial computation, and projected feedback. Through a series of 3D modeling experiments assessed using both quantitative and qualitative methods we determined that Tangible Landscape can improve 3D spatial performance. Participants produced more accurate models that better represented morphological features with tangible modeling than they did with either digital or analog, hand modeling.

CCS Concepts: •**Human-centered computing** → **Human computer interaction (HCI); Laboratory experiments;**

Additional Key Words and Phrases: Human-computer interaction, tangible interfaces, interaction design, physical computation, embodied cognition, spatial thinking, geospatial modeling

ACM Reference Format:
Brendan A. Harmon, Anna Petrasova, Vaclav Petras, Helena Mitasova, Ross K. Meentemeyer, Eugene H. Bressler, and Art Rice, 2016. Cognitively Grasping Topography with Tangible Landscape. *ACM Trans. Comput.-Hum. Interact.* 0, 0, Article 0 (0), 56 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Spatial thinking – 'the mental processes of representing, analyzing, and drawing inferences from spatial relations' [Uttal et al. 2013] – is used pervasively in everyday life for tasks such as recognizing things, manipulating things, interacting with others, and way-finding. Higher dimensional spatial thinking – thinking about form, volume, and processes unfolding in time – plays an important role in science, technology, engineering, the arts, and math. 3-dimensional (3D) spatial thinking is used in disciplines such as geology to understand the structure of the earth, ecology to understand the structure of ecosystems, civil engineering to shape landscapes, architecture to design buildings, urban planning to model cities, and the arts to shape sculpture.

Author's addresses: B. A. Harmon and A. Petrasova and V. Petras and H. Mitasova and R. K. Meentemeyer, Center for Geospatial Analytics, North Carolina State University; B. A. Harmon, E. H. Bressler and A. Rice, Department of Landscape Architecture, North Carolina State University.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 0 ACM. 1073-0516/0/-ART0 $15.00
DOI: 0000001.0000001

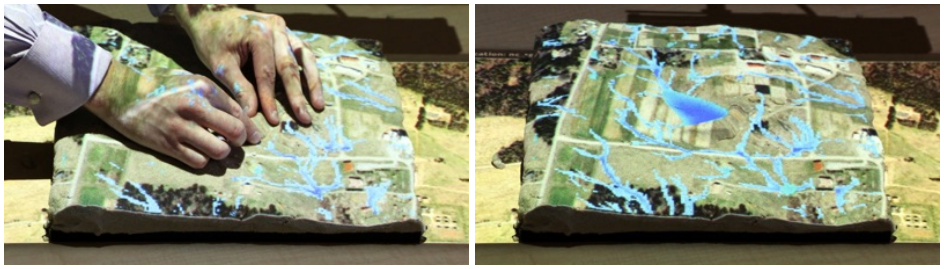


Fig. 1. Modeling the flow of water with Tangible Landscape. A user sculpts a polymeric sand model of a landscape augmented with simulated water flow and an orthophotograph. Changes to the model are scanned into in GIS and the resulting water flow simulation is projected back onto the model in real-time.

Many spatial tasks can be performed computationally enabling users to efficiently store, model, and analyze large sets of spatial data and solve complex spatiotemporal problems. In engineering, design, and the arts computer-aided design (CAD) and 3D modeling software are used to interactively, computationally model, analyze, and animate complex 3D forms. In scientific computing multidimensional spatial patterns and processes can be mathematically modeled, simulated, and optimized using geographic information systems (GIS), geospatial programming, and spatial statistics. GIS can be used to quantitatively model, analyze, simulate, and visualize complex spatial and temporal phenomena – computationally enhancing users' understanding of space. With extensive libraries for point cloud processing, 3D vector modeling, and surface and volumetric modeling and analysis GIS are powerful tools for studying 3D space.

GIS, however, can be unintuitive, challenging to use, and creatively constraining due to the complexity of the software, the complex workflows, and the limited modes of interaction and visualization [Ratti et al. 2004a]. Unintuitive interactions with GIS can frustrate users, constrain how they think about space, and add new cognitive burdens that require highly developed spatial skills and reasoning to overcome. The paradigmatic modes for interacting with GIS today – command line interfaces (CLI) and graphical user interfaces (GUI) – require physical input into devices like keyboards, mice, digitizing pens, and touch screens, but output data visually as text or graphics. Theoretically this disconnect between intention, action, and feedback makes graphical interaction unintuitive [Dourish 2001; Ishii 2008]. Since users can only think about space visually with GUIs, they need sophisticated spatial abilities like mental rotation [Shepard and Metzler 1971; Just and Carpenter 1985] to parse and understand, much less manipulate 3D space.

In order to make 3D GIS more natural and intuitive to use we have designed Tangible Landscape – a tangible interface for GIS – that physically manifests 3D data so that users can feel and manipulate it with their bodies (Fig. 1). Our goal is for users with little or no computer experience to be able to intuitively, collaboratively explore 3D spatial data and interact with scientific models so that they can rapidly test ideas while learning from computational feedback. We have begun to quantitatively analyze how Tangible Landscape mediates 3D spatial cognition. We invite other researchers to collaborate in this ongoing open source and open science project by building Tangible Landscape, contributing to its development, developing new applications, and studying how it mediates cognition.

1.1. Tangible, embodied interaction

In embodied cognition higher cognitive processes are grounded in, built upon, and mediated by bodily experiences such as kinaesthetic perception and action [Hardy-Vallée and Payette 2008]. Tangible interfaces – interfaces that couple physical and digital data [Dourish 2001] – are designed to enable embodied interaction by physically manifesting digital data so that users can cognitively grasp and absorb it, thinking with it rather than about it [Kirsh 2013]. Embodied interaction should be highly intuitive – drawing on existing motor schemas and seamlessly connecting intention, action, and feedback. It should reduce users' cognitive load by enabling them to physically simulate processes and offload tasks like spatial perception and manipulation onto the body [Kirsh 2013]. Distance and physical properties like size, shape, volume, weight, hardness, and texture can be automatically and subconsciously assessed with the body [Jeannerod 1997]. Tangible interfaces should, therefore, enable users to subconsciously, kinaesethically judge and manipulate spatial distances, relationships, patterns, 3D forms, and volumes offloading these challenging cognitive tasks onto their bodies.

1.2. Tangible interfaces for geospatial modeling

With GIS users can computationally offload complex cognitive tasks like analyzing spatial patterns and simulating spatiotemporal processes. Tangible interfaces for geospatial modeling should, therefore, enhance users' spatial performance – their ability to sense, manipulate, and interact with multidimensional space – for challenging tasks like sculpting topography and guiding the flow of water by combining these physical and computational affordances. There are already many tangible interfaces for geospatial modeling. These include actuated pin tables (Table I), augmented architectural models (Table II), augmented clay (Table III), and augmented sandboxes (Table IV).

Actuated pin tables like Relief [Leithinger et al. 2009] are a type of transformable tangible interface [Ishii et al. 2012], dynamic shape display [Poupyrev et al. 2007], or shape changing interface [Rasmussen et al. 2012] that use motor-driven pistons to actuate an array of pins that physically change the shape of a tabletop surface based on computation. These tangible interfaces have three feedback loops – users can feel the physical model for passive, kinaesthetic feedback, the model can be computationally transformed for active, kinaesthetic feedback, and users can see computationally generated, graphical feedback.

Projection-augmented tangible interfaces couple a physical and digital model through a cycle of 3D sensing or object recognition, computation, and projection. Augmented architectural models like Urp [Underkoffler and Ishii 1999] and the Collaborative Design Platform [Schubert et al. 2011b] are a type of 'discrete tabletop tangible interface' [Ishii et al. 2012] with physical models of buildings that are augmented with projected analytics. Augmented clay models like Illuminating Clay [Piper et al. 2002a] and augmented sandboxes like Sandscape [Ishii et al. 2004] are types of 'deformable, continuous tangible interfaces' [Ishii et al. 2012] that users can sculpt. These tangible interfaces have two feedback loops – there is passive, kinaesthetic feedback from grasping the physical model and active, graphical feedback from computation.

1.2.1. Actuated pin tables. Project FEELEX, an early prototype developed in 2001, used an array of 36 actuated pins to deform a 24 x 24 cm rubber sheet. This computationally deformable rubber sheet was augmented with projected graphics to create a shape changing 3D display. User studies for FEELEX included observations of user behavior and shape recognition tests. They found that 85% of their subjects used their entire

hand to interact with the interface. On average subjects had 6 mm of error when perceiving shapes by touch with their system [Iwata et al. 2001].

The XenoVision Mark III Dynamic Sand Table, developed in 2004, and the Northrop Grumman Terrain Table, developed in 2006, were large actuated pin tables designed for military geospatial applications that used thousands of pins to dynamically reshape 1.32 x 1.02 m silicone surfaces representing topography.

The MIT Media Lab developed a family of actuated pin tables – Relief, Recompose, Tangible CityScape, and inFORM – to explore novel modes of interaction with dynamically changing affordances and constraints. Relief used an array to 125 actuated pins to deform a projection-augmented lycra surface representing topography [Leithinger et al. 2009]. A depth sensor was integrated into the next generation of this system, Recompose, so that users could control the surface using gestures [Leithinger et al. 2011; Blackshaw et al. 2011]. inFORM – redesigned as an array of 900 polystyrene pins forming a 381 x 381 mm 3D projection-augmented surface – affords diverse modes of interaction such as sculpting, gesture, indirectly moving passive objects with pins, and pressing user interface elements like buttons formed from pins. [Follmer et al. 2013]. The Tangible CityScape project used inFORM to physically model urban spaces and analyze urban conditions like light and shadow. Building masses formed by pins would dynamically reshape when users panned or zoomed with gestures.

While actuated pin tables afford highly intuitive, bi-directional physical interaction, they are mechanically complex, relatively low resolution, and challenging to maintain and transport. The size of the actuators limits their resolution – Project FEELEX had 40 mm spacing between pins, FEELEX 2 had 8 mm spacing, Relief had 38 mm spacing, and inFORM has 3.175 mm spacing between 9.525 mm^2 pins. These systems tend to be small and expensive to build due to their mechanical complexity.

1.2.2. Augmented architectural modeling interfaces. Urp – a projection augmented interface for urban design – used tag-based objection detection to digitize physical models of buildings on a table. Spatial analyses and simulations such as proximity, wind, shadow, and viewsheds were computed and projected onto the tabletop in real-time so that users could rapidly test different spatial configurations of buildings [Underkoffler and Ishii 1999]. As a case study Urp was used by a urban design class in the MIT School of Architecture and Planning. The researchers observed that it helped students to rapidly explore and test different configurations of space and effectively communicate their designs [Ishii et al. 2002].

The Collaborative Design Platform uses a depth camera to digitize and track physical models of buildings on a rear-projection light table. As users move polystyrene foam models of buildings, the models are 3D scanned updating a digital 3D model of a city. Analyzes like wind, light, shadow, accessibility, distance, and views are projected onto the table in realtime. Views are also rendered in 3D on a wall-mounted touch screen. Users can interact by placing and moving physical models of buldings, touching the screen, or sketching with a digitizing pen [Schubert et al. 2015].

1.2.3. Augmented clay interfaces. Illuminating Clay coupled a clay model and digital model of landscape through a cycle of laser scanning, spatial modeling, and projection. By enriching physical models of urban spaces and landscapes with spatial analyses such as elevation, aspect, slope, cast shadow, profile, curvature, viewsheds, solar irradiation, and water direction it enabled intuitive form-finding, streamlined analog and digital workflows, and enabled multiple users to simultaneously interact in a natural way [Ratti et al. 2004a]. Illuminating Clay had a very limited library of custom implemented spatial analyses. Since many of analyses were adapted from the open source GRASS GIS project [Piper et al. 2002a] there was a call for closer integration with GRASS GIS in order to draw on its extensive libraries for spatial computation [Piper

et al. 2002b]. The effort to couple a physical landscape model with GRASS GIS [Mitasova et al. 2006] led to the development of the Tangible Geospatial Modeling System [Tateosian et al. 2010].

The Tangible Geospatial Modeling System coupled a physical model and GIS model of a landscape through a cycle of laser scanning, geospatial computation in GRASS GIS, and projection giving developers and users access to a sophisticated library for spatial modeling, simulation, visualization, and databasing. It enriched freeform hand modeling with geospatial simulations like diffusive water flow and erosion-deposition so that users could easily explore how changes in topographic form affect landscape processes [Tateosian et al. 2010].

1.2.4. Augmented sandbox interfaces. SandScape used infrared depth sensing to digitize a 'sandbox' of 1 mm glass beads. An infrared camera captured the intensity of infrared light passing through the beads from below in real-time. A digital elevation model computed from the light intensity and derived analyses were projected back onto the sandbox for real-time feedback. SandScape was relatively low resolution due to the quality of the infrared sensing and the size of the glass beads [Ishii et al. 2004; Ratti et al. 2004a].

Phoxel-Space adapted SandScape and Illuminating Clay for physically interacting with voxels, i.e. volumetric pixels. The system coupled a malleable physical model – built of media like clay, plasticine, cubic blocks, or glass beads – with a 3D raster dataset using either a laser scanner or an IR camera. The researchers demonstrated how Phoxel-Space could be used to explore magnetic resonance imaging data, seismic velocity, and computational fluid dynamics [Ratti et al. 2004b].

The Augmented Reality Sandbox developed by the UC Davis W.M. Keck Center for Active Visualization in the Earth Sciences couples a sandbox with a digital model of a landscape through a real-time cycle of 3D scanning with a Kinect sensor. Scans are statistically filtered to remove hands and tools, reduce noise, and fill in areas with no data. The default filtering – 30 frames – results in 1 second of lag. The digital elevation model, contours, and simulated water flow based on the shallow water equations are projected back onto the sand model of the landscape [Kreylos 2012]. It was inspired by a Czech prototype called SandyStation. The code for this open source project, released under the GNU General Public License, and blueprints for building the system are available at https://arsandbox.ucdavis.edu. 280 Augmented Reality Sandboxes have already been built around the world [Kreylos 2017].

Researchers at Eastern Carolina University built an Augmented Reality Sandbox and conducted a qualitative pilot study examining the effect of the technology on learning and engagement in geoscience education. 12 students used the Augmented Reality Sandbox to build terrain models from contours, model fluvial features and processes, and model coastal features and processes. The researchers solicited feedback with an exit survey. Students reported that the sandbox helped them learn about topography, fluvial and coastal processes, and process-form interactions more effectively. Based on the survey and their observations the researchers hypothesized that augmented sandboxes could enable embodied learning and encourage the development of scientific modeling skills [Woods et al. 2016].

Hakoniwa – a projection-augmented sandbox for making generative music and art – was inspired by sandtray therapy. The system was designed to create a playful, embodied experience that could be therapeutic. As users built landscapes in the sandbox by sculpting sand and placing wooden blocks they created music and visual patterns in real-time through a cycle of depth and color sensing, image processing, audio generation, computer graphics, and projection [Kikukawa et al. 2013]. The system evolved from PocoPoco – a tabletop tangible interface for making music [Kanai et al. 2011].

The Landscape Morphologies Lab at the University of Southern California developed the Rapid Landscape Prototyping Machine – a projection-augmented sandbox with robotic fabrication – to design and test strategies for dust control and mitigation for Lake Owens, California. The system used a 6-axis robotic arm to digitally fabricate algorithmically generated landscapes in a sandbox. The sand models were digitized with a laser scanner, the point cloud was triangulated as a terrain mesh, and spatial analyses such as viewsheds, aspect, and flooding were projected back onto the sandbox [Robinson 2014; Cantrell and Holzman 2016]

The Augmented REality Sandtable (ARES) developed by the US Army Research Laboratory is designed for military training and simulation. This system uses a depth camera to continually 3D scan a sandbox and detect gestures. The digital elevation model and contour map derived from the scans and military units created using gestures or tablet input are projected onto the sand model. Units and buildings can be visualized in 3D using tablets or augmented reality glasses. The Augmented REality Sandtable can be linked with other military software to simulate scenarios [Amburn et al. 2015]. A user study comparing users' performance with paper maps, Google Earth, and the Augmented REality Sandtable found that the sandtable was the most effective technology. Participants – especially participants who were veteran video gamers – tended to perform better with the sandtable in landmark identification, distance estimation, and situational judgment tests [Schmidt-Daly et al. 2016].

Inner Garden – a projection-augmented sandbox for contemplation and self-reflection – couples a sand model of topography with simple digital environment with water, plants, clouds, and daylight through a cycle of 3D scanning with a Kinect, biometric sensing, computation, and projection. While a user sculpts topography in the sand creating a digital elevation model, their physiological and emotional state are monitored with an electroencephalogram (EEG) and breathing sensor. Their breath controls the simulated sea level and daylight, their level of frustration controls the simulated cloud cover, and their meditativeness controls simulated plant life [Roo et al. 2016].

Tangible Landscape is a projection-augmented sandbox powered by a GIS for real-time geospatial analysis and simulation [Petrasova et al. 2015]. It evolved from Illuminating Clay [Piper et al. 2002a] and the Tangible Geospatial Modeling System [Tateosian et al. 2010]. While the Tangible Geospatial Modeling System used an expensive laser scanner for 3D sensing [Tateosian et al. 2010], Tangible Landscape – inspired by the open source Augmented Reality Sandbox [Kreylos 2012] – uses a low-cost 3D sensor for real-time depth and color sensing. The 1st generation of Tangible Landscape [Petrasova et al. 2014] used the 1st generation Kinect with structured light sensing [Smisek et al. 2011], while the 2nd [Petrasova et al. 2015] and 3rd generations of Tangible Landscape used the 2nd generation Kinect with time-of-flight sensing [Bamji et al. 2015]. Two pilot studies have quantitatively compared users' 3D spatial performance with Tangible Landscape and digital 3D modeling programs. The first pilot study found that participants tended to more accurately model landscapes using projection augmented modeling with Tangible Landscape than digital modeling with Rhinoceros [Harmon 2016]. The second pilot study found that participants tended to more accurately model topography and hydrology using Tangible Landscape with the water flow analytic than Vue's terrain editor [Harmon et al. 2016c].

1.3. Aims and objectives

Research on tangible interfaces for geospatial modeling has focused on the design of new technologies and prototypes rather than studying how they are used. A review of tangible interfaces for geospatial modeling (Tables I-IV) shows that there have been relatively few case studies [Ishii et al. 2002; Tateosian et al. 2010; Petrasova et al.

Table I. Actuated pin tables

System	Interaction	GIS	User studies	Publications
XenoVision Mark III Dynamic Sand Table	Sculpting			
Northrop Grumman Terrain Table	Sculpting			
Relief	Sculpting			[Leithinger et al. 2009]
				[Leithinger and Ishii 2010]
Recompose	Sculpting			[Leithinger et al. 2011]
	Gesture			[Blackshaw et al. 2011]
Tangible CityScape	Gesture			
inFORM	Sculpting			[Follmer et al. 2013]
	Gesture			
	Object detection			

Table II. Augmented architectural modeling interfaces

System	Interaction	GIS	User studies	Publications
Urp	Object detection		Case studies*	[Underkoffler and Ishii 1999]
				[Ishii et al. 2002]*
Collaborative Design Platform	Object detection			[Schubert et al. 2011b]
	Touch			[Schubert et al. 2011a]
	Sketching			[Schubert et al. 2012]
				[Schubert et al. 2014]
				[Schubert et al. 2015]

Table III. Augmented clay interfaces

System	Interaction	GIS	User studies	Publications
Illuminating Clay	Sculpting		Protocol analysis‡	[Piper et al. 2002a]
				[Piper et al. 2002b]
				[Piper 2002]
				[Shamonsky 2003]‡
				[Ishii et al. 2004]
				[Ratti et al. 2004a]
Tangible Geospatial Modeling System	Sculpting	✓	Case studies*	[Mitasova et al. 2006]
				[Tateosian et al. 2010]*

Table IV. Augmented sandbox interfaces

System	Interaction	GIS	User studies	Publications
SandScape	Sculpting			[Ishii et al. 2004]
				[Ratti et al. 2004a]
PhoxelSpace	Sculpting			[Ratti et al. 2004b]
SandyStation	Sculpting			
Augmented Reality Sandbox	Sculpting		Survey[§]	[Woods et al. 2016][§]
	Gesture			
Hakoniwa	Sculpting			[Kikukawa et al. 2013]
	Object detection			
	Sound			
Rapid Landscape Prototyping Machine	Machining			[Robinson 2014]
Tangible Landscape	Sculpting	✓	Case studies[*]	[Petrasova et al. 2014]
	Object detection		Quantitative experiments[†]	[Petrasova et al. 2015][*]
	Sketching			[Harmon et al. 2016c][†]
				[Harmon et al. 2016a][†]
The Augmented REality Sandtable (ARES)	Sculpting		Quantitative experiments[†]	[Amburn et al. 2015]
	Gesture			[Schmidt-Daly et al. 2016][†]
Inner Garden	Sculpting			[Roo et al. 2016]
	Breathing			
	Emotion			

2015], qualitative user studies [Shamonsky 2003; Woods et al. 2016], and quantitive studies [Harmon et al. 2016c; Harmon et al. 2016a; Schmidt-Daly et al. 2016]. A review of shape-changing interfaces also found that there were relatively few user studies and called for 'more, high-quality data on user experience' in order to understand if interfaces work as designed [Rasmussen et al. 2012].

Many of the theoretical underpinnings of tangibles remain unproven and unexplored. Do current approaches to tangible, embodied interfaces really work as theorized? Can users successfully cognitively grasp digital data as an extension of their bodies, intuitively interact, and offload cognitive processes with tangibles? How does this change how users think and perform? How does it mediate spatial cognition and performance? Can tangible interfaces for geospatial modeling enhance spatial thinking through kinaesthetic interaction with spatial computations? Can users offload enough cognitive work onto their bodies to successfully parse and learn from computational feedback without suffering cognitive overload?

In order to begin to answer some of these questions we designed a tangible interface for geospatial modeling, while simultaneously conducting experiments to study how this tangible interface mediates spatial cognition. Our research objectives were to:

— Design an effective tangible interface for geospatial modeling
— Test whether coupling a physical and digital model of topography can improve 3D spatial performance
— Study how different geospatial analytics mediate users' 3D spatial performance when using a tangible interface for geospatial modeling

2. TANGIBLE LANDSCAPE

2.1. Concept

Tangible interfaces for GIS should ease the cognitive burden of visualizing, interacting with, and reasoning about space by giving spatial data an interactive, physical form that users can cognitively grasp and kinaesthetically explore. Tangible Landscape – a tangible user interface for GRASS GIS – couples a physical and digital model of a landscape through a continuous cycle of 3D scanning, geospatial modeling, and projection so that users can intuitively interact with the modeled landscape in real-time. Conceptually Tangible Landscape physically manifests 3D data so that users can hold a GIS in their hands – so that they can, for example, feel the shape of the earth, sculpt its topography, and direct the flow of water with their hands. It enables users to 3D sketch – to naturally model forms such as topography, draw points and polygons, and interact with simulated physical processes – in a rapid, iterative process of observation, hypothesis generation and testing, and inference. Tangible Landscape is meant to fluidly, seamlessly combine computational science with exploratory modes of creative thinking. Tangible Landscape is unique because it is the only real-time tangible interface for spatial modeling that (1) is powered by a GIS, (2) has extensive libraries for rasters, vectors, time-series, statistics, modeling, analysis, simulation, visualization, and databasing, (3) physically represents space with enough precision for real-world problem solving, (4) has been demonstrated for a wide range of applications, (5) and has been quantitatively tested in user experiments.

2.2. Design

Tangible Landscape was designed to let users naturally explore spatial data, models, and simulations in an engaging, playful way by 3D sketching (Fig. 5). Users can work collaboratively, simultaneously interacting with the physical model (Table V) [Tabrizian et al. 2016]. As users change the physical model the model is 3D scanned as a point cloud, georeferenced, imported into GIS, and either binned or interpolated as a digital elevation model. The digital elevation model is used to compute geospatial analyses, models, and simulations, which are then projected back onto the physical model – all in real-time (Fig. 2). Users can tangibly interact with digital models and simulations by sculpting, placing objects, or sketching (Fig. 3). They can sculpt the model with their hands to shape topography, create or move volumes like buildings or forest canopy, or explore 3D raster data. They can place objects or markers that will be identified using object and color recognition in order to draw points, polylines, polygons, or volumes. They can also use a laser pointer to draw polylines or polygons that will be detected based on light intensity. As the digital models and simulations update the results are projected back onto the model for users to see. The results can also be rendered in 3D on a screen or head-mounted display like an Oculus Rift so that users can immersively visualize the modeled landscape at a human scale (Table V) [Tabrizian et al. 2016]. Because the model is continually scanned users' hands will be digitized as they sculpt or place objects. Scanning users' hands as topography helps users understand how the system works and encourages play. Users can, for example, place their hands on the model and create simulated lakes between their fingers (Fig. 4).

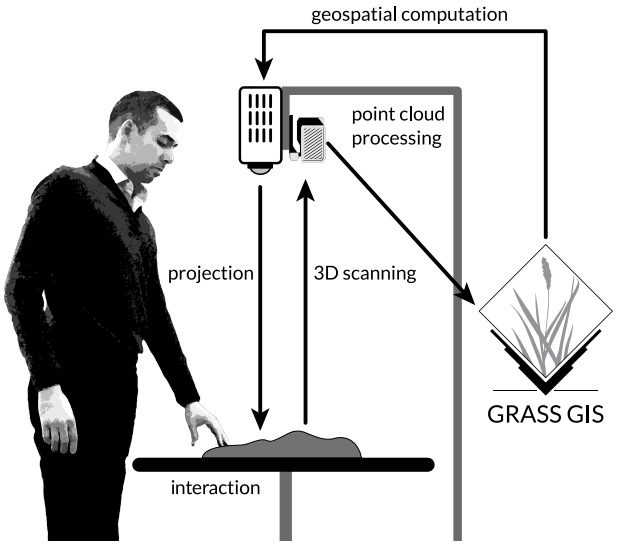


Fig. 2. How Tangible Landscape works: a real-time feedback cycle of physical interaction, 3D scanning, point cloud processing, geospatial computation, and projection.

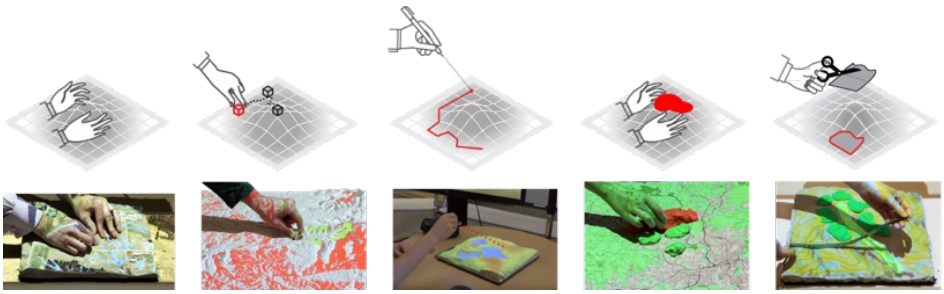


Fig. 3. Modes of interaction with Tangible Landscape: sculpting surfaces, placing points, drawing lines, building volumes, and marking areas.

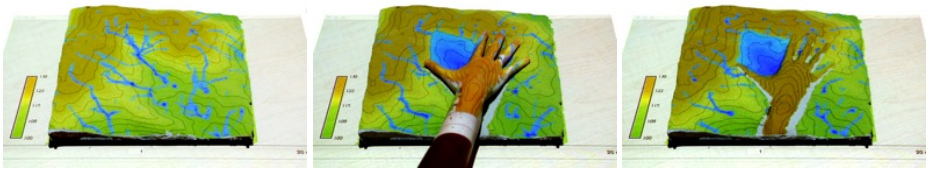


Fig. 4. With Tangible Landscape users can scan their arms and hands as topography creating lakes between their fingers.

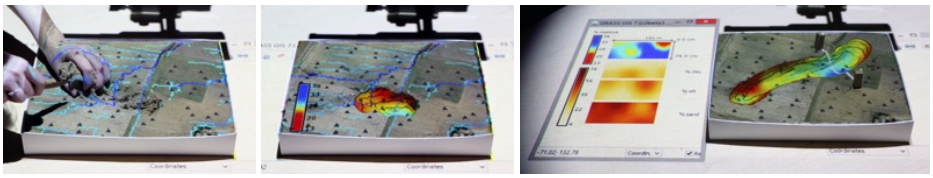
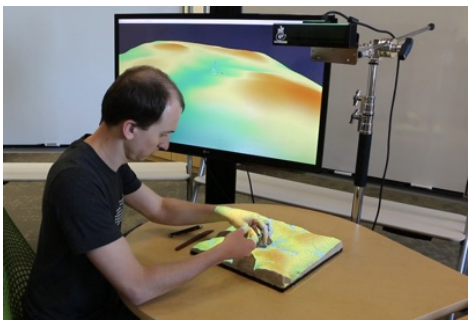
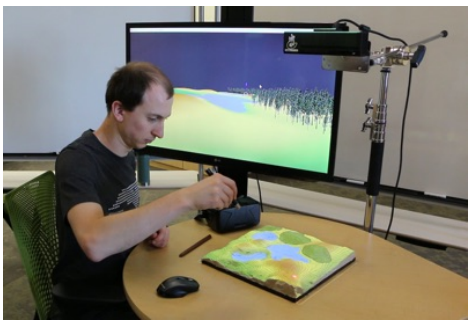
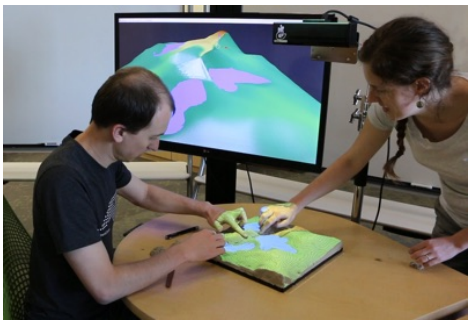
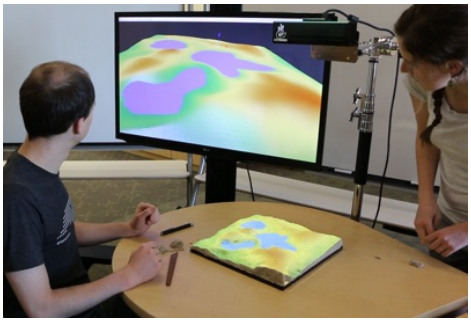


Fig. 5. Naturally exploring subsurface soil moisture and soil types with Tangible Landscape.

Table V. Collaboratively sculpting topography to create lakes, drawing trees with a laser pointer, and visualizing in VR. Source: [Tabrizian et al. 2016].

Sculpting	Drawing & Immersion
	
	
	

Multidimensional sketching. Tangible Landscape was designed to enable tangible multidimensional sketching in geographic space and time. With Tangible Landscape users can sketch in 3D by sculpting surfaces or volumes, drawing across surfaces, or placing and manipulating 3D objects. By directly manipulating 3D form users can indirectly shape other dependent dimensions of data such as simulated processes. Each scan can be stored, timestamped, and registered as a space time raster or vector dataset in GRASS GIS' temporal framework to create a time series of maps. With this time series users can animate the evolution of their model using the module *g.gui.animation*[GRASS Development Team 2016a].

Advances in virtual reality are enabling immersive 3D sketching. The illustrator Wesley Allsbrook, for example, has drawn an animated short film called Dear Angelica by sketching in 3D space using the Oculus Rift and Oculus Touch to combine the affordances of digital painting and physical sculpture [Oculus 2016; Quilez 2016]. While this sort of immersive 3D sketching is situated in a fully virtual space, tangible 3D sketching is situated in a real, but digitally augmented space. Tangible 3D sketching can be situated in a collaborative social context and may not require any new skills – for tangible interactions can be analogous to everyday actions, making use of existing motor schema. While these approaches to 3D sketching should theoretically offer very different affordances in terms of embodied and situated cognition both have the potential to revolutionize how we express ourselves in space and time.

2.3. Implementation

Tangible Landscape uses Microsoft Kinect to continuously capture the shape of the physical model as a point cloud. The point cloud is processed into a raster-based digital elevation model (DEM) in a GRASS GIS database. Since the scanner can be slightly tilted, the scan is horizontally aligned through a calibration process in which a rotation matrix is derived by scanning the horizontal surface of the table where the model will be placed. This rotation matrix is then applied to each scan. The point cloud is automatically trimmed so that it only includes points on the physical model, smoothed, and georeferenced based on the known spatial extent of the area represented by the model. Georeferencing includes translation, rotation by 180° along the z-axis, horizontal scaling, and vertical scaling to account for the vertical exaggeration, if any, of the physical model. The resulting representation of the terrain has real world dimensions and can be combined effortlessly with other GIS data, such as land cover or orthophotographs.

The DEM is reconstructed from the point cloud using either the regularized spline with tension interpolation method [Mitasova et al. 2005] or binning with cell values computed as the mean of the z-coordinates of all of the points falling into that cell. Although binning results in a noisier DEM, it is very fast, and may, therefore, be a better option for some applications. Multiple point clouds can be integrated to increase point density at the cost of increased processing time. For each new DEM a set of customizable analyses and geospatial workflows specified in a Python file is run and the display is updated. Tangible Landscape provides a library of geospatial functions built on top of GRASS GIS modules and an API for developing custom workflows. The modeled analyses and scanning parameters can be modified as the system runs.

Tangible Landscape has been implemented as a set of components – the *r.in.kinect* add-on module [Petrasova 2016b] and the *Tangible Landscape* plugin [Petrasova 2016a] – for the active development version of GRASS GIS (Fig. 6). The *r.in.kinect* add-on module continuously imports and filters point cloud data from the 2nd generation Kinect into GRASS GIS as raster or vector maps. It has been implemented as a separate component so that it can easily be used for other applications. The *Tangible Landscape* plugin provides utilities, a library of analyses, and a GUI dialog for tangible interaction in GRASS GIS. The *Tangible Landscape Immersive Extension* [Tabrizian

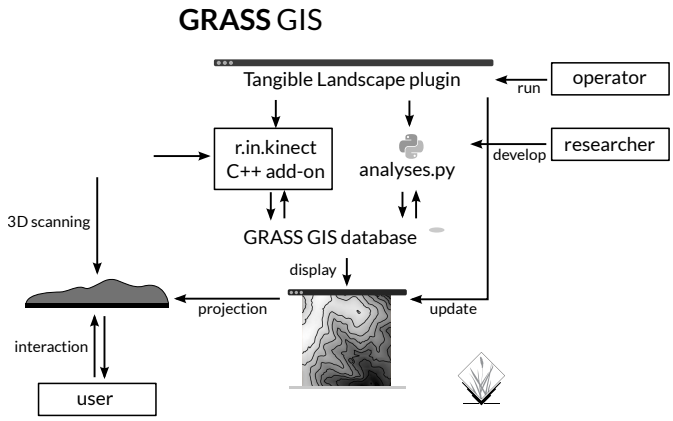


Fig. 6. Software schema for Tangible Landscape.

2016] links Tangible Landscape with Blender, an open source 3D modeling and animation program [Blender Online Community 2016] enabling 3D rendering on screens or head-mounted displays.

Users can develop new analyses for Tangible Landscape using the GRASS GIS Python Scripting Library. Since many tasks in GIS are not appropriate for tangible interaction Tangible Landscape was designed to supplement, not replace GRASS GIS's GUI, CLI, and scripting application program interface (API). Tangible Landscape currently runs on Linux and Mac OSX. Dependencies include the Point Cloud Library [Rusu and Cousins 2011; PCL Development Team 2016], OpenKinect's libfreenect2 [OpenKinect Development Team 2016], OpenCV [OpenCV Development Team 2016], watchdog [Mangalapilly 2011], and GRASS GIS [GRASS Development Team 2016f].

2.4. System resolution, accuracy, and speed

The spatial resolution of the system is determined by the depth camera's parameters and the distance from the camera to the model. Given the Kinect's 70 $\times$ 60 degree field of view and 512 $\times$ 424 pixels depth resolution, the spatial resolution of the system ranges from 1.4 to 2.7 mm when the scanner is 0.5 to 1 m above the model. In the experiments the Kinect was mounted approximately 0.6 m above a 23.5 x 23.5 cm model so we used 2 mm resolution, which translated to 6 m in the real world given the scale of the physical model. Kinect depth measurements suffer from noise as quantified by Sarbolandi et al. [Sarbolandi et al. 2015] We minimized the negative effects of noise by placing the model closer to the center of the camera and by smoothing the data using the Moving Least Squares reconstruction method [Rusu and Cousins 2011].

Fig. 7 and Table VI show the accuracy of the experimental setup. We scanned one of the CNC routed model used in this study and compared it with the original DEM. We removed the systematic error (i.e. the vertical shift) since that is corrected as part the analysis for the experiment and does not influence the results. The difference shown in Fig. 7 was computed from a single frame. Although it is possible to integrate multiple frames, we chose to construct the DEM from a single frame for faster smoothing.

Tangible Landscape's speed – typically ranging from approximately 2 fps to 0.5 fps – depends upon the size of the model, the point cloud processing methods, and the

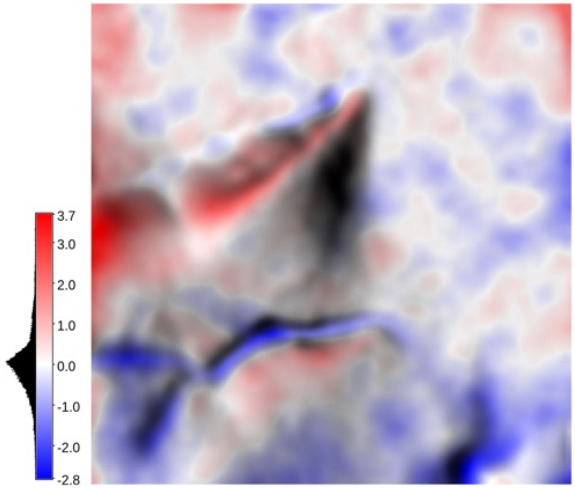


Fig. 7. Accuracy assessment: the difference between original digital elevation model and the scanned elevation of a digitally fabricated physical model of the same landscape. The scanned elevation is higher than the original digital elevation model in blue areas and lower in red areas. Legend values are in millimeters.

Table VI. Scanning accuracy (mm)

Min	Max	Mean	Stdev	1st Q	3rd Q
-2.8	3.7	-0.02	0.7	-0.4	0.3

Table VII. Scanning speed

Size, Process	Small	Medium
Physical size	23.5 cm $\times$ 23.5 cm	34 cm $\times$ 34 cm
Cells	13,456	26,235
Binning	0.51 s	0.71 s
Interpolation	0.74 s	0.97 s
Water flow	0.29 ± 0.01 s	1.05 ± 0.05 s
Contours	0.054 ± 0.004 s	0.061 ± 0.004 s
Difference	0.036 ± 0.002 s	0.042 ± 0.003 s
Landforms	0.034 ± 0.003 s	0.084 ± 0.009 s

analyses chosen. Table VII compares times for different point cloud processing methods (binning and interpolation) and analyses (e.g. simulated water flow) for small and medium sized models. We used the small model size – 23.5 cm x 23.5 cm – in this study. We recommend the medium model size – 34 cm x 34 cm – for new users getting started with Tangible Landscape. The time needed to scan and obtain a model is less than 1 s, but the subsequent analyses may take more than 1 s. It takes 1.03 s to compute water flow over the smaller model with 0.74 s for interpolation and 0.29 s for the water flow simulation. If a user interacts with Tangible Landscape immediately after a scan has been captured, then they will have to wait for that scan to be processed before their change will be processed potentially doubling the total processing time. The current system, reported here, performs approximately twice as fast as the version used in

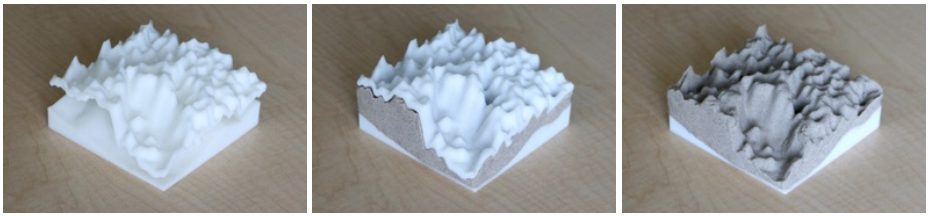


Fig. 8. Casting polymeric sand models with 3D printed molds. Source: [Petrasova et al. 2015].

this study. Benchmarks were performed using a System76 Oryx Pro with a 6th generation Intel Core processor i7-6700HQ (2.6 GHz, up to 3.5 GHz, 6 MB cache, 4 cores, 8 threads), 16 GB dual-channel DDR4 random-access memory (2 $\times$ 8 GB), M.2 SSD storage (540 MB/s sequential read, 520 MB/s write), and NVIDIA GeForce GTX 1060 running Ubuntu 16.04 LTS (64-bit), GRASS GIS 7.2, and Tangible Landscape 2c1ede9.

2.5. Fabrication

We typically use familiar, everyday materials – like sand and wooden blocks – for modeling with Tangible Landscape. Interactions – like sculpting sand and moving wooden blocks – are analogous to everyday tasks so users should subconsciously know what to do and how to do it, leveraging existing sensorimotor schemas. The materiality – the feel, look, and physics – of the media matters. The choice of material can afford different interactions and mediate meaning, emotion, and motivation. We typically use a polymer-enriched sand for the physical terrain model so that users can easily sculpt forms in a deformable medium that will hold its shape, has good plasticity, and has a familiar feel and aesthetic. Digital fabrication technologies like computer numeric control (CNC) manufacturing and 3D printing can be used to create molds for casting polymer-enriched sand into precise yet deformable models (Fig. 8). Cast sand models can precisely represent complex shapes and volumes that are challenging to model by hand and can easily be re-cast after use.

Tangible Landscape can be also be used as a modeling aid for sculpting. Static projections or dynamic analytics like differencing or water flow can be used as guides for hand sculpting terrain models. Users can project their target digital elevation model and contours over their polymeric sand model as a static guide for sculpting. Tangible Landscape can also dynamically compute the difference – i.e. cut and fill – between the target digital elevation model and the scanned model that has been sculpted. The difference can provide a real-time guide where to add or remove sand in order to match the target digital elevation model (Fig. 9).

2.6. Applications

Using GRASS GIS's extensive libraries for geospatial modeling, analysis, and simulation we have developed a wide range of applications for Tangible Landscape [Petrasova et al. 2015]. Design and planning applications include grading, cut and fill analysis, stormwater management, erosion control, trail planning (Fig. 10), viewshed analysis, and the assessment of solar potential. Scientific applications include subsurface visualization (Fig. 5), urban growth modeling, disease management, and invasive species management (Fig. 12). Disaster management applications include flood control, wildfire management, and coastal change and adaptation (Fig. 11). Educational applications include spatial training and serious gaming. See Appendix D for videos of

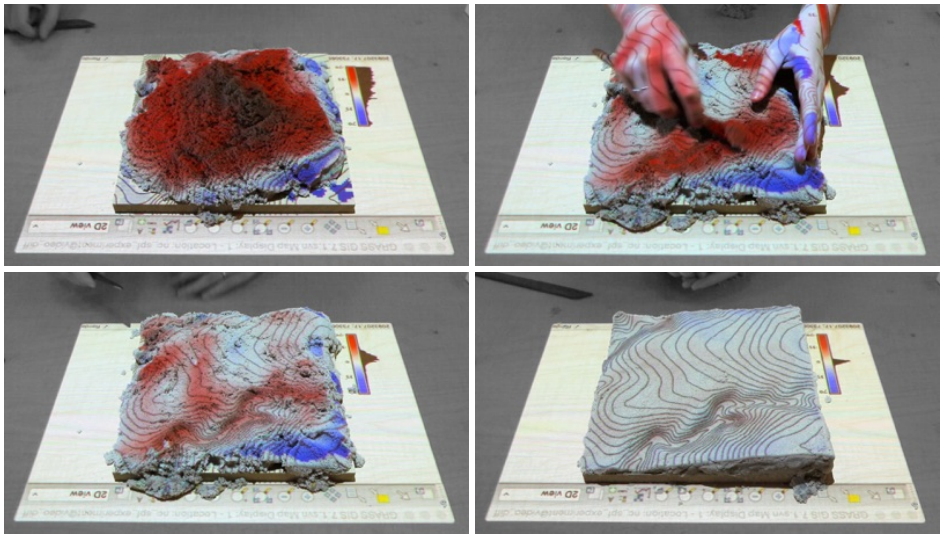


Fig. 9. Sculpting a terrain model using Tangible Landscape's difference analytic. Blue means add sand and red means remove sand.

landscape design (D.1), landscape planning (D.2), subsurface visualization (D.3), viewshed analysis (D.4), wildfire spread (D.5), and urban growth applications (D.6).

3. COUPLING EXPERIMENT

After a pilot study [Harmon 2016] we conducted an experiment to study how coupling digital and physical models of topography mediates 3D spatial performance. This experiment compared how users performed in a terrain modeling task using digital 3D modeling with Rhinoceros, analog modeling by hand, and projection-augmented modeling powered by Tangible Landscape. Digital modeling with a GUI affords precise transformations and dynamic modes of visualization such as 3D orbiting, zooming, and ray traced shading, but is not embodied. Theoretically analog modeling – modeling by hand or with tools – should be embodied, affording subconscious, kinaesthetic sensing and manipulation of form. By sensing form subconsciously with the body users should have more cognitive resources for critiquing their work and strategizing their next moves. Projection-augmented modeling couples digital mapping with physical modeling. Theoretically it should combine affordances of both – enabling enriched visualization and physical sensing and manipulation – to offer more feedback. While more feedback may help users better assess and critique their performance so they can strategize their next moves, too much feedback may be a cognitive overload resulting in distraction, frustration, and demotivation. Physical sensing and manipulation, however, should offload some of this cognitive work onto the body.

3.1. Methods

In this experiment 18 participants tried to accurately model a given study landscape digitally, by hand, and with Tangible Landscape.

Participants. All of the participants – either graduate students, faculty, or professionals in landscape architecture or geographic information science – had experience

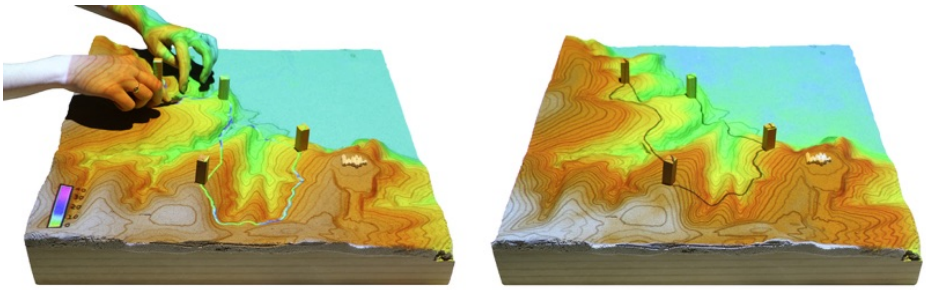


Fig. 10. Trail planning with Tangible Landscape. Users create waypoints for a trail by placing markers on the terrain model. The optimal route between the waypoints is automatically computed based on the least cost path and the solution of the traveling salesman problem. Feedback includes the slope along the trail and viewsheds from waypoints. Users can also change the topography and build bridges. Source: [Petrasova et al. 2015].

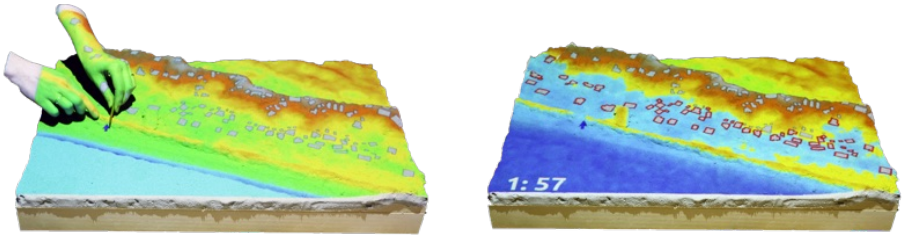


Fig. 11. Testing coastal flood defences with Tangible Landscape. Users try to save houses from coastal flooding by building coastal defenses. Given a small handful of polymer-enriched sand – their budget – they sculpt new dunes as flood defenses. The foredune is then breached at a random location and simulated storm surge is rerun, flooding any vulnerable houses.

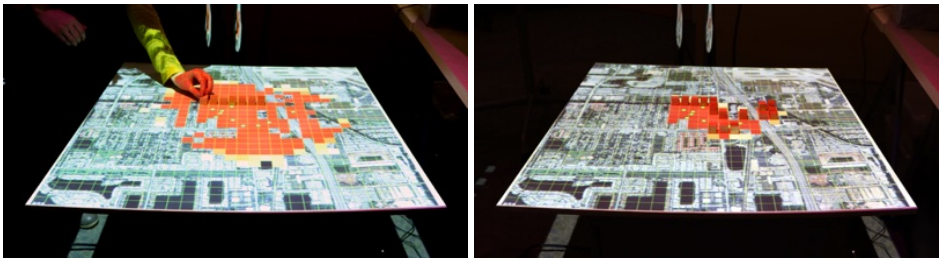


Fig. 12. Managing the simulated spread of termites through a city with Tangible Landscape by treating city blocks. After watching the simulated spread of termites across the city users place preventative treatments to try to contain the invasion. To treat a city block they place wooden cubes representing preventive treatments on the game board leveraging basic motor skills from childhood play. After they treating 10 blocks the stochastic simulation is rerun so users can see how well they contained the invasion.

Table VIII. Participants

Group	No.	GIS training	3D modeling expertise
Landscape architecture students	6	0	0
GIS students	6	6	0
Academics & professionals	6	0	2

thinking spatially. These participants were selected in order to compare the performance of novices and experts in grading (i.e. earthworking), GIS, and 3D modeling. Of the 12 graduate students 6 were in the 2nd semester of the Master of Landscape Architecture program at North Carolina State University. These students were just beginning to learn to read and manipulate contour lines in the class Landform, Site Grading, and Development Systems. They had no experience with GIS, 3 credit hours of training in digital representation, and no experience with Rhinoceros. The other 6 graduate students were in the 2nd semester of the Master of Geospatial Information Science and Technology program at North Carolina State University. They had at least 9 credits hours of training in GIS, but had no experience with contour maps, digital terrain modeling, or 3D modeling. The other 6 participants were either landscape architecture faculty with professional experience or practicing professionals in landscape architecture from around the world. Of these 2 were experts in 3D modeling with over 10 years of experience using Rhinoceros and similar programs including Maya and 3ds Max. These experts felt comfortable and proficient modeling a wide variety of geometries and had developed unique, personalized, and highly adaptable workflows (see Table VIII).

Experimental design. Participants were given three tasks – to digitally sculpt a model of a landscape using the 3D modeling program Rhinoceros, to hand sculpt a model of a landscape in polymer-enriched sand, and to sculpt a model of a landscape using Tangible Landscape. They had 10 minutes to complete each task. The time limit was calibrated based on the time it took the research team to comfortably complete each task. The modeling tasks were reordered for each participant to partially account for learning and fatigue by counterbalancing the experiment using a Latin square design. Participants were asked to model the same landscape for each task so that their performance could be quantitatively assessed using spatial modeling, statistics, analysis, and simulation. Their modeling processes were qualitatively assessed through direct observation supported by photographic and video documentation. After completing the tasks the academics and professionals were interviewed. We intended to conduct semi-structured interviews with all of the participants, but the students were too tired by the end of the experiment. Psychometric tests were not used to assess spatial ability because these do not address geographic scales, spatial relations, domain-specific knowledge and abilities [Lee and Bednarz 2009; Bednarz and Lee 2011; Wakabayashi and Ishikawa 2011], or embodied cognitive ability. See Appendix B for instructions for running the experiment and Appendix C for semi-structured interview guidelines.

We used a region of Lake Raleigh Woods in Raleigh, North Carolina as the study landscape for this experiment. A real landscape was used because computer generated landscapes often look surreal and may lack distinct landforms and clearly defined streams. This study region has distinctive, clearly defined landforms – a stream valley flanked by a ridge and sloping towards the lake. The digital elevation model for this region was derived from a 2013 airborne lidar survey using the regularized spline with tension interpolation method [Mitasova et al. 2005]. We CNC routed a model of the study landscape in Baltic birch using 2 parallel finish cuts with a 3.175 mm diam-

eter ball-nose bit with a 1.27 mm stepover. Participants were given this model at the beginning of the experiment to use as a reference during each task.

Digital modeling. Participants had 10 minutes to digitally model the study landscape in Rhinoceros 5, a non-uniform rational basis spline (NURBS) based 3D modeling program designed for precise freeform curve and surface modeling [McNeel and Associates 2016]. After 10 minutes of training participants modeled the study landscape as a NURBS surface by vertically translating control points (Fig. 13). They started with a flat NURBS surface that was divided into a 10 x 10 grid of control points. As a reference their model space also included a locked representation of the study landscape as 3D contours. At any point participants could rebuild the surface with a higher density of control points for finer, more nuanced control. This method is relatively simple and analogous to basic actions in sculpture – pushing and pulling. We developed and tested this method as a simple, straightforward technique for digital freeform surface modeling that could be taught quickly, yet could produce an accurate model. Through testing we determined that novice users took approximately 10 minutes to model the surface with a 10 x 10 grid of control points before wanting to rebuild the surface with more control points. See Appendix E for videos demonstrating the training (E.1) and digital 3D modeling task (E.2).

Different 3D modeling programs afford very different interactions, modeling tools, data structures, and modes of representations. Developing a simple, intuitive modeling technique drove the choice of software in this study. After testing 3D modeling workflows in Rhinoceros, Vue, Maya, and SketchUp we chose Rhinoceros 5 for this task because it is popular with designers such as architects, has a wide variety of modeling tools, can precisely represent continuous surfaces, and has plugins for importing and exporting geographic data. To make a fair comparison between digital, analog, and tangible modeling, the digital modeling technique needed to be relatively analogous to hand sculpting – i.e. pushing and pulling rather than painting in 3D, drafting in 3D, or stacking voxels. There is an analogous modeling technique – pulling and pushing vertices – using the Sandbox tool in SketchUp [Trimble 2016], but this creates a triangulated irregular network (TIN) rather than a continuous NURBS surface. Vue has a terrain editor designed for intuitive 3D painting and sculpting [e-on software 2012], but it requires continual retuning of tool parameters and generates a TIN. TINs less accurately represent continuous surfaces like topography and would generate artifacts in our analyses. In another pilot study [Harmon et al. 2016c] we tested Vue's terrain editor. We found that participants tended to over-exaggerate distinct features in the landscape like stream channels, while ignoring more subtle features like slopes. We also determined that the resulting models had an obvious triangulated structure that caused exaggerated slopes and discontinuous water flow in our analyses. While voxel modeling programs like Voxel Builder and games like Minecraft [Mojang 2009] are very intuitive to use, they can also be slow to use, create blocky rather than continuous surfaces, and are not typically used in design professions like architecture and landscape architecture.

Analog modeling. After a brief explanation and demonstration participants had 10 minutes to sculpt the study landscape in polymer-enriched sand by hand or with a wooden sculpting tool (Fig. 14). They were given a CNC-routed model of the study landscape as a reference, a sculpting tool, and a 3D scale. Participants were shown how the 3D scale could be used to measure the height of the model. A lamp on the table cast shadows across the model for hillshading. See Appendix E for a video demonstrating the analog 3D modeling task (E.3).

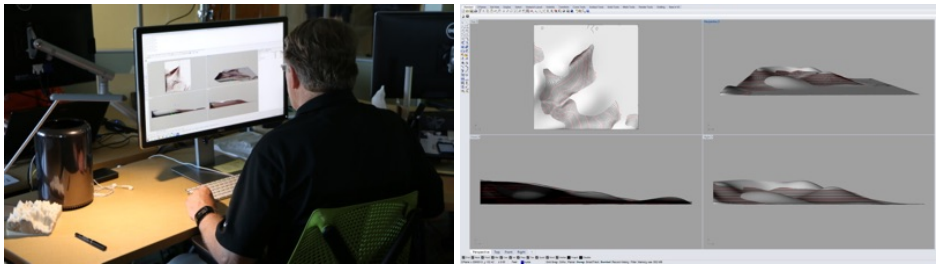


Fig. 13. Coupling experiment – digital modeling. A participant digitally sculpts the study landscape in Rhinoceros using 3D contours as guides.

Fig. 14. Coupling experiment – analog modeling by hand. A participant sculpts the study landscape by hand using a physical model as a reference.

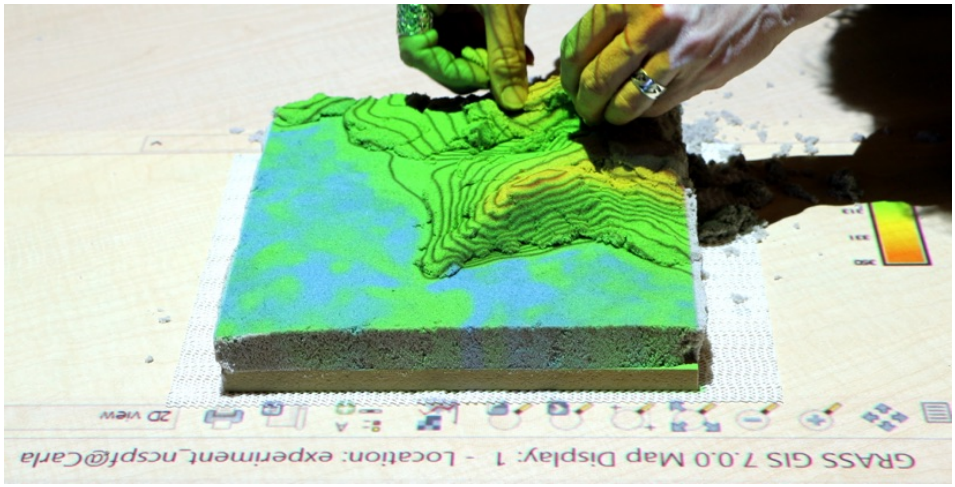


Fig. 15. Coupling experiment – projection-augmented modeling. A participant sculpts the study landscape using the projected elevation and contour maps as guides.

Projection-augmented modeling. After a minute long explanation and demonstration participants had 10 minutes to sculpt a projection-augmented, polymer-enriched sand model of the study landscape by hand or with a wooden sculpting tool (Fig. 15). Tangible Landscape was used to project an elevation map of the study landscape with contours and a legend onto participants' sand models as guide for sculpting. Participants were also given CNC routed reference model and a 3D scale ruled in map units. After an explanation of the contour map, elevation color table, and legend, participants were shown how the 3D scale could be used to measure the elevation of their scale models. See Appendix E for a video demonstrating the projection-augmented 3D modeling task (E.4).

Data collection and analysis. We used Tangible Landscape to scan the finished models built using analog hand modeling and projection-augmented modeling. The scans were captured as point clouds, interpolated as digital elevation models using the regularized spline with tension method, and stored as raster maps in a GRASS GIS database. The NURBS surfaces modeled in Rhinoceros were exported as raster elevation maps, imported into GRASS GIS, randomly resampled, re-interpolated using the regularized spline with tension method, and stored as raster maps in a GRASS GIS database. The data from Rhinoceros was randomly resampled and re-interpolated to account for differences and irregularities in resolution, data density, and point spacing. For each set of models – digital, analog, and augmented – we computed raster statistics (i.e. per cell statistics), topographic and morphometric parameters, and simulated water flow.

Mean elevation. We computed the mean elevation and standard deviation of elevation for each set with the module *r.series* [GRASS Development Team 2016d]. The mean elevation is the average of each cell of all elevation maps in a given set of models.

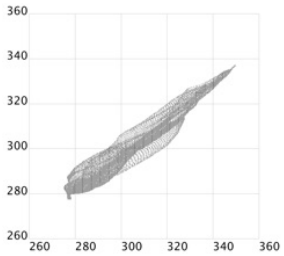
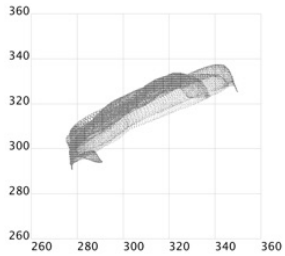
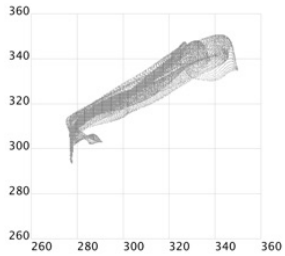
Standard deviation of elevation. The standard deviation of elevations represents the set's consistency. We computed the standard deviation of elevation for each set with the module *r.series* [GRASS Development Team 2016d]. This is the standard deviation of each cell of all elevation maps in a given set of models. We used a sequential color table from Color Brewer [Brewer 1994; Brewer and Harrower 2016].

Mean difference. In order to compare participants' modeling performance between sets we computed the difference between the linearly regressed reference elevation and the mean elevation for each set. The difference between the reference and mean elevation maps should show where the mean elevation values for each set are too low or too high, i.e. where they needed to add (blue) or remove (red) volume to match the reference. There were, however, systematic errors in the scanned models. Table IX shows the vertical shift in the hand sculpted and projection-augmented models caused by scanning and georeferencing. We used linear regression to vertically rescale and translate the reference elevation in order to account for these systematic errors in the difference calculation,

$$\Delta = (a + b * z_0) - \overline{z} \tag{1}$$

where:
- Δ is the difference
- z_0 is the reference elevation map
- $\overline{z}$ is the mean elevation of maps in a set
- a is the intercept of the regression line
- b is the slope of the regression line.

Table IX. Bivariate scatterplots of elevation values

	Digital	Analog	Augmented
Mean			
	Reference	Reference	Reference

Standard deviation of difference. In order to find how consistently the models fit the reference landscape we computed the standard deviation of difference by first calculating the difference between the linearly regressed reference elevation and each modeled elevation map in a set to generate a set of difference maps. Then we calculated the standard deviation for that set of difference maps.

Mean slope. We derived the slope of the mean elevation of each set using differential geometry [Wood 1996] with the module *r.param.scale* [GRASS Development Team 2016c].

Mean landforms. We identified landforms for the mean digital elevation model for each set using geomorphons, a pattern recognition method for landform classification based on the openness of the terrain and implemented as the add-on module *r.geomorphon* [GRASS Development Team 2016b]. Geomorphon works effectively across spatial scales because the neighborhood search size for pattern recognition is spatially variable – it depends on the visibility of the cell [Jasiewicz and Stepinski 2013]. Landforms can be classified as flat, peak, ridge, shoulder, spur, slope, hollow, footslope, valley, or depression (Fig. 16).

Mean minimum distance. We computed the mean minimum distance between cells with water flow, ridges, and valleys on the models and the reference landscape as a measure of positional accuracy.

Systematic errors. Table IX shows systematic errors in the hand sculpted and projection-augmented models. These models are vertically shifted due to scanning and have low values along the borders caused by slumping sand. While we used linear regression to vertically shift and rescale the difference in elevation, we did not mitigate the systematic errors along the borders.

3.2. Results

We used pairwise comparison to study how performance varied between novices and experts (Fig. 17). After analyzing all participants performance (Table X) we compared students with academics and professionals (Table XI - XII). Then we compared landscape architecture students with GIS students (Table XIII - XIV). Last we compared academics and professionals with and without 3D modeling expertise (Table XV - XVI).

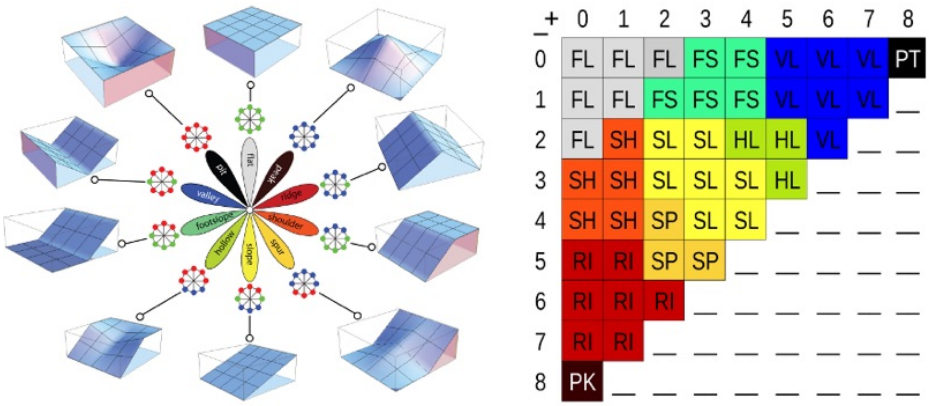


Fig. 16. Landforms identified by *r.geomorphon* – 1) flat, 2) peak, 3) ridge, 4) shoulder, 5) spur, 6) slope, 7) hollow, 8) footslope, 9) valley, and 10) depression. Source: [GRASS Development Team 2016b].

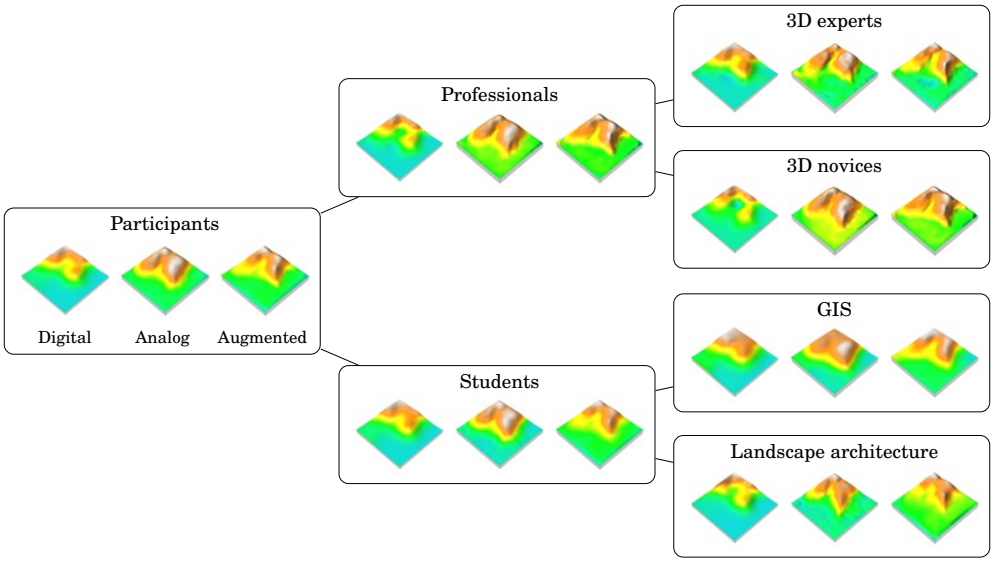
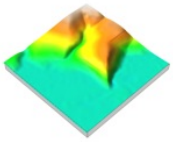
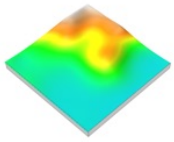
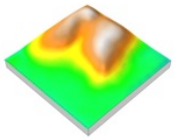
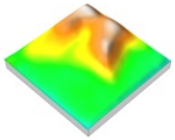
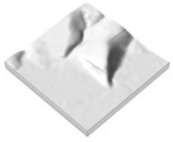
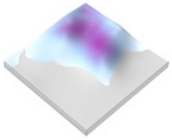
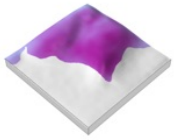
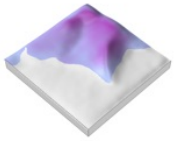
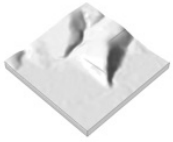
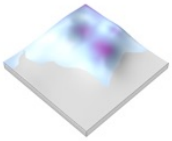
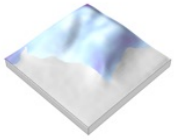
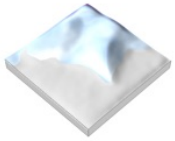
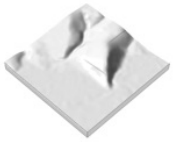
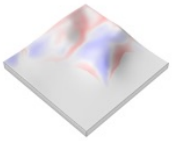
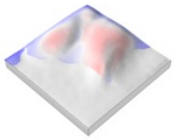
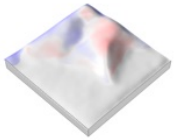
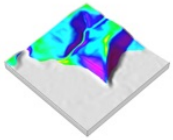
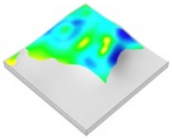
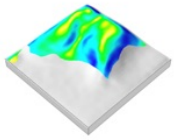
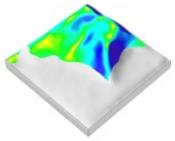
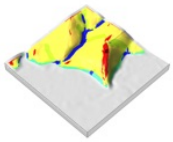
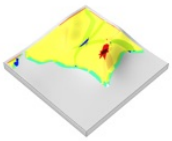
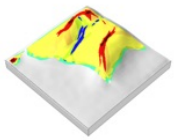
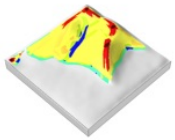


Fig. 17. Pairwise comparison of the mean digital elevation models grouped by category of participants

Overall participants performed worst with digital modeling, better with analog modeling, and best with projection-augmented modeling. Table X compares participants' performance with each technology through 3D maps of raster statistics and geospatial analyses. When digitally modeling with Rhinoceros participants tended to create very approximate massings of the topography with serious errors in the interior space and with indistinct landforms that only hinted at the morphology of the landscape. When they sculpted by hand their models tended to be descriptive – differing substantially from the reference, but accurately representing most of the landforms.

Table X. Coupling experiment: maps of per-cell statistics and geospatial analyses draped over 3D topography for all participants

	Reference	Digital	Analog	Augmented
Mean elevation (275 – 350)				
Stdev. of elevations (0 – 32)				
Stdev. of difference (0 – 24)				
Mean difference (-36 – 36)				
Mean slope (0 – 90)				
Mean landforms (1 – 10)				

Their performance improved when they sculpted projection-augmented models – the resulting models fit the reference better, had more defined topography, and accurately represented most of the landforms.

When participants are analyzed as groups – as GIS students, Landscape Architecture students, academics and professionals without 3D modeling expertise, and academics and professionals with 3D modeling expertise – these trends generally still hold, albeit with some very important exceptions. Unlike all other groups, the 3D modeling experts performed extremely well in the digital modeling task with a very low

Table XI. Students

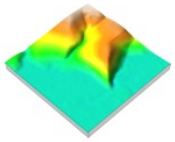
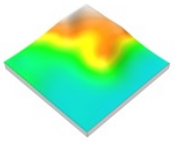
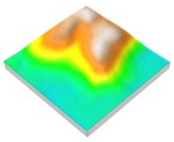
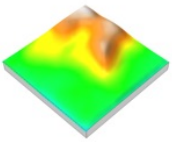
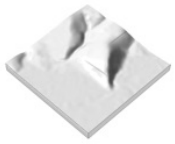
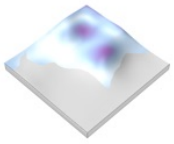
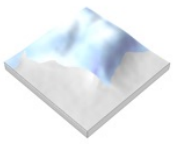
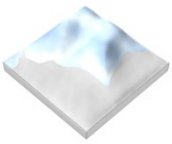
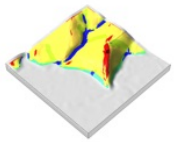
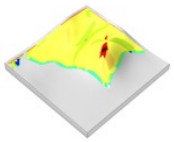
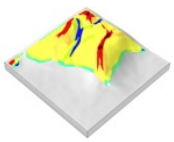
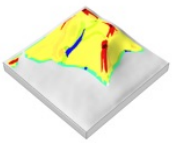
	Reference	Digital	Analog	Augmented
Mean elevation 275 350				
Stdev. of difference 0 24				
Mean landforms 1 10				

Table XII. Academics and professionals

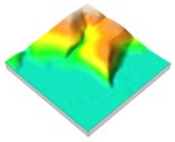
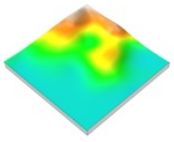
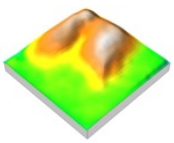
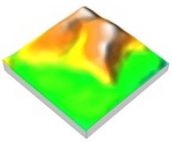
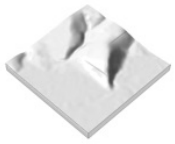
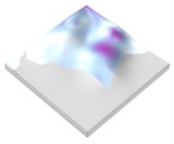
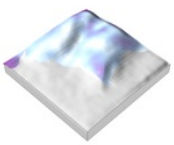
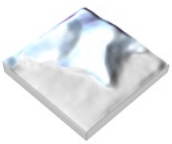
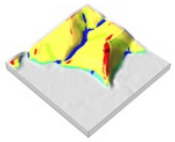
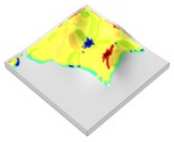
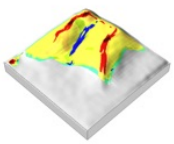
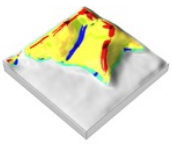
	Reference	Digital	Analog	Augmented
Mean elevation 275 350				
Stdev. of difference 0 24				
Mean landforms 1 10				

Table XIII. Landscape architecture students

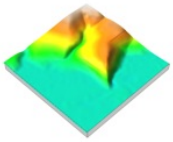
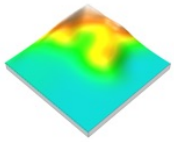
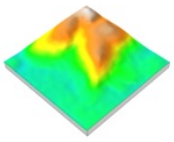
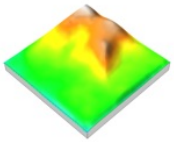
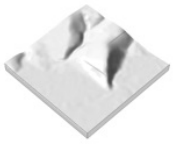
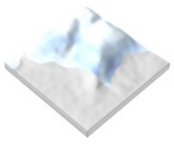
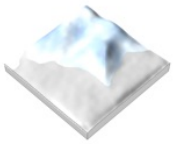
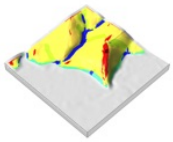
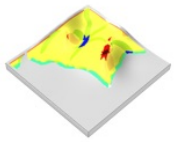
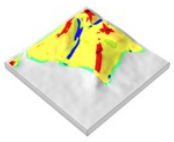
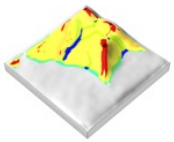
	Reference	Digital	Analog	Augmented
Mean elevation 275 350				
Stdev. of difference 0 24				
Mean landforms 1 10				

Table XIV. GIS students

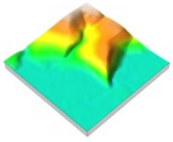
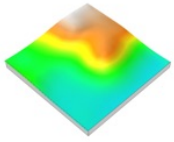
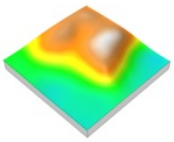
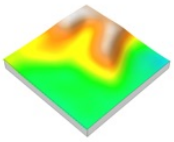
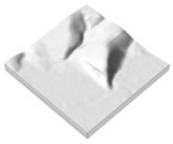
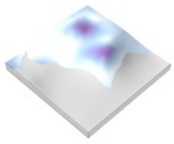
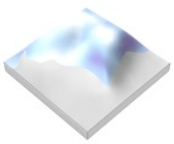
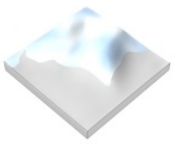
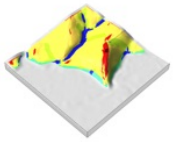
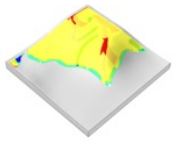
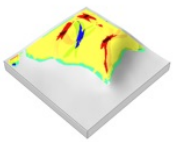
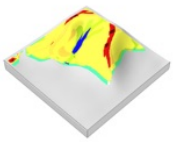
	Reference	Digital	Analog	Augmented
Mean elevation 275 350				
Stdev. of difference 0 24				
Mean landforms 1 10				

Table XV. Academics and professionals without expertise in 3D modeling

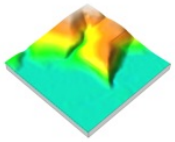
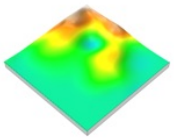
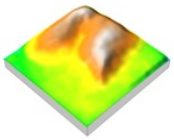
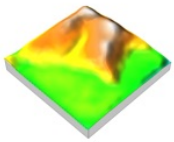
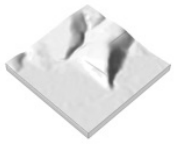
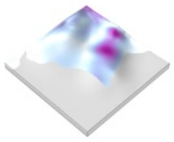
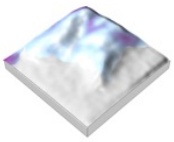
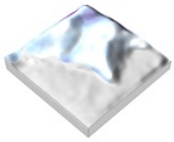
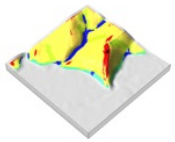
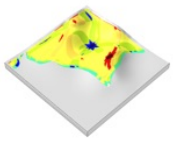
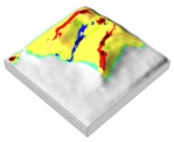
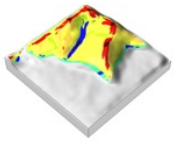
	Reference	Digital	Analog	Augmented
Mean elevation 275 350				
Stdev. of difference 0 24				
Mean landforms 1 10				

Table XVI. Academics and professionals with expertise in 3D modeling

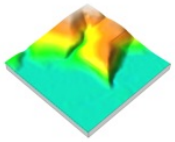
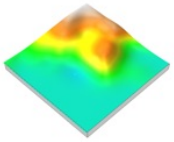
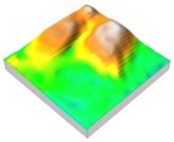
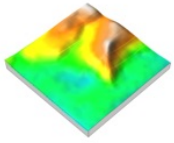
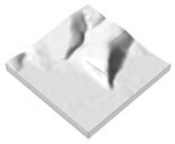
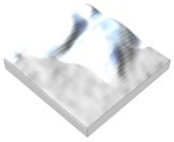
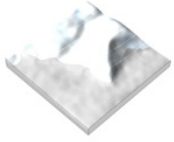
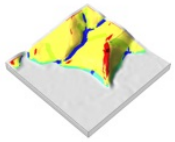
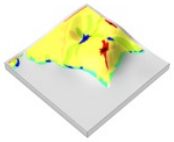
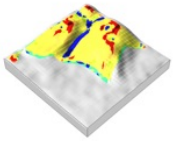
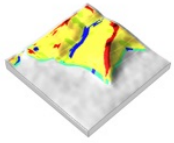
	Reference	Digital	Analog	Augmented
Mean elevation 275 350				
Stdev. of difference 0 24				
Mean landforms 1 10				

Table XVII. Coupling experiment: percent cells

Method	Concentrated flow	Ridges	Valleys
Reference	1.16	2.10	2.90
Digital	1.34	0.69	0.49
Hand	0.97	3.85	1.48
Augmented	1.05	4.18	1.63

Table XVIII. Mean minimum distance (ft) of features from reference features

Method	Concentrated flow	Ridges	Valleys
Digital	19303	105764	110017
Hand	12021	25238	70287
Augmented	11898	20110	30980

standard deviation of difference, but still only hinted at the landforms (Table XVI). This means that even for expert 3D modelers, projection-augmented modeling should be a better, faster tool for quick modeling tasks.

All participants. Tables X, XVII, and XVIII show the results for all participants. When digitally modeling with Rhino participants were relatively consistent and accurate except in the interior space. The interior space – the main valley and the ridge – tended to have serious errors. They built very abstract massing models approximating the general shape of the landscape without any detail. The slopes in these models were consistently too low and gradual. They did not have enough curvature to create distinct landforms. These models only hinted at the morphology of landscape with a small cluster of valley cells and another small cluster of ridge cells. Key features like the stream channel were not represented.

We observed that most participants had trouble judging depth and perceiving interior space when digitally modeling. We also observed most participants using similar digital modeling strategies in a relatively linear fashion. Since the reference data – the 3D contour curves – clearly represented profiles, many participants first modeled the borders of the landscape relying heavily on the front and side viewports. Then in perspective view they began to pull up relative high points to build a rough massing model of the topography. Finally they began to refine its shape by pulling down relative low points to steepen slopes and form valleys. The 3D modeling experts, however, used unique modeling strategies developing their own techniques as they worked. One worked exclusively in the perspective viewport continually orbited above and below the model in order to compare the top and the bottom as he pushed and pulled points in a very freeform, iterative process. Only the experts had time to rebuild the 10 x 10 grid of control points as a 20 x 20 grid. This meant that all of the other models had half the effective resolution and were thus more approximate representations of the landscape.

When sculpting by hand participants used very different modeling strategies and as a result they built very different models as shown by the standard deviation of elevation. While inconsistent, their models were relatively accurate and captured the key landforms. While they tended to over exaggerate the main ridge, they did represent this ridge and most of the central valley. Due to slumping sand these models tended to be too low along the edges.

We observed participants using a wide range of different modeling strategies and techniques when sculpting by hand. They tended to work in a very freeform manner – switching freely between adding, removing, pressing, pushing, pulling, or smoothing sand. Some used only their bare hands, some used only the wooden sculpting tool, and others used their bare hands to sculpt and the tool the refine details. Most participants used the 3D scale to build highest point at the right height – some even buried the 3D scale in their model building around it. Some participants instead picked up the reference model and used its edges – its profiles – to build the sides of the model. Some participants also ran their fingers over the reference model to feel its shape. A few participants, however, ignored both the 3D scale and the reference model.

We observed participants freely and rapidly switch between sculpting the sand with their whole hands, their palms, the blade of their hands, or just their fingertips depending on whether they wanted to make a big move or a fine-tuned refinement. While participants could select and transform a group of control points in Rhinoceros, their control was limited by the grid size, i.e. the number of points, and their speed by the need to continually change the selection. When we tested Vue as an alternative with 3D painting and sculpting tools with parameters like size and intensity we found that these parameters required continually tuning, which interrupted the modeling process and slowed down interaction.

See Table XXVIII for select comments from interviews. One participant said that she felt anxious when digitally modeling, but felt calmer and more relaxed when hand sculpting sand. She found hand sculpting to be more intuitive, saying that while digital modeling had 'a long learning curve,' she could read the sand model 'like braille. . . I could feel the shape with my fingers.'

Overall participants performed best with projection-augmented modeling building more accurate models that represented the major landforms. These models had the lowest mean minimum distance for cells with concentrated water flow, ridges, and valleys (see Tables XVII & XVIII). We observed participants using the same modeling techniques, strategies, and tool use as they did when sculpting by hand. They worked in the same freeform manner, but – with the aid of the projected contours and elevation – had more consistent results as shown by the standard deviation of elevations. Overall, they had more accurate results – albeit with systemic errors along the borders due to slumping sand – as shown by the mean difference and standard deviation of differences. They correctly represented the main ridge and the central valley, but tended to miss details like the y-shaped branch at the head of of the stream channel.

One participant found that with projection-augmented modeling he was able to effectively combine the affordances of hand sculpture with the extra layer of data. He described an iterative strategy of additive modeling in which the projected 'contours were just a guide' – 'My general strategy was additive. I felt with my hands to try to match the contours. If I saw concavity in the contours then I felt the sand and sculpted that concavity. Finding the relative height, however, was challenging – it was subtle.'

Students vs. academics and professionals. Table XI and Table XII compare students' results with academics and professionals. Overall the academics and professionals only performed slightly better than the students. When digitally modeling both groups built very approximate massing models that only hinted at the landforms. While the professionals and academics managed to create a small cluster of valleys cells in the central stream channel, the students tended to miss this feature. When sculpting by hand both groups' performance improved dramatically; they built more accurate models as shown by the standard deviation of difference and better represented the landforms roughly capturing the main ridge and valley. With projection augmented modeling both groups' performance increased even more as shown by the low standard deviation of difference.

Landscape architecture vs. GIS students. Table XIII and Table XIV compare landscape architecture and GIS students' results. The GIS students built more abstract, approximate models than the landscape architecture students who tended to over-exaggerate the shape of the landscape. With each technology the GIS students' performance improved; they built more and more accurate models with more distinct landforms. The landscape architecture students' performance, however, did not improve significantly between analog and projection-augmented modeling. With digital modeling both groups had major errors along the ridge, but the GIS students also had serious problems with the interior space. The GIS students missed the central valley entirely, while the landscape students hinted at it with a small cluster of valley cells. When hand sculpting both groups made major improvements. The GIS students began to form the main ridge and valley. They tended to build too large a ridge with the highest point near its tip and significant errors on its slopes. The landscape architecture students tended to hand sculpt distorted, exaggerated landscapes with extras landforms, but roughly captured the y-shaped stream. With projection augmented modeling the GIS students captured the central valley and ridge, correctly modeled steeper slopes, and had much less errors as shown by the standard deviation of difference. The landscape architecture students, however, did not make significant improvements over their hand sculpted models.

Academics and professionals with and without 3D modeling expertise. Table XV and Table XVI compare landscape architecture academics and professionals with and without 3D modeling expertise. While the other academics and professionals performed poorly with digital modeling, better with analog modeling, and best with projection augmented modeling, the expert 3D modelers built very accurate models with each technology, but represented the landforms best with projection augmented modeling. While the rest of the academics and professionals had major errors on the ridge in the digital modeling task, the expert 3D modelers built accurate models with very low mean difference and standard deviation of difference. The 3D experts worked faster and had time to rebuild denser grids of control points so that they could build more detailed, higher resolution models. Despite their accuracy their digitally sculpted models still only hinted at the landforms with small clusters of valley and ridge cells. The 3D modeling experts represented more complete landforms when sculpting by hand, capturing details like the y-shaped branch of the stream, but missed much of the ridge. With projection augmented modeling the expert 3D modelers performed even better successfully capturing the ridge, the valley, and its y-shaped branch with few anomalous features. They used the wooden modeling tool to cut clean edges around their models minimizing systematic errors caused by slumping sand.

4. DIFFERENCE EXPERIMENT

We conducted an experiment to study how the difference analytic mediates 3D spatial performance when using a tangible interface for GIS. This experiment was designed to explore how participants used the difference analytic – to study their modeling process, rather than compare their performance against other technologies.

4.1. Methods

Difference analytic. The same 18 participants were asked to use Tangible Landscape's difference analytic to model a different region of Lake Raleigh Woods with a large central ridge flanked by valleys and a smaller, secondary ridge. Participants had 10 minutes to model this region in polymer-enriched sand using Tangible Landscape with the difference analytic (Fig. XIX). The difference between the reference elevation and the participant's modeled elevation was computed in near real-time and

projected onto the sand as an interactive guide. The linear regression of the reference and scanned elevation maps was used to correct shifts in scanning and georeferencing. The difference showed where sand needs to added (blue) or removed (red) in order to match the reference landscape (Fig. 9). While this feedback should show participants what to do next, it may provide too much data or be too challenging to rapidly parse. See Appendix E for a video demonstrating the 3D modeling task with the difference analytic (E.5).

Data collection and analysis. The final scan of each model was stored in a GRASS GIS database for analysis. We used raster statistics, the difference in elevation, topographic parameters, and morphometric parameters to compare the reference elevation and the set of modeled elevations. We computed the mean elevation, the standard deviation of elevations, and the standard deviation of difference for the set. To find the standard deviation of differences we first computed the difference between the reference elevation and each elevation and then used raster statistics to find the standard deviation of these differences. We also computed the difference between the reference elevation and the mean elevation for the set. The reference elevation used in the difference calculation was rescaled and shifted based on the linear regression of the reference and mean elevation. We computed the slope and landforms of the mean elevation. We also filmed each session, observed and took notes on participants' modeling processes, and interviewed landscape architecture professionals and academics.

4.2. Results

Participants performed well with the difference analytic – modeling simplified, but relatively accurate approximations of the landscape. Participants with expertise in 3D modeling performed even better than other participants – more accurately modeling the elevation and slopes, but not the landforms. Since these models were sculpted in sand, their edges tend to slump. This caused systematic artifacts in the analysis like low elevation values and steep slopes along the borders. Based on interviews and observations we found that the difference analytic enabled a rapid, iterative process of modeling, learning from computational feedback, re-remodeling, etc.

Per-cell statistics. Table XX shows maps of raster statistics for this experiment. The mean elevation for this set of models has the approximate shape of the reference elevation, but is much simpler, lacking many details. The standard deviation of elevations shows that participants consistently modeled the low point at the base of the secondary ridge in the same way, but modeled the primary ridge in different ways. The standard deviation of difference shows how consistently the models fit the reference. Overall participants performed well – there was little deviation from the best fit. Participants tended to perform poorly near the edges of models, especially in the corners. They also had trouble with the valleys and the low point by the secondary ridge.

Geospatial analyses. Table XXI shows geospatial analyses for this experiment. The mean elevation difference shows that participants tended to add too little to the edges of the model, too much to primary ridge, and too much to the slope of the secondary ridge. The mean slope shows that participants tended to model overly steep slopes for the primary ridge – exaggerating its form – but tended not model steep enough slopes for the secondary ridge. The mean landforms show that participants tended to clearly capture the central ridge and its spurs and the secondary ridge and its valley, but missed the other valleys (sees Tables XXI-XXII). Hollows – transitions between slopes, footslopes, and valleys – on the mean landform map hint at these other valleys in the right locations.

Table XIX. Difference experiment: a participant sculpts the study landscape using Tangible Landscape's difference analytic, which shows where to add sand (blue) and remove sand (red).

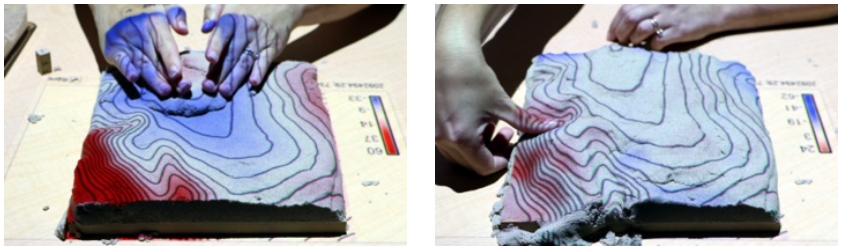


Table XX. Difference experiment: maps of per-cell statistics draped over a 3D rendering of the topography for all participants

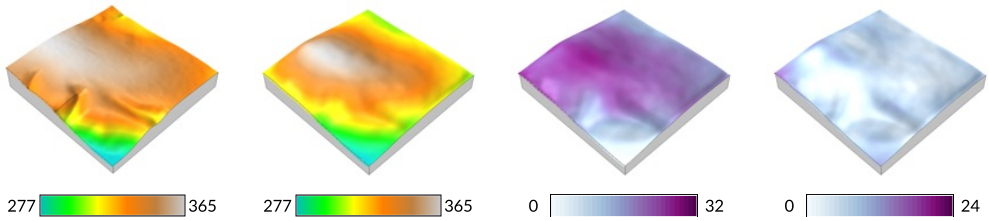
Reference elevation	Mean elevation	Stdev. of elevations	Stdev. of differences
			
277 365	277 365	0 32	0 24

Table XXI. Difference experiment: maps of geospatial analyses draped over a 3D rendering of the topography for all participants

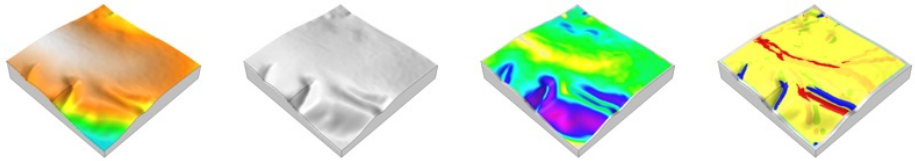
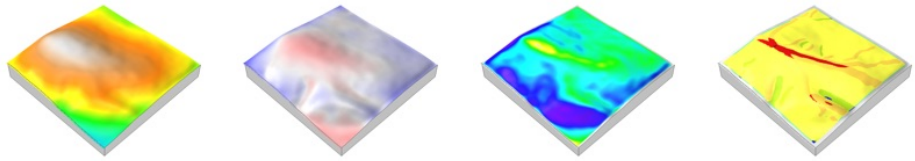
	Elevation	Elevation difference	Slope	Landform
Reference				
Mean				
	277 365	-36 36	0 90	1 10

Table XXII. Difference experiment: percent cells

Method	Concentrated flow		Ridges		Valleys	
	Reference	Mean	Reference	Mean	Reference	Mean
Difference	1.94	0.90	4.27	2.93	2.96	0.13

Table XXIII. Difference experiment: comparison of 3D modeling novices and experts

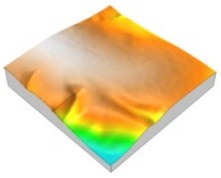
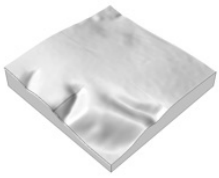
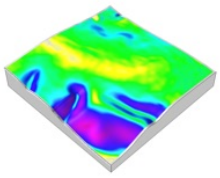
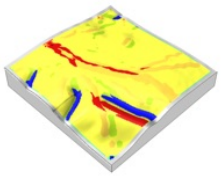
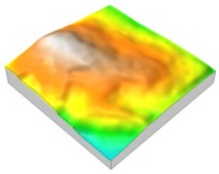
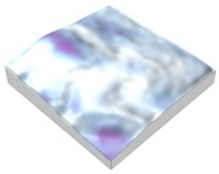
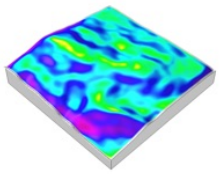
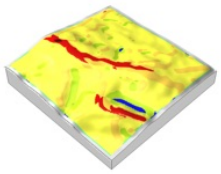
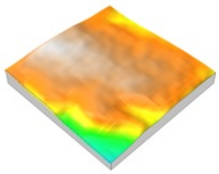
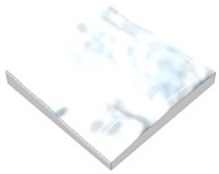
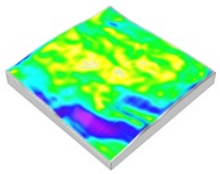
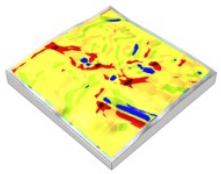
	Elevation	Stdev. of differences	Slope	Landforms
Reference				
3D novices				
3D experts				
	277 365	0 24	0 90	1 10

Table XXIII compares the performance of landscape architecture academics and professionals with and without 3D modeling experience. The 3D modeling experts built more accurate models with a lower standard deviation of differences and more accurate slopes. Since they did not, however, exaggerate the central ridge and did not smooth its slopes, they did not cleanly represent this landform. The others built more exaggerated models that more clearly captured the main ridge.

Interviews and observations. To successfully use the difference analytic participants had to think about topography as volume. They had to either add or excavate sand to make the models match. They described the process of modeling with the difference analytic as continual 'rebuilding to make it match.' They described an iterative process of continual refinement based on critical analysis – similar to Schön's reflection-in-action [Schön 1983] – but enhanced by computational feedback explaining that 'because Tangible Landscape gives immediate results it encourages an iterative process.' See Table XXVIII for select comments about this experiment.

The difference analytic is intuitive – participants were able to learn how to use it effectively without training, producing good, albeit exaggerated approximations of the landscape. Their models tended to have key morphometric characteristics – the primary ridge, its spurs, the low point, the secondary ridge, and one of the valleys – but these characteristics tended to be either exaggerated or under-exaggerated.

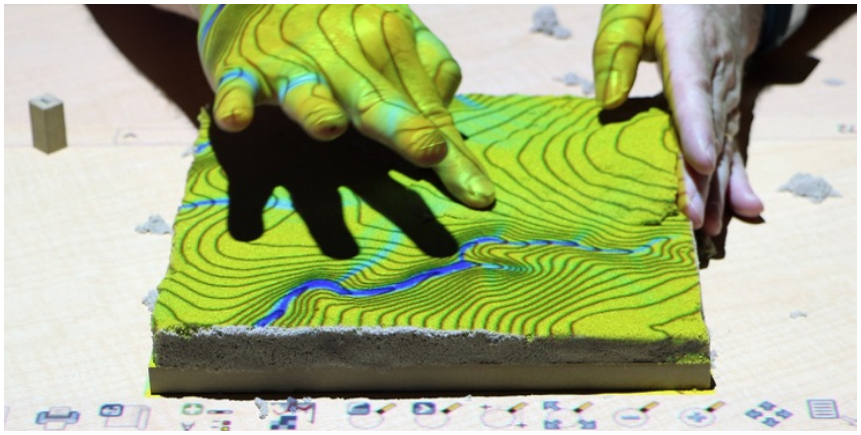


Fig. 18. Water flow experiment: A participant sculpts the study landscape using Tangible Landscape's water flow analytic.

5. WATER FLOW EXPERIMENT

After a pilot study [Harmon et al. 2016c] we conducted an experiment to study how water flow analytics mediate 3D spatial performance when using a tangible interface for GIS. This experiment was designed to study whether participants could link form and process using the water flow analytic – to assess how well they could understand the relationship between topographic form and the flow of water when using Tangible Landscape.

5.1. Methods

Water flow analytic. The same 18 participants were asked to model water flow across another region of Lake Raleigh Woods using Tangible Landscape with the water flow analytic. This region has a central ridge flanked by a large stream on one side and a small stream on the other. A third, smaller stream bisects the ridge. Participants had 10 minutes to model water flow across this region by sculpting polymer-enriched sand using Tangible Landscape with the water flow analytic. They sculpted sand models of the topography to direct the simulated flow of water. Water flow was simulated in near real-time as a diffusive wave approximation of shallow water flow. The module *r.sim.water* [GRASS Development Team 2016e] uses a path sampling technique to solve the shallow water flow continuity equation [Mitasova et al. 2004]. Participants could switch between the precomputed reference water flow – i.e. their target – and the water flow over their scanned model. See Appendix E for a video demonstrating the 3D modeling task with the water flow analytic (E.6).

Data collection and analysis. The final scan of each model was stored in a GRASS GIS database for analysis. We used raster statistics, simulated water flow, and the difference in simulated water depth to compare water flow across the study landscape and the set of models. We computed the mean elevation, the standard deviation of elevations, and the standard deviation of difference for the set. Then we simulated water flow across the mean elevation for the set of models and computed the difference between the reference and mean water flow.

Table XXIV. Water flow experiment: maps of per-cell statistics draped over a 3D rendering of the topography for all participants

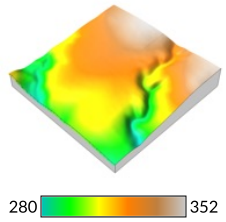
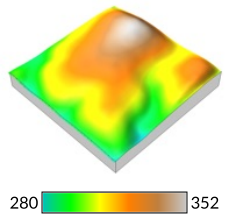
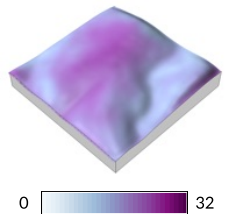
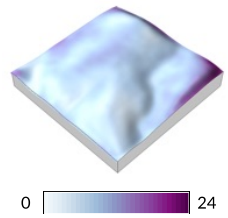
Reference elevation	Mean elevation	Stdev. of elevations	Stdev. of differences
			

Table XXV. Water flow experiment: maps of geospatial analyses draped over a 3D rendering of the topography for all participants

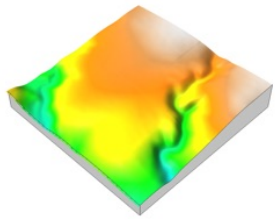
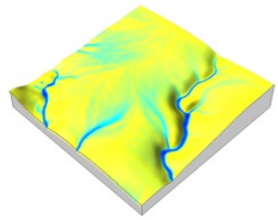
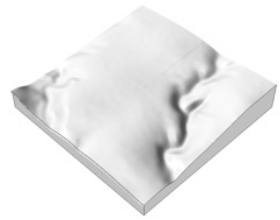
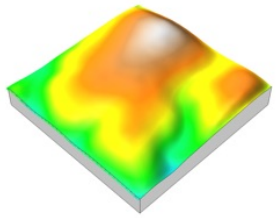
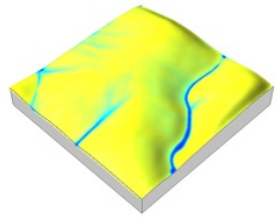
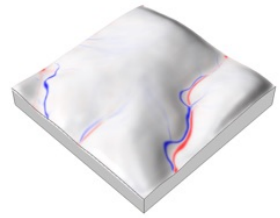
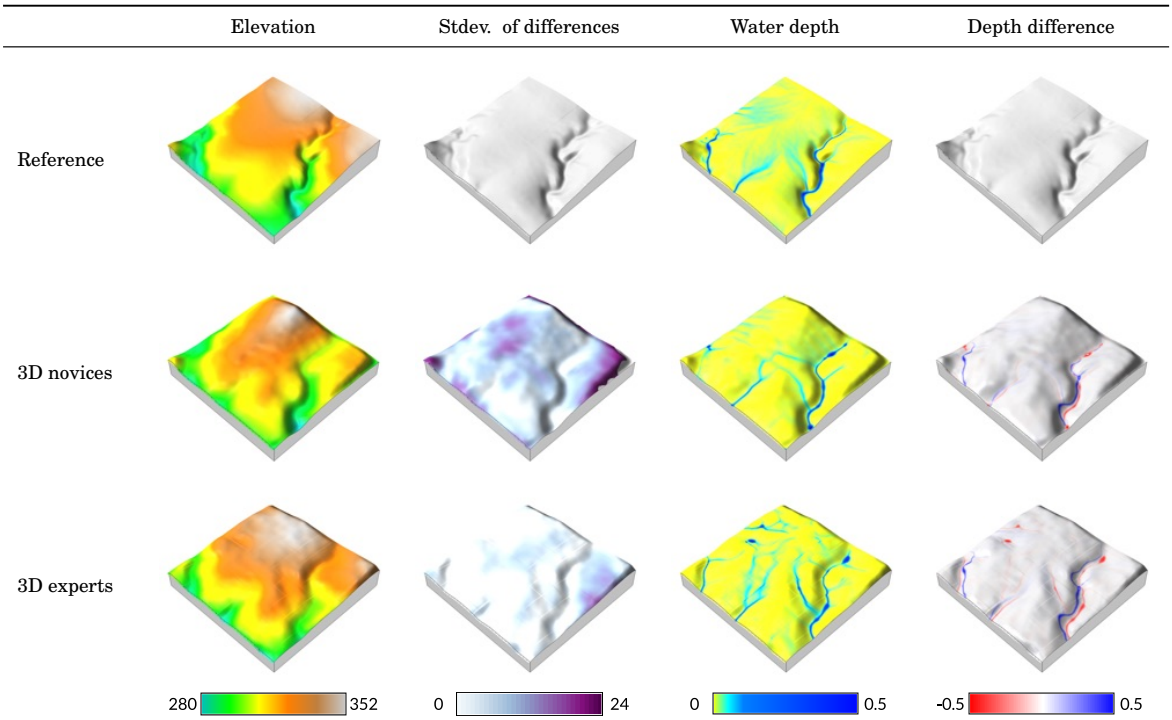
	Elevation	Water depth	Depth difference
Reference			
Mean			

Table XXVI. Water flow experiment: percent cells

method	concentrated flow		ridges		valleys	
	reference	mean	reference	mean	reference	mean
water flow	3.28	2.26	1.18	2.99	4.77	3.70

Table XXVII. Water flow experiment: comparison of 3D modeling novices and experts



5.2. Results

Participants performed well with the water flow analytic – approximately modeling two of the three streams. Participants with expertise in 3D modeling performed better than the others with more accurate models that captured all three streams.

Per-cell statistics. Table XXIV shows maps of raster statistics for this experiment. The mean elevation for this set of models has under-exaggerated valleys and over-exaggerated ridges in the right places. The mean elevation has 22.43% less valley cells than the reference, but 153.40% more ridge cells (see Table XXVI). The standard deviation of elevations shows that participants consistently modeled the streams, but varied greatly in their treatment of the primary ridge. The standard deviation of difference shows that participants performed well with the greatest deviation from the reference in the corners and along the edges. Again this is to be expected as sand slumps along the borders.

Geospatial analyses. Table XXV shows geospatial analyses for this experiment. The mean water depth map shows two of the streams and hints of the third. The streams are simplified, but in the right locations. The simplified stream channels lack micro-topography and thus details. Table XXVI shows that there was substantial concentrated flow (water depth $>= 0.05$ ft) in the streams, albeit 31.10% less than in the reference. This is to be expected since water flow over the sculpted models was only computed over the model region, while the reference water flow was computed over a larger region with the entire contributing watershed in order to produce an accurate

representation of water flow across the study landscape. The water depth difference map shows where water should be added (red) or removed (blue) to match the reference – i.e. where water should flow versus where it was modeled. The mean water flow tightly fit the reference, following similar, albeit simplified routes. Table XXVII compares the performance of landscape architecture academics and professionals with and without 3D modeling experience. With the water flow analytic the 3D modeling experts built more accurate models with a lower standard deviation of differences and captured all of the streams, while the others only captured two of the streams and hinted at the third.

Interviews and observations. With the water flow analytic participants focused on accurately modeling the streams, rather than the general shape of the topography. As a result the primary ridge was inconsistently modeled and over-exaggerated, while the streams were consistently modeled, had high volumes of continuous water flow, and flowed along the correct routes. This experiment required abstract spatial thinking linking form and process. Because water flow is controlled by the shape and gradient of the topography participants had to sculpt topographic form to drive water flow. This modeling process, however, helped them understand topography better because 'seeing the flow took away the mystery of topography.' We observed participants using an iterative modeling process with the water flow analytic – they *a)* sculpted the topography, *b)* observed how the water flow simulation changed, *c)* critiqued their water flow and topography, *d)* and continued to sculpt. Using the water flow analytic with this trial-and-error process participants were able to generate hypotheses, test hypotheses, and draw inferences about the way that water flows over topography. See Table XXVIII for select comments about this experiment.

6. DISCUSSION

6.1. Coupling physical and digital models

The coupling experiment was designed to test whether coupling a physical and digital model of topography can improve 3D spatial performance. We found that both novices and experts were able to take advantage of the affordances of both digital and analog modeling with projection-augmented modeling. All participants' without expertise in 3D modeling performed significantly better with projected-augmented modeling. They were able to combine the extra layers of data afforded by digital modeling with the intuitive interaction, speed, and detail afforded by analog modeling to model more accurate landscapes with distinct landforms. The expert 3D modelers performed well in the digital modeling task, but given the time constraints and limited palette of digital modeling tools were not able to form well defined morphological features. With projection-augmented modeling, however, they were able to work faster, refine their models sooner, and build distinct ridges and valleys.

While the novices' performance demonstrates that projection-augmented modeling can be intuitive – enabling high performance without training or experience – the experts' performance shows that the technology affords faster interaction. Intuitive interaction means that users are able to act, to do what they want to do, to make what they want to make quickly and reliably without training, experience, or even having to consciously think about it. The 3D modeling novices were able to build more accurate models with projection-augmented modeling. Drawing on embodied cognition they were able to feel the 3D shape of their model, while they struggled to read curvature, judge distance, and estimate size and shape based on distance visually using the graphical user interface. They were also able to work faster, knowing automatically, subconsciously how to act, how to feel and shape things with their hands, while they

Table XXVIII. Interviews

Technology	Select comments
Digital	Rhino has a long learning curve.
	Understanding how Rhino works – i.e. the underlying mathematical representation, NURBS – is very important. Once you understand that is a tension field then you understand how to shape it, how to make it do what you want.
Analog	I worked additively, then subtractively, smoothing. The sculpting tool gave sharpness – the sharp edge let me smooth in a way my fingers couldn't. It felt like drawing or laying concrete. Feeling is important – I could feel subtle changes in topography.
	My sense of touch helped me to understand the topography. I could sculpt like reading braille. I could feel the shape with my fingers. It was intuitive and calming.
	I have done lots of sculpture so I knew how to feel the shape of the model. And the desk lamp cast shadows so I could visually perceive depth.
Augmented	The contours were just a guide. My general strategy was additive. I felt with my hands to try to match the contours. If I saw concavity in the contours then I felt the sand and sculpted that concavity. Finding the relative height, however, was challenging – it was subtle. Most people don't understand contours. They have to be taught.
Tangible Landscape	We all already understand how sand works. We understand sand, but not necessarily these analytics – the difference analytic or the water flow simulation.
	The water flow analytic reminds of me of playing in creeks and grading streams as a kid.
	The difference analytic was the best. I tried to make it match. I was constantly rebuilding to make it match. The water flow analytic was useful for thinking about form, about what form does – why water flows where it does. Seeing the flow takes away the mystery of topography. Our students tend to have a linear design process. Because Tangible Landscape gives immediate results it encourages an iterative process.
	Tangible Landscape let me tinker. I could rapidly create, making new iterations. I could try something, see and feel it – directly experience it – and try again. Reinvent it. Tinkering like this is a learning process. Learning through doing. Tangible Landscape lowers the stakes so that you're not too invested. You're ready to fail. So you can intuitively explore, while reflecting on what you've done, what you're doing.

struggled to find buttons, remember keyboard shortcuts, and understand the graphical user interface.

Intuitive interaction mattered less for the 3D modeling experts because they had already acquired the knowledge and skills needed. They still performed better with projection-augmented modeling because they were able to work faster and refine their models earlier. Based on observations during the study we conclude that even without real-time analytics the tangible interface is faster for 3D modeling experts because it affords an almost instantaneous transition between tools – between fingertips, palms, the arches of hands, the blades of hands, the blade of a knife, and the tip of a knife. It is also faster because it affords more feedback by combining kinaesthetic sensing of physical form with computer graphics so users do not have to spend as much time trying to understand the space.

While previous studies of tangible interaction have shown that tangible interfaces can be faster and more reliable than graphical interfaces for simple spatial tasks [Huang 2004; Couture et al. 2008], this study shows that tangible interfaces can be

more intuitive, faster, and more reliable for complex spatial tasks. While research on tangibles has focused on novices and intuition, we have found that tangible interfaces can be useful for experts as well as novices if they have the capabilities, tools, and precision that experts need. A well designed tangible interface for spatial modeling should be more effective than a digital modeling program for short exploratory modeling tasks because it affords faster modeling. With faster modeling experts can explore more ideas, rapidly iterating through many designs. This ability to work quickly and rapidly iterate is the key to ideation.

We hypothesize, however, that digital modeling becomes increasingly efficient as tasks become longer. The longer the modeling task the more the basic affordances of digital modeling – precision, zooming, layering, snapping, copying and pasting, etc. – should help make the process more and more efficient. Affordances like zooming that are challenging to develop for augmented sandbox interfaces have, however, already been implemented in shape changing interfaces like Tangible CityScape [Tang et al. 2013]. Based on our own experiences we further hypothesize that hybrid approaches combining tangible modeling and digital fabrication should be even more efficient and effective for longer, more complex problems.

6.2. How tangible geospatial analytics mediate users' 3D spatial performance

The results of the difference experiment show that the difference analytic enabled an iterative modeling process successfully combining the affordances of physical modeling with near real-time computational feedback. Participants tended to perform well without training; they managed to quickly learn and understand the analytic and successfully used it to adaptively sculpt accurate models. This suggests that they were able to offload the cognitive work of manipulation onto their bodies, while cognitively parsing a rapidly changing, graphical representation of the difference in volume.

We observed, however, that some participants had trouble understanding the color table for the difference analytic. We used a relative color table that assigned colors based on percentage of value. The color table rescaled every scan so participants had a hard time quantifying their overall performance. This caused participants to make exaggerated changes in places where only minor adjustments were needed. After the experiment we tested an absolute color table with colors assigned to specific values. The absolute color table helped – it was much easier to observe gradual, incremental progress throughout the modeling process.

The results of the water flow experiment show that the water flow simulation helped participants understand the relationship between form and process. We observed participants using an iterative process to adaptively sculpt based on the simulated water flow. Through this trial and error process participants learned how topographic form controls the flow of water. Given that they accurately represented the streams without training they must have been able to offload some of the cognitive work of sensing and manipulating 3D form onto their bodies so that they could focus on the water flow.

Tangible Landscape enabled users to model in an iterative process informed by spatial analytics. We theorize that users offloaded the cognitive work of sensing and manipulating 3D form onto their bodies, successfully parsed the projected graphics, and metacognitively adapted their modeling strategy. Given the accuracy of their models participants did not suffer from too great a cognitive overload; they were parse the analytic feedback, critique their performance, and re-strategize.

With both analytics the 3D modeling experts performed better than all of the others including other experienced academics and professionals. First of all this shows that tangible interfaces for geospatial modeling with real-time analytics can be effective tools for experts. It also suggests that advanced spatial abilities and skills developed with digital tools can be transferred to tangible interfaces. If these abilities are trans-

Fig. 19. Attendees playing games with Tangible Landscape at an interactive seminar hosted by NCSU's Library exploring how serious, analytical games can be used to explore real world problems like coastal flooding and the spread of invasive species. See Fig. 11 - 12 for examples of the serious games played at this event.

ferable, then tangible interfaces – given their intuitive, embodied nature – may be a fast and effective way to train them. Future research is needed to explore spatial learning and transfer using tangible interfaces.

6.3. Reflections on the design process

Since we began designing and developing Tangible Landscape we have demonstrated and used various iterations of the system live during lab tours, in classes such as GIS for Designers and the Landscape Dynamics Seminar in NCSU's Department of Landscape Architecture, at conferences such as FOSS4G NA 2016 [Petrasova et al. 2016a; Harmon et al. 2016b], US-IALE 2016, and ACM SIGSPATIAL 2016 [Tabrizian et al. 2016], in public exhibitions at the City of Raleigh Museum and the North Carolina Museum of Natural Sciences, and at special events such as NCSU Library's Coffee & Viz seminar on serious gaming [Harmon et al. 2016d] (Fig. 19). At these events we observed how diverse groups of users interacted with the system as we continued to develop it. During tours we observed that many users had trouble understanding how Tangible Landscape worked until they actually tried it themselves. Once users saw their own hands scanned and rendered as part of the topography with simulated water flowing between their fingers or ponding in the palms of their hands they intuitively understood how to interact with the system (Fig. 4). Early in the design process we considered arms and hands captured by the 3D sensor to be confusing, disruptive artifacts that obfuscated the real data interrupting the desired interaction. We tried to deal with these artifacts by ignoring scans with large changes (i.e. an arm reaching across the model), masking cells that were too high (i.e. a hand hovering above the model), and speeding up the system to minimize lag so that a digitized hand would not linger as topography for too long. However, after observing users' first encounters with Tangible Landscape during tours, exhibits, and other events we found that users more quickly grasped how the system worked by trying it and seeing their hands digitized as part of the topography directing the flow of water. While sculpting a depression and then seeing it fill with simulated water is an intuitive interaction with a clear cause and effect, users appeared to learn how the system worked even faster by instead cupping their hands to create simulated ponds in the hollow of their palms. This interaction is even more direct; we hypothesize that its immediacy and embodied nature

helps users make the conceptual leap that the physical and digital are coupled. For users to effectively see their hands digitized as topography there needs to be a little lag – their digitized hands should trail behind their real hands by a half second or so.

We also observed that many users did not know what they should do even though they understood how to use the tangible interface. Some users hesitated to try the system, made a few indecisive moves, and then gave up, while other users happily played in the sand or with objects like markers or buildings blocks. While free, unstructured play suits some users, others would benefit from a more structured experience. Free, unstructured play is a useful way to introduce a tangible interface like Tangible Landscape, but does not engage some users for long. While Woods et al. argue that 'effective classroom use of the AR sandbox required developing student modeling exercises that took advantage of real-time feedback, virtual water, and physical modeling activities' [Woods et al. 2016], we found that some Tangible Landscape users needed an even more structured experience to learn effectively. Even during classes and workshops in which we had assigned tasks like designing a network of hiking trails many students and attendees still had trouble deciding what to do. An objective was not enough; they needed more feedback and a way to judge their performance – metrics or even a score. When we demonstrated Tangible Landscape for students or the general public we found that they needed a challenge to hold their attention and that simple rules and a score were enough to structure the experience. We designed two serious games about coastal flooding and termite invasion as prototypes and tested them in an interactive workshop as part of NCSU Library's Coffee & Viz series [Harmon et al. 2016d]. Quick rounds with scores encouraged participants to keep on playing, to try new things in hope of getting a better score. While the structured experience of these serious games worked well with the general public, experts wanted more depth, feedback, and customizability – i.e. a structured, yet open experience that would be engaging and encourage in-depth exploration. Experts need to be able to use many different modes of tangible interaction so that they can make use of their extensive, well-developed skills. Experts may also want to compare metrics rather than see a simplified score so that they can make tradeoffs between alternatives. When for example we designed applications for geographers and ecologists for managing termite invasions and controlling the spread of sudden oak death they asked for dashboards with statistics and data visualizations in addition to the projected maps so that they could study how different scenarios would affect groups of impacted stakeholders who valued different things.

The experiments helped us to refine the design of Tangible Landscape. We learned how users performed with different modes of analytic feedback, gained insight about their experiences and needs, thought critically about colors, and reaffirmed the need for speed – the importance of fast (but not too fast) feedback. This informed the development of the next generation of Tangible Landscape and led to the redesign of the difference analytic. After the experiments we developed the 3rd generation Tangible Landscape system using the OpenKinect project for drivers, the Point Cloud Library for point cloud processing, and OpenCV for computer vision. This dramatically increased the speed of the system approximately doubling the rate of feedback. Now that the analytics are faster (see Table VII) users should perform even better than they did in the experiments. As a test a member of our research team performed the task for the difference experiment using the new generation of Tangible Landscape and produced a very accurate model (see Appendix E.7).

6.4. Design guidelines

Based on our experience designing Tangible Landscape, the results of the experiments, and our observations of users during tours, workshops, classes, and other events we propose the following guidelines for designing tangible interfaces for spatial modeling:

— Modes of interaction should be analogous to ubiquitous tasks
— Design intuitive interactions with simple feedback for novices
— Design a wide variety of interactions and tools with rich feedback for experts
— Design free, playful experiences to encourage creative exploration
— Design structured experiences with objectives, rules, and feedback to encourage continued engagement
— Build on existing tools and technologies

We recommend designing tangible interactions that are analogous to ubiquitous, everyday tasks so that users will already subconsciously know what to do. When designing for novices we recommend making interactions as natural and intuitive as possible, while offering relatively simple feedback like a water flow simulation. Free, unstructured experiences will encourage creative exploration, but many users will be more engaged by more structured experiences with clear, well defined objectives, rules, feedback, and even scores. Experts, however, need a wide array of tools available so that they can intuitively use their diverse, existing skills and seamless switch between them. They need richer, more complex feedback so they can make critical, scientifically informed decisions. To efficiently develop the wide variety of tools and interactions that experts need we recommend using existing libraries for scientific computing. Using well documented, peer-reviewed, open source scientific models makes it easier to develop new tangible geospatial interactions and know if they behave properly. Building on the infrastructure of existing open source projects – on their codebase, documentation, binaries, support, and community – helps one efficiently collaborate and involve expert users in the design and development process so that they can contribute their own scientific models to the project [Petrasova et al. 2016b].

6.5. Open science

As a work of open science we invite readers to replicate or build upon this experiment by using or adapting our tangible interface, experimental methodology, code, and data. The python scripts for data processing and analysis are in Appendix F and also on GitHub at https://github.com/baharmon/tangible_topography released under the GNU General Public License (GPL). See Appendix B and the documentation in the GitHub repository for detailed instructions for replicating this experiment. The data used in this experiment and the results – including data, renderings, photographs, and interview notes – are in Appendix G and also on the Open Science Framework at https://osf.io/82gst/ under the Creative Commons Zero license. While the rest of the software used in this research is free and open source, Rhinoceros is proprietary software. Open source 3D modeling programs like Blender, however, could be substituted for Rhinoceros.

To build your own Tangible Landscape refer to Appendix A, visit the project website at http://tangible-landscape.github.io/, see the documentation in the GitHub repository including the Wiki at https://github.com/tangible-landscape/grass-tangible-landscape/wiki, and refer to the book Tangible Modeling with Open Source GIS [Petrasova et al. 2015]. GRASS GIS is available at https://grass.osgeo.org/ under the GPL. Tangible Landscape's components are available at https://github.com/tangible-landscape under the GPL.

7. FUTURE WORK

Our user experiments showed that Tangible Landscape is an effective tangible interface for geospatial modeling. Based on the user experiments and our own reflections and experiences with Tangible Landscape we have identified ways to improve the system. Our users want more tools that are natural to use so that they can do whatever

they imagine with their existing skills. They want a richer, more immersive experience. They want larger models and easier, faster, cheaper ways to build models. We plan to refine existing modes of interaction with new methods for handling, processing, and filtering point clouds. By fusing color and depth data we will develop new modes of interaction including planting individual trees of a given species with a model tree and building structures with building blocks. We plan to support new VR headsets and integrate procedural plant generation, 3D tree libraries, and 3D building libraries for more immersive renderings. We plan to support arrays of multiple 3D sensors for scanning larger models. We plan to design better ways to build physical models using in-situ robotic fabrication and plan to develop GRASS GIS modules for 3D printing and CNC routing. We also plan to improve the system setup so that it is modular, lighter, more portable, more adaptable, less expensive, and more aesthetic. Through the user experiments and our own experiences we have also identified new research questions. How do tangible interfaces for geospatial modeling mediate learning? How can Tangible Landscape be used effectively in classrooms? Our design process is also highly conceptual and vision-driven. Inspired by advances in pervasive sensing, robotics, and automated construction we have developed a new vision for Tangible Landscape as a way to couple physical models and digital models of landscapes with real landscapes.

7.1. Future cognitive studies

This study focused on 3D spatial performance using geospatial analyses to assess modeling results. It did not quantitively address cognitive, affective, motivational, or metacognitive processes. Future research in embodied spatial cognition with tangible interfaces should study these processes using quantitive methods like eye tracking and biometric sensing and qualitative methods like protocol analysis. Future research should also study spatiotemporal performance using time series analysis to assess participants' modeling processes. Suitable methods for spatiotemporal analysis and visualization include space-time cubes, 3D raster statistics and map algebra, vector flow fields, contour evolution, and animation.

7.2. Robotic fabrication and autonomous construction

Emerging technologies are enabling ever closer coupling of the physical and the digital bridging real and virtual geographies. With advances in sensor networking physical things and environmental processes can be digitally tracked, sensed, and analyzed – giving the physical a digital presence [Ratti et al. 2009; Resch et al. 2012]. With advances in sensing technologies 3D forms can be digitized at scales ranging from millimeters to the entire planet. Data can be digitally fabricated with technologies such as 3D printing and robotics. Landscapes and the built environment can be digitally fabricated with technologies such as machine control for construction, earthmoving, and precision agriculture. Tangible interfaces for GIS have the potential to tie these emerging technologies together seamlessly linking physical models, digital models, and real geographies. In the future we intend to extend Tangible Landscape to seamlessly, bidirectionally link a physical model, a digital model, and a real landscape. We plan to integrate real-time streaming data, robotic fabrication, and automated construction to link real and virtual geographies via a tangible interface. We would stream real-time topographic data from sensors in the field – such as unmanned aerial vehicles – to a GIS. A robotic arm would then rapidly update the physical model based on changes to the digital elevation model. Users could then explore potential changes by sculpting the physical model and interacting with the real-time geospatial analyses and simulations. Finally, they could send a computationally optimized design to an autonomous construction vehicle or robot in the field for construction (Fig. 20).

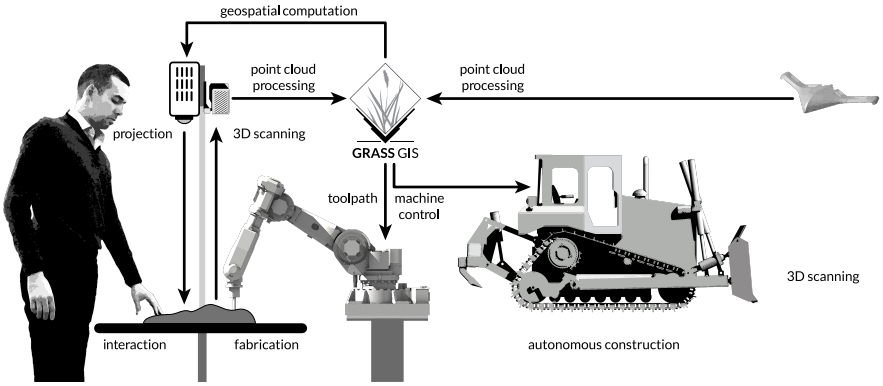


Fig. 20. Tangible Landscape with robotic fabrication, robotic construction, and streaming field data.

8. CONCLUSION

Through a series of experiments we found that tangible interfaces for spatial modeling can enhance 3D spatial performance in terms of speed, accuracy, and process. By comparing digital, analog, and projection-augmented modeling we found that coupling digital and physical models as a tangible interface can combine the affordances of digital and analog tools – enabling an embodied modeling process enriched with digital data – so that users can model intuitively, quickly, and precisely. Even 3D modeling experts performed better with the tangible interface – building more accurate models that better represented the morphology of the landscape – because they could work faster creating and refining details sooner. They were able to transfer and effectively use the spatial skills and abilities they had developed through digital modeling with the tangible interface. We also found that tangible interaction with real-time geospatial analytics can encourage iterative modeling processes. With the real-time difference analytic and water flow simulation users worked in rapid cycles of sculpting and digitally informed critical analysis to build accurate models that correctly represented the topographic and hydrologic morphology. Through this embodied process of reflection-in-action users were able to observe spatial patterns, forms, and processes, generate and test hypotheses, and draw inferences. The experiments showed that users were able to offload enough of the cognitive work of sensing and manipulating space onto their bodies that they could understand the computational analytics and adaptively re-strategize. Further experiments are needed to explore the role of spatial cognition, affect, motivation, and metacognition in tangible modeling.

The experiments show that Tangible Landscape, a tangible interface for GIS, works as theorized and designed – coupling a physical and digital model of a landscape enables users to cognitively grasp topography, intuitively shape and interact with multidimensional space, and offload enough cognitive work to understand real-time geospatial analytics. With Tangible Landscape users can intuitively interact with spatial data and scientific models using their bodies. While novices should be able to effectively learn about multidimensional space and rapidly improve their spatial abilities with Tangible Landscape, experts can effectively use it to rapidly develop, prototype, and test hypotheses about space and spatiotemporal processes.

A. HOW TO BUILD TANGIBLE LANDSCAPE

Table XXIX. Hardware

Type	Product	Cost
Computer	System 76 Oryx Pro	$1500
Projector	Epson PowerLite 1761W	$600
3D sensor	Xbox One Kinect	$100
	Kinect Adapter for Windows	$50
Stand	Avenger 40-Inch C-Stand with Grip Kit	$200
	Kupo 4 Way Clamp	$50
	Kupo Baby 5/8-Inch Receiver for 4 Way Clamp	$12
	Avenger D520L 40-Inch Extension Arm	$45
	Avenger F800 3-Inch Baby Wall Plate	$10
	Avenger F810 3-Inch Baby Wall Plate with Swivel Pin	$40
Peripherals	HDMI cable	$10
	Extension cord	$10
Modeling media	Waba Fun Kinetic Sand 11 Lbs	$50

Hardware & software. To build Tangible Landscape you will need a computer, a projector, a 3D sensor, an armature to hold the projector and sensor, and a physical model of a landscape. The computer should be running Linux or Mac OS X and have GRASS GIS,[1] the grass-tangible-landscape plugin,[2] the r.in.kinect add-on,[3] and their dependencies compiled.

Physical models. To sculpt with Tangible Landscape you need a malleable model made of a soft, deformable medium like polymer enriched sand. These models can be hand sculpted or precisely cast with digitally fabricated molds. When you are using Tangible Landscape for object detection you can use hard, rigid models such as 3D prints.

Setup. Place your physical model of a landscape on a table. Mount the Kinect sensor on baby wall plate attached to a C-stand. Adjust the height of the C-stand so that the Kinect is 50 - 100 cm above the model. Check that the Kinect is level using an instrument like a spirit level. Mount the projector on another baby wall plate attached to the other C-stand. Connect the Kinect and projector to your computer. See https://github.com/tangible-landscape/grass-tangible-landscape/wiki for more detailed instructions.

Calibration. Remove the model, clear the table, and then run Tangible Landscape's automatic calibration function to account for the relative rotation of the scanner and the table.

[1] https://grass.osgeo.org/
[2] https://github.com/tangible-landscape/grass-tangible-landscape
[3] https://github.com/tangible-landscape/r.in.kinect

Fig. 21. Download stickers from https://github.com/tangible-landscape/tangible-landscape-media

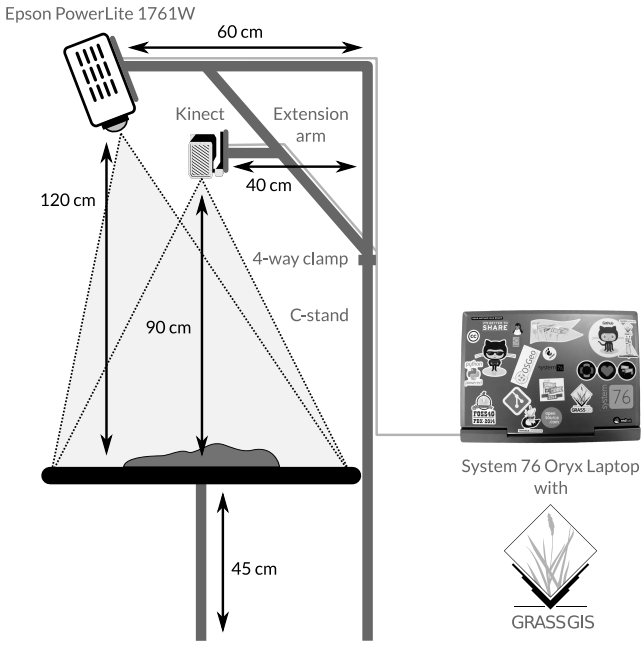


Fig. 22. System setup

B. EXPERIMENTAL PROCEDURE

B.1. Data

Acquire elevation data for your study landscape. We used a 2013 airborne lidar survey of Wake County, North Carolina. Other sources for lidar point clouds and raster elevation data include the National Elevation Dataset,[4] the United States Interagency Elevation Inventory,[5] Open Topography,[6] and the European Environment Agency's Digital Elevation Model over Europe.[7] Import lidar data in GRASS GIS as vector points using the module v.in.lidar[8] and then interpolate with the module v.surf.rst.[9] Find regions of the same size with clearly defined morphological features for each experiment.

When preparing data and running analyses use the following naming conventions. Digital modeling, analog modeling, augmented modeling, tangible modeling with the difference analytic, and tangible modeling with the water flow analytic will be experiments 1-5. For each experiment save the corresponding reference digital elevation model in the *PERMANENT* mapset with the number of the experiment (i.e. dem_1, dem_2, dem_3, dem_4, and dem_5). Save the results of each experiment in the *data* mapset with the participant's name, the type of data, and the experiment number (i.e. name_dem_number).

B.2. Experiments

Introduce the research and explain the experiment to each participants. If they agree to take part in the experiment have them sign an informed consent form. Shuffle the order of experiments for each participant.

Digital modeling. In the 1st experiment participants will sit at a computer and use Rhinoceros, a 3D modeling program, to digitally model the study landscape. They will have 10 minutes of training before they begin. Start by explaining the interface, showing them the basic controls – selecting, panning, zooming, orbiting, and transforming with the gumball – and then letting them try. Then explain how this experiment will work, show them how to vertically translate control points on a NURBS surface using the gumball, and let them try. Begin the experiment. They will have 10 minutes to model the study landscape by vertically translating control points on a NURBS surface using the gumball. They will begin with a flat surface subdivided into a 10 x 10 grid of control points. They can use the *rebuild* command to create a higher density of control points. They will raise and lower these control points with the gumball to change the surface. They can used a locked layer with 3D contours as a reference. Snapping should be disabled. When 10 minutes have elapsed save their model as a *.3dm* file and then export it as a GeoTIFF (*.tif)*. Import this *.tif* raster into GRASS GIS using r.in.gdal,[10] randomly resample it using r.random,[11] and then interpolate it using v.surf.rst to create a digital elevation model. Copy this to the *data* mapset as *name_dem_1*.

Analog modeling. In the 2nd experiment each participant will have 10 minutes to build a polymer-enriched sand model of the study landscape using a CNC-routed model as a reference. They will be given a 3D scale – a wooden block ruled with map units –

[4]https://viewer.nationalmap.gov
[5]https://coast.noaa.gov/inventory/
[6]http://www.opentopography.org/
[7]http://www.eea.europa.eu/data-and-maps/data/eu-dem
[8]https://grass.osgeo.org/grass72/manuals/v.in.lidar.html
[9]https://grass.osgeo.org/grass72/manuals/v.surf.rst.html
[10]https://grass.osgeo.org/grass72/manuals/r.in.gdal.html
[11]https://grass.osgeo.org/grass72/manuals/r.random.html

as an additional guide. They can use a wooden modeling tool or their hands to sculpt the model. A desk lamp will illuminate both models from an angle casting shadows like a hillshade. When 10 minutes have elapsed save their model by scanning it with Tangible Landscape. Copy the scanned elevation to the *data* mapset as *name_dem_2*.

Projection-augmented modeling. In the 3rd experiment each participant will have 10 minutes to build a projection-augmented polymer-enriched sand model of the study landscape. The digital elevation model and contours of the study landscape will be projected on the sand model as a reference. They should also be given a CNC-routed model and a 3D scale for additional guidance. When 10 minutes have elapsed save their model by scanning it with Tangible Landscape. Copy the scanned elevation to the *data* mapset as *name_dem_3*.

Tangible modeling with the difference analytic. In the 4th experiment each participant will have 10 minutes to build a polymer-enriched sand model of the study landscape using Tangible Landscape with the difference analytic. This experiment will use a different region of the study landscape. Use Tangible Landscape's *run_difference* function in the *current_analyses.py* script to show each participant where to add and remove sand in order to match the study landscape The difference between their model and the study landscape will be projected onto the sand model in near real-time to guide them. The contours of the study landscape should layered over the difference map. They should also be given a CNC-routed model and a 3D scale for additional guidance. When 10 minutes have elapsed save their model by scanning it with Tangible Landscape. Copy the scanned elevation to the *data* mapset as *name_dem_4*.

Tangible modeling with the water flow analytic. In the 5th experiment each participant will have 10 minutes to build a polymer-enriched sand model of the study landscape using Tangible Landscape with the water flow analytic. This experiment will use a different region of the study landscape. Each participant can choose to see either simulated water flow for the study landscape or simulated water flow for their model. When a participant asks switch between these layers in GRASS's layer manager. Use Tangible Landscape's *run_simwe* function in the *current_analyses.py* script to simulate water flow across their model in near-real time. The contours of the study landscape should layered over the precomputed reference water flow, while the contours of the scanned model should be layered over the water flow for the scanned model. They should also be given a CNC-routed model and a 3D scale for additional guidance. When 10 minutes have elapsed save their model by scanning it with Tangible Landscape. Copy the scanned elevation to the *data* mapset as *name_dem_5*.

Documentation. Record each experiment with photographs or video. Observe each participant as they work and take notes about their modeling processing.

Interviews. Interview select participants after they finish the experiments. See Appendix C for semi-structured interview guidelines.

B.3. Geospatial analysis

Copy the scanned maps from the *data* mapset to a new *analysis* mapset for processing. Open a session of GRASS GIS in the appropriate location (i.e. *tangible_topography_mapset*) with the *analysis* mapset. Create new directories *results* and *render_3d* inside the location directory. Run the script *analysis.py* in this GRASS session to analyze the data. This will generate new maps with analyses and simulations in the *analysis* mapset and will render the maps as *.png* images in the *results* directory. Then run the script *render_3d_images.py* to generate 3D renderings of the maps in the *render_3d* directory.

C. SEMI-STRUCTURED INTERVIEW GUIDELINES

C.1. Aim

Understand how tangible interfaces for geospatial modeling change how users model.

C.2. Interview goals

— Map participants' analog, hand modeling processes
— Map participants' digital modeling processes
— Map participants' augmented modeling processes
— Map participants' tangible modeling processes with the difference analytic
— Map participants' tangible modeling processes with the water flow analytic

C.3. *Topic:* Modeling process

— Please describe your modeling process with each technology
— Did you work additively or subtractively? A mix?
— Did you work in a linear or iterative, exploratory process?
— How did this technology aid you? What did it let you to do?
— Did this technology constrain you in any way?

C.4. *Topic:* Intuition

— How intuitive was it?
— Could you model what you intended?
— Did you have to think about how to modeling? Or could you just act?

C.5. *Topic:* Metacognition

— We asked you to sculpt a model of the study landscape. Please describe your thought process while sculpting.
— Did you strategize about how to model? If so what was your modeling strategy?
— Did your modeling strategy evolve as you worked?

C.6. *Topic:* perception and experience

— How did it feel to sculpt a 3D model with this technology?
— Was it stressful? Was it fun?
— Did the technology change how you perceived distance, depth, form, or volume?

D. VIDEOS: APPLICATIONS

https://www.youtube.com/playlist?list=PLiNYVMuYqTHXzgHK0Y_yQ7v9EsnnMaMwg

A collection of videos demonstrating applications for Tangible Landscape.

D.1. Landscape design with Tangible Landscape and Virtual Reality

https://youtu.be/pYbpEMjME1Y

We coupled Tangible Landscape with an immersive virtual environment so that users can virtually walk around the modeled landscape and visualize it at a human-scale. As users shape topography, draw trees, define viewpoints, or route a walkthrough, they can see the results on the projection-augmented model, rendered on a display, or rendered on a head-mounted display.

D.2. Landscape planning with Tangible Landscape

https://youtu.be/Cd3cCQTGer4

As users sculpt a physical model – building dams, forming depressions, and carving stream channels in polymeric sand – they see simulated water flow and ponding update in real-time. Computed using the modules r.sim.water and r.fill.dir in GRASS GIS.

D.3. Subsurface visualization with Tangible Landscape

https://youtu.be/__fWppH9aqQ

Users visualize subsurface properties such as the percentage of soil moisture by digging to excavate soil or placing markers to take soil core samples.

D.4. Viewshed analysis with Tangible Landscape

https://youtu.be/tGFNHFoHMYM

Users place markers on a physical model to digitize viewpoints and compute viewsheds in real-time. Viewshed are computed using the module r.viewshed in GRASS GIS.

D.5. Wildfire spread modeling with Tangible Landscape

https://youtu.be/EJc57GFJeZI

Users interact with a fire spread simulation by removing fuel from a physical model of a landscape to create firebreaks. Fire spread is computed using the modules r.ros and r.spread in GRASS GIS.

D.6. Urban growth modeling with Tangible Landscape

https://youtu.be/SeBoGwSqmRM

Users collaboratively interact with the FUTURES urban growth simulation by delineating protected areas (with green sand) and areas where they want future development to occur (with red sand). Participants can also draw new roads, which will attract more development.

E. VIDEOS: USER EXPERIMENTS

https://www.youtube.com/playlist?list=PLiNYVMuYqTHVCISj3CS9Og2Q5GZZ8o065

A collection of videos demonstrating the training and user experiments in this study.

E.1. Training for digital 3D modeling

https://youtu.be/dSyrHAuu698

E.2. Digital 3D modeling with Rhinoceros

https://youtu.be/vA1xwMSaGV4

E.3. Analog 3D modeling

https://youtu.be/STYHUHNaWdY

E.4. Projection-augmented 3D modeling with Tangible Landscape

https://youtu.be/1uEvzMJWh_E

E.5. 3D modeling with Tangible Landscape's difference analytic

https://youtu.be/VXmW9-yASVU

E.6. 3D modeling with Tangible Landscape's water flow analytic

https://youtu.be/61hsXgb3MLY

E.7. Good performance with Tangible Landscape's difference analytic

https://youtu.be/Q3elMIRCYSk

F. SCRIPTS

F.1. analysis.py

https://github.com/baharmon/tangible_topography

The python scripts for data processing, analysis, and rendering used in this experiment are available on GitHub under the GNU General Public License (GPL).

G. DATA

G.1. tangible_topography_mapset

https://osf.io/82gst/

The data used in this experiment and the results including analyses and renderings are available on the Open Science Framework under the Creative Commons Zero license. The data is stored in a GRASS GIS location with a collection of mapsets with GIS data and directories with renderings.

REFERENCES

e-on software. 2012. Vue. (2012). Retrieved August 6, 2016 from http://www.e-onsoftware.com/products/vue/

Charles R Amburn, Nathan L Vey, Michael W Boyce, and Jerry R Mize. 2015. *The Augmented REality Sandtable (ARES)*. Technical Report October. US Army Research Laboratory. DOI:http://dx.doi.org/10.13140/RG.2.1.2685.0006

Cyrus S. Bamji, Patrick O'Connor, Tamer Elkhatib, Swati Mehta, Barry Thompson, Lawrence A. Prather, Dane Snow, Onur Can Akkaya, Andy Daniel, Andrew D. Payne, Travis Perry, Mike Fenton, and Vei Han Chan. 2015. A 0.13 μm CMOS System-on-Chip for a 512 x 424 Time-of-Flight Image Sensor with Multi-Frequency Photo-Demodulation up to 130 MHz and 2 GS/s ADC. *IEEE Journal of Solid-State Circuits* 50, 1 (2015), 303–319. DOI:http://dx.doi.org/10.1109/JSSC.2014.2364270

Robert S. Bednarz and Jongwon Lee. 2011. The components of spatial thinking: Empirical evidence. *Procedia - Social and Behavioral Sciences* 21 (2011), 103–107. DOI:http://dx.doi.org/10.1016/j.sbspro.2011.07.048

Matthew Blackshaw, Anthony DeVincenzi, David Lakatos, Daniel Leithinger, and Hiroshi Ishii. 2011. Recompose: Direct and Gestural Interaction with an Actuated Surface. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11)*. ACM, New York, NY, USA, 1237–1242. DOI:http://dx.doi.org/10.1145/1979742.1979754

Blender Online Community. 2016. Blender – a 3D modelling and rendering package. (2016). Retrieved July 25, 2016 from http://www.blender.org

Cynthia A. Brewer. 1994. Color Use Guidelines for Mapping and Visualization. In *Visualization in Modern Cartography*, Alan M. MacEachren and D.R. Fraser Taylor (Eds.). Pergamon, New York, 123–147.

Cynthia A. Brewer and Mark Harrower. 2016. ColorBrewer 2.0. (2016). Retrieved December 17, 2016 from http://colorbrewer2.org/

Bradley Cantrell and Justine Holzman. 2016. Rapid Landscape Prototyping Machine. In *Responsive Landscapes*. Routledge, New York, Chapter 6.6, 159–166.

Nadine Couture, Guillaume Rivière, and Patrick Reuter. 2008. GeoTUI: A Tangible User Interface for Geoscience. In *Proceedings of the 2Nd International Conference on Tangible and Embedded Interaction (TEI '08)*. ACM, New York, NY, USA, 89–96. DOI:http://dx.doi.org/10.1145/1347390.1347411

Paul Dourish. 2001. *Where the action is: the foundations of embodied interaction*. MIT Press, Cambridge, MA.

Sean Follmer, Daniel Leithinger, Alex Olwal, Akimitsu Hogge, and Hiroshi Ishii. 2013. inFORM: Dynamic Physical Affordances and Constraints Through Shape and Object Actuation. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 417–426. DOI:http://dx.doi.org/10.1145/2501988.2502032

GRASS Development Team. 2016a. *Geographic Resources Analysis Support System (GRASS GIS 7.2) Reference Manual: g.gui.animation*. Open Source Geospatial Foundation, USA. https://grass.osgeo.org/grass72/manuals/g.gui.animation.html

GRASS Development Team. 2016b. *Geographic Resources Analysis Support System (GRASS GIS 7.2) Reference Manual: r.geomorphon*. Open Source Geospatial Foundation, USA. https://grass.osgeo.org/grass70/manuals/addons/r.geomorphon.html

GRASS Development Team. 2016c. *Geographic Resources Analysis Support System (GRASS GIS 7.2) Reference Manual: r.param.scale*. Open Source Geospatial Foundation, USA. https://grass.osgeo.org/grass72/manuals/r.param.scale.html

GRASS Development Team. 2016d. *Geographic Resources Analysis Support System (GRASS GIS 7.2) Reference Manual: r.series*. Open Source Geospatial Foundation, USA. https://grass.osgeo.org/grass72/manuals/r.series.html

GRASS Development Team. 2016e. *Geographic Resources Analysis Support System (GRASS GIS 7.2) Reference Manual: r.sim.water*. Open Source Geospatial Foundation, USA. https://grass.osgeo.org/grass72/manuals/r.sim.water.html

GRASS Development Team. 2016f. *Geographic Resources Analysis Support System (GRASS GIS) Software*. Open Source Geospatial Foundation, USA. http://grass.osgeo.org

Benoit Hardy-Vallée and Nicolas Payette. 2008. *Beyond the Brain: Embodied, Situated and Distributed Cognition*. Cambridge Scholars Publishing, Newcastle, UK.

Brendan A Harmon. 2016. Embodied Spatial Thinking in Tangible Computing. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '16)*. ACM, New York, NY, USA, 693–696. DOI:http://dx.doi.org/10.1145/2839462.2854103

Brendan A Harmon, Anna Petrasova, Vaclav Petras, and Helena Mitasova. 2016a. Computational Landscape Architecture: Procedural, Tangible, and Open Landscapes. In *Innovations in Landscape Ar-*

chitecture, Jonathan R Anderson and Daniel Ortega (Eds.). Routledge. https://www.routledge.com/Innovations-in-Landscape-Architecture/Anderson-Ortega/p/book/9781138860681

Brendan A Harmon, Anna Petrasova, Vaclav Petras, Helena Mitasova, and Ross K Meentemeyer. 2016b. Tangible interaction for GIS. (2016). http://baharmon.github.io/foss4g-na-2016/ FOSS4G NA 2016.

Brendan A Harmon, Anna Petrasova, Vaclav Petras, Helena Mitasova, and Ross K Meentemeyer. 2016c. Tangible Landscape: cognitively grasping the flow of water. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLI-B2 (2016), 647–653. DOI:http://dx.doi.org/10.5194/isprs-archives-XLI-B2-647-2016

Brendan A Harmon, Anna Petrasova, Vaclav Petras, Helena Mitasova, Jelena Vukomanovic, and Ross K Meentemeyer. 2016d. Serious Gaming with Tangible Landscape. (2016). https://www.lib.ncsu.edu/event/coffee-viz-serious-gaming-tangible-landscape NCSU Coffee & Viz.

Chen-Je Huang. 2004. Not Just Intuitive: Examining the Basic Manipulation of Tangible User Interfaces. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04)*. ACM, New York, NY, USA, 1387–1390. DOI:http://dx.doi.org/10.1145/985921.986071

Hiroshi Ishii. 2008. Tangible Bits: Beyond Pixels. In *Proceedings of the 2Nd International Conference on Tangible and Embedded Interaction (TEI '08)*. ACM, New York, NY, USA, xv–xxv. DOI:http://dx.doi.org/10.1145/1347390.1347392

Hiroshi Ishii, Eran Ben-Joseph, John Underkoffler, Luke Yeung, Dan Chak, Zahra Kanji, and Ben Piper. 2002. Augmented Urban Planning Workbench: Overlaying Drawings, Physical Models and Digital Simulation. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality (ISMAR '02)*. IEEE Computer Society, Washington, DC, USA, 203–211. http://dl.acm.org/citation.cfm?id=850976.854980

Hiroshi Ishii, Dávid Lakatos, Leonardo Bonanni, and Jean-Baptiste Labrune. 2012. Radical Atoms: Beyond Tangible Bits, Toward Transformable Materials. *interactions* 19, 1 (Jan. 2012), 38–51. DOI:http://dx.doi.org/10.1145/2065327.2065337

Hiroshi Ishii, Carlo Ratti, Ben Piper, Yao Wang, Assaf Biderman, and Eran Ben-Joseph. 2004. Bringing Clay and Sand into Digital Design – Continuous Tangible User Interfaces. *BT Technology Journal* 22, 4 (Oct. 2004), 287–299. DOI:http://dx.doi.org/10.1023/B:BTTJ.0000047607.16164.16

Hiroo Iwata, Hiroaki Yano, Fumitaka Nakaizumi, and Ryo Kawamura. 2001. Project FEELEX: Adding Haptic Surface to Graphics. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 469–476. DOI:http://dx.doi.org/10.1145/383259.383314

Jarosław Jasiewicz and Tomasz F. Stepinski. 2013. Geomorphons a pattern recognition approach to classification and mapping of landforms. *Geomorphology* 182 (2013), 147–156. DOI:http://dx.doi.org/10.1016/j.geomorph.2012.11.005

Marc Jeannerod. 1997. *The Cognitive Neuroscience of Action*. Blackwell, Cambridge, MA. 236 pages.

Marcel A Just and Patricia A Carpenter. 1985. Cognitive coordinate systems: Accounts of mental rotation and individual differences in spatial ability. *Psychological Review* 92, 2 (1985), 137–172. DOI:http://dx.doi.org/10.1037/0033-295X.92.2.137

Takaharu Kanai, Yuya Kikukawa, Tatsuhiko Suzuki, Tetsuaki Baba, and Kumiko Kushiyama. 2011. PocoPoco: A Tangible Device That Allows Users to Play Dynamic Tactile Interaction. In *ACM SIGGRAPH 2011 Emerging Technologies (SIGGRAPH '11)*. ACM, New York, NY, USA, Article 12, 1 pages. DOI:http://dx.doi.org/10.1145/2048259.2048271

Yuya Kikukawa, Megumi Kato, Tetsuaki Baba, and Kumiko Kushiyama. 2013. Hakoniwa: A Sonification Art Installation Consists of Sand and Woodblocks. In *Proceedings of the 19th International Conference on Auditory Display (ICAD 2013)*. International Community for Auditory Display, Lodz, Poland, 283–286. http://www.eletel.p.lodz.pl/icad2013/paper/37

David Kirsh. 2013. Embodied Cognition and the Magical Future of Interaction Design. *ACM Trans. Comput.-Hum. Interact.* 20, 1, Article 3 (April 2013), 30 pages. DOI:http://dx.doi.org/10.1145/2442106.2442109

Oliver Kreylos. 2012. Augmented Reality Sandbox. (2012). http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/index.html

Oliver Kreylos. 2017. Augmented Reality Sandbox. (2017). Retrieved 01-20-2017 from https://arsandbox.ucdavis.edu/

Jongwon Lee and Robert S. Bednarz. 2009. Effect of GIS Learning on Spatial Thinking. *Journal of Geography in Higher Education* 33, 2 (2009), 183–198. DOI:http://dx.doi.org/10.1080/03098260802276714

Daniel Leithinger and Hiroshi Ishii. 2010. Relief: A Scalable Actuated Shape Display. In *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '10)*. ACM, New York, NY, USA, 221–222. DOI:http://dx.doi.org/10.1145/1709886.1709928

Daniel Leithinger, Adam Kumpf, and Hiroshi Ishii. 2009. Relief. (2009). http://tangible.media.mit.edu/project/relief/

Daniel Leithinger, David Lakatos, Anthony DeVincenzi, Matthew Blackshaw, and Hiroshi Ishii. 2011. Direct and Gestural Interaction with Relief: A 2.5D Shape Display. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 541–548. DOI:http://dx.doi.org/10.1145/2047196.2047268

Yesudeep Mangalapilly. 2011. Watchdog. (2011). Retrieved August 6, 2016 from https://github.com/gorakhargosh/watchdog

Robert McNeel and Associates. 2016. Rhinoceros. (2016). Retrieved August 6, 2016 from https://www.rhino3d.com/

Helena Mitasova, Lubos Mitas, and Russell S. Harmon. 2005. Simultaneous spline approximation and topographic analysis for lidar elevation data in open-source GIS. *IEEE Geoscience and Remote Sensing Letters* 2 (2005), 375–379. DOI:http://dx.doi.org/10.1109/LGRS.2005.848533

Helena Mitasova, Lubos Mitas, Carlo Ratti, Hiroshi Ishii, Jason Alonso, and Russell S. Harmon. 2006. Real-time landscape model interaction using a tangible geospatial modeling environment. *IEEE Computer Graphics and Applications* 26, 4 (2006), 55–63. DOI:http://dx.doi.org/10.1109/MCG.2006.87

Helena Mitasova, Chris Thaxton, Jaroslav Hofierka, Richard McLaughlin, Amber Moore, and Lubos Mitas. 2004. Path sampling method for modeling overland water flow, sediment transport, and short term terrain evolution in Open Source GIS. *Developments in Water Science* 55 (2004), 1479–1490.

Mojang. 2009. Minecraft. (2009). Retrieved December 15, 2016 from https://minecraft.net/

Oculus. 2016. Illustrative filmmaking with Oculus Story Studio. Video. (2016). Retrieved July 24, 2016 from https://youtu.be/-rvwcGxEUGM

OpenCV Development Team. 2016. OpenCV. (2016). Retrieved August 6, 2016 from http://opencv.org/

OpenKinect Development Team. 2016. OpenKinect. (2016). Retrieved August 6, 2016 from https://github.com/OpenKinect

PCL Development Team. 2016. Point Cloud Library. (2016). Retrieved August 6, 2016 from http://pointclouds.org/

Anna Petrasova. 2016a. grass-tangible-landscape. (2016). Retrieved August 6, 2016 from https://github.com/tangible-landscape/grass-tangible-landscape

Anna Petrasova. 2016b. r.in.kinect. (2016). Retrieved August 6, 2016 from https://github.com/tangible-landscape/r.in.kinect

Anna Petrasova, Brendan Harmon, Vaclav Petras, and Helena Mitasova. 2015. *Tangible Modeling with Open Source GIS*. Springer.

Anna Petrasova, Brendan A Harmon, Vaclav Petras, and Helena Mitasova. 2014. GIS-based environmental modeling with tangible interaction and dynamic visualization. In *Proceedings of the 7th International Congress on Environmental Modelling and Software*, D.P. Ames and N. Quinn (Eds.). IEMSS. http://www.iemss.org/society/index.php/iemss-2014-proceedings

Anna Petrasova, Vaclav Petras, Brendan A Harmon, and Helena Mitasova. 2016a. Using GRASS GIS through Python and tangible interfaces. (2016). https://grasswiki.osgeo.org/wiki/Using_GRASS_GIS_through_Python_and_tangible_interfaces_(workshop_at_FOSS4G_NA_2016) FOSS4G NA 2016.

Anna Petrasova, Vaclav Petras, Derek Van Berkel, Brendan A Harmon, Helena Mitasova, and Ross K Meentemeyer. 2016b. Open source approach to urban growth simulation. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2016), 953–959. DOI:http://dx.doi.org/10.5194/isprs-archives-XLI-B7-953-2016

Ben Piper. 2002. *The Illuminated Design Environment : A 3-D Tangible Interface for Landscape Analysis*. Master's thesis. Massachusetts Institute of Technology.

Ben Piper, Carlo Ratti, and Hiroshi Ishii. 2002a. Illuminating clay: a 3-D tangible interface for landscape analysis. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '02*. ACM Press, Minneapolis, 355. DOI:http://dx.doi.org/10.1145/503376.503439

Ben Piper, Carlo Ratti, and Hiroshi Ishii. 2002b. Illuminating Clay: A Tangible Interface with potential GRASS applications. In *Proceedings of the Open Source GIS - GRASS users conference 2002*. Trento, Italy.

Ivan Poupyrev, Tatsushi Nashida, and Makoto Okabe. 2007. Actuation and Tangible User Interfaces: The Vaucanson Duck, Robots, and Shape Displays. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction (TEI '07)*. ACM, New York, NY, USA, 205–212. DOI:http://dx.doi.org/10.1145/1226969.1227012

Inigo Quilez. 2016. Quill: VR Drawing in the Production of Oculus Story Studio's New Movie. In *ACM SIGGRAPH 2016 Real-Time Live! (SIGGRAPH '16)*. ACM, New York, NY, USA, Article 41, 1 pages. DOI:http://dx.doi.org/10.1145/2933540.2933549

Majken K. Rasmussen, Esben W. Pedersen, Marianne G. Petersen, and Kasper Hornbæk. 2012. Shape-changing Interfaces: A Review of the Design Space and Open Research Questions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 735–744. DOI:http://dx.doi.org/10.1145/2207676.2207781

Carlo Ratti, Bernd Resch, Manfred Mittlboeck, Fabien Girardin, and Rex Britter. 2009. Live Geography Embedded Sensing for Standardised Urban Environmental Monitoring. *International Journal on Advances in Systems and Measurements* 2, 2-3 (2009), 156–167. http://hdl.handle.net/1721.1/55980

Carlo Ratti, Yao Wang, Hiroshi Ishii, Ben Piper, Dennis Frenchman, John P Wilson, A Stewart Fotheringham, and Gary J Hunter. 2004a. Tangible User Interfaces (TUIs): A Novel Paradigm for GIS. *Transactions in GIS* 8, 4 (2004), 407–421. DOI:http://dx.doi.org/10.1111/j.1467-9671.2004.00193.x

Carlo Ratti, Yao Wang, Ben Piper, Hiroshi Ishii, and Assaf Biderman. 2004b. PHOXEL-SPACE: An Interface for Exploring Volumetric Data with Physical Voxels. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '04)*. ACM, New York, NY, USA, 289–296. DOI:http://dx.doi.org/10.1145/1013115.1013156

Bernd Resch, Rex Britter, and Carlo Ratti. 2012. Live Urbanism - Towards Senseable Cities and Beyond. In *Sustainable Architectural Design: Impacts on Health*, Panos M Pardalos and Stamatina Rassia (Eds.). Vol. 56. Springer-Verlag New York, 175–184. DOI:http://dx.doi.org/10.1007/978-1-4419-0745-5

Alexander Robinson. 2014. Calibrating Agencies in a Territory of Instrumentality: Rapid Landscape Prototyping for the Owens Lake Dust Control Project. In *ACADIA 14: Projects of the 34th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*. Los Angeles, 143–146. http://papers.cumincad.org/cgi-bin/works/Show?acadia14projects

Joan Sol Roo, Renaud Gervais, and Martin Hachet. 2016. Inner Garden: An Augmented Sandbox Designed for Self-Reflection. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '16)*. ACM, New York, NY, USA, 570–576. DOI:http://dx.doi.org/10.1145/2839462.2856532

Radu Bogdan Rusu and Steve Cousins. 2011. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China.

Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. 2015. Kinect Range Sensing: Structured-Light versus Time-of-Flight Kinect. *Computer Vision and Image Understanding* 139 (2015), 1–20.

Tarah N. Schmidt-Daly, Jennifer M. Riley, Charles R. Amburn, Kelly S. Hale, and P. David Yacht. 2016. Video Game Play and Effect on Spatial Knowledge Tasks Using an Augmented Sand Table. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 60, 1 (2016), 1429–1433. DOI:http://dx.doi.org/10.1177/1541931213601328

Donald A. Schön. 1983. *The reflective practitioner : how professionals think in action*. Basic Books, New York.

Gerhard Schubert, Eva Artinger, Frank Petzold, and Gudrun Klinker. 2011a. Bridging the Gap: A (Collaborative) design platform for early design stages. In *Education and Research in Computer Aided Architectural Design in Europe*, Vol. 29. Ljubljana, 187–193.

Gerhard Schubert, Eva Artinger, Frank Petzold, and Gudrun Klinker. 2011b. Tangible tools for architectural design seamless integration into the architectural workflow. In *Proceedings of Association for Computer Aided Design in Architecture*. Banff, Canada, 1–12.

Gerhard Schubert, Eva Artinger, Violin Yanev, Frank Petzold, and Gudrun Klinker. 2012. 3D Virtuality Sketching: Interactive 3D-sketching based on real models in a virtual scene. *Proceedings of the 32nd Annual Conference of the Association for Computer Aided Design in Architecture* 32 (2012), 409–418.

Gerhard Schubert, David Schattel, Gudrun Klinker Marcus Tönnis, and Frank Petzold. 2015. Tangible Mixed Reality On-Site: Interactive Augmented Visualisations from Architectural Working Models in Urban Design. In *Computer-Aided Architectural Design Futures. The Next City - New Technologies and the Future of the Built Environment*. Vol. 527. Springer-Verlag Berlin Heidelberg, 55–74. DOI:http://dx.doi.org/10.1007/978-3-662-47386-3

Gerhard Schubert, Marcus Tönnis, Violin Yanev, Gudrun Klinker, and Frank Petzold. 2014. Dynamic 3d-sketching. *Proceedings of the 19th International Conference on Computer-Aided Architectural Design Research in Asia* 19 (2014), 107–116.

Dorothy J Shamonsky. 2003. *Tactile, spatial interfaces for computer-aided design: superimposing physical media and computation*. Ph.D. Dissertation. Massachusetts Institute of Technology. http://hdl.handle.net/1721.1/67172

Roger N Shepard and Jacqueline Metzler. 1971. Mental Rotation of Three-Dimensional Objects. *Science* 171, 3972 (1971), 701–703. http://www.jstor.org/stable/1731476

Jan Smisek, Michal Jancosek, and Tomas Pajdla. 2011. 3D with Kinect. In *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, Barcelona, 1154–1160. DOI:http://dx.doi.org/10.1109/ICCVW.2011.6130380

Payam Tabrizian. 2016. Tangible Landscape Immersive Extension. (2016). Retrieved August 6, 2016 from https://github.com/tangible-landscape/tangible-landscape-immersive-extension

Payam Tabrizian, Anna Petrasova, Brendan Harmon, Vaclav Petras, Helena Mitasova, and Ross Meentemeyer. 2016. Immersive Tangible Geospatial Modeling. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '16)*. ACM, New York, NY, USA, Article 88, 4 pages. DOI:http://dx.doi.org/10.1145/2996913.2996950

Sheng Kai Tang, Yusuke Sekikawa, Daniel Leithinger, Sean Follmer, and Hiroshi Ishii. 2013. Tangible CityScape. (2013). http://tangible.media.mit.edu/project/tangible-cityscape/

Laura Tateosian, Helena Mitasova, Brendan A Harmon, Brent Fogleman, Katherine Weaver, and Russell S Harmon. 2010. TanGeoMS: tangible geospatial modeling system. *IEEE transactions on visualization and computer graphics* 16, 6 (2010), 1605–12. DOI:http://dx.doi.org/10.1109/TVCG.2010.202

Trimble. 2016. SketchUp. (2016). Retrieved December 15, 2016 from http://www.sketchup.com/

John Underkoffler and Hiroshi Ishii. 1999. Urp: A Luminous-tangible Workbench for Urban Planning and Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 386–393. DOI:http://dx.doi.org/10.1145/302979.303114

David H. Uttal, D. I. Miller, and Nora S. Newcombe. 2013. Exploring and Enhancing Spatial Thinking: Links to Achievement in Science, Technology, Engineering, and Mathematics? *Current Directions in Psychological Science* 22, 5 (2013), 367–373. DOI:http://dx.doi.org/10.1177/0963721413484756

Yoshiki Wakabayashi and Toru Ishikawa. 2011. Spatial thinking in geographic information science: A review of past studies and prospects for the future. *Procedia - Social and Behavioral Sciences* 21 (2011), 304–313. DOI:http://dx.doi.org/10.1016/j.sbspro.2011.07.031

Joseph Wood. 1996. *The geomorphological characterisation of Digital Elevation Models*. Doctoral dissertation. University of Leicester. http://hdl.handle.net/2381/34503

Terri L. Woods, Sarah Reed, Sherry Hsi, John A. Woods, and Michael R. Woods. 2016. Pilot Study Using the Augmented Reality Sandbox to Teach Topographic Maps and Surficial Processes in Introductory Geology Labs. *Journal of Geoscience Education* 64, 3 (08 2016), 199–214. http://proxying.lib.ncsu.edu/index.php?url=http://search.proquest.com.prox.lib.ncsu.edu/docview/1816624044?accountid=12725 Copyright - Copyright National Association of Geoscience Teachers Aug 2016; Document feature - ; Tables; Photographs; Last updated - 2016-09-05.

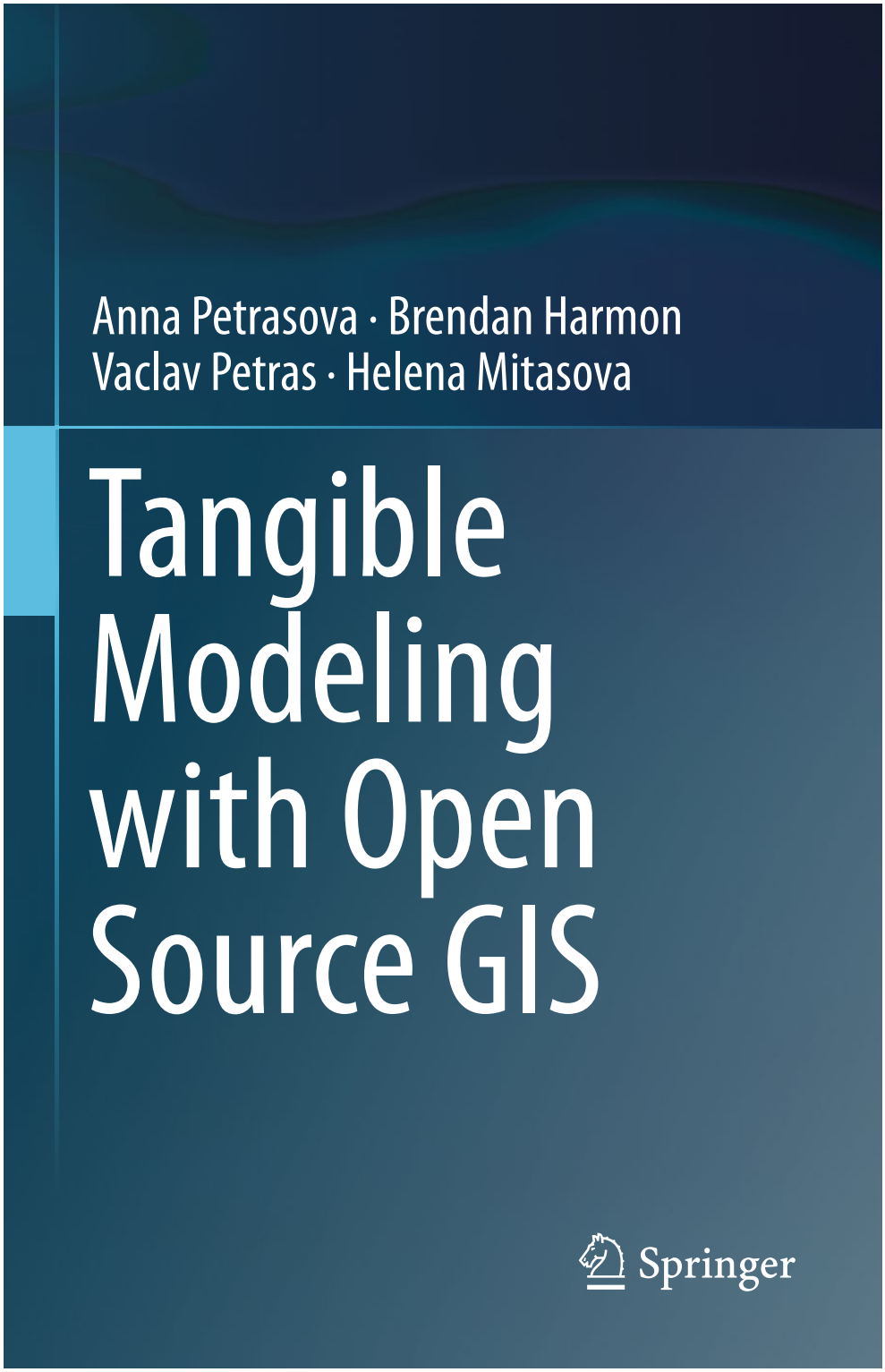
5 Tangible Modeling with Open Source GIS

Reprint

Petrasova, A., Harmon, B., Petras, V., & Mitasova, H. (2015). Tangible Modeling with Open Source GIS. Springer International Publishing. `http://dx.doi.org/10.1007/978-3-319-25775-4`

Attribution

This book was a collaborative effort with my colleagues Anna Petrasova, Vaclav Petras, and Helena Mitasova. I was the sole author of chapter 1 and chapter 3. I co-authored chapter 7 and chapter 10 being responsible for approximately half of the content. I also contributed extensively to the Appendix being responsible again for approximately half of the content. My co-authors and I extensively edited the content, imagery, style, and structure of the entire book.


Anna Petrasova · Brendan Harmon
Vaclav Petras · Helena Mitasova
Tangible Modeling with Open Source GIS
Springer

Tangible Modeling with Open Source GIS

Anna Petrasova • Brendan Harmon • Vaclav Petras
Helena Mitasova

Tangible Modeling with Open Source GIS

Springer

Anna Petrasova
Center for Geospatial Analytics
North Carolina State University
Raleigh, NC, USA

Brendan Harmon
Center for Geospatial Analytics
North Carolina State University
Raleigh, NC, USA

Vaclav Petras
Center for Geospatial Analytics
North Carolina State University
Raleigh, NC, USA

Helena Mitasova
Center for Geospatial Analytics
North Carolina State University
Raleigh, NC, USA

ISBN 978-3-319-25773-0 ISBN 978-3-319-25775-4 (eBook)
DOI 10.1007/978-3-319-25775-4

Library of Congress Control Number: 2015955733

Springer Cham Heidelberg New York Dordrecht London
© The Author(s) 2015
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Contents

1 Introduction 1
1.1 The Evolution of Tangible User Interfaces 1
1.1.1 The Principles of Tangible User Interfaces 2
1.1.2 Dynamic Shape Displays 4
1.1.3 Continuous Shape Displays 7
1.2 Tangible Landscape 8
1.2.1 Modes of Interaction 12
1.2.2 Developing Tangible Landscape 12
1.3 The Organization of This Book 13
References 13

2 System Configuration 17
2.1 Hardware 17
2.1.1 3D Scanner 17
2.1.2 Projector 19
2.1.3 Computer Requirements 22
2.1.4 Physical Setup 23
2.2 Software 24
2.2.1 GRASS GIS 24
2.2.2 GRASS GIS Python API 25
2.2.3 Scanning Component 26
2.2.4 Tangible Landscape Plugin for GRASS GIS 27
References 30

3 Building Physical 3D Models 31
3.1 Handmade Models 31
3.2 Digitally Fabricated Models 33
3.2.1 Digital Models 34
3.2.2 Laser Cutting 36
3.2.3 CNC Routing 37
3.2.4 3D Printing 39
3.3 Molding and Casting 41

3.4 Workflows 43
3.4.1 Selecting a 3D Model Scale 43
3.4.2 Sculpting a Malleable Model from Data 45
3.4.3 CNC Routing a Topographic Model from Contour Data 46
3.4.4 CNC Routing Topographic and Surface Models from Lidar Data 47
3.4.5 3D Printing Topographic and Surface Models from Lidar Data 48
3.4.6 Casting a Malleable Topographic Model with a CNC Routed Mold Derived from Lidar Data 49
References 51

4 Basic Landscape Analysis 53
4.1 Processing and Analyzing the Scanned DEM 53
4.1.1 Creating DEM from Point Cloud 54
4.1.2 Interpolation with the RST Function 54
4.1.3 Analyzing the DEM 55
4.2 Case Study: Topographic Analysis of Graded Landscape 59
4.2.1 Site Description and Model 59
4.2.2 Basic Workflow with DEM Differencing 60
4.2.3 The Impact of Model Changes on Topographic Parameters 61
4.2.4 Changing Landforms 63
References 64

5 Surface Water Flow and Soil Erosion Modeling 65
5.1 Foundations in Flow Modeling 65
5.1.1 Overland Flow 65
5.1.2 Dam Break Modeling 67
5.1.3 Sediment Transport, Erosion, and Deposition 68
5.2 Case Study: The Hydrological Impact of Development 69
5.2.1 The Impact of Development on Surface Water Flow 69
5.2.2 The Impact of Grading on Erosion and Deposition 71
5.3 Case Study: Dam Breach 72
5.3.1 Site Description and Input Data Processing 73
5.3.2 The Impact of the Road on Flooding 74
References 75

6 Viewshed Analysis 77
6.1 Line of Sight Analysis 77
6.2 Case Study: Viewsheds Around Lake Raleigh 78
6.2.1 Site Description and Model 78
6.2.2 Identifying View Points 79
6.2.3 Visibility Analysis on DSM Using Markers 80
6.2.4 Modeling Viewsheds from a New Building 81
References 82

7 Trail Planning 83
7.1 Trail Design Methodology 83
7.1.1 Least Cost Path Analysis 84
7.1.2 Network Analysis 85
7.1.3 Trail Slope Extraction 85
7.2 Case Study: Designing a Recreational Trail 87
7.2.1 Input Data Processing 87
7.2.2 Computing the Trail Using the Least Cost Path 88
7.2.3 Finding the Optimal Trail 90
7.2.4 Mapping Trail Slopes 91
7.2.5 Alternative Trail Scenarios 92
References 95

8 Solar Radiation Dynamics 97
8.1 Solar Radiation Modeling 97
8.2 Case Study: Solar Irradiation in Urban Environment 99
8.2.1 The Impact of Building Configuration on Cast Shadows 100
8.2.2 The Impact of Building Configuration on Direct Solar Irradiation 100
References 102

9 Wildfire Spread Simulation 105
9.1 Fire Spread Modeling Methods 105
9.1.1 Input Data 105
9.1.2 Fire Spread Algorithm 106
9.2 Case Study: Controlling Fire with Firebreaks 107
9.2.1 Data Preparation 108
9.2.2 Scenario with Multiple Firebreaks 109
References 113

10 Coastal Modeling 115
10.1 Modeling Potential Inundation 115
10.2 Case Study: Simulating Flood Risks for Barrier Island Communities 116
10.3 Case Study: Designing Resilient Coastal Architecture 117
Reference 119

A Appendix 121
A.1 Applications of Tangible Landscape 121
A.2 Data Sources 128
A.3 Starting with GRASS GIS 130
References 132

Index 133

Acronyms

2D	Two-dimensional
2.5D	Two-and-a-half-dimensional
3D	Three-dimensional
API	Application programming interface
CAD	Computer aided design
CAM	Computer aided manufacturing
CLI	Command line interface
CNC	Computer numerical control
DEM	Digital elevation model
DSM	Digital surface model
GIS	Geographic information system
GNU GPL	GNU General Public License
GRASS	Geographic Resources Analysis Support System
GUI	Graphical user interface
HCI	Human-computer interaction
MDF	Medium density fiberboard
NCSU	North Carolina State University
NURBS	Non-uniform rational b-spline
RST	Regularized spline with tension
USLE	Universal Soil Loss Equation
SDK	Software development kit
SED	Simplified erosion and deposition
TanGeoMS	The Tangible Geospatial Modeling System
TIN	Triangulated irregular network
TSP	Traveling salesman problem
TUI	Tangible user interface
UNC	University of North Carolina
USGS	United States Geological Survey
USDA	United States Department of Agriculture
UI	User interface

Chapter 1
Introduction

The complex, 3D form of the landscape—the morphology of the terrain, the structure of vegetation, and built form—is shaped by processes like anthropogenic development, erosion by wind and water, gravitational forces, fire, solar irradiation, or the spread of disease. In the spatial sciences geographic information systems (GIS) are used to computationally model, simulate, and analyze these processes and their impact on the landscape. Similarly in the design professions GIS and computer aided design (CAD) programs are used to help study, re-envision, and reshape the built environment. However, because of the unintuitive nature of understanding and manipulating 3D forms in abstract, digital space via a graphical user interface (GUI), technologies like GIS are often so challenging to learn and use that they restrict creativity. The limitations of the technology constrain how we think. Being able to interact more naturally with digital space enhances our spatial thinking, encouraging creativity, analytical exploration, and learning. This is critical for designers as they need to intuitively understand and manipulate information in iterative, experimental processes of creation. With tangible user interfaces (TUIs) like Tangible Landscape one can work intuitively by hand with all the benefits of computer modeling and analytics. This chapter discusses the evolution of TUIs and the development of Tangible Landscape, exploring its motivations, applications, and future directions. This chapter also describes the organization of this book.

1.1 The Evolution of Tangible User Interfaces

Inspired by prototypes like Durrell Bishop's Marble Answering Machine (Poynor 1995) and concepts like Fitzmaurice et al.'s Graspable User Interface (1995) Ishii and Ullmer (1997) proposed a new paradigm of human-computer interaction—the TUI. They envisioned that TUIs could make computing more natural and intuitive by coupling digital bits with physical objects as Tangible Bits. In their

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_1

vision Tangible Bits bridge the physical and digital, affording more manual dexterity and kinaesthetic intelligence and situating computing in physical space and social context (Dourish 2001; Ishii and Ullmer 1997). Recently, the development of TUIs has gained momentum thanks to new developments in 3D technologies such as 3D scanning and 3D printing.

1.1.1 The Principles of Tangible User Interfaces

We can easily, intuitively understand and manipulate space physically, but our understanding is largely qualitative. We can also precisely and quantitatively model and analyze space computationally, but this tends to be less intuitive and requires more experience. Intuition allows us to perceive, think, and act in rapid succession; it allows us to creatively brainstorm and express new ideas. TUIs like Tangible Landscape aim to make the use of computers more intuitive combining the advantages of physicality and computation.

Physical models are used to represent landscapes intuitively. With a physical model we can not only see its volume and depth just as we would perceive space in a real landscape, but also feel it by running our hands over the modeled terrain. We can shape physical models intuitively—for example we can sculpt landforms by hand, place models of buildings, or draw directly on the terrain. With a physical model, however, we are constrained to a single scale, simple measurements, and largely qualitative impressions.

Digital models are used to visualize, quantitatively analyze, and precisely design landscapes. With digital models we can dynamically visualize different scales and layers of information, study the landscape by computing geospatial analyses and simulations, and precisely edit digital models. Digital models, however, can be unintuitive. A high level of abstraction separates us from a digital model—to interact with this digital abstraction of physical space with a GUI we must translate our intent through our hands to the computer via input devices like a keyboard and mouse and then visualize the results graphically on a display. The digital model itself is a high level abstraction of physical space represented by arrays of values or by mathematical functions. We have to think abstractly and make great conceptual leaps to work through so many layers of abstraction. Understanding and working directly with a digital model is a hard learned skill, a skill that constrains our thinking and transforms creativity.

Tangible user interfaces aim to make interaction between humans and computers more natural and intuitive by giving digital data a physical form and physical form a digital dimension (Fig. 1.1). By coupling the physical and the digital through a feedback loop, we can interact with our physical data directly and very naturally with our bodies—drawing on our highly developed kinaesthetic intelligence.

A landscape modeled as a physical-digital hybrid through a TUI such as Tangible Landscape has the potential to be intuitively modeled, creatively explored, dynamically computed, richly layered with data, and rigorously analyzed. It can have

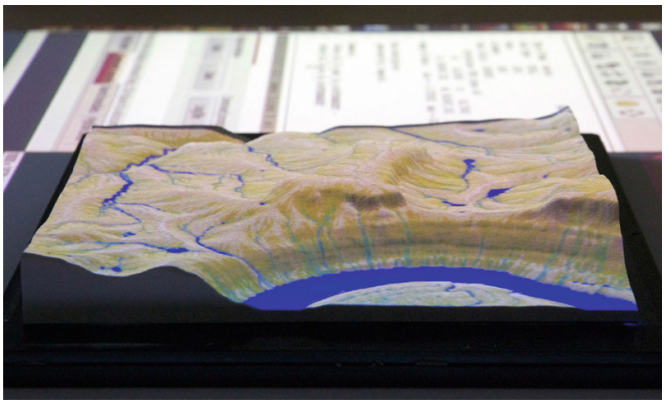


Fig. 1.1 Tangible Landscape: simulated water flow projected over a physical model of a landscape

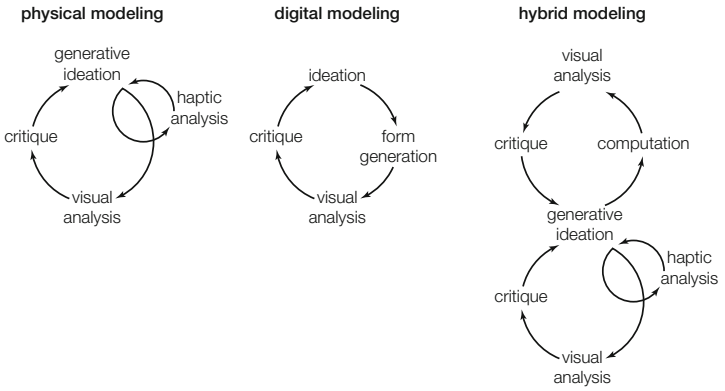


Fig. 1.2 Physical, digital, and hybrid physical-digital modeling processes

all of the advantages of physical landscape models, geospatial models, and CAD combined. With Tangible Landscape we can shape a hybrid model naturally with our hands to creatively explore ideas as if sketching in multiple dimensions. With a physical model we are constrained to modeling in three-dimensions, to modeling for example topographic form. With a hybrid landscape model we are not limited to modeling topographic form—we can dynamically explore and compute many dimensions of geospatial data in rapid succession with the same model (Fig. 1.2).

1.1.2 Dynamic Shape Displays

Dynamic shape displays or shape changing displays—another type of TUI—are computer controlled, interactive, physically responsive surfaces. As we interact with the physical surface it changes the digital model and, conversely, as we interact with the digital model the physical surface changes (Ishii 2008a; Poupyrev et al. 2007). Dynamic shape displays tend to be arrays of pistons and actuated pins that form kinetic, 2.5D surfaces (Petrie 2006) although there is experimental research into continuous, moving surfaces made of shape changing materials driven by heat, magnetic, or electrical stimuli (Coelho and Zigelbaum 2010).

Aegis Hyposurface The Aegis Hyposurface, an early example of a dynamic shape display, is a generative art installation that uses pneumatic actuators to move a triangulated mesh surface according to an algorithm. It can be either preprogrammed or interactive, moving in response to sensed sound, light, or movement. As it was designed and built at an architectural scale the Aegis Hyposurface has a very coarse resolution for an actuated shape display (Goulthorpe 2000).

FEELEX The resolution of actuated shape displays is constrained by the size and arrangement of the piston motors and piston rods or pins that move the surface. Project FEELEX, another early dynamic shape display, used linear actuators to deform a rubber plate. The size of the motors—4 cm—meant that the resolution of the shape display was very coarse. Since the motors are larger than the pins, FEELEX 2 used a piston-crank mechanism to achieve a relatively high 8 mm resolution by clustering the pins while offsetting the motors below. A rubber sheet was stretched over the array of pins to create a 2.5D display for projection. When a user touched the surface they would depress the pins and the pressure of their touch would be recorded as a user interaction (Iwata et al. 2001).

Dynamic Sand Table and Terrain Table The XenoVision Mark III Dynamic Sand Table, developed in 2004, and the Northrop Grumman Terrain Table, developed in 2006, were actuated shape displays that represented topography in 2.5D. In the Terrain Table thousands of pins driven by a motor shaped a silicone surface held taut by suction from a vacuum below into a terrain. The Terrain Table recorded touches as user interactions such as panning and zooming. As users panned, zoomed, or loaded new geographic data, the actuated surface would automatically reshape within seconds (Petrie 2006).

Relief Relief is a relatively low-cost, scalable, 2.5D actuated shape display based on open source hardware and software. Given the complexity and thus the cost, maintenance, and unadaptability of earlier dynamic shape displays like FEELEX and the Northrop Grumman Terrain Table, Leithinger and Ishii (2010) aimed to design a simpler, faster system that was easier to build, adapt, scale, and maintain. In the first prototype of Relief an array of 120 actuated pins driven by electric slide potentiometers stretch a Lycra sheet into a shape display. Users can reshape the shape display by pressing or pulling on the actuated pins. The actuators are

controlled with open source Arduino boards and a program written in the open source language Processing controls, senses, and tracks all of the pins and their relation to the digital model (Leithinger et al. 2009; Leithinger and Ishii 2010). The transparency and freedom of open source solutions should make it relatively easy to reconfigure and adapt this system.

Recompose While Relief was initially designed for a simple, highly intuitive interaction—direct physical manipulation (Leithinger and Ishii 2010)—its next iteration, Recompose, added gesture recognition (Blackshaw et al. 2011; Leithinger et al. 2011). While with Relief users can only directly sculpt the shape display with their hands, with Recompose they can also use gestures to select, translate, rotate, and scale regions of the display. The size and coarse resolution of the actuated shape display mean that only small datasets or subsets of larger datasets can be modeled with useful fidelity. Furthermore, Leithinger et al. (2011) found that only a very limited range of touch interactions could be recognized at the same time and that it can be challenging to manipulate individual pins as they may be out of reach. They augment touch with gestures by adding a Kinect as a depth camera so that users can easily change the context and explore larger datasets. While gestures are less precise than direct physical manipulation, they greatly expand the scope of possible interactions (Blackshaw et al. 2011). Interactions via external devices such as a mouse may be less ambiguous than gestures, but Leithinger et al. argue that they draw users' focus away from the shape display. Therefore they choose to combine touch interactions with gestures rather than pointing devices so that the transition from sculpting to selection, translation, rotation, and scaling would be fluid and seamless given the physical directness of both modes of interaction (Leithinger et al. 2011).

Tangible CityScape Tangible CityScape, a system built upon Recompose, is an example of how this type of TUI can be applied to a specific domain—urban planning. It used a 2.5D dynamic shape display to model and study urban massing. Building masses were modeled by clusters of pins and the model dynamically reshaped as users panned or zoomed with gestures (Tang et al. 2013).

inFORM With inFORM Follmer et al. (2013) developed a dynamically changing user interface capable of diverse, rich user interactions. Building on the Relief and Recompose systems, they developed a 2.5D actuated shape display that supports object tracking, visualization via projection, and both direct and indirect physical manipulation. The shape display is moved by a dense array of pins linked by connecting rods to a larger array of actuators below (Fig. 1.3). The pins, pushing and pulling with variable pressure, offer nuanced haptic feedback to users. A Kinect depth sensor is used to track objects and users' hands. The actuated surface—a grid of pins—can be manipulated directly by pushing and pulling pins. Furthermore, users can interact with the system indirectly via object tracking. The surface can respond to interactions—both direct and indirect—and reshape itself. The actuated surface can also move objects placed on it, enabling indirect physical manipulations. Follmer et al. used this system to explore how shape-changing displays can

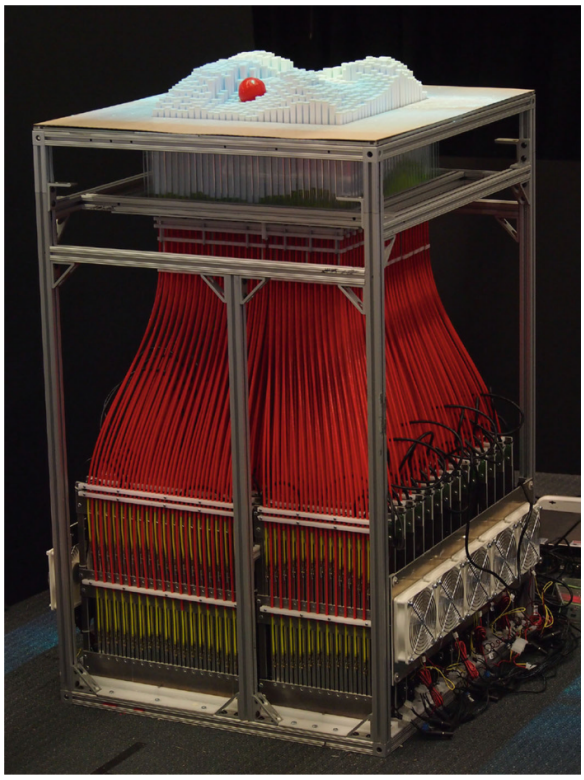


Fig. 1.3 inFORM: a 2.5D actuated shape display designed by the MIT Media Lab (image source: http://tangible.media.mit.edu/project/inform/)

dynamically model content and offer novel modes of interaction based on dynamically changing constraints and opportunities. As a responsive physical user interface inFORM enables rich and varied mode of interaction such as responsive sculpting, moving passive objects, painting changes, and physically instantiated UI elements like buttons (Follmer et al. 2013).

Dynamic shape displays that can be shaped both by touch and computation enable physical interactions from human to computer and computer to human. With a dynamic shape display the digital model is physically instantiated and this shape changing physical model is used for both input and output. When a user changes the surface of the shape display the computer reads the changes; when the computer changes the surface the user sees and feels the changes. Hypothetically this should radically reduce the level of abstraction in human-computer interaction. This novel

mode of bidirectional, tangible interaction should be highly intuitive because it is so direct—human and computers communicate via the same medium. However, while dynamic shape displays may be highly intuitive, they have a relatively coarse resolution, are expensive, maintenance intensive, and hard to transport, all due to the actuators. The resolution of the display for example is constrained by the size of the actuators that make the display kinetic. The coarse resolution makes these displays' representations approximate and abstract, limiting their possible applications.

1.1.3 Continuous Shape Displays

Continuous shape displays couple a continuous physical model with a digital model through a cycle of sculpting, 3D scanning, computation, and projection. We can sculpt the display and then see the resulting computation projected onto the model in near real-time. The feedback from digital modeling is visual—unlike dynamic shape displays, these shape display are passive and do not physically respond to computations. Because there are no actuators the model can be built of a malleable, continuous material such as sand or clay that can be easily sculpted and can accurately represent forms such as landscapes (Ishii et al. 2004; Ishii 2008b).

Projection Augmented Relief Models Currently continuous shape displays rely on projection for representing digital data. While projected imagery has long been used to augment physical terrain models the approach has recently been dubbed Projection Augmented Relief Models (Priestnall et al. 2012). This approach enhances a static physical representation of a landscape with a dynamic visual representation of digital data.

Illuminating Clay With Illuminating Clay (Piper et al. 2002a)—an early continuous shape display from the MIT Media Lab—a clay model of a landscape was continuously scanned with a laser to generate a point cloud of x, y, and z coordinates which were then binned into a digital elevation model (DEM). The DEM or a derived topographic parameter such as slope, aspect, or cast shadow was then projected back onto the clay model so that users could see the impact of their changes in near real-time. Illuminating Clay's library of geospatial analyses was limited as it used custom implementations of elevation, aspect, slope, cast shadow, profile, curvature, viewshed, solar irradiation, and drain direction algorithms. Because Illuminating Clay used a laser scanner the scans were relatively fast—1.2 s each—and accurate to less than 1 mm, but the system was very expensive (Piper et al. 2002a,b). Piper et al. (2002a) also tried to use TUI elements based on object tracking to select algorithms and parameters, but found that too many parameters were involved in geospatial modeling for this approach to be feasible, so they ultimately used standard graphical user interactions with a mouse and keyboard.

Sandscape In an effort to make a more affordable continuous shape display Ishii et al. (2004) designed a system called Sandscape based on infrared depth sensing. In Sandscape a 'sandbox' filled with 1 mm glass beads was lit from beneath by infrared LEDs; an infrared camera above the 'sandbox' captured the intensity of infrared light passing through the glass beads. A DEM based on the light intensity was projected back onto the 'sandbox' so that users would have real-time feedback as they sculpted. While this system was very fast and much less expensive than Illuminating Clay, the scans were low resolution and the physical medium—glass beads—could only be coarsely sculpted (Ishii et al. 2004; Ishii 2008b).

Augmented Reality Sandbox SandyStation (Altman and Eckstein 2014), developed in 2011, and the Augmented Reality Sandbox (Kreylos 2015), developed in 2012, are continuous shape display sandboxes. Kreylos's Augmented Reality Sandbox, inspired by the Czech SandyStation, couples a sandbox with a DEM through a real-time cycle of 3D scanning, modeling and simulation, and projection. As users sculpt the sand a Kinect sensor continually scans the sand surface generating a stream of depth maps. The depth stream is filtered to remove noise and the user's hands. A DEM, contours, and simulated water flow are computed from the depth and projected back on the sandbox. The water flow simulation is a custom implementation based of the shallow water flow equations. The Augmented Reality Sandbox is powered by Kreylos's Kinect 3D Video Capture Project and Vrui VR Toolkit, open source software released under the GNU General Public License. The system is highly intuitive and very fast with each cycle of scanning, processing, and simulation completing in just a couple seconds. However, given that the system only models elevation, contours, and water flow on an abstract landscape it is limited in its scope of potential applications.

Tangible Geospatial Modeling System The Tangible Geospatial Modeling System (TanGeoMS) extended the concept of Illuminating Clay by integrating GRASS GIS (Fig. 1.4). Integrating a full GIS into a continuous shape display gave users a wide range of sophisticated, scientific tools for geospatial modeling and visualization in a highly intuitive environment. By combining freeform hand modeling with geospatial modeling, simulation, and visualization users could easily explore how changes in topographic form affect landscape processes (Mitasova et al. 2006). Like Illuminating Clay the system used a laser scanner to 3D scan a clay model. The scanned point cloud was then interpolated as a DEM, select geospatial analyses or simulations were computed from the DEM, and the results were projected back onto the clay model (Tateosian et al. 2010) (Fig. 1.4b).

1.2 Tangible Landscape

The next generation of TanGeoMS—Tangible Landscape—was designed to intuitively 3D sketch landscapes. This continuous shape display is more tightly integrated with GRASS GIS and has new modes of interaction based on object recognition so that users cannot only sculpt, but also sketch in 3D—rapidly

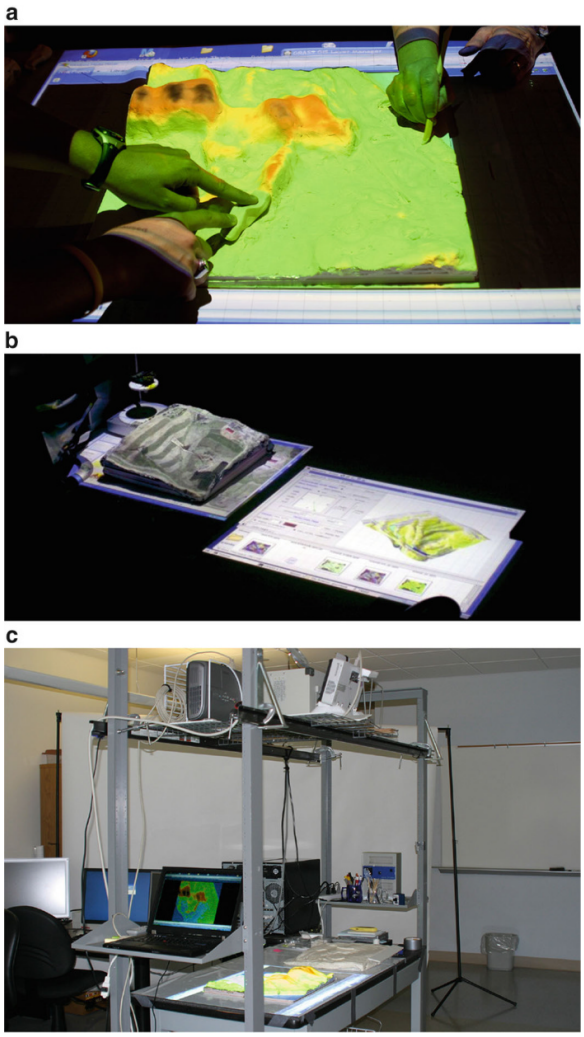


Fig. 1.4 The Tangible Geospatial Modeling System: a continuous shape display powered by GRASS GIS. (**a**) Multiple users interacting. (**b**) The shape display paired with a 3D rendering. (**c**) The hardware setup with a laser scanner mounted above the model

exploring ideas while learning from near real-time feedback (Fig. 1.5). Tangible Landscape is more affordable than its predecessors because it uses a Kinect sensor to 3D scan physical models. As a continuous shape display with object recognition

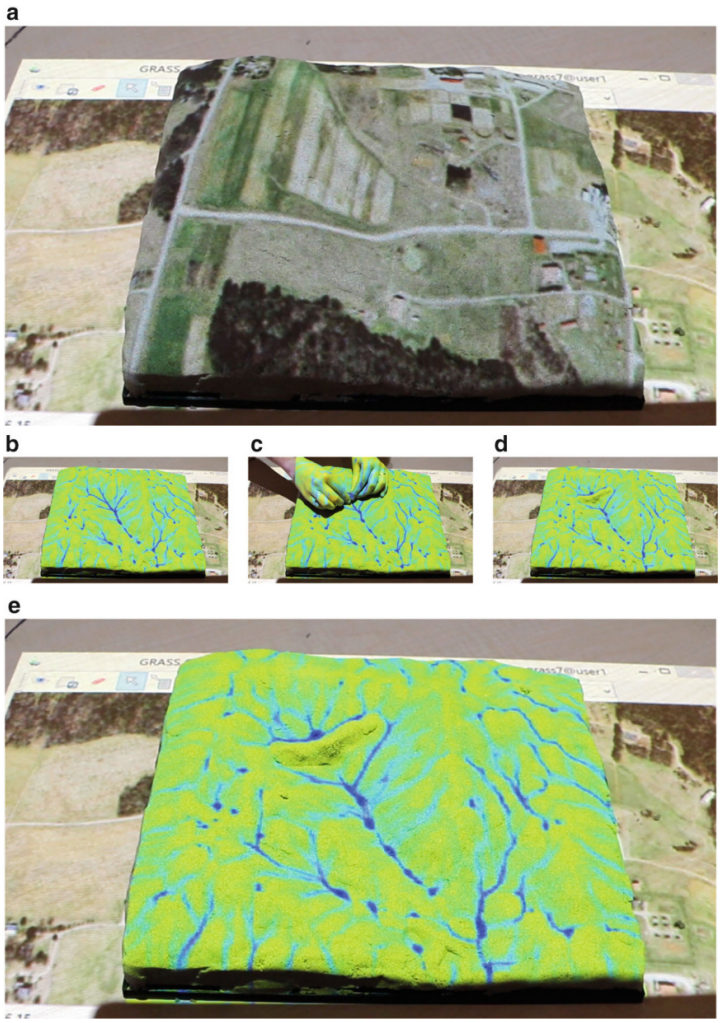


Fig. 1.5 3D sketching a check dam with Tangible Landscape: (**a**) orthoimagery, (**b**) water flow, (**c**) sculpting a check dam, (**d**) updated water flow, and (**e**) a dammed valley with water flow simulated by *r.sim.water*. http://grass.osgeo.org/grass70/manuals/r.sim.water.html

a wide range of media can be used such as polymer-enriched sand, clay, 3D prints, CNC-machined models, and architectural models. It is more tightly integrated with GIS, using a GRASS GIS plugin to automatically scan, process, georeference,

Fig. 1.6 Collaboratively sculpting a retention pond

import, and analyze the model. Because it is so tightly integrated with GRASS GIS users can also use the GUI, the command line, and scripting as advanced controls for tasks not suited to a TUI (Petrasova et al. 2014).

Conceptually, Tangible Landscape couples a physical model with a digital model in a real-time feedback cycle of 3D scanning, geospatial modeling and simulation, and projection. For example, by sculpting the terrain of the physical model, we can see how the changes affect processes like the flow of water, flooding, erosion, and solar irradiation. Thus we can easily and rapidly test ideas while being guided by scientific feedback, exploring a much a larger solution space and make more creative and informed decisions. And since many users can interact with the physical model at once, Tangible Landscape encourages collaboration and interdisciplinary exchange (Fig. 1.6). Tangible Landscape combines real-time interaction with extensive scientific tools for modeling, analysis, and visualization at the precision needed for real-world design and planning applications. Tangible Landscape can be used for applications such as grading landforms (Chap. 3), analyzing topography (Chap. 4), modeling water flow and soil erosion (Chap. 5), analyzing viewsheds (Chap. 6), planning trail networks (Chap. 7), analyzing solar radiation dynamics (Chap. 8), simulating and managing fire (Chap. 9), and modeling inundation and flooding (Chap. 10).

Tangible Landscape is inexpensive, portable, adaptable, and easy to implement. As a high resolution continuous shape display it can be used for both rapid ideation and precise modeling. However, unlike a dynamic shape display the physical model is passive and cannot be changed computationally in real-time—trading real-time response for precision, complexity, and cost. Rapid prototyping can be used to precisely shape or reshape the physical model allowing users to work with real world landscapes and easily reset the conditions.

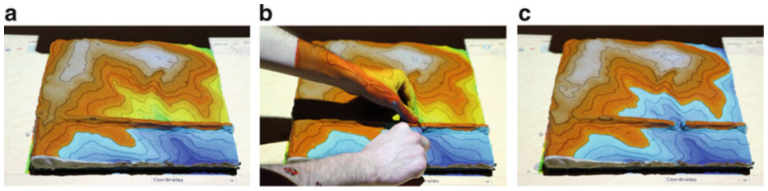


Fig. 1.7 3D sketching a levee breach with feedback on the resulting extent of flooding: (**a**) a flooded landscape, (**b**) breaching the levee, and (**c**) updated flooding

1.2.1 Modes of Interaction

There are many ways to 3D sketch with Tangible Landscape. It combines a couple modes of tangible user interaction. As a continuous shape display—a 2.5D interface that physically instantiates and displays continuous data in real-time—we can use it sculpt surfaces and volumes. It can also use object recognition to detect and compare changes, model volumes, designate points, select data, and sketch lines. By sculpting a physical model made of polymeric sand we can intuitively model topography (Fig. 1.7), forest canopy, buildings, and cost surfaces. We can also explore volumes such as subsurface soil moisture by digging the into polymeric sand. With change detection we can visualize the difference between models to study how we have changed it. With object recognition we can place makers to create points such as observer points for viewshed analysis, lines such as section cuts for visualizing volumes, and polylines such as the least cost path between points for route analysis. Since Tangible Landscape has so many modes of interaction and draws on GRASS GIS' extensive modules for data management, geospatial modeling, and visualization, it can easily be adapted to new uses. Since both Tangible Landscape and GRASS GIS are open source projects we can easily develop new modes of interaction and new analyses.

1.2.2 Developing Tangible Landscape

Tangible Landscape is a free, open source project with source code hosted on GitHub.[1] The system is constantly evolving in response to new developments in its hardware and software components and the needs of its expanding range of applications. Enhanced point cloud processing, optimization of the core functions, and migration to more efficient hardware and operating systems will improve real-time interaction and allow us to incorporate more sophisticated dynamic

[1] https://github.com/ncsu-osgeorel/grass-tangible-landscape

modeling. User interaction and experience can be further improved with dashboards and steered simulations. Computationally controlled shape generation with in-situ digital fabrication will combine the advantages of dynamic and continuous shape displays. By providing Tangible Landscape as a free, open source project, along with this book, we aim to build a community of developers and users of this technology for the benefit of education, research and collaborative decision making in communities worldwide.

1.3 The Organization of This Book

This book describes tangible geospatial modeling using open source GIS and its applications. For each application we explain the underlying theory and algorithms, provide the workflows used, and describe case studies. Simple GRASS GIS workflows consisting of module calls are presented in boxes with green backgrounds, while more complex workflows written in Python are presented in boxes with blue backgrounds. 3D modeling workflows in Rhinoceros are presented in boxes with grey backgrounds.

Chapter 2 describes how Tangible Landscape works and how it is set up. This chapter details the 3D scanning technology, the hardware and software used, and the physical setup of the system. It provides information about where to download GRASS GIS and the Tangible Landscape plugin and explains how to read and use the workflows. Chapter 3 explains how to build physical models for use with Tangible Landscape. This chapter covers handmade modeling techniques, digital fabrication techniques, and techniques for molding and casting models. Chapter 4 covers the basic processes of scanning, interpolating, and analyzing a terrain model. The following chapters describe a range of scientific applications for Tangible Landscape. Each chapter first covers the relevant theory and algorithms and then demonstrates the application in a case study. Chapter 5 covers surface water flow and soil erosion, Chap. 6 covers viewsheds, Chap. 7 covers trail planning, Chap. 8 covers solar irradiation dynamics, Chap. 9 covers the spread of wildfire, and Chap. 10 covers coastal inundation and flooding. The Appendix includes case studies of additional applications of Tangible Landscape, a summary of the data used in this book with download links, a list of useful on-line data sources, and a guide to getting started with GRASS GIS.

References

Altman, P. and Eckstein, R. SandyStation [online]. (2014) [Accessed 2015-08-12]. http://en.sandystation.cz/.

Blackshaw, M., DeVincenzi, A., Lakatos, D., Leithinger, D., and Ishii, H. (2011). Recompose: direct and gestural interaction with an actuated surface. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11*, page 1237, Vancouver. ACM Press.

Coelho, M. and Zigelbaum, J. (2010). Shape-changing interfaces. *Personal and Ubiquitous Computing*, 15(2):161–173.

Dourish, P. (2001). *Where the action is : the foundations of embodied interaction*. MIT Press, Cambridge, MA.

Fitzmaurice, G. W., Ishii, H., and Buxton, W. (1995). Bricks: laying the foundations for graspable user interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 442–449.

Follmer, S., Leithinger, D., Olwal, A., Hogge, A., and Ishii, H. (2013). inFORM: dynamic physical affordances and constraints through shape and object actuation. In *UIST '13 Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 417–426, St. Andrews, UK. ACM Press.

Goulthorpe, M. Aegis Hyposurface [online]. (2000) [Accessed 2015-04-10]. http://hyposurface.org/.

Ishii, H. (2008a). Tangible bits: beyond pixels. In *Proceedings of the 2nd international conference on Tangible and embedded interaction - TEI '08*, pages xv–xxv, Bonn, Germany. ACM Press.

Ishii, H. (2008b). The tangible user interface and its evolution. *Communications of the ACM*, 51(6):32–36.

Ishii, H., Ratti, C., Piper, B., Wang, Y., Biderman, a., and Ben-Joseph, E. (2004). Bringing Clay and Sand into Digital Design – Continuous Tangible user Interfaces. *{BT} Technology Journal*, 22(4):287–299.

Ishii, H. and Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, pages 234–241, New York, New York, USA. ACM Press.

Iwata, Yano, Nakaizumi, and Kawamura (2001). Project FEELEX: adding haptic surface to graphics. *Proceedings of SIGGRAPH 2001*, pages 469–475.

Kreylos, O. Augmented Reality Sandbox [online]. (2015) [Accessed 2015-08-12]. http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/index.html.

Leithinger, D. and Ishii, H. (2010). Relief: a scalable actuated shape display. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction - TEI '10*, page 221, Cambridge, MA. ACM Press.

Leithinger, D., Kumpf, A., and Ishii, H. Relief [online]. (2009). http://tangible.media.mit.edu/project/relief/.

Leithinger, D., Lakatos, D., Devincenzi, A., Blackshaw, M., and Ishii, H. (2011). Direct and Gestural Interaction with Relief : A 2 . 5D Shape Display. *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 541–548.

Mitasova, H., Mitas, L., Ratti, C., Ishii, H., Alonso, J., and Harmon, R. S. (2006). Real-time landscape model interaction using a tangible geospatial modeling environment. *IEEE Computer Graphics and Applications*, 26(4):55–63.

Petrasova, A., Harmon, B. A., Petras, V., and Mitasova, H. (2014). GIS-based environmental modeling with tangible interaction and dynamic visualization. In Ames, D. and Quinn, N., editors, *Proceedings of the 7th International Congress on Environmental Modelling and Software*, San Diego, California, USA. International Environmental Modelling and Software Society.

Petrie, G. (2006). TouchTable and TerrainTable. *Geoinformatics*, pages 40–41.

Piper, B., Ratti, C., and Ishii, H. (2002a). Illuminating clay: a 3-D tangible interface for landscape analysis. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '02*, page 355, Minneapolis. ACM Press.

Piper, B., Ratti, C., and Ishii, H. (2002b). Illuminating Clay: A Tangible Interface with potential GRASS applications. In *Proceedings of the Open Source GIS - GRASS users conference 2002*, Trento, Italy.

Poupyrev, I., Nashida, T., and Okabe, M. (2007). Actuation and tangible user interfaces: the Vaucanson duck, robots, and shape displays. *Proceedings of TEI 2007*, pages 205–212.

Poynor, R. (1995). The Hand That Rocks the Cradle. *ID Magazine*, pages 60–65.

Priestnall, G., Gardiner, J., Way, P., Durrant, J., and Goulding, J. (2012). Projection Augmented Relief Models (PARM): Tangible Displays for Geographic Information. *Proceedings of Electronic Visualisation and the Arts, London*, pages 1–8.

Tang, S. K., Sekikawa, Y., Leithinger, D., Follmer, S., and Ishii, H. Tangible CityScape [online]. (2013) [Accessed 2014-03-27]. http://tangible.media.mit.edu/project/tangible-cityscape/.

Tateosian, L., Mitasova, H., Harmon, B. A., Fogleman, B., Weaver, K., and Harmon, R. S. (2010). TanGeoMS: tangible geospatial modeling system. *IEEE transactions on visualization and computer graphics*, 16(6):1605–12.

Chapter 2
System Configuration

The setup of the Tangible Landscape system consists of four primary components: (a) a physical model that can be modified by a user, (b) a 3D scanner, (c) a projector, and (d) a computer installed with GRASS GIS for geospatial modeling and additional software that connects all the components together. The physical model, placed on a table, is scanned by the 3D scanner mounted above. The scan is then imported into GRASS GIS, where a DEM is created. The DEM is then used to compute selected geospatial analyses. The resulting image or animation is projected directly onto the modified physical model so that the results are put into the context of the modifications to the model.

2.1 Hardware

Tangible Landscape can be built with affordable, commonly available hardware: a 3D scanner, a projector and a computer. We describe some of the current hardware options while noting that the technology develops rapidly and alternative, more effective solutions may emerge.

2.1.1 3D Scanner

In the Tangible Landscape workflow the 3D scanner captures the physical model as it is modified by users. Therefore it is important that the device scans as accurately as the technology allows. Few affordable 3D scanners are currently available. However, due to growing demand this technology is developing rapidly and new, improved 3D scanners may become available soon. The Tangible Landscape system was developed using the Kinect for Windows v2 and thus has the disadvantage of

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_2

being bound to the Microsoft Windows operating system. Being bound to a specific operating system limits the choice of environment in which to run the application. Therefore, the selection of the 3D scanner may change in the future as new, more flexible technologies become available.

The Technology Behind 3D Scanning We describe the basic principles behind current 3D depth sensing technologies in order to explain both the potential and the limitations of current scanning devices. More detailed information can be found for example in Mutto et al. (2012).

Several currently available depth sensors such as Apple Primesense Carmine, Asus Xtion PRO LIVE,[1] and Kinect for Windows v1, are based on a Primesense proprietary light-coding technique. This technique uses triangulation to map 3D space in manner similar to how the human visual system senses depth from two slightly different images. Rather than using two cameras it triangulates between a near-infrared camera and a near-infrared laser source. Corresponding objects need to be identified in order to triangulate between the images. A light coding (or structured light) technique is used to identify the objects. The laser produces a pseudo-random dot pattern which is then used to find the matching dot pattern in the projected pattern. In this way the final depth image can be constructed.

Optical triangulation is also used by high resolution 3D scanners such as SCANIFY by Fuel3D (2015). The scanner acquires a series of stereoscopic photographs with different lighting directions. When scanning multi-color objects with some texture this technology can create highly accurate 3D models with high levels of detail. The major limitation is that only relatively small models that can be scanned making it unsuitable for most applications using physical landscape models. These models usually need to be relatively large to clearly represent relief at a landscape scale without too much vertical exaggeration.

Kinect for Windows v2 uses *time-of-flight*, a technique widely used in lidar technology. Kinect v2 has a sensor that indirectly measures the time it takes for pulses of near-infrared laser light to travel from a laser source, to a target surface, and then back to the image sensor. Time-of-flight sensors are generally considered to be more precise, but also more expensive.

Since the scanning and 3D modeling algorithms and their implementations for most sensors are proprietary, the specific behavior and precision of particular sensors is subject to many experimental studies. Generally, sensors using near-infrared light are sensitive to lighting conditions, so outdoor usage is typically not recommended. The depth resolution decreases with the distance from the sensor and also at close range (i.e. between a couple of millimeters to tens of centimeters). The range and field of view of the sensors can vary; Tangible Landscape requires short range sensors with a minimum distance of 0.5 m to keep the highest possible resolution. When scanning with one sensor the size of the physical model is limited by the required accuracy because the sensor must be far enough away to

[1] https://www.asus.com/us/Multimedia/Xtion_PRO_LIVE/

capture the entire model in the sensor's field of view, which for the Kinect for Windows v2 is 60° vertical by 70° horizontal. Detailed information about accuracy and precision of Kinect for Windows v1 and v2 can be found in Andersen et al. (2012); Lachat et al. (2015) and a comparison to Asus Xtion can be found in Gonzalez-Jorge et al. (2013).

2.1.2 Projector

The projector projects the background geospatial data and results of analyses onto the the 3D physical model. Therefore it is important to select a projector with sufficient resolution and properties that minimize distortion and generate a bright image.

Resolution and brightness are two important criteria to be considered. We recommend higher resolution projectors offering at least WXGA (1280 × 800). The brightness depends upon where Tangible Landscape is used and whether the room's ambient light can be reduced for the sake of brighter projected colors. We recommend brighter projectors (at least 3000 lumens) since we project on a variety of materials which are not always white and reflective.

There are two important types of projectors—standard and short-throw projectors. They differ in throw ratio values, which are defined as the distance from the projector's lens to the screen, divided by the width of the projected image (short throw and ultra short throw projectors have ratios 0.3–0.7, while the standard projectors have throw ratio values around 2). At the same distance the projector with a lower throw ratio can display a larger image. In other words a projector with a lower throw ratio (short-throw) can project an image of the same size as the higher throw ratio projector, but from a shorter distance. Tangible Landscape can be set up using both types of projectors; each has advantages and disadvantages.

The placement and configuration of the projector is important because it affects the coverage, distortion, and visibility of the projected data. For example in some setups the 3D scanner may be caught in the projection and would thus cast a shadow over the model. Therefore it can be practical to use a short-throw projector because it can be placed to the side of the physical model at a height similar to the 3D scanner (Figs. 2.1b and 2.2). Since the projection is cast from the side the projection beam does not cross the 3D scanning device and no shadow is cast. However, with a short-throw projector there is a certain level of visible distortion when projecting on a physical model that has substantial relief. The distortion occurs because the light rays reach the model at a very acute angle. The horizontal position at which the projected light intersects the model is shifted from the position at which it would intersect with a flat surface. Larger differences in height increase the distortion. Theoretically we can remove the distortion by either warping the projected data itself or using the projector to automatically warp the projected image. The first solution would require an undistorted dataset for geospatial modeling and a warped copy of that data for projecting. That is impractical especially when working with

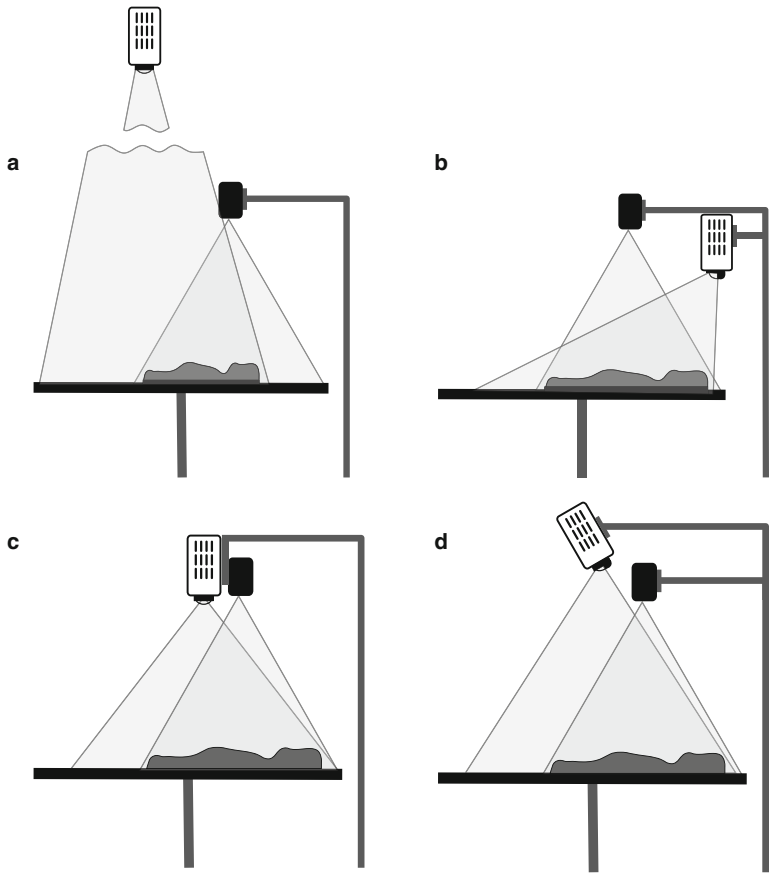


Fig. 2.1 Possible placements of the projector and scanner according to projector type. (**a**) Standard projector mounted in the ceiling. (**b**) Standard short throw projector. (**c**) Short throw projector capable of projecting above the center. (**d**) Short throw projector tilted to project above the center with keystone correction

many different raster and vector layers. The other solution requires the projector itself to warp the image; while this technology exists, it is only offered by a few projector manufacturers and such projectors are typically more expensive. Moreover, as the landscape is modified, the warping pattern should change as well. Currently it is not possible to find this feature in the off-the-shelf projectors.

With standard projectors the distortion is usually negligible since the incidence angle of the rays is relatively small. A standard projector needs to be placed much

Fig. 2.2 Mobile system setup: projector, scanner, model and laptop

higher above the model than a short-throw projector does due to the difference in throw ratios (Figs. 2.1a and 2.3). This creates several challenges. It is hard to mount and manipulate the projector when is it so high above the model. Furthermore it is challenging to align projector and the 3D scanner without casting a shadow. Since the 3D scanner has a limited field of view it must be placed close to the horizontal center of the model. When the projector is mounted above the scanner, the scanner is caught in the projector's beam, casting a shadow over part of the model. For small to mid sized models, depending on the particular setup and the height of the scanner and projector, this may not be a problem.

The shadow of the scanner can also be avoided with specialized short throw projectors that allow greater installation flexibility through lens shifting. These projectors (for example Canon WUX400ST) are capable of projecting from the horizontal center of the projected area at the same height as the scanner (Fig. 2.1c). A device which combines the scanner and projector would make this setup easy; an appropriate device, however, was not available at the time of writing.

To minimize the distortion when using projectors with lower throw ratios, we can project from the center by tilting the projector and correcting the resulting keystone distortion (Fig. 2.1d). With this setup we recommend projectors that have throw ratios around 1.0 (for example Epson PowerLite 1700 Series) as a lower ratio can result in additional distortion. When testing the projector setup it is useful to project the rectangular grid on a flat surface in order to quickly check if there is any distortion.

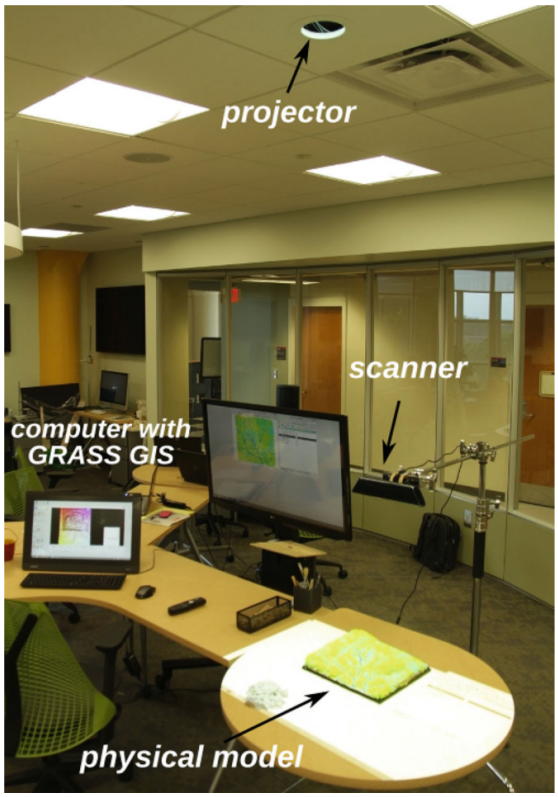


Fig. 2.3 Laboratory system setup with the projector mounted in the ceiling

2.1.3 Computer Requirements

System requirements depend largely upon the sensor and its associated library or software development kit (SDK). Certain sensors, such as Kinect for Windows, are designed to work only on specific operating systems. The computer should be configured for 3D scanning and geospatial modeling, both of which are performance- and memory-intensive processes. The hardware requirements are very similar to the requirements for gaming computers: a multi-core processor, at least 4 GB of system memory and a good graphics card are necessary to achieve real-time interaction with the model. The specific parameters required for the scanner device should be reviewed on the manufacturer's website.

2.1.4 Physical Setup

The setup for Tangible Landscape system is quite flexible. However, there are some crucial components and some specifics that make a setup more usable. When building Tangible Landscape in the laboratory or when bringing it to the community the following items are necessary:

- a table for a model
- a laptop or desktop computer with a table
- a scanner
- a projector
- 1–2 stands for a projector and a scanner
- 3–4 power plugs (and/or extension cable)

Ideally the table should be either a 90 cm × 90 cm table with rounded corners, a rounded table 100 cm in diameter, or a teardrop table of similar size. To freely interact with the model a person should be able to almost reach the other side of the table; this is not possible with larger tables. Smaller tables, on the other hand, have less space for tools, additional sand, and application windows. Ideally application windows with additional information should be projected onto the table beside the model. A large model can be placed on top of a smaller table or stand, provided that this base is stable. In this case if any application windows are needed, they have to be projected directly onto the model.

A round table is quite advantageous because people can stand at equal distances from the model and can easily move around the table. Unfortunately, one side of the table is typically occupied by the metal stands for projector and scanner since ceiling mounts are rarely possible and render the system immobile. The table should be stable enough to hold sand and models and sturdy enough to withstand their manipulation. We recommend putting the computer on a separate table so that the modeling table is not cluttered. The more space there is around the modeling table, the more access users have to the model. The computer, however, needs to be situated so that its operator has easy access to the model as well.

An alternative is to use a rectangular table (ideally with rounded corners) that is 140 cm × 90 cm for both the model and the laptop. A narrow table that is 150 cm × 75 cm may work as well. This setup makes the model less accessible as at least two sides of the model are blocked. However, the whole model should be accessible from any of the remaining sides, so this setup does not necessarily limit interaction, but rather the number of users.

Sometimes it is advantageous to have a large screen showing additional data or 3D views to enhance users' understanding of the processes. However, such screen should not limit access to the physical model. Tangible Landscape could be also extended with additional devices such as 3D displays, head-mounted displays, and hand-held devices.

There should be at least 80 cm of space on each side of the modeling table so that there is room for walking, standing, and placing the stands. The whole area required is about 2.5 m × 2.5 m but in practice the area is often larger depending on the size

of the tables and the surroundings. Generally, it is better to have a larger open space. With more space people can easily come and cluster around Tangible Landscape spreading around the different sides of the setup.

The projector and scanner can be mounted on one or two stands according to projector's capabilities as discussed in detail in Sect. 2.1.2. Two separate stands or arms give enough flexibility to accommodate various types of projectors. In any case, the stand for the projector must be robust because some of the projectors with adequate parameters can be heavy. However, there exist some portable projectors weighting less than 2 kg as well. A mobile setup with two separate stands and two separate tables is shown in Fig. 2.2. For permanent setups the projector can be mounted in the ceiling as shown in Fig. 2.3.

2.2 Software

The software for Tangible Landscape consists of several different components: a scanning program dependent upon the Kinect for Windows SDK libraries, GRASS GIS for processing scans and running geospatial analyses, and a custom library that automates the entire process. Here we briefly describe the software components with the main focus on GRASS GIS as the processing engine of Tangible Landscape. Tangible Landscape software is available in the GitHub repository[2] under the GNU GPL license.

2.2.1 GRASS GIS

GRASS GIS (Neteler and Mitasova 2008) is a general purpose, cross-platform, open-source geographic information system with raster, vector, 3D raster and image processing capabilities. It includes more than 400 modules for managing and analyzing geographical data and many more user contributed modules available in the add-on repository. GRASS GIS modules can be run using a command-line interface (CLI) or a native graphical user interface (GUI) called wxGUI which offers a seamless combination of GUI and CLI native to the operating system.

Modules are organized based on the data type they handle and they follow naming conventions explained in Table 2.1. Each module has a set of defined options and flags which are used to specify inputs, outputs, or different module settings. Most core modules using GRASS GIS C library are written in C for performance and portability reasons. Other modules and user scripts are written in Python using the GRASS GIS Python Scripting Library which provides a Python interface to GRASS GIS modules.

[2] https://github.com/ncsu-osgeorel/grass-tangible-landscape

Table 2.1 Naming conventions for GRASS GIS modules with examples

Name	Data type	Examples
g.*	General data management	g.list, g.remove, g.manual
r.*	Raster data	r.neighbors, r.viewshed, r.cost
r3.*	3D raster data	r3.colors, r3.to.rast, r3.cross.rast
v.*	Vector data	v.net, v.surf.rst, v.generalize
db.*	Attribute data	db.tables, db.select, db.dropcolumn
t.*	Temporal data	t.register, t.rast.aggregate, t.vect.extract
i.*	Imagery data	i.segment, i.maxlik, i.pca

GRASS GIS software can be downloaded freely from the main GRASS project website.[3] The download web page offers easy to install binary packages for GNU/Linux, Mac OS X, and Microsoft Windows as well as the source code. The GRASS GIS website also provides additional documentation including manual pages, tutorials, information about the externally developed modules (add-ons) and various publications. Support for developers and users is provided by several mailing lists. The following tutorial provides a quick introduction to GRASS GIS: http://grass.osgeo.org/grass70/manuals/helptext.html.

2.2.2 GRASS GIS Python API

GRASS GIS provides several Python application programming interfaces (API) to access different functionalities and accommodate many different use cases. Tangible Landscape uses the GRASS GIS Python Scripting Library to easily build new workflows by chaining together existing GRASS GIS modules. We will show an example how to use this library to automate tasks and build new functionality.

GRASS GIS modules are called with the following command syntax:

$$\underbrace{\texttt{r.colors}}_{\text{module name}}\ \underbrace{\texttt{-e}}_{\text{flag}}\ \underbrace{\texttt{raster}}_{\text{option name}}\texttt{=}\underbrace{\texttt{elev_state_500m}}_{\text{option value}}\ \texttt{color=elevation}$$

This command assigns a predefined color ramp `elevation` to the raster map `elev_state_500m`. It can be executed in a GRASS GIS session from a terminal or from a GRASS GIS GUI command console. The same command looks very similar when written using the Python Scripting Library—it is just adjusted to Python syntax. To run it we have to be in a GRASS GIS session and must first import the necessary library.[4] We will use the function `run_command` to call individual modules:

[3] http://grass.osgeo.org

[4] All further Python code snippets assume the library is already imported to avoid code duplication.

```
import grass.script as gscript
gscript.run_command('r.colors', flags='e',
    raster='elev_state_500m', color='elevation')
```

Besides `run_command`, other important functions are `read_command` for reading the raw module output, `parse_command` for reading module output already parsed into a Python dictionary, and `write_command` for passing strings from standard input. All GRASS GIS modules can be called using these functions. However, for some commonly used modules the library conveniently provides wrappers simplifying the syntax, such as the `mapcalc` function, a wrapper for the *r.mapcalc* module.

The following code snippet provides a more complex example using these functions. Here, we extract low-lying areas from a DEM where the elevation z is lower than the difference $z_{mean} - z_{stddev}$:

```
import grass.script as gscript

def main():
   input_raster = 'elevation'
   output_raster = 'low_areas'
   stats = gscript.parse_command('r.univar', map=input_raster,
       flags='g')
   mean = float(stats['mean'])
   stddev = float(stats['stddev'])
   low = mean - stddev
   gscript.mapcalc('{out} = {inp} <
       {lim}'.format(out=output_raster, inp=input_raster,
       lim=low))

if __name__ == "__main__":
   main()
```

More information on writing Python scripts in GRASS GIS can be found in the online GRASS GIS Python Scripting Library documentation. [5]

2.2.3 Scanning Component

The scanning component is currently a C++ application derived from the sample application KinectFusionExplorer shipped with the Kinect for Windows SDK. It reconstructs a mesh based on multiple scans using the KinectFusion algorithm (Newcombe et al. 2011) and writes the point cloud as a text file. It allows us to set the scanning properties such as the resolution of the reconstructed mesh and to limit the volume of the mesh. We recommend using a higher volume resolution such

[5] GRASS GIS Python Scripting Library documentation: http://grass.osgeo.org/grass70/manuals/libpython/script_intro.html

Fig. 2.4 A mesh with a resolution of 512 voxels/m constructed using the KinectFusion algorithm. The mesh captures a sand model, the area outside of the model including a pile of sand and a modeling tool, and the hands of a person interacting with the model

as 512 or 640 voxels/m. It is important to make the volume of the mesh as small as possible to increase processing speed and decrease memory demands. Since the size of the mesh can only be controlled in predefined steps we cannot limit the scanned area to the model area in this way—further processing of the point cloud to filter the points outside of the model area is needed (Fig. 2.4).

2.2.4 Tangible Landscape Plugin for GRASS GIS

The Tangible Landscape plugin connects the scanning component with GRASS GIS and automates the loop of scanning, importing scans, and geoprocessing in the GRASS GIS environment. It has a graphical user interface which allows the adjustment of the different processing parameters that are necessary to properly georeference and extract the model. First it can perform an initial calibration. Then with each new scan the plugin performs the following basic steps:

1. acquiring the scan as a point cloud,
2. extracting only relevant points from the point cloud (filtering points outside the model area)
3. georeferencing (horizontal rotation, horizontal and vertical scaling, translation) the scanned data to known geographic coordinates on the edges of the model,
4. importing the point cloud into GRASS GIS and creating a digital elevation model,
5. running selected geospatial analyses, and
6. re-rendering the resulting raster or vector data that are projected on top of the model.

The conversion from the scanned point cloud to a DEM is an automated process; the details and assumptions that have to be met to produce good results are described in the following paragraphs.

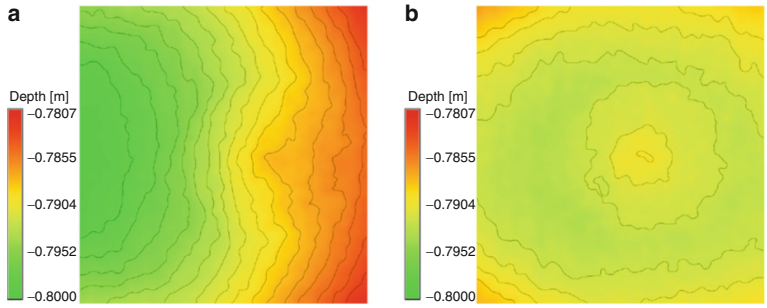


Fig. 2.5 Even slight angular deviation from the vertical scanner axis will cause a tilted scan as seen in (**a**) on a scan of a flat plane with a small angular deviation of 1.7°. A calibration based on the initial scan of a flat plane can compensate for such angular deviations resulting in a horizontally aligned scan as shown in (**b**) although a small radial distortion will still be present

Calibration During scanning the scanner axis should be oriented precisely perpendicular to the table with the model in order to avoid a tilted scan. Even a small deviation of 1° can cause centimeters of height difference depending on the size of the model (see Fig. 2.5a). Therefore, we first calibrate the system by scanning the empty table and computing the angular deviation. We then use this information to automatically rotate each scanned point cloud.

From the scan of the empty table, we compute the plane that best fits the scan by applying the linear regression in 3D. We compute the angular deviation between the unit vector along the scanner axis and the vector perpendicular to the plane. We then use Rodrigues' rotation formula (Wikipedia 2015) to derive a rotation matrix to rotate the point cloud by this angular deviation around a vector given as a cross product of the two vectors. After calibration, the derived rotation matrix is used for rotating every scan resulting in perfectly horizontal scan, see Fig. 2.5b. If the scanner is moved the calibration must be repeated.

For more precise results, we can also minimize the radial distortion of the scanner visible in Fig. 2.5b. The simplest way to do this is to acquire a scan of a flat surface with the dimensions of the physical model and then subtract it from the raster representation of the model.

Model Extraction Since the size of the scanned area cannot be easily limited to just the physical model (see Fig. 2.4), we need to find the boundaries of the model in order to georeference the scanned data.

The first part of the process is to identify points that represent the underlying table. These points have very similar (but due to noise nonidentical) low values of z coordinates. If enough of the table is captured in the scan, we can find the interval of z coordinates containing the points of the table as a first local maximum in a histogram of z values. These points can then be discarded.

After removing these points, we attempt to remove points outside of the model area such as leftover sand or arms reaching over the model in Fig. 2.4. This can be successfully done when the model is rectangular, since we can estimate the horizontal dimensions of the model from the scan and then trim the areas that do not contain enough of points from each side. If the model is not rectangular we can manually choose how many centimeters to trim from each side. Manual trimming, however, requires adjustments anytime that the model is moved even slightly.

Georeferencing Georeferencing the scanned model is an important step when we need to combine it with our geographic data and to ensure that any geospatial analyses are performed on a DEM with real-world dimensions. We need to designate the DEM raster map that the model represents and specify the vertical exaggeration of the physical model to scale the elevation values properly. The scale factors S_x, S_y and S_z are computed:

$$S_x = \frac{Y_{north} - Y_{south}}{y_{max} - y_{min}}, \quad S_y = \frac{X_{north} - X_{south}}{x_{max} - x_{min}}, \quad S_z = \frac{(S_x + S_y)/2}{e} \tag{2.1}$$

where X, Y are DEM (real-world) coordinates, x, y are coordinates on the physical model and e is the specified vertical exaggeration. The vertical scale of the model typically differs from the horizontal because we vertically exaggerate the physical model to enhance our perception of the landscape and simplify our interaction with the model. Since Tangible Landscape users usually interact with the physical model from the side opposite to the scanner we have to rotate the scan by 180°. We can georeference the point cloud with the following equation:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.2}$$

which rotates the points around the z axes by angle α in the counterclockwise direction, scales to real-world dimensions, and translates the points by adding t_x and t_y computed so that the lower left corner of the model matches the south-west corner of the DEM. The vertical translation t_z is then similarly computed to match the lowest point of the model and the minimum height of the DEM.

DEM Processing The plugin allows users to select the method for creating the DEM from the point cloud, which can be either binning or interpolation (see Sect. 4.1). Once the DEM is created all geospatial analyses specified in a dedicated Python file are sequentially performed. Each separate analysis is represented as one function. These functions are reloaded for each new processing loop so that we can easily run different algorithms and change their parameters. This flexible system enables us to dynamically develop and test new workflows during scanning. The plugin provides a library containing often used analyses and workflows which can be readily used and combined with other GRASS GIS functionality using the GRASS GIS Python Scripting Library.

References

Andersen, M., Jensen, T., Lisouski, P., Mortensen, A., Hansen, M., Gregersen, T., and Ahrendt, P. (2012). Kinect Depth Sensor Evaluation for Computer Vision Applications. Technical report, Aarhus University, Department of Engineering. Denmark.

Fuel3D. SCANIFY: The low-cost, high-quality handheld 3D Scanner from Fuel3D [online]. (2015) [Accessed 2015-08-13]. https://www.fuel-3d.com/.

Gonzalez-Jorge, H., Riveiro, B., Vazquez-Fernandez, E., Martínez-Sánchez, J., and Arias, P. (2013). Metrological evaluation of Microsoft Kinect and Asus Xtion sensors. *Measurement*, 46(6):1800–1806.

Lachat, E., Macher, H., Mittet, M.-a., Landes, T., and Grussenmeyer, P. (2015). First Experiences With Kinect V2 Sensor for Close Range 3D Modelling. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W4(February):93–100.

Mutto, C., Zanuttigh, P., and Cortelazzo, G. (2012). Introduction. In *Time-of-Flight Cameras and Microsoft Kinect™*, SpringerBriefs in Electrical and Computer Engineering, pages 1–15. Springer US.

Neteler, M. and Mitasova, H. (2008). *Open source GIS: a GRASS GIS approach*. New York: Springer, third edition.

Newcombe, R. a., Molyneaux, D., Kim, D., Davison, A. J., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). KinectFusion : Real-Time Dense Surface Mapping and Tracking. *IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136.

Wikipedia. Rodrigues' rotation formula — wikipedia, the free encyclopedia [online]. (2015) [Accessed 2015-08-13]. https://en.wikipedia.org/w/index.php?title=Rodrigues%27_rotation_formula&oldid=671556479.

Chapter 3
Building Physical 3D Models

Tangible Landscape works with many types of physical 3D models. When used as a continuous shape display the physical model should be built of a malleable material such as sand, clay, or wax so that users can easily deform the surface. When used for object recognition the physical model can be built of a rigid material such as a wood product, plastic, or resin. When both modes of interaction are combined the physical model should use malleable materials for the base and rigid materials for the objects. These models can be built by hand or digitally fabricated using using 3D printing or computer numerically controlled (CNC) manufacturing (Fig. 3.1).

The final model should be opaque, have a light color, and have a matte finish so that the projected image is crisp and vivid. Transparent materials like acrylic cannot be 3D scanned with a Kinect. Some 3D printing and casting materials like resin may appear opaque, but have translucent properties—this will diffuse the projection. If we desire a very crisp and vivid image on a rigid model made of wood products or resins we recommend painting the model white.

In this chapter we discuss different types of physical models and explain how to fabricate them.

3.1 Handmade Models

Rigid physical models can be handmade by hand-cutting contours and malleable physical models can be handmade by sculpting sand or clay.

Contour Models Contour models can be precise if they are finely cut, but they are inaccurate as they depict abstract, stepped landscapes that are discrete rather than continuous. They are also very legible—one can easily count the contours and read the height—but again they represent abstracted landscapes. While one can easily read the contours and then calculate slope, it is hard to intuitively visualize the slope. Furthermore this abstract, discontinuous representation may obfuscate the

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_3

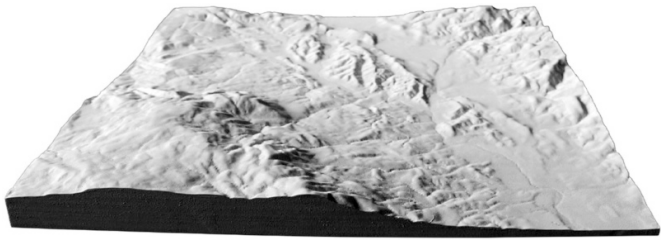


Fig. 3.1 A CNC routed model of the Sonoma Valley, California

relationship between form and process. Furthermore they are time consuming and complicated to construct especially for complex topographies. Hand cutting contour models can be dangerous as the knife may slip or jump.

Contour models can be made out of stacked boards of paperboard, cardboard, foam, or a soft wood like balsa or basswood. The thickness of the sheets should correspond to the vertical interval between contours at the desired map scale. To build a contour model by hand start by plotting a contour map at the desired map scale. Then spray the plotted paper map with a spray mount adhesive, and stick the plotted map onto a board. As an alternative to plotting and mounting, we can instead project the contour map. Cut out the lowest contour level with a precision knife such an x-acto. Glue this contour level onto the base or the level below. Repeat this process until all of the contour levels have been cut out and stacked together.

Hand Sculpted Models Hand sculpted sand and clay models are natural, intuitive, and even fun to make, but are imprecise and hard to read quantitatively (Fig. 3.2). When sculpting by hand we not only see, but also feel the three-dimensional form, using our highly developed kinaesthetic intelligence to better understand the space. While tools can help us to cut sharp edges, shape certain forms, and smooth surfaces, sculpting with our hands alone lets us feel the shape of the topography and get an intuitive understanding of its form undistorted by perspective and depth perception. Because these models are continuous they could—if sculpted well—accurately represent the topography, but as they lack discrete intervals it is challenging to quantitively judge distances and heights. Because it is so natural and easy to make, but challenging to read quantitatively freeform hand modeling is better suited to ideation than detailed design or presentation.

While hand sculpted models have traditionally been made out of sand or clay, we recommend using polymer enriched sand. Clay holds form well, but can be sticky and hard to work with. Sand while easy to move does not hold a form well—sculpt too steep a slope and it will slump, the grains avalanching down. Polymeric sand is easy to work with and holds together well without sticking to our hands. It is not only easy to sculpt, but also easy to cast into precise forms. We will discuss

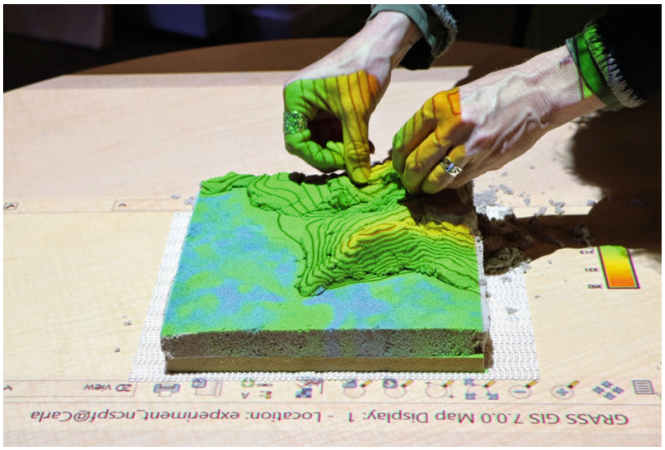


Fig. 3.2 Sculpting a polymeric sand model

methods for casting sand in digitally fabricated molds in Sect. 3.3. We use the product DeltaSand, which is also marketed as Kinetic Sand. It is made by mixing sand with a low-viscosity polymer and a cross-linking agent, resulting in a sand with the plasticity, moldability, adhesiveness, and longevity required for sculpting (Modell and Thuresson 2009).

3.2 Digitally Fabricated Models

Complex physical models for Tangible Landscape can be digitally fabricated with CNC routing or 3D printing. Digitally fabricated models are rigid and precise—while they cannot easily be sculpted, they are ideal presentation models, base models for object recognition, and molds for casting malleable models. CNC machines such as laser cutters, routers, and 3D printers require digital instructions, a controller to decode the instructions, and a machining tool. Machining processes include subtraction with routers, addition with 3D printers, and deformation and molding with vacuum formers (Schodek et al. 2004, Chap. 13). In this section we will explain how to prepare GIS data for digital fabrication and discuss various methods of digital fabrication. Section 3.4 describes workflows for creating models using these methods.

3.2.1 Digital Models

Terrain and other continuous, 3D geographic data can be digitally represented as 3D points, 2.5D rasters, 3D rasters, contours, meshes, or surfaces. In GRASS GIS our workflows for landscape modeling and analysis are raster based. 3D printers and CNC routers, however, require vector data, typically meshes. After acquiring elevation data we import them into GRASS GIS and if necessary we interpolate a raster digital elevation model (DEM). This raster will later be used as the reference data for Tangible Landscape. Next, we export the raster as a point cloud and import it into a 3D modeling program to compute a surface, mesh, or solid from the point cloud. Finally, we generate a toolpath from the surface or mesh for the 3D printer or CNC router.

Digital Elevation Models The first step in modeling a landscape is to acquire data. Entire landscapes can be precisely 3D scanned as point clouds with airborne lidar. As pulses of light are emitted from the aircraft they hit and reflect back from vegetation, structures, and topography. When a pulse of light hits a tree part of the light is reflected and recorded as the first return, while the rest penetrates the outer canopy. The residual pulse, recorded as a series of returns, reflects off of leaves, branches, shrubs, and the ground. Lidar data can be used to compute very precise, high resolution models. By filtering lidar returns and classes we can extract points for the bare earth topography and interpolate a raster DEM or we can extract points with all of the structures and vegetation and interpolate a raster digital surface model (DSM). Lidar point clouds or ready to use raster DEMs are available for many locations and are well suited for building topographic models (see Appendix A.2 for links to lidar data and DEM repositories).

DEMs—raster maps of bare earth topography—and DSMs—raster maps of topography with structures and vegetation—are commonly used in GIS to represent and analyze terrain as they can easily be mathematically transformed. For example topographic parameters such as slope, aspect, and curvature can be computed from the partial derivatives of a function approximating a DEM or DSM (see Chap. 4).

Lidar, especially when filtered by classes or returns, can produce scattered, spatially heterogeneous data points that are challenging to accurately interpolate as terrain surfaces. Furthermore, other sources of elevation data such as digitized contours and surveying data can be clustered and have a highly spatially heterogeneous distribution. The regularized spline with tension (RST) interpolation function, implemented as the *v.surf.rst* module in GRASS GIS, can be used to accurately model terrain surfaces from clustered and heterogeneous data such as lidar with some experimentation and tuning (see Sect. 4.1.2).

Contour data are challenging to interpolate because the data are spatially heterogenous—they are often very dense along the contour lines with large unsampled areas in regions with flat topography. If we acquire digital contour data we can either linearly interpolate between contours with *r.surf.contour* or thin the points along contours using *v.generalize*, convert them to a point representation and then interpolate them using a function like RST with the module *v.surf.rst* (The GRASS GIS Community 2015).

Meshes Terrain can also be modeled in 3D vector representations such as polygon meshes or mathematically defined surfaces. In a mesh a vector network of polygons such as triangles or quadrilaterals forms a shape. Terrain meshes are typically triangulated irregular networks (TINs) that connect the data points from which they are computed. Triangulation was used to manually interpolate contours from surveyed spot elevations before the advent of computers. Now TINs are typically computed using the Delaunay triangulation algorithm which maximizes the smallest angle of every triangle in order to minimize the occurrence of very thin triangles. Since TINs connect the data points the input data is preserved. When meshes are low resolution, i.e. computed from small datasets, they reveal their polygonal structure with angled planes where there should be curved surfaces. When meshes are high resolution, i.e. computed from large datasets, they can be very detailed and represent highly complex forms. Even a high resolution mesh, however, is inaccurate when used to represent curved slopes as planar faces. Meshes are very useful for modeling structures and engineered topography with discontinuous slopes.

NURBS Non-uniform rational B-splines (NURBS) are parametric approximation curves and surfaces defined by a series of polynomials. The curves are basis splines defined by a knot vector, weighted control points, and the curve's order or degree (Piegl and Tiller 1995). A NURBS surface is the tensor product of two NURBS curves (Martisek and Prochazkova 2010). Since NURBS surfaces are mathematically defined they can precisely describe a continuous geometry.

Processed lidar data and rasters can be exported as point clouds from GIS and then imported into a 3D modeling program for meshing or surface generation. Toolpaths for CNC routing can be generated for a mesh or NURBS surface once it has been scaled. 3D printing, however, requires closed, solid volumes so the terrain surface should be offset or extruded to give it a thickness. Most 3D printers read the stereolithography (`.stl`) format, a mesh representation of a solid.

We typically use Rhinoceros for 3D modeling[1] with the RhinoTerrain plugin for terrain modeling[2] and the RhinoCAM plugin for CNC routing.[3] This proprietary 3D modeling program is useful for its interoperability—it can write and read a wide range of formats and can model NURBS, polygon meshes, solids, and point clouds. The RhinoTerrain plugin has modules for importing and exporting geographic data and efficiently computing large terrain meshes. While many computer aided manufacturing (CAM) programs require polygon meshes, the RhinoCAM plugin enables us to generate and simulate CNC toolpaths in Rhinoceros using NURBS, polygon meshes, or solids. The Rhino3DPRINT plugin can be used to prepare models for 3D printing.[4] Alternatives include Maya, a proprietary 3D modeling and animation program,[5] or MeshLab, an open source mesh processing program.[6]

[1] Rhinoceros: http://www.rhino3d.com/

[2] RhinoTerrain: http://www.rhinoterrain.com/

[3] RhinoCAM: http://mecsoft.com/rhinocam-software/

[4] Rhino3DPRINT: http://mecsoft.com/rhino3dprint/

[5] Maya: http://www.autodesk.com/products/maya/overview

[6] MeshLab: http://meshlab.sourceforge.net/

Fig. 3.3 A laser cut chipboard contour model with basswood buildings fabricated by David Koontz and Faustine Pastor

3.2.2 Laser Cutting

Laser cutting can be used to efficiently build precise contour models of moderate complexity (Fig. 3.3). Laser cutters are CNC machines that use a laser moving along a gantry in the x and y axes to make 2D cuts. To laser cut a contour model we separate each contour step onto different layers of a 2D vector drawing in a computer aided design (CAD) format (like `.dwg`, `.dxf`, or `.ai`). We cut each contour step out of a sheet of our material of choice and glue them together. We can also laser etch elevation values, patterns, and line work like road outlines and building footprints onto each layer. While it is easy to precisely cut complex topographies it is still challenging to construct them as we have to lay and glue each piece by hand.

The materials determine the cost. As this is a subtractive method a lot of material is wasted. Contour models cut from thick, low-cost materials like cardboard are relatively inexpensive, but have large contour steps and thus low resolution. Models cut from thin, low-cost materials like chipboard allow for small contour steps at

Fig. 3.4 A CNC routed terrain model of the Jockey's Ridge dune complex on the Outer Banks of North Carolina

moderate cost, but tend to warp with time especially if too much glue has been applied. When aesthetics and archivability are important museum board or acrylic can be cut to create presentation-quality laser cut contour models. Paper-based media such as chipboard or museum board are not suitable for casting polymeric sand as the polymer will stick and soak into them.

3.2.3 CNC Routing

CNC routing or milling is a subtractive fabrication process that can be used to precisely manufacture contour models and surface models of great complexity (Fig. 3.4). This type of digital fabrication is a precise, accurate, inexpensive, and scalable way to build terrain models. Three-axis CNC routers and milling machines move a spindle with a machining bit along the x, y, and z axes in a programmed path to carve a shape out of a block of material (Fig. 3.5). Because 3-axis routers can only carve vertically they produce 2.5D surface models on a solid base, rather than full 3D volumetric models. It is possible, however, to manufacture fully 3D volumes with 3-axis routers; after each cut one can reorient the model and then route the sides or bottom of the model (Schodek et al. 2004, Chap. 14).

To CNC route a terrain model we first prepare a solid block of our material. We can use materials such as medium to high density foam, wax, and wood

Fig. 3.5 A 3-axis CNC router carving a terrain model

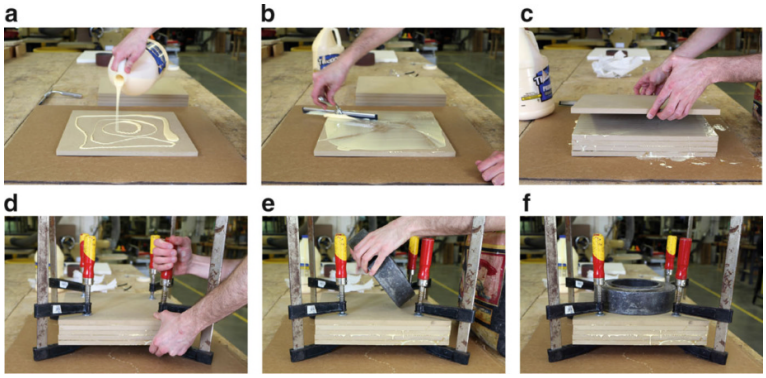


Fig. 3.6 Prepare MDF for CNC routing: (**a**) pour glue onto a layer of MDF, (**b**) spread glue evenly over a layer of MDF, (**c**) stack layers of MDF, (**d**) clamp the stack of MDF, (**e**) add a weight on top of the stack of MDF, and (**f**) wait for the glue to dry

products. We typically use medium density fiberboard (MDF) or baltic birch, engineered wood products with high strength, good dimensional stability, and a fine, uniform grain with minimal voids. To prepare MDF we cut the sheet into tiles that match the desired extent of our scale model, spread wood glue evenly over each tile and stack the tiles together, and firmly clamp the stack of tiles together until the glue sets (Fig. 3.6).

In a CAD program we prepare a digital model of the geometry we wish to carve. Then we generate a toolpath for this digital model in a CAM program. To make a

Table 3.1 CNC settings: horizontal rough cut with MDF

Bit ∅	Speed	Plunge	Approach	Engage	Cut	Retract	Depart.	Step down	Stepover
0.5 in.	16,000	65	135	205	275	800	800	0.32 in.	48 %
0.25 in.	16,000	50	75	125	175	800	800	0.25 in.	40 %

Table 3.2 CNC settings: parallel finish cut with MDF

Bit ∅	Speed	Plunge	Approach	Engage	Cut	Retract	Departure	Stepover
0.5 in.	16,000	65	160	230	325	800	800	20 %
0.25 in.	16,000	50	100	150	225	800	800	20 %

contour model we use closed, 3D contour curves as our data and then carve with a contour cut. Contour cutting can also be used to carve buildings. To carve a surface model from a mesh or NURBS surface we use parallel cuts. For a surface model we typically start with a horizontal rough cut with a 0.5 in. diameter, carbide bit to remove the bulk of the excess material (Table 3.1). Then we use a parallel finish cut with a 0.25 in. diameter, carbide, ball-nose bit to carve the terrain as a continuous surface (Table 3.2). If we need a more refined presentation-quality model we then make two more parallel finish cuts in alternative cutting directions with a 0.125 in. diameter, carbide, ball-nose bit to smooth the surface and capture more detail. Finally a contour cut along the border can be used to neatly trim and remove the model from the base material. The toolpath is written as a sequence of instructions in a CNC programming language often as a `.nc` file in G-Code. The CNC machine's controller reads this code and drives the tool along the toolpath carving our model. We have a streamlined workflow using Rhinoceros with the RhinoCAM plugin for both 3D modeling and the generation, simulation, and visualization of toolpaths. We do our CAD and CAM in the same environment—Rhinoceros—so that we can easily edit our models and work with both meshes and NURBS.

Once the model has been routed we may want to finish it by sanding, priming, and painting the model. Sanding the model, a prerequisite for priming and painting, with a fine grit sandpaper produces a smoother surface. After routing and sanding we clean the model with an air hose. Applying several coats of magnetic primer to our model weakly magnetizes it so that magnetized markers stick to the slopes (Fig. 3.7). A magnetized model is useful for Tangible Landscape applications relying on object recognition rather than sculpting. Finally we may want to spray paint our model white to enhance the brightness of projected imagery (Fig. 3.8).

3.2.4 3D Printing

With 3D printing, a type of solid freeform fabrication, we can make a complex, 3D volume in a single run. While the models can be precise, accurate, and highly complex, they are also expensive and small as 3D printers have restrictively small

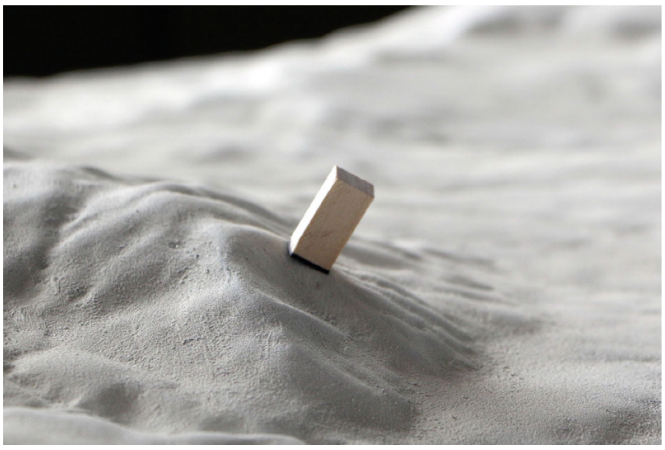


Fig. 3.7 A marker with a magnetic base sticking to a CNC-routed terrain model primed with magnetic paint

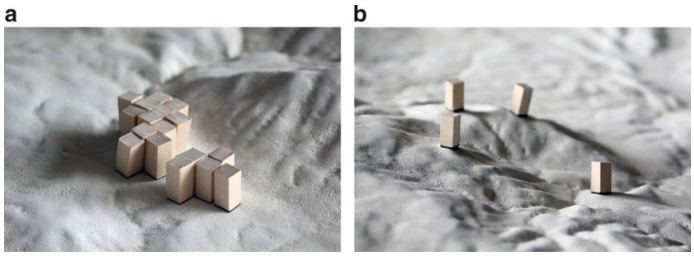


Fig. 3.8 Modes of interaction using markers: (**a**) rasterizing an area and (**b**) drawing a polygon

build areas and the materials are relatively expensive. 3D printing is an ideal process for fabricating small, complex models such as buildings or small, high quality presentation models for use with Tangible Landscape (Fig. 3.9a).

While CNC milling and routing are subtractive processes, 3D printing is an additive process in which a model is built up layer by layer. A digital, 3D, solid model is divided into a stack of horizontal layers or slices and a toolpath is computed for each slice. The physical model is then built slice by slice by depositing or hardening material along the toolpath. By dividing the model into cross sections, each as thin as the technology allows, a complex volume can be formed. There are a variety of different 3D printing processes including stereolithography, selective laser sintering, fused deposition modeling, and 3D ink-jet printing each with tradeoffs in build speed, quality, strength, cost, resolution, color, and material (Schodek et al. 2004, Chap. 14).

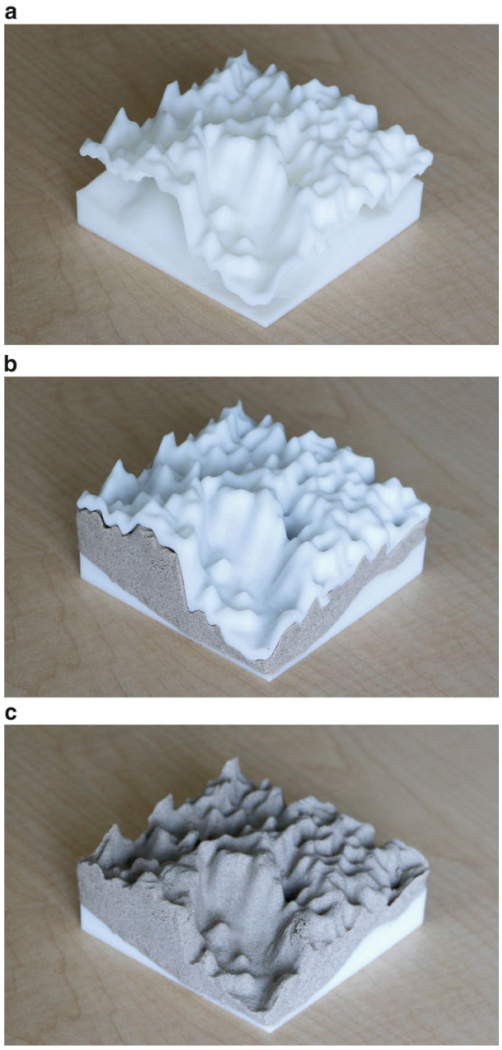


Fig. 3.9 Casting polymeric sand from 3D printed molds. (**a**) 3D prints of the terrain and canopy. (**b**) Casting sand with 3D prints. (**c**) The canopy cast in sand

3.3 Molding and Casting

To sculpt with Tangible Landscape we need a malleable model made of a soft, deformable medium. A model made of polymer enriched sand is easy to sculpt, holds its form well, and can be cast into precise forms. CNC routed and 3D printed models can be used as molds for casting polymeric sand into malleable models.

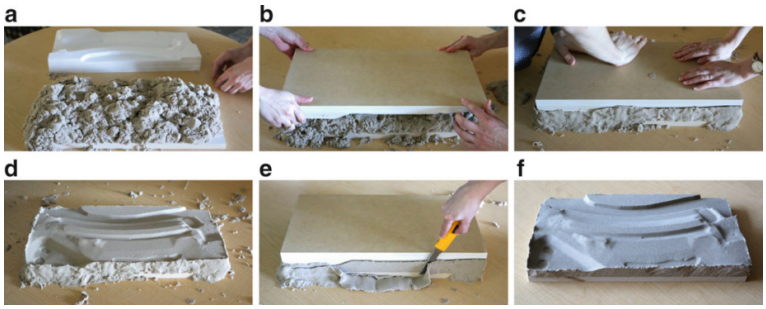


Fig. 3.10 Cast polymeric sand from CNC routed molds. (**a**) Pour polymeric sand onto the base. (**b**) Stack the inverse model on top of the sand. (**c**) Apply pressure. (**d**) Check the cast and remold if necessary. (**e**) Trim the excess sand. (**f**) Remove the inverse model

Fig. 3.11 A thermoformed polystyrene model of Lake Raleigh Woods

The mold should be the inverse of the surface that will be cast. Cast models can precisely represent complex forms that are too challenging to model by hand and can easily be recast.

To cast a terrain model either CNC route the inverse of the terrain (Fig. 3.10) or 3D print the terrain as a volume extruded with enough thickness to survive the casting process (Fig. 3.9). We press the mold into the polymeric sand to cast the form. We have to check the cast and repeat the process if necessary—sand may need to be added or removed in places to get a good cast. Clamps can be used to get strong, even pressure on a CNC routed mold when casting.

Thermoforming or vacuuming forming can be used to quickly make lightweight copies or imprints of a CNC routed terrain model in a thermoplastic. Since they are lightweight thermoformed models are very portable and are ideal for casting polymeric sand while traveling (Fig. 3.11). To thermoform a terrain model we heat a thermoplastic sheet in a vacuum former until it becomes soft. Use the vacuum to pull the hot plastic over a mold deforming the plastic into the desired shape. Once

the plastic cools into the cast shape we release the vacuum. We may need to drill small holes through our mold to make a complete vacuum (Schodek et al. 2004, Chap. 14).

3.4 Workflows

This section describes workflows for creating physical models for Tangible Landscape. Our examples use lidar point cloud data for Lake Raleigh Woods on North Carolina State University's (NCSU) Centennial Campus acquired during a 2013 lidar survey by Wake County. The data are provided as a single tile in the standard LAS format (`tile_0793_015_spm.las`). We processed this data within a GRASS GIS data set `nc_spm_tl`. All examples in the following subsections assume running GRASS GIS with this data set; see Appendix A.3 for information on how to download the data, start GRASS GIS with this location data set, and perform basic display and other operations (Fig. 3.12).

3.4.1 Selecting a 3D Model Scale

The scale of the physical 3D model depends on the extent of the data, the desired model size, and the desired scale. The scale may also depend on the material of the model, the scanning technology, and the resolution of the data. Here we assume that we know the spatial extent of our data and want our model to be approximately half a meter long on each side. If we do not already know the extent of our data we can determine the values once the data has been imported into GRASS GIS for example using *v.info* or *r.info*.

We can perform this computation in the Python interactive console. First, we enter the spatial extent of the data and approximate the size of the model (in meters):

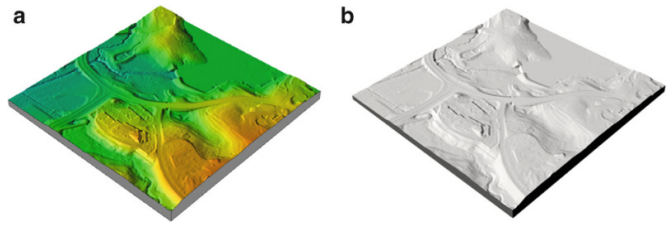


Fig. 3.12 3D renderings of terrain models derived from lidar tile 0793_015 using (**a**) GRASS GIS and (**b**) Rhinoceros with the RhinoTerrain plugin (see Sect. 3.4.4)

```
n = 224028.45
s = 223266.45
e = 639319.28
w = 638557.28
desired_model_x = 0.50
```

Then we compute the scale using the equation:

$$s = \frac{d_m}{d_r} \tag{3.1}$$

where s is the scale of the model, d_m is the distance measured on the model, and d_r is the real-world distance. For convenience we can also compute the scale number using the equation:

$$s_n = \frac{1}{s} \tag{3.2}$$

where s_n is the scale number of the model and s is the scale of the model.

We then type the following in the Python console:

```
scale = desired_model_x / (n - s)
scale_number = 1. / scale
```

The scale is now 1:1524 and the scale number is 1524. Since we want the scale to be a round number and do not want to add or remove data we need to modify the scale and reverse the computation in the previous step so that we can compute the new size of the model at the selected scale. We select a scale of 1:1500 (a round value close to the previously computed scale). In Python we then write:

```
scale_number = 1500
scale = 1. / scale_number
model_x = (n - s) * scale
```

The resulting model size is 50.8 cm.

So far we have only used the spatial extent along the north-south direction. We skipped the east-west direction because the extent of both the data and the model were squares. In a region with an elongated rectangular shape we might need to determine a suitable scale for each of the horizontal directions and then decide on a compromise.

Next we determine the vertical size of the model and its vertical exaggeration. First, we enter the minimum and maximum elevation in the dataset (called top and bottom in GRASS GIS) and we also include the desired height of the model (in meters). In our example we set the desired height of the model to 0.04 m:

```
t = 109.33
b = 76.54
desired_model_h = 0.04
```

Then we compute the model height without vertical exaggeration and the exaggeration based on the desired model height using the equation:

$$e = \frac{h_e}{h} \tag{3.3}$$

where e is the exaggeration, h_e is the exaggerated height (in our case the desired height of the model) and h is the height of the model according to the scale in case the model would not be exaggerated.

Since we already know the scale of the model from the previous computations we can now compute the exaggeration and model height without vertical exaggeration:

```
model_h = scale * (t - b)
exaggeration = desired_model_h / model_h
```

With the given the inputs the exaggeration is now approximately 1.8. Since we want a round number we set the vertical exaggeration to 2 and compute the actual height of the model using Eq. (3.3):

```
exaggeration = 2
actual_model_h = exaggeration * model_h
```

With the exaggeration set to 2 the model height (or more precisely the maximum height difference of the top surface of the model) will be approximately 4.4 cm. The base of the model will increase the actual model height. Depending on how it is constructed the base may or may not be an integral part of the model. If we set the exaggeration to 1.5 the model height difference would be approximately 3.3 cm, which might be too shallow. At a different scale, however, an exaggeration of 1.5 might be the right choice.

We have used the same scale in both the x and y horizontal directions (1:1500). However, in the vertical direction (z) we exaggerated the scale by a factor of 2. As a result the vertical scale (1:750) is different than horizontal scale (1:1500). Depending on the audience it may be advantageous to note either the horizontal and vertical scales or the scale and exaggeration. When the scale and exaggeration are known the vertical scale can be computed using the equation:

$$s_v = se \tag{3.4}$$

where s_v is the vertical scale, s is the horizontal scale and e is the exaggeration. Once we have computed the size of our model we can build the model using any of the methods described in the following sections.

3.4.2 Sculpting a Malleable Model from Data

A simple way to create a malleable 3D model is to sculpt the landscape in polymer-enriched sand using projected contours derived from DEM as a guide.

In this workflow we used a point cloud of elevation data in the LAS format (`tile_0793_015_spm.las`) and worked in the GRASS GIS location `nc_spm_tl`. To sculpt a malleable model from lidar data perform the following steps:

Import the lidar points classified as ground (standard class 2) into GRASS GIS using the module *v.in.lidar*:

```
v.in.lidar -t input=tile_0793_015_spm.las output=ground \
    class_filter=2
```

Set the region to the spatial extent of the imported tile and resolution to 1 m. Then interpolate the DEM from the processed lidar data:

```
g.region vector=ground res=1
v.surf.rst -t input=ground elevation=dem tension=100 npmin=250 \
    dmin=2
```

Check the results, adjust the parameters, and rerun with the overwrite flag if necessary. Then compute the 1 m interval contours from the DEM:

```
r.contour input=dem output=contours_1m step=1
```

We can now sculpt the terrain in polymer-enriched sand using the projected DEM and contours as a guide (Fig. 3.2). Once we have made at least a rough approximation of the form, we can use DEM differencing to critique and refine the modeled form (see Sect. 4.2.2).

3.4.3 CNC Routing a Topographic Model from Contour Data

Import the contours provided in the file `contours.dxf` into Rhinoceros:

```
_Import contours.dxf _Enter
```

Zoom to the extent of all viewports:

```
Zoom All Extents
```

Select the contours and interpolate a terrain mesh:

```
_RtMeshTerrainCreate _Accept _Enter
```

Scale the model uniformly in the x, y, z direction to our chosen map scale by selecting the mesh, setting the origin to 0,0,0 and setting the scale factor:

```
_Scale 0,0,0 1/450 _Enter
```

Optionally scale the model in the z-direction to exaggerate the relief by selecting the mesh, setting the origin to 0,0,0, setting the scale factor to 2, and setting the scale direction by drawing a line in the z-axis with ortho mode, using the gumball, or entering the coordinates 0,0,0 and 0,0,1:

```
_Scale1D 0,0,0 2 0,0,0 0,0,1 _Enter
```

Use RhinoCAM to generate the toolpath. Then CNC route the model.

3.4.4 CNC Routing Topographic and Surface Models from Lidar Data

Import the lidar data classified as ground (standard class 2) into GRASS GIS:

```
v.in.lidar -t input=tile_0793_015_spm.las output=ground \
    class_filter=2
```

Import the first return lidar data into GRASS GIS filtering unnecessary points:

```
v.in.lidar -t input=tile_0793_015_spm.las output=surface \
    return_filter=first class_filter=1,2,3,4,5,6,9
```

Set the region and resolution and interpolate the DEM from the ground points:

```
g.region vector=ground res=1
v.surf.rst -t input=ground elevation=dem tension=100 npmin=250 \
    dmin=2
```

Check the results, adjust the parameters, and rerun with the overwrite flag if necessary. With the region already set in the previous step, interpolate the DSM from the surface points:

```
v.surf.rst -t input=surface elevation=dsm tension=200 \
    smooth=0.5 npmin=120 dmin=2
```

Check the results, adjust the parameters, and rerun with the overwrite flag if necessary. Export the DEM to a space delimited text file with x, y, and z values:

```
r.out.xyz input=dem output=dem.xyz separator=space
```

Export the DSM to a space delimited text file with x, y, and z values:

```
r.out.xyz input=dsm output=dsm.xyz separator=space
```

Import the DEM text file into Rhinoceros as a point cloud. In the import options, set delimiter to ‘space’ and check ‘create point cloud’:

```
_Import dem.xyz
```

Import the DSM text file as a point cloud. In the import options, set delimiter to ‘space’ and check ‘create point cloud’:

```
_Import dsm.xyz
```

Zoom to the extent of all viewports:

```
Zoom All Extents
```

Select the DEM point cloud, interpolate a terrain mesh, and delete or hide the point cloud:

```
_RtMeshTerrainCreate _Accept _Enter
```

Select the DSM point cloud and interpolate a terrain mesh. Delete or hide the point cloud:

```
_RtMeshTerrainCreate _Accept _Enter
```

Scale both the DEM and DSM meshes uniformly in the x, y, z direction to our chosen map scale by selecting the meshes, setting the origin to 0,0,0 and setting the scale factor:

```
_Scale 0,0,0 1/1500 _Enter
```

Optionally scale both the DEM and DSM meshes in the z-direction to exaggerate the relief by selecting the meshes, setting the origin to 0,0,0, setting the scale factor to 2, and setting the scale direction by drawing a line in the z-axis with ortho mode, using the gumball, or entering the coordinates 0,0,0 and 0,0,1:

```
_Scale1D 0,0,0 2 0,0,0 0,0,1 _Enter
```

Optionally, create a vector curve delineating the border of the DEM or DSM:

```
_RtMeshFindBorder _Enter
```

Use RhinoCAM to generate the toolpaths. CNC route the models.

3.4.5 3D Printing Topographic and Surface Models from Lidar Data

Import the lidar data classified as ground (standard class 2) into GRASS GIS:

```
v.in.lidar -t input=tile_0793_015_spm.las output=ground \
    class_filter=2
```

Set the region to the given tile and set a relatively low resolution of 10 m. Interpolate the DEM from the ground points:

```
g.region vector=ground res=10
v.surf.rst -t input=ground elevation=dem tension=100 npmin=250
```

Export the DEM to a space delimited text file with x, y, z values:

```
r.out.xyz input=dem output=dem.xyz separator=space
```

Import the DEM text file into Rhinoceros as a point cloud. In the import options, set delimiter to 'space' and check 'create point cloud':

```
_Import dem.xyz _Enter
```

Zoom to the extent of all viewports:

```
Zoom All Extents _Enter
```

Select the DEM point cloud and interpolate a terrain mesh. Delete or hide the point cloud:

```
_RtMeshTerrainCreate _Accept _Enter
```

Scale the mesh uniformly in the x, y, z direction to our chosen map scale by selecting the mesh, setting the origin to 0,0,0 and setting the scale factor:

```
_Scale 0,0,0 1/3000 _Enter
```

Optionally scale the mesh in the z-direction to exaggerate the relief by selecting the mesh, setting the origin to 0,0,0, setting the scale factor to 2, and setting the scale direction by drawing a line in the z-axis with ortho mode, using the gumball, or entering the coordinates 0,0,0 and 0,0,1:

```
_Scale1D 0,0,0 2 0,0,0 0,0,1 _Enter
```

Create a vector curve delineating the border of the DEM:

```
_RtMeshFindBorder _Enter
```

Prepare the mesh for 3D printing. Select the border curve as the boundary and the DEM mesh as the mesh. Set offset to relative or absolute. Set a base height:

```
_RtMesh3dPrintPrepare
```

Export the selected mesh as a stereolithography file (`.stl`) and send it to the 3D printer:

```
_SaveAs
```

3.4.6 Casting a Malleable Topographic Model with a CNC Routed Mold Derived from Lidar Data

Import the lidar data classified as ground (standard class 2) into GRASS GIS:

```
v.in.lidar -t input=tile_0793_015_spm.las output=ground \
    class_filter=2
```

Set the region and resolution, then interpolate the DEM from the ground points:

```
g.region vector=ground res=1
v.surf.rst -t input=ground elevation=dem tension=100 npmin=250 \
    dmin=2
```

Check the results, adjust the parameters, and rerun with the overwrite flag if necessary. Export the DEM to a space delimited text file with x, y, z values:

```
r.out.xyz input=dem output=dem.xyz separator=space
```

Import the DEM text file into Rhinoceros as a point cloud. In the import options, set delimiter to 'space' and check 'create point cloud':

```
_Import dem.xyz _Enter
```

Zoom to the extent of all viewports:

```
Zoom All Extents _Enter
```

Select the DEM point cloud and interpolate a terrain mesh. Delete or hide the point cloud:

```
_RtMeshTerrainCreate _Accept _Enter
```

Optionally, create a vector curve delineating the border of the DEM:

```
_RtMeshFindBorder _Enter
```

Invert the DEM by rotating it 180° on the z axis. Set the 'first reference point' by drawing a vertical line with ortho mode on and then set the 'second reference point' to '180' or by or entering the coordinates 0,0,0 and 0,0,1 and then 180 in the command line:

```
_Rotate 0,0,0 0,0,1 180 _Enter
```

Scale the mesh uniformly in the x, y, z direction to our chosen map scale by selecting the mesh, setting the origin to 0,0,0 and setting the scale factor:

```
_Scale 0,0,0 1/1500 _Enter
```

Optionally scale the mesh in the z-direction to exaggerate the relief by selecting the mesh, setting the origin to 0,0,0, setting the scale factor to 2, and setting the scale direction by drawing a line in the z-axis with ortho mode, using the gumball, or entering the coordinates 0,0,0 and 0,0,1:

```
_Scale1D 0,0,0 2 0,0,0 0,0,1 _Enter
```

Optionally, create a vector curve delineating the border of the DEM or DSM:

```
_RtMeshFindBorder _Enter
```

Use RhinoCAM to generate the toolpath, CNC route the inverted terrain model, and use this routed model as a mold to cast polymeric sand into a solid, malleable model of the topography.

References

Martisek, D. and Prochazkova, J. (2010). Relation between algebraic and geometric view on nurbs tensor product surfaces. *Applications of Mathematics*, 55(5):419–430. Copyright - Institute of Mathematics of the Academy of Sciences of the Czech Republic, Praha, Czech Republic 2010; Last updated - 2014-08-22.

Modell, J. and Thuresson, S. (2009). Material composition and method for its manufacture. EP Patent App. EP20,070,794,114.

Piegl, L. and Tiller, W. (1995). *The NURBS book*. Springer, New York.

Schodek, D., Bechthold, M., Griggs, K., Kao, K. M., and Steinberg, M. (2004). *Digital Design and Manufacturing*. John Wiley & Sons, Inc., Hoboken, New Jersey.

The GRASS GIS Community. Contour lines to DEM [online]. (2015) [Accessed 2015-05-28]. http://grasswiki.osgeo.org/wiki/Contour_lines_to_DEM.

Chapter 4
Basic Landscape Analysis

Tangible Landscape allows us to explore the spatial patterns of topographic parameters and their relation to basic surface geometry. We can easily sculpt a physical model of a landscape with our hands changing its topography. With Tangible Landscape we can analyze the topography of a landscape model and how it changes by continually 3D scanning the model and computing DEMs from the scanned point clouds using binning or interpolation. By computing basic topographic parameters, morphometric units, and DEM differencing we can map changes in elevation, slope, and landform as the model is modified. These maps are then projected over the physical model of the landscape so that we have near real-time feedback and can understand the impact of the changes we are making as we make them.

4.1 Processing and Analyzing the Scanned DEM

To start working with Tangible Landscape, the setup is first calibrated using a flat surface, typically a table, to minimize the scanning distortion (see Sect. 2.2.4). After successful calibration the model can be scanned continuously. With each scan the edges of the model are detected and georeferenced and the point cloud is converted into a DEM via binning or interpolation. Standard GRASS GIS modules are then used to compute maps of topographic parameters or landforms that are then projected over the model. This chapter explains how the DEM is derived and demonstrates the results in a case study.

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_4

4.1.1 Creating DEM from Point Cloud

There are two ways to derive a raster DEM from a set of points in Tangible Landscape:

Binning A cell value is assigned based on the univariate statistics of the points that fall inside the particular cell. When constructing a DEM the mean of the z coordinates is typically used. Binning is fast, but creates a rough surface so it is only useful for certain types of analyses.

Interpolation Cell values are estimated from available point data using a selected mathematical or statistical interpolation technique.

4.1.2 Interpolation with the RST Function

The point clouds produced by the scanning process are usually noisy. The regularized spline with tension (RST) function, implemented as the *v.surf.rst* module in GRASS GIS, can be used to interpolate smooth DEMs from noisy point data and simultaneously compute topographic parameters such as slope and curvature. RST approximates a surface from data points by minimizing a smoothness seminorm and the deviations between the given points and the resulting surface (Mitasova et al. 2005). The RST smoothness seminorm includes derivatives of all orders with their weights decreasing with the increasing derivative order leading to the function:

$$z(\mathbf{r}) = a_1 + \sum_{j=1}^{N} \lambda_j R(\varrho_j) \tag{4.1}$$

$$R(\varrho_j) = -[E_1(\varrho_j) + \ln(\varrho_j) + C_E] \tag{4.2}$$

where $z(\mathbf{r})$ is the elevation at a point $\mathbf{r} = (x, y)$, a_1 is a trend, λ_j are coefficients, N is a number of given points, $R(\varrho_j)$ is a radial basis function, $\varrho_j = (\varphi r_j/2)^2$, φ is a generalized tension parameter, $r_j = |\mathbf{r} - \mathbf{r}_j|$ is a distance, $C_E = 0.577215$ is the Euler constant, and $E_1(\varrho_j)$ is the exponential integral function (Abramowitz and Stegun 1965; Mitasova and Mitas 1993). The coefficients a_1, $\{\lambda_j\}$ are obtained by solving the system of linear equations:

$$\sum_{j=1}^{N} \lambda_j = 0 \tag{4.3}$$

$$a_1 + \sum_{j=1}^{N} \lambda_j \left[R(\varrho_j) + \delta \frac{w_0}{w_j} \right] = z(\mathbf{r}_i), \qquad i = 1, \ldots, N \tag{4.4}$$

where w_0/w_j are positive weighting factors representing a smoothing parameter at each given point $\mathbf{r_j} = (x_j, y_j)$.

Tension influences the detail of the surface and smoothing influences the deviations between the given points and the resulting surface. A higher smoothing value can be used to reduce the noise in the data. Because this is a global operation acting on the entire dataset it can be processing intensive for large datasets so the *v.surf.rst* implementation uses a quadtree segmentation algorithm with smooth overlaps to separately process overlapping neighborhoods of points. A quadtree is a tree data structure that recursively partitions processing regions into smaller processing regions by dividing them into quadrants. Depending on the distribution of the input points in relation to the surface complexity, interpolation with *v.surf.rst* may require experimental, iterative tuning of the parameters. By adjusting the parameters which control the RST function properties and distribution of points used for interpolation, this function can be used to accurately model smooth or rough topography (Mitasova and Mitas 1993; Mitasova et al. 2005).

4.1.3 Analyzing the DEM

The geometry of an elevation surface at any point can be described by topographic parameters—slope, aspect, and several types of curvatures (Mitasova et al. 2005; Olaya 2009). Elevation surface can be partitioned into units with specific geometric properties; one approach to defining these units is geomorphons.

Basic Topographic Parameters The steepest slope angle β in degrees or percent and the aspect angle α in degrees are the most commonly used topographic parameters topographic parameters (Fig. 4.2). They represent the magnitude and direction of the surface gradient vector $\nabla z = (f_x, f_y)$ (its direction is upslope) and are computed as follows:

$$\beta = \arctan\sqrt{f_x^2 + f_y^2} \qquad \beta[\%] = 100\sqrt{f_x^2 + f_y^2} \tag{4.5}$$

$$\alpha = \arctan -\frac{f_y}{f_x} \tag{4.6}$$

where $f_x = \partial z/\partial x$ and $f_y = \partial z/\partial y$ are the first order partial derivatives of elevation surface function $z = f(x, y)$. The slope values range from 0° to 90° and the aspect values range from 0° to 360°. Therefore, computing the correct angle for aspect requires the evaluation of all possible combinations of negative, positive and zero values of f_x,f_y in relation to the selected direction of 0° (usually east or north).

For applications in geosciences the curvature in the gradient direction (profile curvature) is important because it reflects the change in the slope angle and thus controls the change of the velocity of mass flowing downwards along the slope curve. The equation for the profile curvature is

$$\kappa_s = \frac{f_{xx}f_x^2 + 2f_{xy}f_xf_y + f_{yy}f_y^2}{p\sqrt{q^3}} \tag{4.7}$$

where κ_s is the profile curvature in m^{-1} and f_{xx}, f_{xy}, f_{yy} are the second order partial derivatives of the elevation surface function $z = f(x, y)$, $p = f_x^2 + f_y^2$ and $q = p + 1$. The curvature in a direction perpendicular to the gradient (tangential curvature) reflects the change in the aspect angle and influences the divergence/convergence of water flow. The equation for tangential curvature at a given point is

$$\kappa_t = \frac{f_{xx}f_y^2 - 2f_{xy}f_xf_y + f_{yy}f_x^2}{p\sqrt{q}} \tag{4.8}$$

where κ_t is the tangential curvature in m^{-1}. Partial derivatives can be computed from a suitable approximation function $z = f(x, y)$ such as local second order polynomial or certain types of splines.

Partial Derivatives from a Raster DEM We can use a second order polynomial approximation of a surface defined by a given grid point and its 3×3 neighborhood to estimate the topographic parameters (Horn 1981):

$$z(x, y) = a_0 + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2 \tag{4.9}$$

By fitting this polynomial to the nine grid points (the given point $z_{i,j}$ and its 3×3 neighborhood as shown below in Fig. 4.1) using weighted least squares we can derive the coefficients of this polynomial as well as its partial derivatives ($f_x = a_1$, $f_y = a_2$, $f_{xx} = 2a_4$, $f_{yy} = 2a_5$, $f_{xy} = a_3$):

$$f_x = \frac{(z_{i-1,j-1} - z_{i+1,j-1}) + 2(z_{i-1,j} - z_{i+1,j}) + (z_{i-1,j+1} - z_{i+1,j+1})}{8\Delta x} \tag{4.10}$$

$z_{i-1,j+1}$	$z_{i,j+1}$	$z_{i+1,j+1}$
$z_{i-1,j}$	$z_{i,j}$	$z_{i+1,j}$
$z_{i-1,j-1}$	$z_{i,j-1}$	$z_{i+1,j-1}$

Fig. 4.1 The elevation values of a grid cell and its 3×3 neighborhood

$$f_y = \frac{(z_{i-1,j-1} - z_{i-1,j+1}) + 2(z_{i,j-1} - z_{i,j+1}) + (z_{i+1,j-1} - z_{i+1,j+1})}{8\Delta y} \tag{4.11}$$

where Δx and Δy is the resolution (grid spacing) in the east-west and north-south directions respectively. To compute the second order partial derivatives we first denote $D(i,\delta) = z_{i,j+1} + z_{i,j-1} - 2z_{i,j}$ and $D(\delta,j) = z_{i+1,j} + z_{i-1,j} - 2z_{i,j}$. Then we can write:

$$f_{xx} = \frac{D(\delta,j+1) + (4z_{i-1,j} + 4z_{i+1,j} - 8z_{i,j}) + D(\delta,j-1)}{6(\Delta x)^2} \tag{4.12}$$

$$f_{yy} = \frac{D(i-1,\delta) + 4(z_{i,j+1} + z_{i,j-1} - 2z_{i,j}) + D(i+1,\delta)}{6(\Delta y)^2} \tag{4.13}$$

$$f_{xy} = \frac{(z_{i-1,j-1} - z_{i+1,j-1}) - (z_{i-1,j+1} - z_{i+1,j+1})}{4\Delta x\Delta y} \tag{4.14}$$

where $z_{i,j}$ is the elevation value at row j and column i, Δx is the east-west grid spacing, and Δy is the north-south grid spacing (resolution).

Landforms Landforms can be identified automatically by fitting a quadratic function to the elevation values in a given neighborhood or 'moving window' using least squares. This method for identifying landforms using differential geometry is implemented in GRASS GIS as a module *r.param.scale* (Fig. 4.3). While real-world landforms are scale-dependent and may be nested, this method can only identify landforms at a single scale based on the size of the moving window.

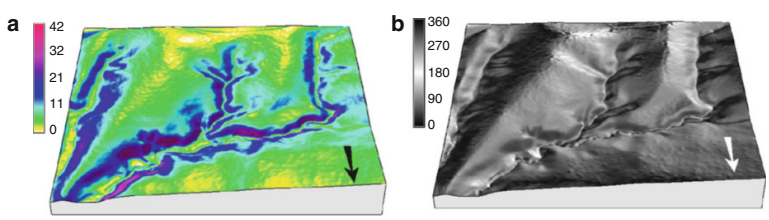


Fig. 4.2 Slope (**a**) and aspect (**b**) maps in degrees draped over a DEM

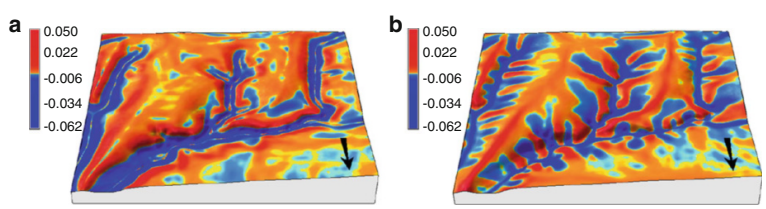


Fig. 4.3 Profile (**a**) and tangential (**b**) curvature maps draped over a DEM

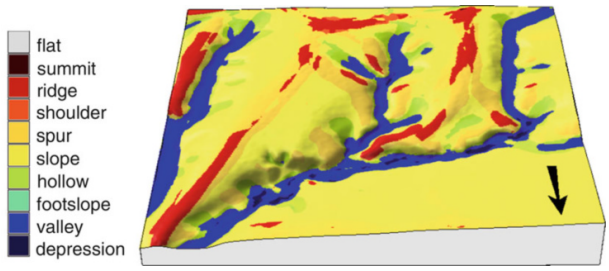


Fig. 4.4 Landforms identified by geomorphons (see *r.geomorphon* manual page for explanation of landform types)

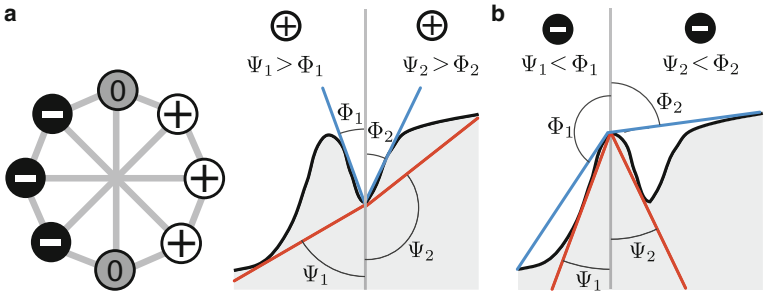


Fig. 4.5 Geomorphons: local ternary patterns and line-of-sights concepts applied to landform identification. The relationship between a cell and its neighbors in terms of relative elevation (higher than, equal to, or lower than) is expressed as a ternary pattern (**a**). Ternary patterns are derived using the line-of-sight method and comparing the zenith angle Φ and nadir angle Ψ (**b**)

Geomorphons—geomorphologic phonotypes—is a novel method for identifying landforms using pattern recognition developed by Jasiewicz and Stepinski (2013) and implemented in GRASS GIS as the add-on module *r.geomorphon* (Fig. 4.4). A geomorphon is an abstract unit of terrain described by the local ternary pattern rather than by relief. The relationship between a cell and its neighborhood is described using an 8-tuple pattern of lower (−), equal (0), higher (+) elevation values (Fig. 4.5a). Neighborhoods are based on visibility rather than the predefined dimensions of a moving window. Geomorphons uses the line-of-sight principle to dynamically determine the optimal size of the search radius for a neighborhood based on openness of the terrain (see Fig. 4.5b). Thus, unlike methods based on differential geometry, the geomorphons approach is able to efficiently classify landforms across a range of spatial scales (Jasiewicz and Stepinski 2013).

4.2 Case Study: Topographic Analysis of Graded Landscape

In this section we demonstrate how changes in elevation impact topographic parameters and landforms in a small study area on NCSU's Centennial Campus.

4.2.1 Site Description and Model

Our 56 ha case study area is located on NCSU's Centennial Campus on the southwest side of Lake Raleigh (Fig. 4.6). The topography is largely natural, but there are some anthropogenic features, such as informal trails and the Chancellor's house with its access road, parking, and retention pond. There is rapid development on the campus; several new buildings are currently under construction. At the same time there is a desire to preserve part of the campus as a park and a natural teaching area. In this case study we developed part of the natural area as student housing and constructed an access road. Since significant grading would be required to develop the site given the steep slopes we extensively modified the topography and mapped the change in elevation (cut and fill), slope, aspect, curvature, and landform.

We CNC routed the base and mold for the model from MDF based on a DEM derived from a 2013 lidar survey. We used the base and the mold to cast a sculptable sand model. Each side of the 1:1500 scale model is 50 cm and represents 750 m. The model is two times vertically exaggerated and has 44 mm of relief which represent 33 m of elevation change.

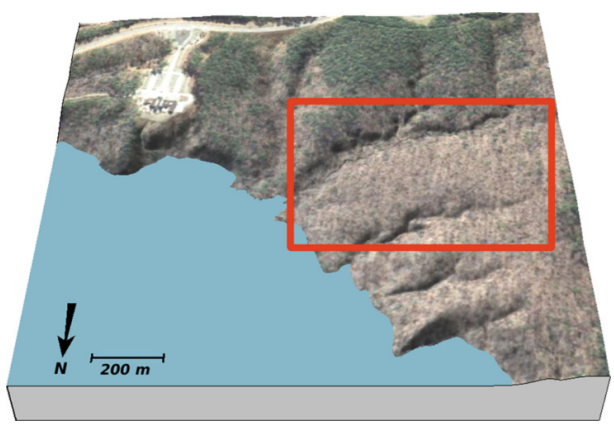


Fig. 4.6 Lake Raleigh Woods with our study site highlighted

We sculpted the sand model to grade a road and building sites. With Tangible Landscape we were able to analyze these modifications and understand how we had changed the slope, aspect, shape, and volume of the landscape.

4.2.2 Basic Workflow with DEM Differencing

To explore the impact of the new development we graded building sites and a road for the future neighborhood. We start with the original molded sand model, scan it and save the scan under different name so that it is not overwritten by the next scan. Then we compute and display 1 m countours:

```
g.rename raster=scan,scan_before
g.region raster=scan_before
r.contours input=scan_before output=contours_1m step=1
```

Figure 4.7a shows the terrain conditions before grading. While sculpting the model, we can pause the continuous scanning or keep it running to get ongoing feedback on our design as we progress.

We can visualize the magnitude and extent of our changes by differencing the scanned elevation before and after the change and then assigning a diverging color scheme to helps us distinguish between cut and fill (Fig. 4.7d):

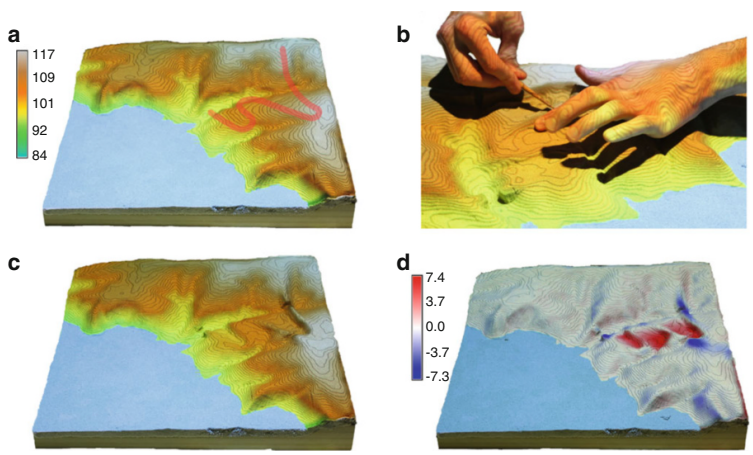


Fig. 4.7 Basic workflow: (**a**) the scanned elevation with contours, a *red curve* indicates the planned road; (**b**) modifying the terrain; (**c**) the finished design; (**d**) and the difference in meters between the elevation before and after grading, *red* representing soil removal (cut) and *blue* representing soil deposition (fill)

```
g.rename raster=scan,scan_after
r.mapcalc "diff = scan_after - scan_before"
r.colors -n map=diff color=differences
```

Because the scans are noisy, they do not align precisely in vertical direction. When computing difference between scans, any vertical shift is undesirable and should be minimized. The following workflow uses linear regression to match one scan to the other on the assumption that there has only been a moderate change in elevation. The linear regression of the form $y = a + bx$ estimates the vertical shift a and scale coefficient b (which should be close to 1) for adjusting the scan. Here we define a function in Python for adjusting the scan to match the other; we will refer to this function throughout the book:

```
def adjust_scan(scan_before, scan_after, scan_adjusted):
   coeff = gscript.parse_command('r.regression.line',
       mapx=scan_after, mapy=scan_before, flags='g')
   gscript.mapcalc("{scan_adjusted} = {a} + {b} *
       {scan_after}".format(scan_adjusted=scan_adjusted,
       a=coeff['a'], b=coeff['b'], scan_after=scan_after))
```

Here we use the function for adjusting scans to compute the difference:

```
adjust_scan('scan_before', 'scan_after', 'scan_adjusted')
gscript.mapcalc("diff = scan_adjusted - scan_before")
gscript.run_command('r.colors', flags='n', map='diff',
    color='differences')
```

4.2.3 The Impact of Model Changes on Topographic Parameters

In our case study the new neighborhood is planned on an east-facing hillside with a gentle slope of about 5° (Fig. 4.8). The slope values are steep on two sides of the hillside where it changes into parallel valleys; the profile curvature increases at the edge of the valleys (Fig. 4.9). The profile curvature on the hillside varies between positive and negative values denoting convex (ridges) and concave shapes (valleys).

We computed the slope and aspect using the *r.slope.aspect* module based on the standard polynomial approximation using 3×3 window, see Eqs. (4.9)–(4.14). When computing curvature, we use a more flexible method with a larger window (21×21 grid cells) to capture the larger scale morphology associated with the main valleys and ridges at our study site. This method is implemented in the module *r.param.scale* and we used this module to compute profile and tangential curvature. To visualize the variability in the curvature values we used a dedicated color ramp with divergent scheme:

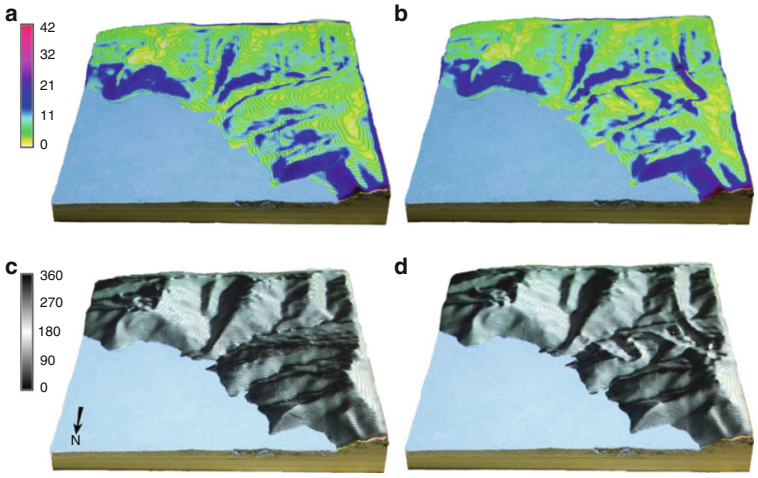


Fig. 4.8 Slope (**a**, **b**) and aspect (**c**, **d**) in degrees computed from the scanned model before (**a**, **c**) and after (**b**, **d**) modification

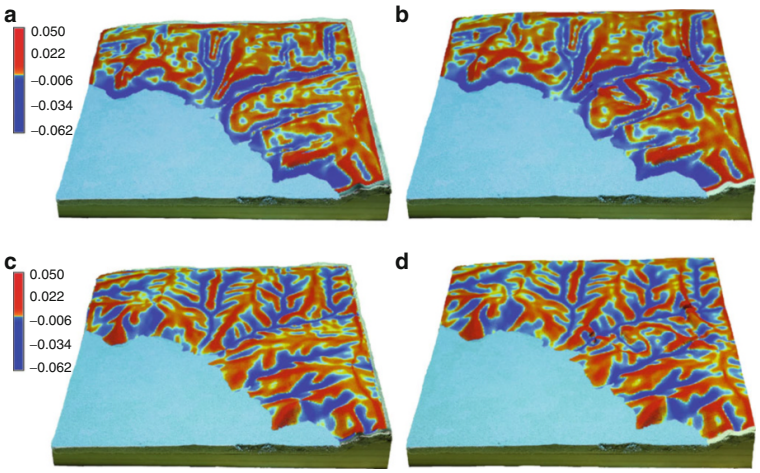


Fig. 4.9 Profile (**a**, **b**) and tangential (**c**, **d**) curvature computed from the scanned model before (**a**, **c**) and after (**b**, **d**) modification

```
r.slope.aspect elevation=scan aspect=scan_aspect \
    slope=scan_slope
r.param.scale input=scan output=scan_pcurv size=21 method=longc
r.param.scale input=scan output=scan_tcurv size=21 method=crosc
r.colors map=scan_pcurv color=curvature
r.colors map=scan_tcurv color=curvature
```

To build a new road with gentle slopes we constructed a series of switchbacks with low areas raised on embankments. As a result there are steep slopes beside the road on the faces of the embankments (Fig. 4.8). We also flattened parts of the hillside as building sites. When we analyzed the curvature of the modified landscape the flat roads were mapped as concave areas with high curvature due to the larger neighborhood size used in the analysis (Fig. 4.9).

4.2.4 Changing Landforms

The new development would change the landforms creating new ridges and valleys. We explored these changes by running the *r.geomorphon* add-on module which can be installed using *g.extension*. We can tune the results with parameters such as *search* and *skip* that determine the appropriate scale of analysis. By increasing these parameters we can skip small terrain variations in order to capture larger landforms:

```
g.extension r.geomorphon
r.geomorphon dem=elevation forms=landforms search=16 skip=6
```

In the resulting landform classification the new road has formed a major ridge (Fig. 4.10).

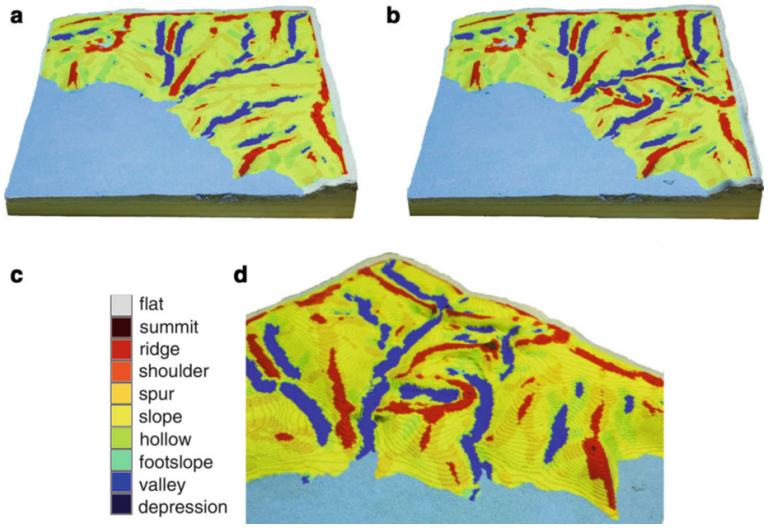


Fig. 4.10 Landform classification: (**a**) before development, (**b**) after development with the new elevated road classified as a ridge, (**c**) legend and (**d**) a detailed view of the modification done in Fig. 4.7

References

Abramowitz, M. and Stegun, I. (1965). *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Dover publications.

Horn, B. K. P. (1981). Hillshading and the reflectance map. *Proceedings of the IEEE*, 69(1):41–47.

Jasiewicz, J. and Stepinski, T. F. (2013). Geomorphons – a pattern recognition approach to classification and mapping of landforms. *Geomorphology*, 182:147–156.

Mitasova, H. and Mitas, L. (1993). Interpolation by regularized spline with tension: I. Theory and implementation. *Mathematical Geology*, 25(6):641–655.

Mitasova, H., Mitas, L., and Harmon, R. (2005). Simultaneous spline approximation and topographic analysis for lidar elevation data in open-source GIS. *IEEE Geoscience and Remote Sensing Letters*, 2:375–379.

Olaya, V. (2009). Basic land-surface parameters. In *Geomorphometry Concepts, Software, Applications, Developments in Soil Science*, volume 33, pages 141–169. Elsevier.

Chapter 5
Surface Water Flow and Soil Erosion Modeling

The topography of the Earth's surface controls the flow of water and mass over the landscape. Modifications to the surface geometry of the land redirect water and mass flows influencing ecosystems, crop growth, the built environment, and many other phenomena dependent on water and soil. We used Tangible Landscape to explore the relationship between overland flow patterns and landscape topography by manually changing the landscape model while getting near real-time feedback about changing flow patterns. We coupled Tangible Landscape with a sophisticated dam break model to investigate flood scenarios after a dam breach.

5.1 Foundations in Flow Modeling

Water flow over complex terrain can be described by a bivariate form of the shallow water continuity equation. The continuity equation can be coupled with momentum (Navier-Stokes) equations to simulate flooding due to dam failure.

5.1.1 Overland Flow

For shallow water flow the spatial variation in velocity with respect to depth can be neglected and the overland water flow during a rainfall event can be approximated by the following bivariate continuity equation (Julien et al. 1995):

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\,\mathbf{v}) = i_e \tag{5.1}$$

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_5

where:

h is the depth of overland flow in m
t is the time in s
$\mathbf{v}$ is the flow velocity vector $\mathbf{v} = (v_x, v_y)$ in m/s
i_e is the rainfall excess (rainfall − infiltration − vegetation intercept) in m/s.

If we assume that the dynamic friction slope of the water surface can be approximated by the static bare ground slope the flow velocity can be estimated by the Manning's equation:

$$\mathbf{v} = \frac{1}{n} h^{2/3} |\mathbf{s}|^{1/2} \mathbf{s_0} \tag{5.2}$$

where:

n is the Manning's coefficient expressed here with units $\mathrm{m}^{-\frac{1}{3}}\mathrm{s}$
$\mathbf{s}$ is the negative elevation gradient $-\nabla z = -(f_x, f_y)$ of the bare ground surface
$|\mathbf{s}|$ is the magnitude of elevation gradient (slope, see Sect. 4.1.3)
$\mathbf{s_0}$ is the unit vector in the flow direction (the negative elevation gradient direction).

For steady state water flow and a steady rainfall excess rate the continuity equation has the following form:

$$\partial h / \partial t = 0 \quad \longrightarrow \quad \nabla \cdot (h\,\mathbf{v}) = i_e \tag{5.3}$$

We introduce a diffusion-like term proportional to $\nabla^2[h^{5/3}]$ to incorporate the diffusive wave effects in at least an approximate way. The spatially distributed, steady overland water flow is then expressed as follows:

$$-\frac{\varepsilon}{2} \nabla^2 [h^{5/3})] + \nabla \cdot (h\,\mathbf{v}) = i_e \tag{5.4}$$

where ε is a spatially variable diffusion coefficient. The diffusion term, which depends on $h^{5/3}$ instead of h, makes the equation [Eq. (5.4)] linear in the function $h^{5/3}$ which enables us to solve it by the Green's function method using a stochastic technique referred to as path sampling method (Mitasova et al. 2004).

The path sampling method is based on the duality between the particle and field representation of spatially distributed phenomena. In this concept the density of particles in a space defines a field and vice versa, i.e. a field is represented by particles with a corresponding spatial distribution. Using this duality, processes can be modeled as the evolution of fields or the evolution of spatially distributed particles (Mitasova et al. 2004).

The flow evolution (accumulation process) can be also interpreted as an approximation of a dynamical solution for shallow water flow, in which velocity is mostly controlled by terrain slope and surface roughness rather than water depth and friction slope. Therefore the change of velocity over time at a given location

is negligible. The robustness of the path sampling method enables us to simulate complex, spatially variable conditions and efficiently explore flow patterns for a wide range of landscape configurations. This method is implemented in GRASS GIS as the module *r.sim.water*.

5.1.2 Dam Break Modeling

Surface water flow and flooding due to a dam breaking are also represented by the shallow water flow equation, but in this application we assume that there is no rainfall and the only source is water from the reservoir. In order to ensure a realistic representation given a large mass of moving water the shallow water continuity equation:

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\,\mathbf{v}) = 0 \tag{5.5}$$

is coupled with the momentum (Navier-Stokes) equation of fluid motion (Cannata and Marzocchi 2012):

$$\frac{\partial h v_x}{\partial t} + \frac{\partial h v_x^2}{\partial x} + \frac{\partial h v_x v_y}{\partial y} = S_x \tag{5.6}$$

$$\frac{\partial h v_y}{\partial t} + \frac{\partial h v_y v_x}{\partial x} + \frac{\partial h v_y^2}{\partial y} = S_y \tag{5.7}$$

where $\mathbf{S} = (S_x, S_y)$ is the source vector (water flowing from the lake):

$$S_x = -gh\left(\frac{\partial z_w}{\partial x} + \frac{n^2 v_x |\mathbf{v}|}{h^{4/3}}\right) \qquad S_y = -gh\left(\frac{\partial z_w}{\partial y} + \frac{n^2 v_y |\mathbf{v}|}{h^{4/3}}\right) \tag{5.8}$$

and

g is the gravitational acceleration in $\mathrm{m\,s^{-2}}$
z_w is the water level expressed as elevation above sea level in m
n is the Manning's roughness coefficient expressed here with units $\mathrm{m^{-\frac{1}{3}}\,s}$
$|\mathbf{v}|$ is the flow velocity magnitude $|\mathbf{v}| = (v_x^2 + v_y^2)^{1/2}$ in m/s.

The resulting partial differential equations are hyperbolic and non-linear and therefore must be solved numerically. The finite volume method with an upwind conservative scheme proposed by Ying et al. (2004) is used to solve the equations on a regular grid which makes the model suitable to GIS-based implementation. Green's theorem is used to obtain the discrete equations. The numerical method explicitly solves the governing equations in two separate steps. First, the continuity equation (5.5) is evaluated deriving the water depth at time $t + \Delta t$ and later

on these values are used in the source term [Eq. (5.8)] to solve the momentum equations (5.6), (5.7) and evaluate the flow velocities at time $t + \Delta t$. More details about the numerical solution can be found in Cannata and Marzocchi (2012). The model is implemented in GRASS GIS as an add-on module *r.damflood*.

5.1.3 Sediment Transport, Erosion, and Deposition

The *Simplified Erosion/Deposition model* (*SED*) model estimates sediment transport across a complex landscape and the resulting pattern of erosion and deposition using the idea originally proposed by Moore and Burch (1986). The model assumes that the sediment flow can be approximated by sediment transport capacity and the net erosion and deposition is transport capacity limited. It combines the Universal Soil Loss Equation parameters and upslope contributing area per unit width A to estimate the sediment flow T:

$$T \approx RKCPA^m (\sin\beta)^n \tag{5.9}$$

where:

T is the sediment flow rate in $\mathrm{kg\,m^{-1}\,s^{-1}}$
R is the rainfall factor in $\mathrm{MJ\,mm\,(ha\,h\,year)^{-1}}$
K is the soil erodibility factor in $\mathrm{ton\,ha\,h\,(ha\,MJ\,mm)^{-1}}$
A is the upslope contributing area per unit width in m
β is the steepest slope angle
C is the dimensionless land-cover factor
P is the dimensionless prevention-measures factor.

The net erosion/deposition D in $\mathrm{kg\,m^{-2}s^{-1}}$ is then computed as a divergence of sediment flow:

$$D = \nabla \cdot (T\mathbf{s_0}) = \frac{\partial(T\cos\alpha)}{\partial x} + \frac{\partial(T\sin\alpha)}{\partial y} \tag{5.10}$$

where $\mathbf{s_0} = (\cos\alpha, \sin\alpha)$ is the unit vector of the steepest slope (flow) direction (direction of the negative gradient $-\nabla z$) given by the aspect angle α (see Sect. 4.1.3). The exponents m, n control the relative influence of the water and slope terms and reflect the impact of different types of flow. The typical range of values is $m = 1.0$–1.6, $n = 1.0$–1.3 with the higher values reflecting the pattern for prevailing rill erosion with more turbulent flow when erosion sharply increases with the amount of water. Lower exponent values close to $m = n = 1$ better reflect the pattern of the compounded, long term impact of both rill and sheet erosion and averaging over a long term sequence of large and small events.

5.2 Case Study: The Hydrological Impact of Development

The development on North Carolina State University's Centennial Campus raises concerns about the impact of stormwater runoff and erosion during construction. We used Tangible Landscape to explore changes in overland flow and erosion patterns caused by potential construction site grading and to provide feedback for the design of stormwater and erosion control measures.

In this case study we worked with the same area and physical model used in Sect. 4.2.1. To familiarize ourselves with the basic hydrologic conditions of this site we computed flow accumulation and delineated watershed boundaries. First, we set the computational region to match our study area and the resolution to 1 m and then we ran the least cost path flow tracing implemented in the module *r.watershed* to derive the flow accumulation raster and watersheds. We set the threshold for the approximate size of the watershed areas to 1000 grid cells and we converted the areas to a vector representation so that we could display the watershed boundaries:

```
g.region n=223765 s=223015 e=638450 w=637700 res=1
r.watershed elevation=dem accumulation=flow_accum \
    basin=watersheds threshold=1000
r.to.vect input=watersheds output=watersheds type=area
```

The resulting map that combines the flow accumulation raster with the vector representation of watershed boundaries and contours was then projected over the model in Fig. 5.1a.

5.2.1 The Impact of Development on Surface Water Flow

To explore the potential impact of new development we graded sites for new apartment buildings with flat slopes and constructed an access road (see Fig. 4.7d). We carved a narrow culvert to allow water to flow under the road. By calculating flow accumulation and watersheds for the modified topography (Fig. 5.1b) we could immediately see that the watershed boundaries in the developed area changed significantly due primarily to the new road acting as an artificial watershed boundary.

Flow accumulation derived by the module *r.watershed* is based on the least cost path algorithm and is not designed to represent water depth and pooling of water in depressions. Therefore, we used the shallow water flow model implemented in the module *r.sim.water* to simulate the overland flow depth during a storm event for the current conditions and after the grading. We first derived the components of elevation surface gradient $\nabla z = (f_x, f_x)$ (parameters dx, dy in the module *r.slope.aspect*) and then ran the water flow simulation with the rainfall excess value of 150 mm/h:

```
r.slope.aspect elevation=scan dx=scan_dx dy=scan_dy
r.sim.water elevation=scan dx=scan_dx dy=scan_dy \
    rain_value=150 depth=depth
```

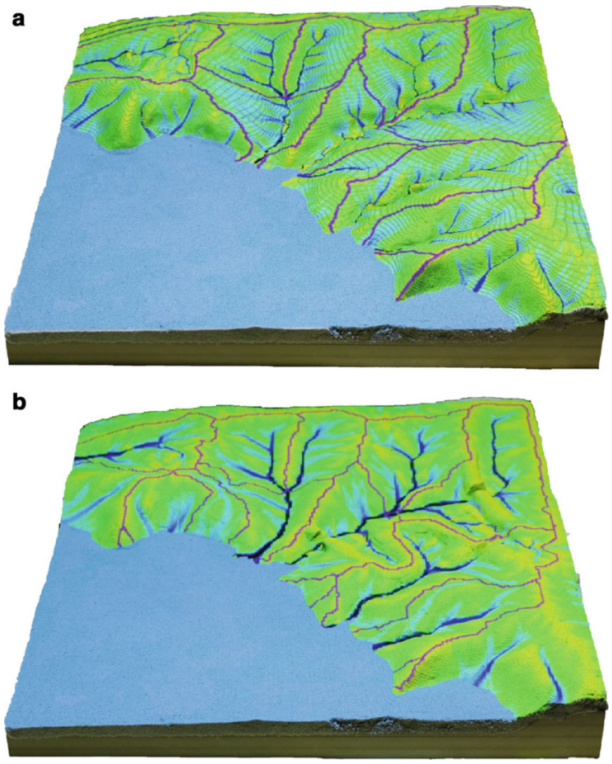


Fig. 5.1 Flow accumulation and watershed boundaries computed with *r.watershed* before and after the terrain modifications shown in Fig. 4.7

The resulting water depth maps were projected over the model with the initial terrain conditions and then in near real time as the terrain was modified with the final water depth pattern shown in Fig. 5.2b. Note that we have used an extremely high value of rainfall excess to provide rapid maps of water depth as the model was modified. Once we have the final design we can run the simulation with a selected storm and obtain realistic water depth estimates.

We can also run the module *r.sim.water* in a dynamic mode (*-t* flag) and project the evolution of water depth during the storm event as an animation using the GRASS GIS animation tool.

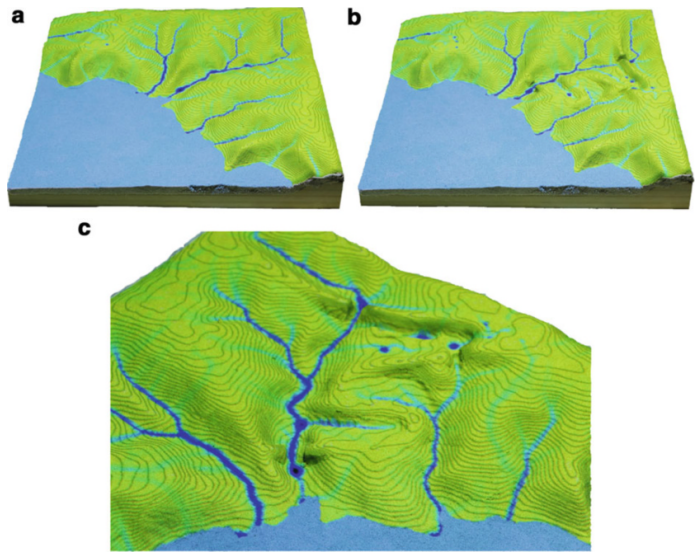


Fig. 5.2 Water depth computed with *r.sim.water* (**a**) before and (**b**) after the terrain modifications shown in Fig. 4.7. A detailed view of the modified water flow is shown in (**c**)

5.2.2 The Impact of Grading on Erosion and Deposition Pattern

We also explored the impact of grading on sediment transport and erosion/deposition patterns using the SED model. We assumed a uniform rainfall factor $R = 270$ in (hundreds of ft-tonf.in)/(acre.h.year) as well as uniform soils and land cover with the soil erodibility factor $K = 0.2$ and the land cover factor $C = 0.01$ (grass, no trees). The topographic factor exponents were set to $m = n = 1$ to represent prevailing sheet flow. Since we used uniform values for the R-, K- and C-factors, the pattern of erosion and deposition depends solely upon the slope of the terrain and the upslope contributing area. In the following workflow we first computed the slope and aspect maps of the scanned model. Then we computed flow accumulation and combined it with the slope using map algebra to estimate the topographic factor and sediment transport capacity [see Eq. (5.9)].[1] Finally, we computed the divergence of the sediment flow vector field with the add-on module *r.divergence* to generate the erosion-deposition map:

[1] For a general case when m, n and resolution are not set to 1 see the erosion modeling tutorial: http://ncsu-osgeorel.github.io/erosion-modeling-tutorial/grassgis.html

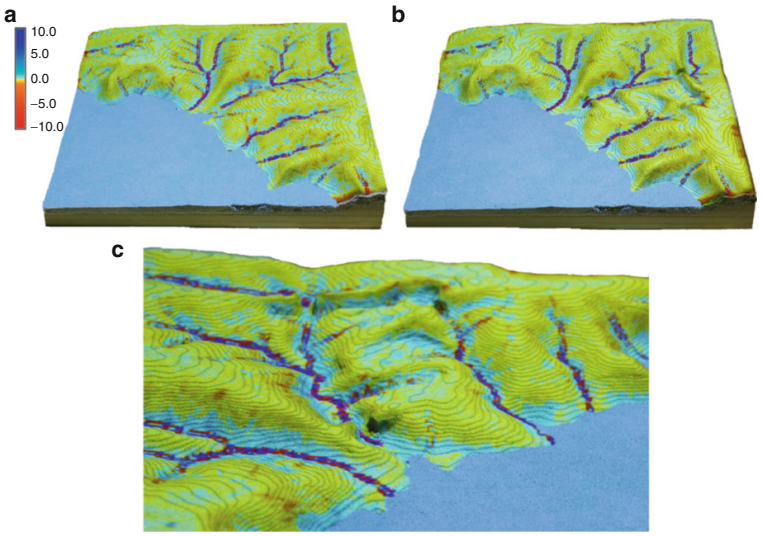


Fig. 5.3 Soil erosion and deposition maps draped over the model (**a**) before and (**b**) after grading shown in Fig. 4.7. Detailed view of the erosion-deposition pattern is in (**c**)

```
r.slope.aspect elevation=scan aspect=scan_aspect \
    slope=scan_slope
r.flow elevation=scan flowaccumulation=scan_flowacc
r.mapcalc "scan_sedflow = 270. * 0.2 * 0.01 * scan_flowacc * \
    sin(scan_slope)"
r.divergence magnitude=scan_sedflow direction=scan_aspect \
    output=scan_usped
r.colors map=scan_usped rules=color_erdep.txt
```

The resulting erosion-deposition map is projected over the model in Fig. 5.3.

5.3 Case Study: Dam Breach

The Lake Raleigh dam located within our Centennial Campus study area broke in September of 1996 after Hurricane Fran. In this case study we simulated how flood water would spread across the current landscape if the dam was breached again. We also modified the terrain to explore how different morphologies would influence the spread of flooding. We used the add-on *r.damflood*, which was designed specifically to simulate dam breaks and analyze the subsequent flooding.

Fig. 5.4 An overview of Lake Raleigh's surroundings with the dam highlighted in *red*. The digital elevation model on which the orthophoto is draped was created by fusing the latest 2014 data with the lake bathymetry data from 2001

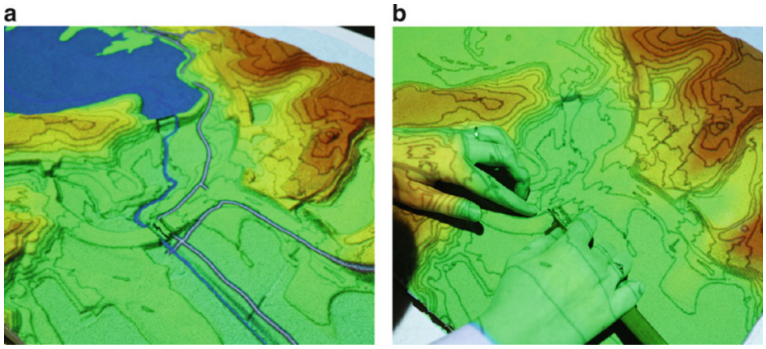


Fig. 5.5 (**a**) The current conditions with a stream and a greenway below the dam; (**b**) a scenario in which the road has been removed

5.3.1 Site Description and Input Data Processing

Figure 5.4 shows the terrain and the most recent orthophotograph of our Lake Raleigh study area including the dam. The stream flowing out from the lake merges with the stream coming from the south in the area just below the dam, which is actively used for recreation providing space for disc golf, running, and cycling (Fig. 5.5a).

The dam break simulation requires several input layers including the elevation raster that describes the lake bathymetry and the surrounding topography, the depth of the lake water, the Manning's roughness coefficient, and the geometry of the dam break. The elevation raster was created by fusing the latest DEM based on the 2014 lidar survey with a DEM from the 2001 survey which we used as a proxy for the lake bathymetry. The 2001 survey captured the topography while the lake was drained after the dam failure in 1996 and we considered the 2001 DEM in the lake area a suitable approximation of the current lake bathymetry.

The lake depth raster was computed with the module *r.lake* using the fused DEM as an input. This module fills an area with water for a designated elevation level starting from a given seed point and outputs a map of the water depth for the flooded area. We determined that the current water level elevation for the Lake Raleigh was 85 m by querying the DEM along the outline of the lake. We selected an arbitrary point inside the lake area as a seed point. While *r.lake* assigns null values to the cells outside of the filled area, *r.damflood* requires zero values outside of the lake, so we converted null values to zeros with the *r.null* module and limited the flooded area to the lake mask:

```
g.region n=224026 s=223007 e=639480 w=637798 res=3
r.lake elevation=lake_bottom_dem water_level=85 \
    lake=lake_depth_tmp coordinates=638792,223659
r.null map=lake_depth_tmp null=0
r.mapcalc "lake_depth = if(lake_mask, lake_depth_tmp, 0)
```

The Manning's roughness coefficient influences the speed of water. It depends on landcover and varies between 0 and 1 with higher values resulting in slower flows. To derive Manning's coefficients we could for example reclassify a landcover map. In our example for simplicity's sake we created a raster map with a uniform Manning's value:

```
r.mapcalc "manning = 0.01"
```

Finally, the module *r.damflood* requires a dam breach raster which represents the height of the breach from the top of the dam. To obtain this raster we scanned the model with the dam. Then we carved a breach into the dam and rescanned the model. We computed the difference using the function defined in code snippet 4.2.2. We used a vertical limit of 1 m to avoid detecting differences due to noise from scanning:

```
adjust_scan('scan_before', 'scan_after', 'scan_adjusted')
gscript.mapcalc("breach = if(scan_before - scan_adjusted > 1,
    scan_before - scan_adjusted, 0)")
```

5.3.2 The Impact of the Road on Flooding

For our first flood simulation we used the current conditions. A major road with a culvert crosses the stream below the dam. We carved a culvert into the sand model by removing a narrow channel of sand from the road and scanned this model to provide input for our simulation.

Since our sand model does not incorporate the lake bathymetry we had to combine the fused DEM that includes the bathymetry with the scan. We then ran the *r.damflood* module with the input map layers and set the simulation parameters such as the length of the simulation, the time step for creating output maps, and the

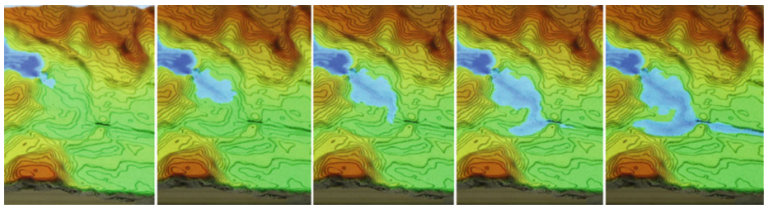


Fig. 5.6 The flood simulation with current conditions

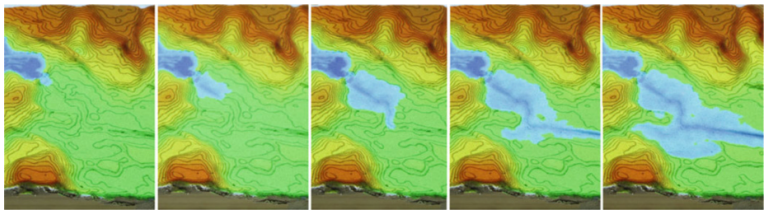


Fig. 5.7 The flood simulation after the road has been removed

computational time step. Since the simulation is computationally intensive it may take several minutes to compute depending on the settings and resolution.

```
r.mapcalc "scan_combined = if(lake_mask > 0, lake_bottom_dem, \
    scan_adjusted)"
r.damflood elev=scan_combined lake=lake_depth dambreak=breach \
    manning=manning h=flood timestep=0.1 tstop=1000 deltat=10
```

The series of output raster maps generated by the simulation can be registered in the GRASS GIS Temporal Framework and then visualized using module *g.gui.animation*. The results for the current conditions are shown in Fig. 5.6. Clearly, the road creates a significant obstacle for the spread of water and causes pooling below the dam. Contained by the road the water floods upstream areas to the west.

In the next simulation we were interested in the conditions before the road was built in 2004. We simply removed the road (Fig. 5.5b) and then rescanned and recomputed the simulation (Fig. 5.7). Removing the road changes the flow pattern and as a result water spreads freely at lower depth thus impacting a larger area, but potentially reducing risk.

References

Cannata, M. and Marzocchi, R. (2012). Two-dimensional dam break flooding simulation: A GIS-embedded approach. *Natural Hazards*, 61(3):1143–1159.

Julien, P. Y., Saghafian, B., and Ogden, F. L. (1995). Raster-based hydrologic modelling of spatially-varied surface runoff. *Water Resources Bulletin*, 31(3):523–536.

Mitasova, H., Thaxton, C., Hofierka, J., McLaughlin, R., Moore, A., and Mitas, L. (2004). Path sampling method for modeling overland water flow, sediment transport, and short term terrain evolution in Open Source GIS. *Developments in Water Science*, 55:1479–1490.

Moore, I. D. and Burch, G. J. (1986). Physical basis of the length-slope factor in the Universal Soil Loss Equation. *Soil Science Society of America journal*, 50:1294–1298.

Ying, X., Khan, A. A., and Wang, S. S. (2004). Upwind conservative scheme for the Saint Venant equations. *Journal of hydraulic engineering*, 130(10):977–987.

Chapter 6
Viewshed Analysis

Viewshed (visibility) analysis is used in many different fields for both practical and aesthetic applications. It can play an important role when planning new buildings or roads especially in urban settings where obstructed views may raise safety concerns. In recreation areas views of beautiful landscapes are highly valued and protected with great passion. Visibility analysis is also crucial when planning location of monitoring cameras or communication towers in order to maximize coverage. With the increasing availability of high-resolution DEMs and DSMs derived from lidar visibility analysis is becoming more accurate, broading the range of its applications. We used Tangible Landscape to analyze viewsheds on North Carolina State University's (NCSU) Centennial Campus from different observer positions and explored how future development would affect the viewsheds. We introduced object recognition to collaboratively designate observer positions.

6.1 Line of Sight Analysis

Line-of-sight analysis is used to map visible areas. If the line of sight between point A and point B does not intersect the terrain then these points are mutually visible. More precisely, with the slope of the line between A and B defined as $(z_B - z_A)/d_{AB}$ where z is the height and d is the horizontal distance, points A and B are considered visible to each other if the line does not cross any location C such that the slope of AC is larger than the slope of AB (Fig. 6.1). The viewshed of A on a raster DEM is the set of all cells of the DEM that are visible from A. The height of the observer A above terrain has significant influence on the viewshed extent. The observer height can be changed to model the view of a standing person, the view from a multi-story building, or the view from the top of a cell tower. The resulting viewshed can be represented for example as a binary raster (containing only 0 and 1s) or a raster with the values of vertical angles with regard to the viewpoint.

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_6

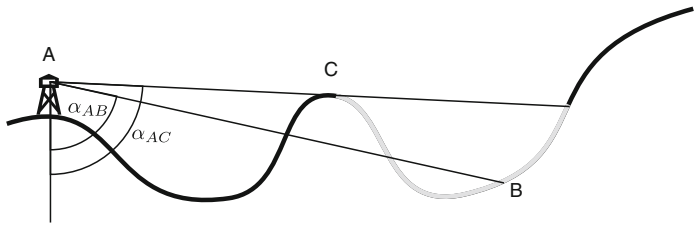


Fig. 6.1 Line of sight analysis: point B is not visible from point A because the line of sight between A and B has a smaller slope $\tan \alpha_{AB}$ than the line of sight between A and C

Viewsheds from multiple locations can be computed and combined as the sum of binary viewsheds into a cumulative viewshed (Wheatley 1995) that allows us to assess the overall visibility of a place from selected locations. The concepts of visibility analysis have been applied to diverse geospatial applications. For example landform identification using geomorphons combines visibility analysis with computer vision (Jasiewicz and Stepinski 2013) (see Sect. 4.1.3). Several terrain relief visualization techniques take advantage of visibility analysis to highlight subtle terrain features. For example sky-view factor—the visible part of the sky unobscured by relief—can be used to render topographic relief as if it was diffusely illuminated in order to clearly visualize the relative height of features. Sky-view factor uses openness as a proxy for uniform diffuse illumination (Zakšek et al. 2011). It is implemented in GRASS GIS as the add-on module *r.skyview*. Haverkort et al. (2009) describe an efficient method implemented in the module *r.viewshed* for computing viewsheds on a raster DEM using a technique called line sweeping (Fig. 6.1).

6.2 Case Study: Viewsheds Around Lake Raleigh

We used object recognition with Tangible Landscape to explore different viewsheds on NCSU's Centennial Campus by placing markers representing observer positions on a physical model of the landscape. We also evaluated how changes to the topography, the forest canopy, and the buildings would impact these viewsheds.

6.2.1 Site Description and Model

Our study area shown in Fig. 6.2 is the Lake Raleigh landscape on NCSU's Centennial Campus. Key features of the landscape include Lake Raleigh Woods in the southwest, the Chancellor's house and State Club in the south, and the Hunt Library in the northwest.

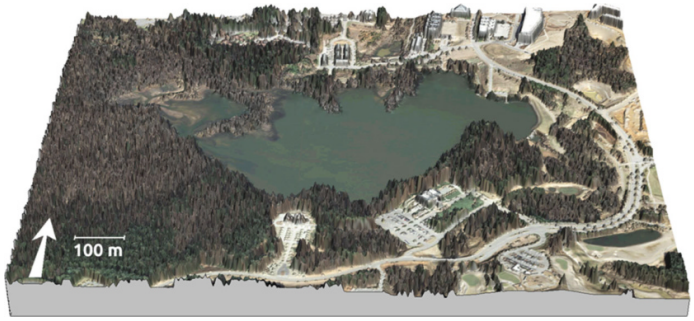


Fig. 6.2 Lake Raleigh study site: a lidar-based DSM with an orthophoto draped over the surface

We CNC routed a base model and a mold of the landscape surface from MDF based on a lidar-derived DEM and DSM. Both models are approximately 1:2400 scale without vertical exaggeration. We used the base and the mold to cast a malleable layer of polymer enriched sand representing the forest canopy and buildings on top of the terrain model.

6.2.2 Identifying View Points

We modeled the views from different locations by placing markers on the model for Tangible Landscape to detect and recognize. The position of each marker was registered as a vector point and its coordinates were then used as an input for automated computation of viewshed from this location.

Given the complex 3D form and steep slopes of some terrains it may be hard to place markers on a rigid model such as a 3D printed or CNC routed terrain. If the model is coated with magnetic paint then a magnetized marker will hold to the slopes, normal to the surface (see Fig. 3.7). If the model has been sculpted or cast in polymeric sand then markers can simply be stuck vertically into the sand.

There are multiple ways to detect markers. The most reliable approach is to detect the markers based on the difference between a base scan of the terrain model without the markers and the scan with the markers placed on the model. We computed the difference using the function defined in the code snippet in Sect. 4.2.2. A threshold value for the difference in elevation must be set in order to distinguish markers from noise. The threshold can be specified in millimeters and since we know the horizontal scale and vertical exaggeration during the processing it can be automatically converted into the real world elevation difference required in the following code snippet. Areas with difference values above the threshold are vectorized as points for viewshed analysis as follows. We first convert the areas from

raster to vector areas (polygons) using the module *r.to.vect* and then extract the area centroids as vector points using the module *v.to.points*.

```
adjust_scan('scan_before', 'scan_after', 'scan_adjusted')
gscript.mapcalc("change = if(scan_adjusted - scan_before >
    threshold, 1, null())")
gscript.run_command('r.to.vect', flags='st', input='change',
    output='change_vector', type='area')
gscript.run_command('v.to.points', flags='t',
    input='change_vector', type='centroid', output='markers')
```

6.2.3 Visibility Analysis on DSM Using Markers

We explored viewsheds from different points of interests such as the Chancellor's house, the North Shore neighborhood, the Hunt Library's observation deck, the lakeside pier, and the middle of the lake. The viewsheds were computed on the lidar-derived DSM. To identify the markers we first scanned the model and saved the scanned raster. We continued scanning while placing the markers at the locations of interest (Fig. 6.3). The markers were then identified and stored as vector points. We used the following workflow to compute the viewsheds and set the desired color scheme. In this case visible areas are highlighted in yellow.

```
coordinates = gscript.read_command('v.out.ascii',
    input='change_vector', separator=',').strip().splitlines()
for i, line in coordinates:
   output = 'viewshed_%s' % i
   coordinate = [float(c) for c in line.split(',')[0:2]]
   gscript.run_command('r.viewshed', flags='b',
       input='elevation', output=output,
       observer_elevation=1.75, coordinates=coordinate)
   gscript.run_command('r.null', map=output, setnull=0)
   gscript.write_command('r.colors', map=output, rules='-',
       stdin='1 yellow')
```

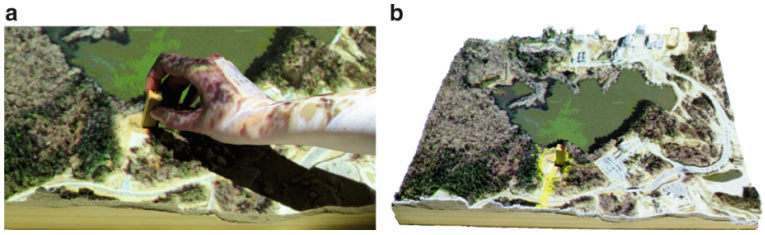


Fig. 6.3 Placing a marker to identify a viewpoint

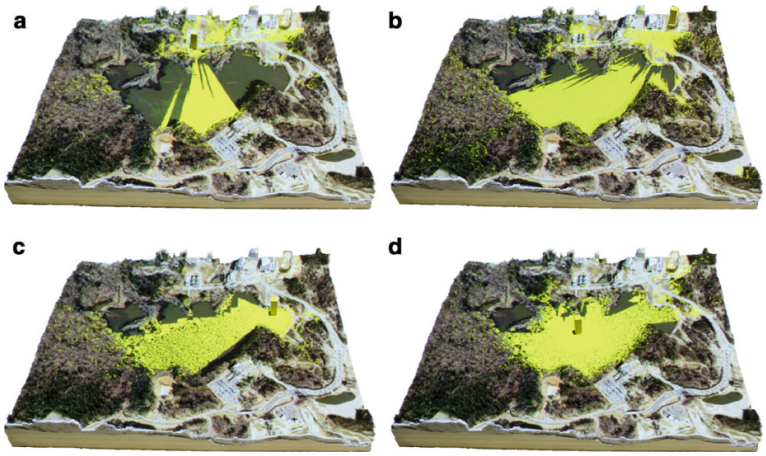


Fig. 6.4 Viewsheds computed on lidar-based DSM from viewpoints at (**a**) the North Shore neighborhood, (**b**) Hunt Library's observation deck, (**c**) a lakeside pier, (**d**) and a canoe on the lake

The viewsheds computed from different marker positions are shown in Fig. 6.4. The best views of the lake are from the Hunt Library's observation deck and the lakeside pier.

6.2.4 Modeling Viewsheds from a New Building

In the previous section we computed viewsheds on the lidar-derived DSM. In this section we used the scanned surface to model viewsheds on a modified landscape. We explored a scenario in which we built a hotel on the southeastern side of the lake and assessed the viewshed from the new hotel (Fig. 6.5a).

We began by sculpting the new building in sand. Then we scanned the modified sand surface and saved the scan. We placed a marker that was automatically detected and digitized as a viewpoint. We computed the viewshed using the base scan rather than the newer scan with the marker because the marker would act as an obstacle blocking the observer's view. Fig. 6.5c shows the viewshed from the southwest side of the new hotel. The hotel is oriented so that it has nice views of the lake. Since the hotel is a tall building it is visible from many places around the lake such as Hunt Library (Fig. 6.5d).

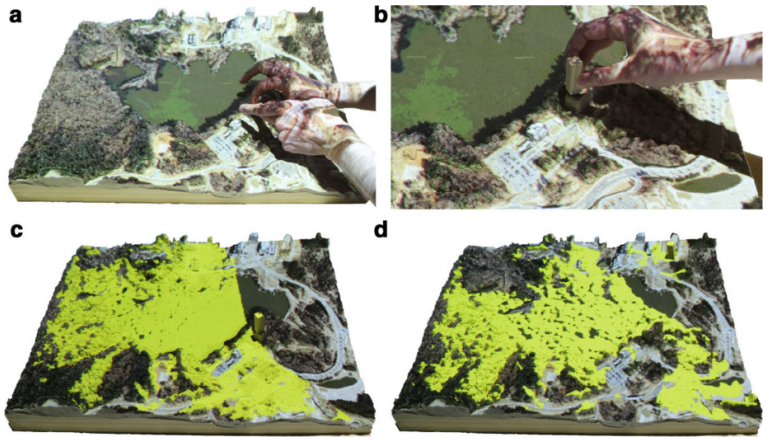


Fig. 6.5 Modeling viewsheds from a new building: (**a**) building a new hotel, (**b**) digitizing a viewpoint, and evaluating viewsheds from (**c**) the hotel and (**d**) the library

References

Haverkort, H., Toma, L., and Zhuang, Y. (2009). Computing visibility on terrains in external memory. *J. Exp. Algorithmics*, 13:5:1.5–5:1.23.

Jasiewicz, J. and Stepinski, T. F. (2013). Geomorphons – a pattern recognition approach to classification and mapping of landforms. *Geomorphology*, 182:147–156.

Wheatley, D. (1995). Cumulative viewshed analysis: a GIS-based method for investigating intervisibility, and its archaeological application. *Archaeology and geographical information systems: a European perspective*, pages 171–185.

Zakšek, K., Oštir, K., and Kokalj, v. (2011). Sky-View Factor as a Relief Visualization Technique. *Remote Sensing*, 3(2):398–415.

Chapter 7
Trail Planning

The design of a walking or hiking trail is based on fine scale topographic conditions and varied criteria specific to the particular context such as aesthetics, views, construction cost, and environmental sensitivity. As a result trail planning is typically a product of expert knowledge, field surveys, and creative design decisions—often made on site. However, when high resolution data is available geospatial modeling can be used to identify routes optimized for travel time and suitability. To design trails with Tangible Landscape we can hand place waypoints on a physical model and then the optimal network connecting the waypoints is computed in near real-time. This approach—hand placing tangible waypoints and computationally networking the waypoints—combines creative, collaborative decision making with mathematical optimization. In this chapter we explain the theory and methodology for designing trails with Tangible Landscape and then discuss a case study, the design of hiking trail scenarios for Lake Raleigh Woods, North Carolina.

7.1 Trail Design Methodology

Our approach for designing trail networks with Tangible Landscape combines the creative identification and siting of key waypoints like trailheads and scenic spots with computationally optimized routing between these points. With Tangible Landscape we can place tangible markers by hand to automatically digitize waypoints. This allows us to work intuitively, feel the slopes and curvature with our hands, and easily collaborate.

To find the optimal routes we compute the least cost path between pairs of waypoints over the terrain and a cost surface as a function of walking energetics. First we create a cost surface that represents friction—the additional time required to cross a cell. Next we designate waypoints, points through which we want the trail to pass. For each unique pair of waypoints we use the *r.walk* module to

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_7

compute the anisotropic cumulative cost of walking between these points across the terrain and the friction surface based on Naismith's rule for hiking times. Then we compute the least cost path between those points across this cumulative cost surface. All of the least cost paths are combined into a network of potential routes. Finally we solve the traveling salesman problem to identify the best route across this network.

7.1.1 Least Cost Path Analysis

With least cost path analysis we can determine the most cost-effective route between a source and destinations on a cost surface. The cost surface is a raster representing cost for traversing a cell and can be derived as a function of distance, slope, land cover or other relevant criteria. To find the least cost path between locations we need to compute a cumulative cost surface where each cell contains the lowest cumulative cost of traversing the space between each cell and the specified location. We also need to generate a movement direction raster tracing the movements that created the cumulative cost surface.

First we compute the walking cost raster using Naismith's rule for walking times (Naismith 1892) with further adjustments (Aitken 1977; Langmuir 1984) of the cost for specific slope intervals. Considering only the elevation surface the cost to walk across a grid cell, expressed as time T in seconds is computed as follows:

$$T = a \cdot \Delta S + b \cdot \Delta H_u + c \cdot \Delta H_{md} + d \cdot \Delta H_{sd} \tag{7.1}$$

where:

ΔS is the horizontal distance in m
ΔH is the height difference in m
a is the time in seconds it takes to walk for 1 m on flat surface
b is the additional walking time in seconds, per meter of elevation gain ΔH_u on uphill slopes
c is the additional walking time in seconds, per meter of elevation loss ΔH_{md} on moderate downhill slopes
d is the additional walking time in seconds, per meter of elevation loss ΔH_{sd} on steep downhill slopes

Up to a specific slope value threshold, walking downhill is faster; after that it becomes more difficult and adds to the time needed to cross the cell. The slope value threshold (slope factor) derived from experiments is -0.2125 corresponding to $12°$ downslope ($\tan 12° = 0.2125$).

Taking into account land cover conditions the total cost T_{total} in seconds is estimated as a linear combination of movement and friction costs using the dimensionless weight λ:

$$T_{\text{total}} = T + \lambda \cdot F \cdot \Delta S \tag{7.2}$$

where F is the friction in s/m. It represents the additional time in seconds that it takes to walk 1 m within a given cell due to its land cover conditions. The friction map may simply be a landcover map that has been recoded as time costs. If landcover data is not available at an appropriate resolution or if there are other important factors that need to be considered, the friction map can be created through map overlay analysis. Parameters such as soils, hydrology, roads, and existing trails can then be incorporated into the cost surface using the friction map.

We used the total cost surface given by Eq. (7.2) to compute a cumulative cost surface where each cell contains the lowest cumulative cost of traversing the space between each cell in the region and the target location. The movement direction raster is also generated tracing the movements that created the cumulative cost surface. We can then compute least cost path between any point in the study area and the target location based on the cumulative cost raster and movement directions raster.

7.1.2 Network Analysis

Using least cost path analysis it is possible to create many combinations of routes between the given waypoints. This can result in a large number of connections and potential trails. We can use network analysis to find a trail loop that goes through all given waypoints in an optimal order, avoiding connections with high costs (Fig. 7.1). The solution for this well-known optimization problem—the traveling salesman problem (TSP)[1]—is implemented in the module *v.net.salesman*, which uses a heuristic algorithm yielding a good solution in a reasonable amount of time.

7.1.3 Trail Slope Extraction

When planning trails we are typically interested in the average and maximum slope of a trail. However, the extracted values are the slope values in the direction of steepest slope rather than in the direction of the trail. To compute that, we need to know the direction of the trail, which can be derived from its vector representation. The slope along the trail β_t can then be computed as the steepest slope β [see Eq. (4.5)] multiplied by the cosine of the angle between the direction of the trail α_t and the steepest slope direction α [i.e. aspect, see Eq. (4.6)]:

$$\tan \beta_t = \tan \beta \cos(\alpha - \alpha_t) \tag{7.3}$$

[1]See: https://en.wikipedia.org/wiki/Travelling_salesman_problem

Fig. 7.1 The TSP applied to a network of potential trails. *Black lines* represent potential routes. The *red line* is the most cost-effective combination of routes that connects all of the waypoints

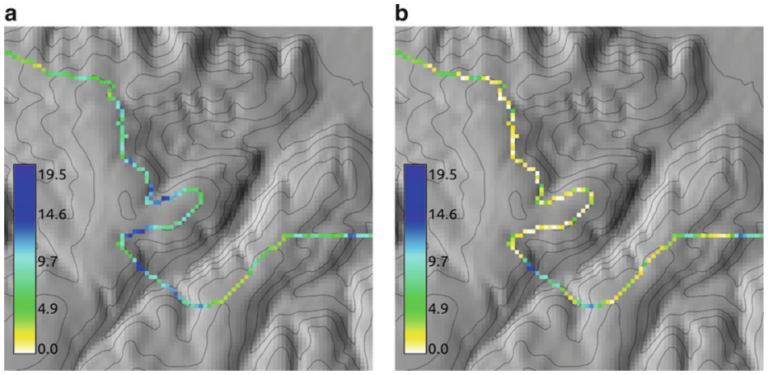


Fig. 7.2 The difference between the trail slope extracted (**a**) in the direction of the steepest slope and (**b**) in the direction of the trail (in degrees)

The difference is visible in Fig. 7.2 where part of a trail along a contour line has very low slope values along the trail, but has higher slope values in the direction of the steepest slope. Both types of slope computation can be useful depending upon our objectives. When building and maintaining a trail we may be interested in the steepness of the terrain in that area, while a hiker may be more interested in the directional slope of a trail.

7.2 Case Study: Designing a Recreational Trail

We used Tangible Landscape to design alternative trail scenarios for Lake Raleigh Woods on North Carolina State University's Centennial Campus. This old growth woodland on the south shore of Lake Raleigh is rich in biodiversity with approximately 200 species of vascular plants and regionally important stands of mesic mixed hardwood and dry-mesic oak-hickory forest (Blank et al. 2010). Informal hiking and mountain biking trails have caused significant disturbance in the woods and erosion along the lakefront. A formal trail system could concentrate people on low-impact routes and thus conserve the landscape while enhancing opportunities for recreation.

7.2.1 Input Data Processing

We derived a bare earth DEM and DSM with buildings and vegetation from a lidar survey acquired in 2013. Then we CNC routed 1:1500 scale models of the DEM, the DSM, and their inverses (see Sect. 3.4.4). We used the inverse models as molds for casting polymeric sand models.

We created the friction map—a required input for computing the cost surface—by computing the average friction cost for each cell from a set of friction maps based on the recoded rasterized maps of buildings, hydrology, floodplains, roads, soils, and trails with *r.series* using the average function.

First we converted all of the necessary vector maps to raster. Before converting linear features we set the resolution to 3 m in order to give these features reasonable widths.

```
g.region n=224134 s=223005 w=637732 e=639085 res=1

v.to.rast input=lake_raleigh output=lake_raleigh use=val
v.to.rast input=chancellors output=chancellors use=val
v.to.rast input=buildings output=buildings use=val
v.to.rast input=floodplain output=floodplain use=val
v.to.rast input=hydrology_areas output=hydrology_areas use=val
v.to.rast input=trail_areas output=trails use=val
v.to.rast input=roads output=roads use=val

g.region res=3
v.to.rast input=greenways output=greenways use=val
v.to.rast input=streets output=streets use=val
v.to.rast input=hydrology output=hydrology use=val
g.region res=1
```

In order to limit processing to publicly accessible areas we made a composite map of inaccessible areas by combining maps of private land, buildings, and the lake using *r.mapcalc*[2] and then created an inverted mask from this map.

```
r.mapcalc "masking = if((buildings ||| chancellors ||| \
    lake_raleigh), 1, null())"
r.mask -i raster=masking
```

Next, we recoded all maps as friction (time penalties—additional time it takes to walk 1 m), and then overlaid the maps using *r.mapcalc* and *r.series*.

```
r.mapcalc "hydro = if(isnull(hydrology_areas ||| hydrology), \
    1, 10)"
r.mapcalc "flood = if(isnull(floodplain), 1, 10)"
r.mapcalc "paths = if(isnull(trails ||| greenways), 10, 1)"
r.mapcalc "transit = if(isnull(streets ||| roads), 1, 10)"

r.series input=flood,hydro,paths,transit,soils output=friction \
    method=average
r.colors -n map=friction color=ryg
r.mask -r
```

The resulting friction map is shown in Fig. 7.3.

7.2.2 Computing the Trail Using the Least Cost Path

In our case study we computed least cost paths between waypoints to create a trail network. Least cost paths were computed for each combination of two points. The following code snippet shows how such a network can be generated. For each waypoint we called the function `trail` defined below, which computes a raster representing the cumulative cost (time) from this waypoint and then multiple least cost paths from the remaining waypoints.

```
def trail(elevation, friction, walk_coeff, lambda_,
    slope_factor, point_from, points_to, vect_paths):
   """Computes least cost path to multiple locations based on
       walking time"""
   gscript.run_command('r.walk', flags='k',
       elevation=elevation, friction=friction, lambda_=lambda_,
       walk_coeff=walk_coeff, slope_factor=slope_factor,
       start_coordinates=point_from,
       stop_coordinates=points_to, output='tmp_walk',
       outdir='tmp_walk_dir')

   for i in range(len(points_to)):
```

[2]Note that ||| is a symbol for logical OR which ignores NULL values and treats them as logical false.

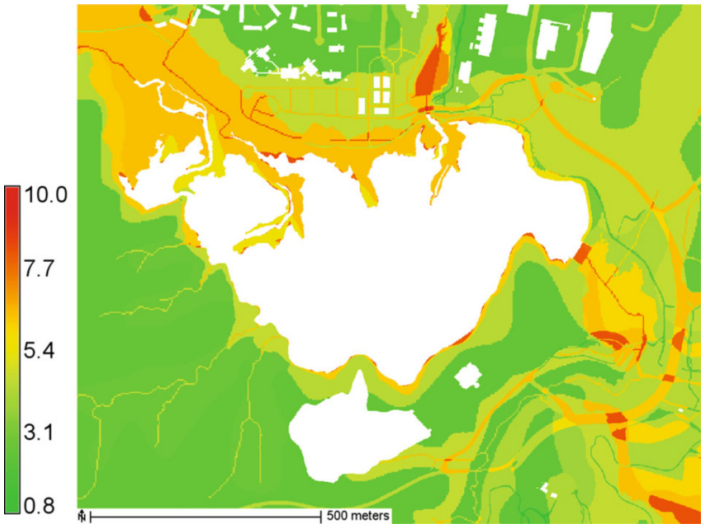


Fig. 7.3 The friction map in seconds per meter created by combining layers of buildings, hydrology, floodplains, roads, and trails

```
        gscript.run_command('r.drain', flags='d',
            input='tmp_walk', direction='tmp_walk_dir',
            output='tmp_drain', drain=vect_paths[i],
        start_coordinates=points_to[i], overwrite=True)

    # remove temporary maps
    gscript.run_command('g.remove', type=['raster', 'vector'],
        name=['tmp_walk', 'tmp_walk_dir', 'tmp_drain'],
        flags='f')
```

This is repeated for all waypoints except the last.

```
def trails_combinations(elevation, friction, walk_coeff,
    lambda_, slope_factor, points, vector_routes):
    import itertools

    coordinates = gscript.read_command('v.out.ascii',
        input=points, format='point', separator=',').strip()
    coords_list = []
    for coords in coordinates.split():
        coords_list.append(coords.split(',')[:2])

    combinations = itertools.combinations(coords_list, 2)
    combinations = [list(group) for k, group in
        itertools.groupby(combinations, key=lambda x: x[0])]
```

```
    i = k = 0
    vector_routes_list = []
    for points in combinations:
       i += 1
       point_from = ',',.join(points[0][0])
       points_to = [',',.join(pair[1]) for pair in points]
       vector_routes_list_drain = []
       for each in points_to:
          vector_routes_list_drain.append('route_path_' +
              str(k))
          k += 1
       vector_routes_list.extend(vector_routes_list_drain)

       trail(elevation, friction, walk_coeff, lambda_,
           slope_factor, point_from, points_to,
           vector_routes_list_drain)

    gscript.run_command('v.patch', input=vector_routes_list,
        output=vector_routes)
```

The path, however, is only computed for combinations that were not covered in previous steps in order to reduce computation time. This means that the least cost path is only computed in one direction. The direction matters because the speed of walking uphill and downhill differs. Therefore, the walking coefficients and slope factor of *r.walk* should be adjusted to affect uphill and downhill speed equally. This can be achieved by setting the slope factor to zero and using the same absolute number for coefficients b and d.

7.2.3 Finding the Optimal Trail

Using least cost path analysis we created all of the combinations of routes between waypoints. This can result in a large number of connections and potential trails. Therefore we used network analysis to find a trail loop that goes through all our waypoints in an optimal order avoiding connections with high costs. The solution for this well-known optimization problem—the traveling salesman problem (TSP)—is implemented in the module *v.net.salesman*.

All GRASS GIS network analysis modules require a network vector map containing connected lines (arcs) and points (nodes). This vector map can be prepared using the module *v.net* which topologically connects the given points (our waypoints) and lines (our trail combinations). Since we wanted to route the trail through all our waypoints we found the categories of all waypoints in the network and then passed them to module *v.net.salesman*:

```
def trails_salesman(trails, points, output):
   gscript.run_command('v.net', input=trails, points=points,
       output='tmp_net', operation='connect', threshold=10)
```

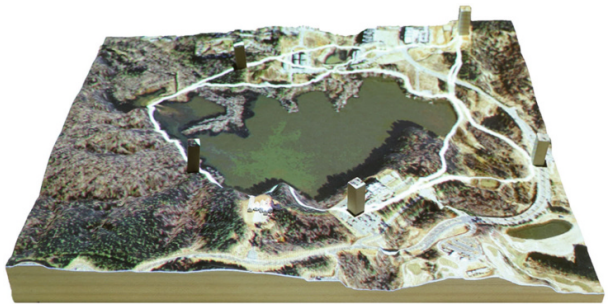


Fig. 7.4 A trail around Lake Raleigh

```
    cats = gscript.read_command('v.category', input='tmp_net',
        layer=2, option='print').strip().split()
    gscript.run_command('v.net.salesman', input='tmp_net',
        output=output, center_cats=','.join(cats), arc_layer=1,
        node_layer=2)
    # remove temporary map
    gscript.run_command('g.remove', type='vector',
        name='tmp_net', flags='f')
```

The final computation combines above defined functions and takes the scanned DEM and the detected points as input:

```
trails_combinations('scan', friction='friction',
    walk_coeff=[0.72, 6, 0, 6], lambda_=0.5, slope_factor=0,
    points='markers', vector_routes='route_net')
trails_salesman(trails='route_net', points='markers',
    output='route_salesman')
```

Examples of final output are shown in Figs. 7.1 and 7.4.

7.2.4 Mapping Trail Slopes

The simplest way to map the slope of a trail is to rasterize the trail using *v.to.rast* and then use *r.mapcalc* to extract the slope values from a slope raster computed with the module *r.slope.aspect*:

```
r.slope.aspect elevation=dem slope=slope
v.to.rast input=trail output=raster_trail type=line use=cat
r.mapcalc "trail_slope = if(raster_trail, slope)"
```

This extracts the slope values in the direction of the steepest slope. To compute the slope in the direction of the trail we first need to rasterize its vector representation by deriving raster values from the vector direction. The slope along the trail can then be computed as the slope multiplied by the cosine of the angle between the direction of the trail and the steepest slope (i.e. aspect):

```
r.slope.aspect elevation=dem slope=slope aspect=aspect
v.to.rast input=trail output=raster_trail_dir type=line use=dir
r.mapcalc "trails_slope_dir = abs(atan(tan(slope) * cos(aspect \
    - raster_trail_dir)))"
```

7.2.5 Alternative Trail Scenarios

We designed the initial trail scenario for Lake Raleigh Woods by placing markers at key points of interest—potential trailheads and scenic viewpoints (for more information about marker detection see Sect. 6.2.2). The siting of trailheads was informed by a previous 1:2500 scale study of potential routes around the entire lake (Fig. 7.4). We used realtime viewshed analysis to evaluate potential viewpoints (Fig. 7.5a). To test the viewsheds we cast the DSM in polymeric sand, carved clearings in the woodland canopy for viewpoints, placed markers to designate the viewpoints, and computed the viewsheds. We also computed the slope along the trail so that we could critique the routes (Fig. 7.5b).

Our trail needed to cross three large gullies with steep slopes. In the initial trail scenario the route follows the contours and ridge lines to avoid the gullies with their challenging slopes. This route formed two loops that intersected at the mouth of the central gully near the lakefront with moderately steep slopes (Fig. 7.6a). We, however, wanted to experiment and see what a route with a single loop would look like so we built bridges across the gullies (Fig. 7.6b). The trail rerouted across the bridges to form a single, shorter loop with a lower average slope (Fig. 7.6c).

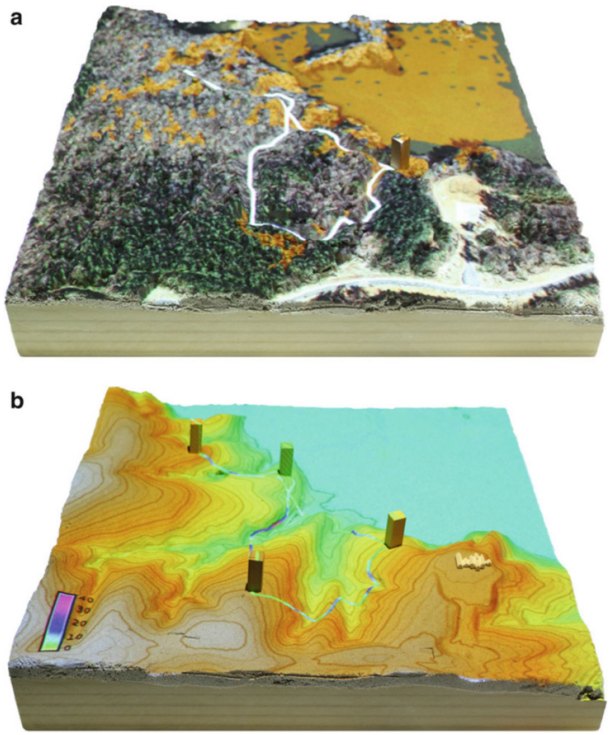


Fig. 7.5 Trail analytics: (**a**) the viewshed from a waypoint with areas visible represented in orange and (**b**) the slope of the trail

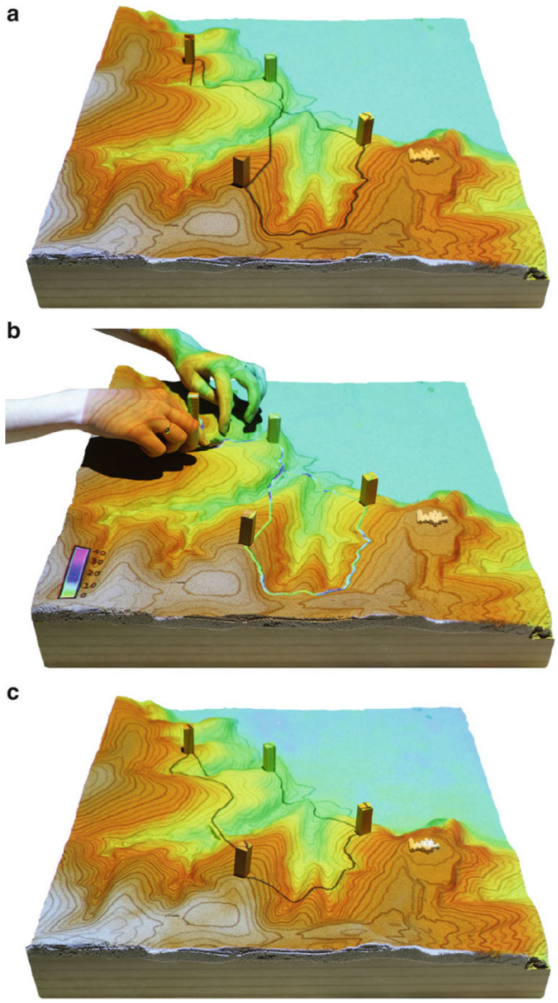


Fig. 7.6 Iteratively designing trail scenarios for Lake Raleigh Woods. (**a**) The initial route for a trail in Lake Raleigh Woods. (**b**) Building bridges to reroute the trail across gullies with steep slopes. (**c**) The updated route for the trail across the new bridges

References

Aitken, R. (1977). *Wilderness Areas in Scotland*. PhD thesis, University of Aberdeen.

Blank, G., Rudder, C., Dombrowski, A., Cser, H., Lawler, M., Kollar, C., Shifflett, S. D., Fortney, J., Sullivan, R., and Canavin, J. (2010). General Management Plan for Lake Raleigh Woods. Technical report, College of Natural Resources, North Carolina State University, Raleigh.

Langmuir, E. (1984). Mountaincraft and leadership. The Scottish Sports Council, Edinburgh.

Naismith, W. W. (1892). Untitled. Scottish Mountaineering Club Journal.

Chapter 8
Solar Radiation Dynamics

Solar radiation (insolation) is the primary driving force for Earth's atmospheric, biophysical, and hydrologic processes. Knowing the amount of radiation at different geographic locations at different times is therefore necessary in many fields including energy production, agriculture, meteorology, ecology, and urban planning. We modeled direct solar radiation and cast shadows in an urban setting to show how different spatial configurations of buildings change the amount of sunlight available throughout the day and the year.

8.1 Solar Radiation Modeling

Solar radiation modeling is critical for the optimal site selection of solar power plants (Carrion et al. 2008; Janke 2010) and for placing photovoltaic systems in often complex urban environments (Freitas et al. 2015; Hofierka and Kaňuk 2009; Jakubiec and Reinhart 2013). In urban design dynamic solar radiation simulations are used to study the effect of urban geometry and building configurations on solar access and shading conditions in order to get insight into the "urban canyon" phenomenon (Arnfield 1990; Lobaccaro and Frontini 2014).

On a global scale solar radiation depends spatially and temporally on the orientation of the Earth relative to the Sun. At a local scale solar radiation is influenced by topography (i.e. an elevation surface's inclination, orientation, and shadows), atmospheric conditions, and surface properties such as land cover. We can analyze the spatial distribution and dynamics of solar radiation and cast shadows using the equations that relate the sun position to the geometry of the terrain surface (Hofierka and Suri 2002). The clear-sky solar radiation model applied in this chapter is based on work undertaken for the development of the European Solar Radiation Atlas (Page et al. 2001; Rigollier et al. 2000; Scharmer and Greif 2000). We include here only the equations for direct solar radiation—beam irradiance—which

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_8

is influenced by the local topography. For a complete set of equations see Hofierka and Suri (2002). The diffuse and reflected components as well as atmospheric parameters can be also found in Rigollier et al. (2000).

First we compute position of the Sun in the sky as a function of time and location on Earth. The Sun declination angle depends on the day number (1–365 or 366):

$$\delta = \arcsin(0.3978 \sin(j' - 1.4 + 0.0355 \sin(j' - 0.0489))) \tag{8.1}$$

where δ is the Sun declination angle in radians and j' is expressed as day angle in radians:

$$j' = 360^\circ \cdot j/365.25, \quad j = 1, 2, \ldots, 365\,(366) \tag{8.2}$$

The position of the Sun in respect to a horizontal plane is defined by the solar altitude and solar azimuth. The solar altitude angle γ_s is a function of the solar hour angle ω, the solar declination angle δ and the latitude φ of the given location:

$$\sin \gamma_s = \cos \varphi \cos \delta \cos \omega + \sin \varphi \sin \delta \tag{8.3}$$

Given that the solar hour angle ω changes at 15° per hour and equals zero at noon it can be computed from local solar time t in decimal hours on the 24 h clock:

$$\omega = 15^\circ \cdot (t - 12) \tag{8.4}$$

The solar azimuth angle α_s (a horizontal angle between the sun and meridian measured from east) can be expressed as a function of the solar altitude angle γ_s, the solar declination angle δ, and the latitude φ of the given location:

$$\cos \alpha_s = (\sin \varphi \sin \gamma_s - \sin \delta)/ \cos \varphi \cos \gamma_s \tag{8.5}$$

$$\sin \alpha_s = \cos \delta \sin \omega / \cos \gamma_s \tag{8.6}$$

The position of the Sun in respect to an inclined plane with slope β and aspect α (e.g., a hillslope or a building roof) is described by a solar incidence angle ν:

$$\cos \nu = \cos \gamma_s \sin \beta \cos(\alpha_s - \alpha) + \sin \gamma_s \cos \beta \tag{8.7}$$

To estimate the duration of solar radiation we compute the sunrise and sunset angle $\omega_{r,s}$ over a horizontal plane:

$$\cos \omega_{r,s} = -\tan \varphi \tan \delta \tag{8.8}$$

and the corresponding time of sunrise T_{sr} and sunset T_{ss} in hours is then:

$$T_{sr} = 12 - \omega_{r,s}/15^\circ \qquad T_{ss} = 12 + \omega_{r,s}/15^\circ \tag{8.9}$$

The sunrise/sunset angle over a south-facing inclined plane (for a location in the northern hemisphere) is computed as follows:

$$\cos \omega'_{r,s} = -\tan(\varphi - \beta) \tan \delta, \qquad \omega''_{r,s} = \min(\omega_{r,s}, \omega'_{r,s}) \tag{8.10}$$

where $\omega''_{r,s}$ is the sunrise and sunset angle over the inclined plane. More general equations can be found in Hofierka and Suri (2002).

Solar irradiation analysis has been implemented in GRASS GIS by Hofierka and Suri (2002) in the module *r.sun*. In addition to the direct solar irradiation described above the module incorporates diffuse and reflected irradiation and cast shadows. The module *r.sun* works in two modes. In the first mode it computes the solar incident angle in degrees and solar irradiance values in $\mathrm{W\,m^{-2}}$ for a specific instance in time. In the second mode it computes the daily sum of solar irradiation in $\mathrm{Wh\,m^{-2}\,day^{-1}}$ and the duration of the direct irradiation. To compute a series of direct irradiance maps for a given day with a selected time interval we use the add-on *r.sun.hourly*, which calls *r.sun* in a convenient loop. Similarly we use the add-on *r.sun.daily* to compute daily sums of direct irradiation for the entire year.

8.2 Case Study: Solar Irradiation in Urban Environment

In this case study we studied how different spatial configuration of buildings changed direct solar radiation and cast shadows dynamics throughout the day. We placed wooden blocks of different shapes and sizes representing hypothetical buildings on a flat surface with an approximate scale of 1:300. We located this abstract site in Raleigh, North Carolina at latitude 36°. We examined two building configurations—one with four medium-sized houses with gabled roofs and the other with three medium-sized houses with gabled roofs and one tall building with a flat roof (Fig. 8.1a, b).

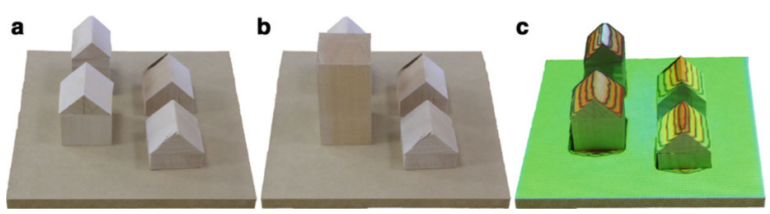


Fig. 8.1 Two configurations of buildings at approximately 1:300 scale viewed from the south. (**c**) Shows the DSM and contours derived from a scan and projected over buildings

8.2.1 The Impact of Building Configuration on Cast Shadows

We compared the shadows cast by the buildings in both configurations during summer and winter solstice. After scanning the scenes we used the add-on *r.sun.hourly* to derive binary raster time series of cast shadows from 6:00 to 22:00 (local solar time) with a 30-minute time step. We had to specify the selected date as a day of year number. For example if we pick the summer solstice in 2015 we can easily compute the day of year using the Python *datetime* module:

```
import datetime
datetime.datetime(2015, 6, 21).timetuple().tm_yday
```

which gives us the number 172.

Running the following commands results in a time series of binary rasters with zeros representing cast shadow based on the direct (beam) radiation. The time series is registered in a spatio-temporal dataset and can be easily rendered as an animation and projected over the physical model.

```
r.slope.aspect elevation=scan aspect=aspect slope=slope
r.sun.hourly -t -b elevation=scan aspect=aspect slope=slope \
    start_time=6 end_time=22 time_step=0.5 day=172 year=2015 \
    beam_rad_basename=shadows_summer
g.gui.animation strds=shadows_summer
```

By changing the day parameter to 356 we obtained the time series of cast shadows for the winter solstice. Figure 8.2 compares the cast shadows at 16:30 and 19:00 (local solar time) during the summer and winter solstice for the two different building configurations.

There is a clear difference in the length of cast shadows during winter and summer. The orientation of the cast shadows at the time close to sunset changes during the year as shown in Fig. 8.2a, c. During the summer the Sun sets below the horizon later (after 19:00) far to the northwest, while it sets earlier (after 16:30) and far to the southwest during winter. As a result the house in the southeast corner only has a view of the sunset during part of the year as the summer view is blocked by another building.

By adding a tall building to the scene we cast more shadow in the space between the houses and on houses in the east. Depending upon the season one of the houses on the east side becomes shadowed by the high building sooner than the other.

8.2.2 The Impact of Building Configuration on Direct Solar Irradiation

Cast shadows give us information about the availability of direct solar radiation at a particular time on particular days. However, we are often interested in the cumulative solar irradiation, which tells us the amount of solar energy received on

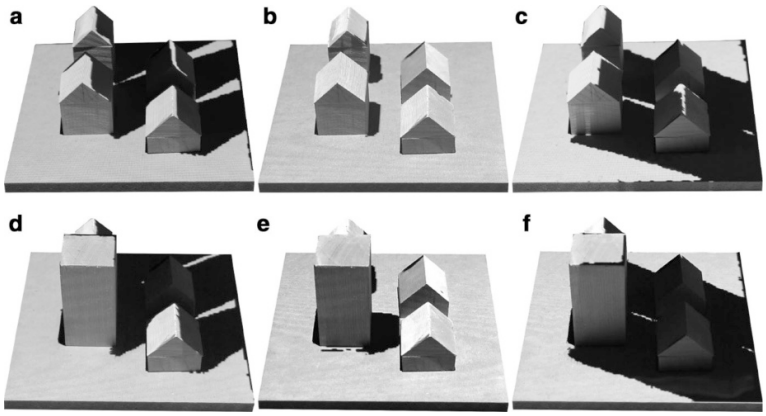


Fig. 8.2 A comparison of cast shadows viewed from south. The *first and second rows* show two different building configurations. The *first and second columns* show cast shadows at 16:30 local solar time during the winter (**a**), (**d**) and summer solstice (**b**), (**e**), respectively. The *third column* shows cast shadows at 19:00 local solar time during the summer solstice. A figure for winter solstice at 19:00 is not shown as it would be after sunset

a given surface during a given time interval. Such information can then be used to assess the suitability of a location for solar energy applications.

To compute direct solar irradiation for individual days we called the add-on *r.sun.daily* with parameters similar to those we used for *r.sun.hourly*. We specified the first and last day of the year resulting in 365 daily irradiation maps. Additionally by specifying the `beam_rad` parameter we computed the sum of all of the daily maps. Again we can animate the time series projected over the scene to get more insight into the spatial patterns of solar radiation over the course of a year.

```
r.slope.aspect elevation=scan aspect=aspect slope=slope
r.sun.daily -t elevation=scan aspect=aspect slope=slope \
    start_day=1 end_day=365 day_step=1 \
    beam_rad=radiation_year_sum \
    beam_rad_basename=radiation_day_sum
g.gui.animation strds=radiation_day_sum
```

Figure 8.3 shows direct solar irradiation in $\mathrm{Wh\,m^{-2}\,day^{-1}}$ during the winter solstice, spring equinox, and summer solstice with a common color ramp. The color ramp was specified in a plain text file `solar_color.txt` as:

```
100% red
70% yellow
0% gray
```

and then this color ramp was set for the entire time series:

```
t.rast.colors input=radiation_day_sum rules=solar_color.txt
```

Fig. 8.3 Direct solar irradiation in Wh m^{-2} day^{-1} during the winter solstice, spring equinox, and summer solstice

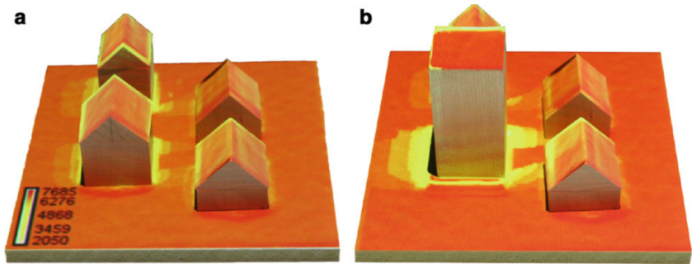


Fig. 8.4 Solar irradiation patterns in March for different building configurations

Figure 8.4a shows the solar irradiation on a particular day in March in order to compare the cumulative solar irradiation for the two building configurations. In the first scenario there is variable irradiation in the space between the buildings as a result of the interaction between the spatial configuration of the buildings and the dynamic solar geometry. In the second scenario the pattern is even more pronounced because the street gets less direct sunlight. The slope of the buildings' roofs affects the amount of irradiation as a visual comparison of the color of the flat and tilted roofs clearly demonstrates. Furthermore, the shadow cast by the tall building has lowered the irradiation values on part of the roof of the southeastern house.

References

Arnfield, A. (1990). Street design and urban canyon solar access. *Energy and Buildings*, 14(2):117–131.

Carrion, J. A., Estrella, A. E., Dols, F. A., Toro, M. Z., Rodríguez, M., and Ridao, A. R. (2008). Environmental decision-support systems for evaluating the carrying capacity of land areas: Optimal site selection for grid-connected photovoltaic power plants. *Renewable and Sustainable Energy Reviews*, 12(9):2358–2380.

Freitas, S., Catita, C., Redweik, P., and Brito, M. (2015). Modelling solar potential in the urban environment: State-of-the-art review. *Renewable and Sustainable Energy Reviews*, 41:915–931.

Hofierka, J. and Kaňuk, J. (2009). Assessment of photovoltaic potential in urban areas using open-source solar radiation tools. *Renewable Energy*, 34(10):2206–2214.

Hofierka, J. and Suri, M. (2002). The solar radiation model for Open source GIS: implementation and applications. In *Proceedings of the Open source GIS-GRASS users conference*, pages 1–19.

Jakubiec, J. A. and Reinhart, C. F. (2013). A method for predicting city-wide electricity gains from photovoltaic panels based on LiDAR and GIS data combined with hourly Daysim simulations. *Solar Energy*, 93:127–143.

Janke, J. R. (2010). Multicriteria GIS modeling of wind and solar farms in Colorado. *Renewable Energy*, 35(10):2228–2234.

Lobaccaro, G. and Frontini, F. (2014). Solar energy in urban environment: how urban densification affects existing buildings. *Energy Procedia*, 48:1559–1569.

Page, J., Albuisson, M., and Wald, L. (2001). The European Solar Radiation Atlas: a valuable digital tool. *Solar Energy*, 71(1):81–83.

Rigollier, C., Bauer, O., and Wald, L. (2000). On the clear sky model of the ESRA – European Solar Radiation Atlas – with respect to the heliosat method. *Solar Energy*, 68(1):33–48.

Scharmer, K. and Greif, J. (2000). *The European solar radiation atlas, vol. 2: database and exploitation software*. Presses del'Ecolle des Mines de Paris.

Chapter 9
Wildfire Spread Simulation

Forest fires, whether naturally occurring or prescribed, are potential risks for ecosystems and human settlements. These risks can be managed by monitoring the weather, prescribing fires to limit available fuel, and creating firebreaks. With computer simulations we can predict and explore how fires may spread. We can explore scenarios and test different fire management strategies under different weather conditions. Using Tangible Landscape and the GRASS GIS wildfire toolset we simulated several wildfire scenarios. We tested different configurations of firebreaks on the physical model and evaluated their effectiveness.

9.1 Fire Spread Modeling Methods

Fire spread across landscapes is a complex, highly dynamic process influenced by weather, topography, and fuel. The fire spread models predict the rate of fire spread and fire perimeter growth using a combination of physical principles such as energy conservation and empirical parameters derived from observations and experimental data. For reviews of physical and empirical models of fire spread and methods of implementing fire simulation please refer to Sullivan (2009a,b,c).

9.1.1 Input Data

The fuel model is one of the most important input variables for fire simulation. Albini (1976) and Anderson (1982) describe 13 classes of fuel (Table 9.1), which differ in fuel loads and the distribution of fuel particle size classes. The size of individual pieces of fuel influences the fire; heat is absorbed faster by small twigs

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_9

Table 9.1 The fuel model by Anderson (1982) describes 13 classes of fuel

Fuel class	Description
Grass and grass-dominated	
1	Short grass (1 foot)
2	Timber (grass and understory)
3	Tall grass (2.5 feet)
Chaparral and shrub fields	
4	Chaparral (6 feet)
5	Brush (2 feet)
6	Dormant brush, hardwood slash
7	Southern rough
Timber litter	
8	Closed timber litter
9	Hardwood litter
10	Timber (litter and understory)
Slash	
11	Light logging slash
12	Medium logging slash
13	Heavy logging slash

due to their large surface to volume ratio. The moisture content of the fuel affects the amount of heat needed to ignite the fuel. The smaller the fuel size, the faster the fuel dries out and becomes more combustible.

The slope of the terrain and the speed and direction of the wind control the rate of spread and direction of the fire. Steeper slopes cause faster ignition in the upslope direction. This rule has been observed and quantified in several laboratory experiments (Silvani et al. 2012; Viegas 2004; Weise and Biging 1996).

Wind is specified by speed and direction at a given height. Unlike the slope of the terrain, wind is difficult to characterize due to its variability in space and time and spatially and temporally averaged values are used in the models. Sparks and embers can be carried great distances by strong winds; this process known as spotting allows fire to spread beyond firebreaks (Albini 1983).

9.1.2 Fire Spread Algorithm

In our simulation we used the GRASS GIS modules *r.ros* and *r.spread*, implemented by Xu (1994), to compute the rate of spread and simulate the spread of fire.

The rate of spread computation is based on the BEHAVE model (Andrews 1986). The inputs are the fuel model, fuel moisture, wind speed and direction and terrain slope and aspect. The rate of spread is computed using the following equation:

$$R = \frac{I_R \xi (1 + \Phi_W + \Phi_S)}{\rho_b \varepsilon Q_{ig}} \tag{9.1}$$

where:

R	is the rate of spread in m/s
I_R	is the reaction intensity in kW/m
ξ	is the propagating flux ratio
Φ_W	is the wind coefficient
Φ_S	is the slope factor
ρ_b	is the oven-dry fuel per cubic meter of fuel bed in kg/m^3
ε	is the effective heating number
Q_{ig}	is the heat of preignition in kJ/kg.

For a detailed description of the underlying mathematical model and an explanation of the input data refer to Rothermel (1972).

The module *r.spread* uses Huygens' principle to simulate elliptically anisotropic spread where each cell center is a potential origin of spread and the local spread is ellipsoidal (Anderson et al. 1982). The sizes and orientations of the ellipses vary by cell. Local wind and slope directions determine the orientation of ellipses. The module *r.spread* uses a specific implementation of the shortest path algorithm.

We used the GRASS GIS add-on module *r.fire.spread* to call the modules *r.ros* and *r.spread* streamlining the fire spread computation and visualization. With this add-on we can use temporally variable input conditions and export the fire spread for each time step so that we can animate the simulation with the module *g.gui.animation*.

9.2 Case Study: Controlling Fire with Firebreaks

With Tangible Landscape we can intuitively create new fire management scenarios, rapidly testing how different sizes, shapes, and placements of firebreaks affect the spread of fire. To test different management strategies we simulated how a fire might spread across NCSU's Centennial Campus and tried to contain it by creating firebreaks of different sizes and alignments. In this case study we modeled a phenomenon—fire spread—that started outside the extent of the physical model, but spread towards the model (Fig. 9.1). Once the fire spread onto the model we could interact with it by creating firebreaks. Geospatial models and simulations need not be constrained to the boundary of the physical model, i.e. the extent of the tangible user interface. The processing extent can extend beyond the interactive area allowing us to work across scales.

We built a malleable physical model of the tree canopy by casting polymeric sand. To build the molds for casting we interpolated a DEM and a DSM from 2013 lidar data and then CNC routed the DEM and the inverse of the DSM. Then we cast a

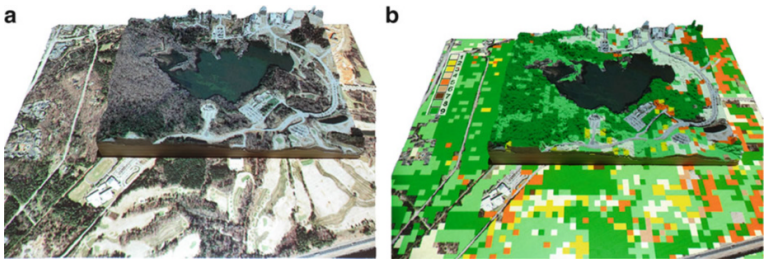


Fig. 9.1 The study area on Centennial Campus: (**a**) an orthophotograph and (**b**) available fuel map projected over the table with the tangible 3D model

layer of sand representing the tree canopy between the two molds. Since this layer of sand represents tree canopy and thus fuel availability we could model clearcutting simply by removing sand. Thus we were able to intuitively design firebreaks by carving into the sand layer and reducing the fuel load. Please refer to Sect. 6.2.1 for a more detailed description of the site and model.

9.2.1 Data Preparation

To simulate fire spread we prepared several input raster layers for the module *r.ros* including the data about terrain, wind, fuel, and moisture conditions. We also specified the coordinates of location where the fire started.

Terrain Terrain slope influences the speed of spread and aspect influences the spread direction. After we set the region to our study area we derived the slope and aspect raster maps from the provided DEM:

```
g.region n=224134 s=222501 e=639326 w=637211 res=3
r.slope.aspect elevation=dem slope=slope aspect=aspect
```

The DEM is also used as input for computing the maximum spotting distance.

Wind Wind has two components—midflame velocity and direction—and both can be spatially variable. In this case we used the prevailing wind speed and direction obtained from the nearest weather station through the State Climate Office of North Carolina.[1] The module *r.ros* requires velocity in feet per minute so the wind velocity data acquired from the State Climate Office of North Carolina must be converted from meters per second to feet per minute (to convert multiply by a factor of approximately 197). Wind direction is typically reported by the direction *from* which

[1]State Climate Office of North Carolina: www.nc-climate.ncsu.edu/

it originates, clockwise from the north. The module *r.ros*, however, requires the "to" direction (also clockwise from north). To simulate spatial variability we applied a random effect, for example, using the module *r.surf.gauss* to produce a raster map of Gaussian deviates with a specified mean and standard deviation:

```
r.surf.gauss output=wind_speed_avg mean=542 sigma=30
r.surf.gauss output=wind_dir mean=75 sigma=20
```

Fuel The module *r.ros* uses Anderson's 13 standard fire behavior fuel models. Fuel data layers at 30 m resolution for the USA are publicly accessible via the LANDFIRE website.[2] Most of our study site falls into fuel classes 8 and 9, i.e. timber litter (Fig. 9.1b).

Moisture Since fuel moisture data were not readily available for our site we generated the required, spatially variable raster maps of 1-h fuel moisture and live fuel moisture percentage for dry conditions:

```
r.surf.gauss output=moisture_1h mean=10 sigma=5
r.surf.gauss output=moisture_live mean=20 sigma=5
```

Starting Location The starting sources of the fire are represented as raster cells and can be created by digitizing points or importing coordinates from a file and converting them to a raster. We provided the fire starting location in a text file with the point coordinates (for example `638097,222934`) and converted them to a raster with the module *v.to.rast*:

```
v.in.ascii input=source.txt output=source separator=comma
v.to.rast input=source output=source type=point use=cat
```

9.2.2 Scenario with Multiple Firebreaks

In our first scenario the fire started near Trailwood Drive and was drawn to the northeast by a southwesterly wind. The fire jumped over the road and quickly spread towards the Chancellor's House (Fig. 9.2).

Using the following workflow we simulated and then animated the spread of the fire. The output raster maps generated by this workflow represent the relative time of ignition in minutes. To better visually represent the spread of the fire we used a red-yellow-gray color ramp. The steepness of the color gradient shows the relative rate of spread. The steeper the color gradient, the slower the spread. To prepare the suggested color ramp we saved the color rules in a plain text file `fire_colors.txt`:

[2]LANDFIRE: http://www.landfire.gov/

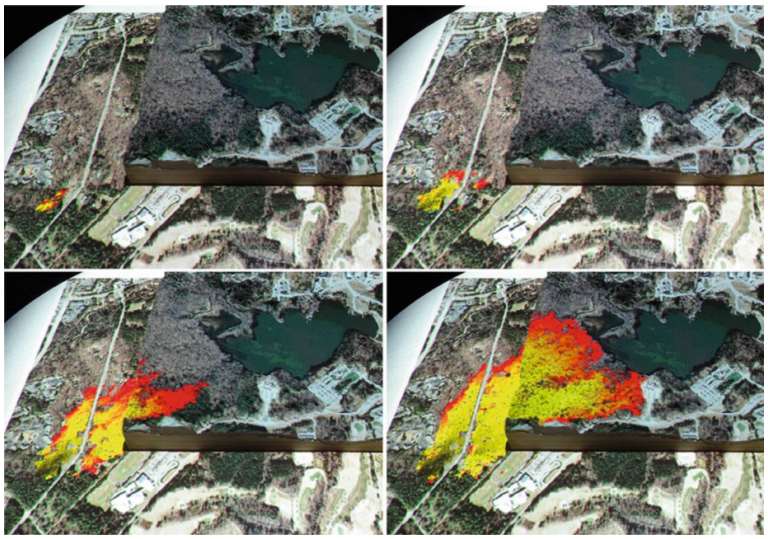


Fig. 9.2 The spread of the fire without intervention

```
0% 50:50:50
60% yellow
100% red
```

To run the simulation we called the module *r.ros* once and then *r.spread* with the desired time lag parameter specifying the length of the simulation. To see the intermediate states we can run the module *r.spread* multiple times and assign our color ramp to the resulting spread raster map:

```
r.ros model=fuel moisture_1h=moisture_1h \
    moisture_live=moisture_live velocity=wind_speed_avg \
    direction=wind_dir slope=slope aspect=aspect \
    elevation=elevation base_ros=out_base_ros \
    max_ros=out_max_ros direction_ros=out_dir_ros \
    spotting_distance=out_spotting

r.spread -s base_ros=out_dir_ros max_ros=out_max_ros \
    direction_ros=out_dir_ros start=source \
    spotting_distance=out_spotting wind_speed=wind_speed_avg \
    fuel_moisture=moisture_1h output=spread_1 lag=20
r.spread -s -i base_ros=out_dir_ros max_ros=out_max_ros \
    direction_ros=out_dir_ros start=spread_1 \
    spotting_distance=out_spotting wind_speed=wind_speed_avg \
    fuel_moisture=moisture_1h output=spread_2 lag=20
r.spread ...
```

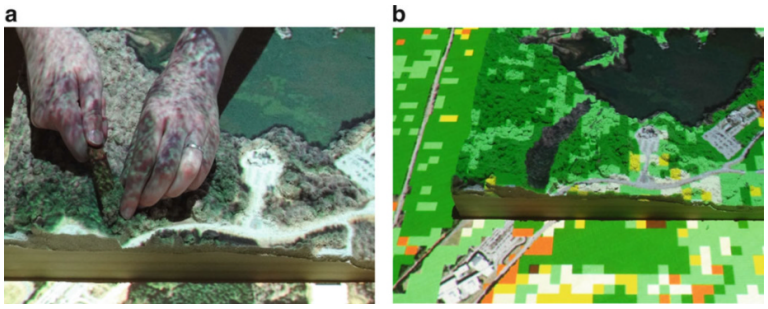


Fig. 9.3 Creating a firebreak by (**a**) manually removing sand and then (**b**) scanning and detecting the change

```
r.null map=spread_1 setnull=0
r.colors map=spread_1 rules=fire_colors.txt
```

Alternatively we can use the add-on module *r.fire.spread*, which conveniently wraps the previous sequence of commands into a single command:

```
r.fire.spread -s start=source times=0 end_time=1600 \
    time_step=20 output=spread model=fuel \
    moisture_1h=moisture_1h moisture_live=moisture_live \
    direction=wind_dir slope=slope aspect=aspect \
    elevation=elevation speed=wind_speed_avg
```

After simulating the initial spread of the fire we attempted to prevent fire from spreading towards Chancellor's House by creating firebreaks. First we scanned the model to save the unmodified state. Next we manually removed sand (representing canopy) from the location where we want to have a firebreak (see Fig. 9.3a). We scanned the modified model and vertically matched the new scan to the unmodified scan using the function defined in code snippet 4.2.2. We derived the new fuel raster layer based on the difference between the two scans introducing no data values into the copy of the original fuel model using a simple raster algebra expression:

```
adjust_scan('scan_before', 'scan_after', 'scan_adjusted')
gscript.mapcalc("changed_fuel = if(scan_before - scan_adjusted
    > 0, null(), fuel)")
```

We repeated our simulation using the new fuel raster layer. The resulting fire spread in Fig. 9.4 shows that our attempt was only a partial success. While we significantly slowed the spread of the fire, potentially giving firefighters more time to act, the fire jumped over the firebreak and started to spread towards Chancellor's House suggesting that a wider firebreak would be needed. With more precise data about fire behavior we could control and limit the spotting effect in the simulation.

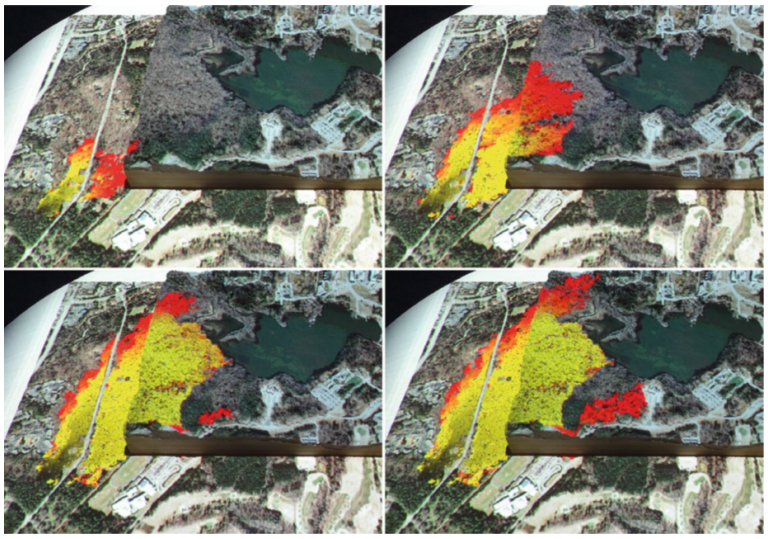


Fig. 9.4 The spread of the fire after creating a firebreak

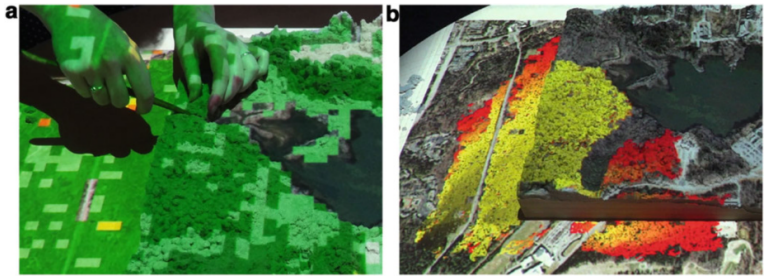


Fig. 9.5 The simulation with additional firebreak: (**a**) the creation of a new firebreak by removing sand and (**b**) the result of the new simulation

In our second scenario we added another firebreak to protect the neighborhood north of Lake Raleigh. The lake creates a natural barrier that constrains the spread of the fire. To protect the community to the north we extended this barrier by creating an additional firebreak starting on western shore of the lake (Fig. 9.5a). After we rescanned the model and reran the simulation, we found that the additional firebreak was only locally effective—while the fire was significantly slowed, it would have eventually reached the neighborhood.

References

Albini, F. A. (1976). Estimating wildfire behavior and effects. Technical report, Intermountain Forest and Range Experiment Station, Forest Service, U.S. Dept. of Agriculture.

Albini, F. A. (1983). Potential spotting distance from wind-driven surface fires. Technical report, Intermountain Forest and Range Experiment Station, Forest Service, U.S. Dept. of Agriculture.

Anderson, D. H., Catchpole, E. A., De Mestre, N. J., and Parkes, T. (1982). Modelling the spread of grass fires. *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 23:451–466.

Anderson, H. E. (1982). Aids to determining fuel models for estimating fire behavior. Technical report, Intermountain Forest and Range Experiment Station, Forest Service, U.S. Dept. of Agriculture.

Andrews, P. L. (1986). BEHAVE: fire behavior prediction and fuel modeling system – BURN subsystem, Part 1. page 130.

Rothermel, R. C. (1972). A mathematical model for predicting fire spread in wildland fuels. Technical report, Intermountain Forest and Range Experiment Station, Forest Service, U.S. Dept. of Agriculture.

Silvani, X., Morandini, F., and Dupuy, J.-L. (2012). Effects of slope on fire spread observed through video images and multiple-point thermal measurements. *Experimental Thermal and Fluid Science*, 41:99–111.

Sullivan, A. L. (2009a). Wildland surface fire spread modelling, 1990–2007. 1: Physical and quasi-physical models. *International Journal of Wildland Fire*, 18(4):349–368.

Sullivan, A. L. (2009b). Wildland surface fire spread modelling, 1990–2007. 2: Empirical and quasi-empirical models. *International Journal of Wildland Fire*, 18(4):369–386.

Sullivan, A. L. (2009c). Wildland surface fire spread modelling, 1990–2007. 3: Simulation and mathematical analogue models. *International Journal of Wildland Fire*, 18(4):387–403.

Viegas, D. X. (2004). Slope and wind effects on fire propagation. *International Journal of Wildland Fire*, 13(2):143–156.

Weise, D. R. and Biging, G. S. (1996). Effects of wind velocity and slope on flame properties. *Canadian Journal of Forest Research*, 26(10):1849–1858.

Xu, J. (1994). *Simulating the spread of wildfires using a geographic information system and remote sensing*. Phd dissertation, Rutgers University, New Brunswick.

Chapter 10
Coastal Modeling

In this chapter we discuss how to simulate inundation and flooding in coastal landscapes, explain workflows for developing storm surge and sea level rise scenarios, and present two case studies. The first case study is a series of storm surge and dune breach scenarios for a populated barrier island. The second is a design—informed by flood and erosion modeling—for resilient, sustainable guest housing at a coastal research institute.

10.1 Modeling Potential Inundation

Inundation and flooding in coastal landscapes is caused by storm surge and sea level rise. It is very complex and depends upon topography and bathymetry, tides, sea level surge and waves, and wind and rainfall. The spatial extent of flooding can be approximated by "spreading" the observed or predicted water level throughout the landscape (Clinch et al. 2012).

To approximate inundation and flooding we can spread the given water level from a predefined seed located in the ocean using the module *r.lake*. This approach respects the obstacles, so if a foredune protects a low lying area, this area will not be flooded even if the water level exceeds its elevation. We describe the algorithm as follows: an increase in water level approximates the dynamics of inundation, but we assume that the water spreads at an infinite speed and thus floods everything that it can reach instantly. Therefore this method is best used to map the maximum possible extent assuming that there are no changes in topography during the process.

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4_10

10.2 Case Study: Simulating Flood Risks for Barrier Island Communities

We explored the impact of topography on inundation and flooding at the Jockey's Ridge sand dune complex in Nags Head on the Outer Banks barrier islands of North Carolina (Fig. 10.1) with a series of a storm surge scenarios. We carved breaches into the foredune of our sand model to identify which homes were most vulnerable to storms.

For this study we selected a study area including the beach, foredune, homes, road, dune complex, maritime forest, and soundside shore. We derived a bare earth DEM and a DSM with buildings and vegetation for our study from a 2009 lidar survey. Then we cast a polymeric sand model of the DEM from a CNC routed mold. We projected an orthophotograph, elevation data, the shoreline, and building footprints as context. Based on the building footprints we added massing models of the buildings to our sand model.

To study the potential impacts of a hurricane we simulated storm surge at increasing levels from 1 to 4 m using the add-on module *r.lake.series*. Then we breached the foredune at a point where it has been compromised by the construction of boardwalk for beach access and compared the extent of flooding at different surge levels (Fig. 10.2).

```
r.lake.series elevation=elev_2009_1m output=flood \
    start_water_level=1 end_water_level=4 water_level_step=0.1 \
    coordinates=913525,249507
g.gui.animation strds=flood
```

Fig. 10.1 The Jockey's Ridge dune complex in Nags Head, North Carolina

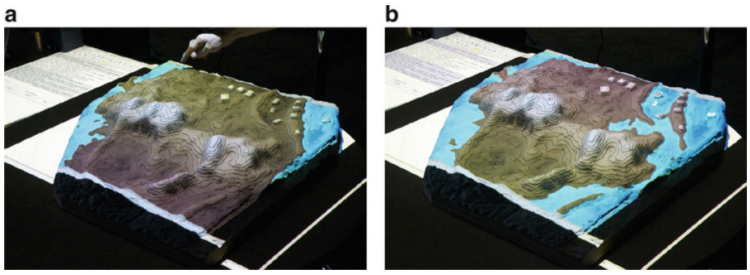


Fig. 10.2 Inundation extent simulations for Jockey's Ridge area for (**a**) 1 m and (**b**) 2 m of storm surge

10.3 Case Study: Designing Resilient Coastal Architecture

Two architecture students—David Koontz and Faustine Pastor—used Tangible Landscape to site a series of residential pavilions (Fig. 10.3) for visitors staying at the UNC Coastal Studies Institute on Roanoke Island, North Carolina as part of an joint architecture-engineering studio. They used storm surge and water flow simulations to inform the selection of building sites. Then they spaced pilings for the buildings and adaptively graded a biofiltration swale based on near real-time feedback about water flow, sediment flux, and erosion-deposition.[1]

The UNC Coastal Studies Institute—the site for the studio—is a research institute on Roanoke Island in the Outer Banks of North Carolina. Students in this North Carolina State University studio course led by Professors Joe DeCarolis, David Hill, and Ranji Ranjithan were tasked with designing sustainable housing for guests and researchers visiting the institute.[2] The site is at risk of inundation from sea level rise and storm surge and is in a sensitive back-barrier salt marsh.

Given the risk of flooding David and Faustine used Tangible Landscape to visualize different levels of inundation and test the vulnerability of potential building sites. They placed basswood massing models of buildings on a cast sand model of the landscape and then used the module *r.lake* to simulate 4 m storm surge and check if their buildings would stay dry (Fig. 10.4). In their design the buildings would be raised on piers and connected by boardwalks over a bioswale. They simulated water flow to find the right spacing for the piers over swale. Then they adaptively graded the swale using feedback from water flow, erosion-deposition, and sediment flux simulations, sculpting the channel so that water and sediment flowed towards a bioretention pond (Fig. 10.5).

[1] Watch the video on Youtube: https://youtu.be/PbbzWymGvLo

[2] Read more about the studio on the Coastal Dynamics Design Lab website: http://design.ncsu.edu/coastal-dynamics-design-lab/

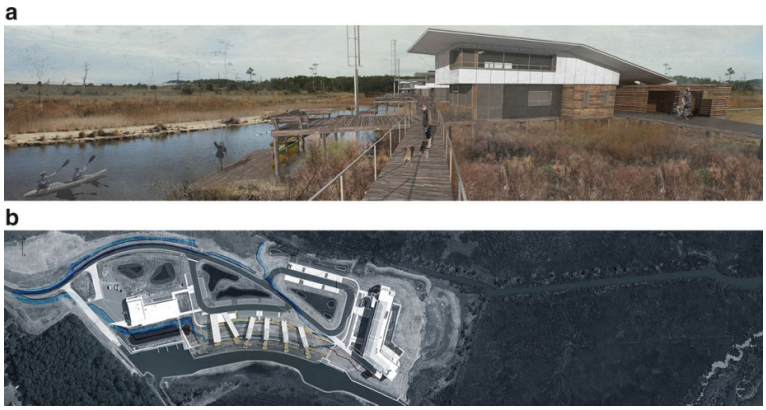


Fig. 10.3 The students' design for a series of pavilions for the UNC Coastal Studies Institute on Roanoke Island, NC. Images by David Koontz and Faustine Pastor. (**a**) A rendering of the proposed design for a residential pavilion. (**b**) A masterplan of the UNC Coastal Studies Institute with the proposed residential pavilions

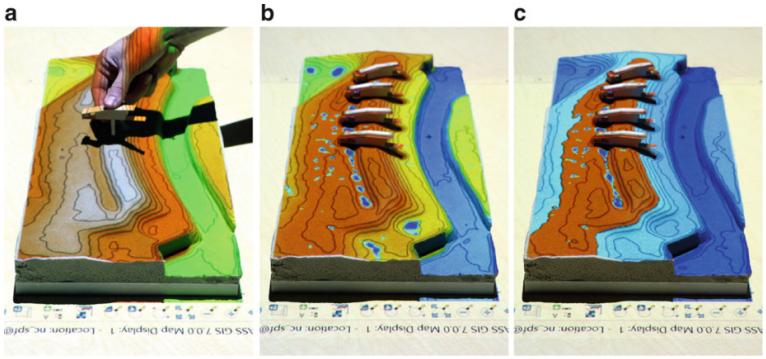


Fig. 10.4 Students using water flow and flood simulations to adaptively inform the placement of a series of pavilions for the Coastal Studies Institute on Roanoke Island, NC: (**a**) siting buildings by hand on an artificial ridge, (**b**) studying the flow of water around the buildings and (**c**) testing whether the buildings stay dry with 4 m storm surge

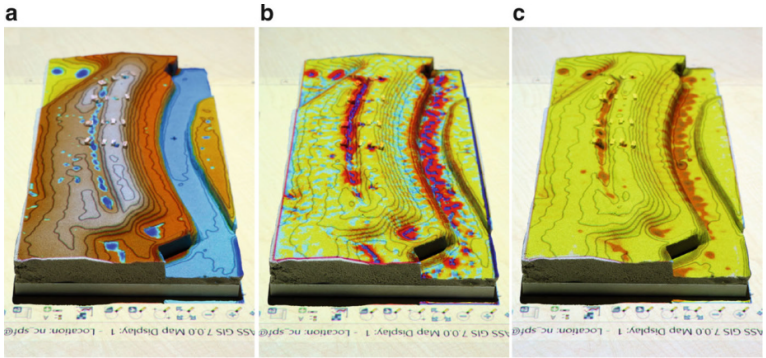


Fig. 10.5 Students using (**a**) water flow, (**b**) erosion-deposition, and (**c**) sediment flux simulations to space structural piers and adaptively grade a biofiltration swale for the Coastal Studies Institute on Roanoke Island, NC

Reference

Clinch, A. S., Russ, E. R., Oliver, R. C., Mitasova, H., and Overton, M. F. (2012). Hurricane Irene and the Pea Island breach: prestorm site characterization and storm surge estimation using geospatial technologies. *Shore & Beach*, 80(2):38–46.

Appendix A

A.1 Applications of Tangible Landscape

In this section we briefly describe additional applications of Tangible Landscape that we, our students, and our colleagues have developed. For each project we explain the aim, used methods and studied processes.

Modeling Avalanches in High Tatras

Eva Stopkova modeled snow avalanches in the High Tatras mountain range in Slovakia (Stopková 2007, 2008). Avalanche potential is closely related to landscape topography so Stopkova built a physical model of her study area in High Tatras in order to visualize the spatial pattern of topographic parameters and landforms (Fig. A.1). She was able to project the results of avalanche danger assessment over the model, but she was not able usefully compute the assessment using the modified and scanned physical model. The scanned models were not accurate enough for practical applications due to their small scale and the steepness of their slopes. Modeling avalanches with Tangible Landscape would require a larger physical model at a larger map scale. CNC routing can be used to precisely and efficiently fabricate the size of model needed for such an application.

Visualizing the Evolution of Oregon Inlet

Liliana Velasquez Montoya created a projection augmented sand model of Oregon Inlet where she was modeling how storms drive the morphological evolution of barrier islands. To visualize and present the results of her research Montoya

© The Author(s) 2015

A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4

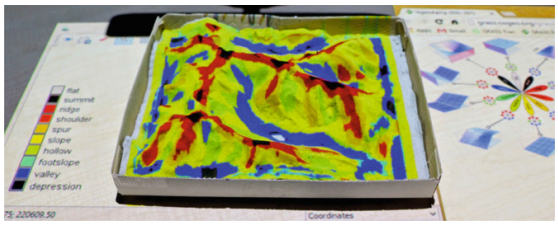


Fig. A.1 A sand model of High Tatras in Slovakia augmented with a projection of the landforms computed by the module *r.geomorphon*; a legend and a diagram of the landform types are projected on the table beside the model

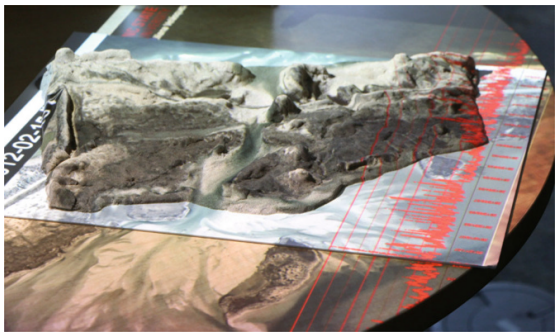


Fig. A.2 A sand model of Oregon Inlet with a projected video showing its formation and evolution along with a graph of wave height changing over time

projected a video depicting the evolution of the inlet in response to storm events onto a sand model. She built the sand model to physically represent the present 3D form of the inlet. Since the sand model was not dynamic it only matched the end of animation. This was a useful mode of representation as the model acted as a datum that helped us compare the present and past morphology of the inlet. Since the small sand model was only large enough to show the most active part of the inlet she also projected the surrounding coastal landscape (Fig. A.2).

Designing Disaster Relief Housing for Rodanthe

Logan Free designed disaster relief shelters for an emergency ferry terminal at Rodanthe on the Outer Banks of North Carolina. The ferry terminal is used for evacuation and relief when there are disasters like hurricanes or bridge failures. Free was tasked with designing temporary housing structures and a community service

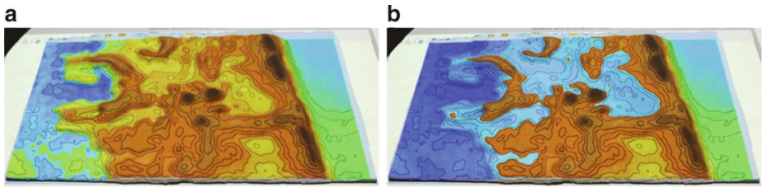


Fig. A.3 Simulating storm surge at the Rodanthe emergency ferry terminal on the Outer Banks of North Carolina: (**a**) the mean high water level on the sound side of the island and (**b**) the approximate water level during high storm surge

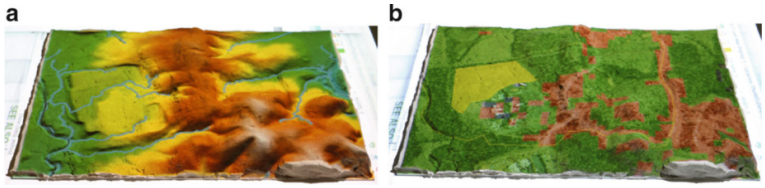


Fig. A.4 Landscape change modeling for a watershed in Charlotte, North Carolina: (**a**) a sand model of the terrain at the Charlotte study area augmented with a projected hillshade to enhance perception of relief; (**b**) using a mouse and the digitizer tool to draw a polygon to designate a development area

center for disaster recovery at the ferry terminal. To identify vulnerable hotspots and select safe sites for the shelters he built a sand model of the site and used Tangible Landscape with the module *r.lake* to simulate the potential extent of storm surge flooding (Fig. A.3).

Simulating Landscape Change in Charlotte

Douglas A. Shoemaker created a model of a small watershed in Charlotte, North Carolina to test how the FUTure Urban-Regional Environment Simulation (FUTURES) framework could be integrated with Tangible Landscape. FUTURES uses a stochastic patch growing algorithm to simulate urban growth and land use change (Meentemeyer et al. 2013). It has been integrated with GRASS GIS as the add-on module *r.futures* (Petrasova et al. 2015). Shoemaker used the raster digitizer tool in the GRASS GIS GUI to designate a site for development and change the land cover. He also sculpted the sand to grade the topography for the new development. Then he ran FUTURES to explore how the new development would influence longterm landscape change in the watershed-scale study area (Fig. A.4).

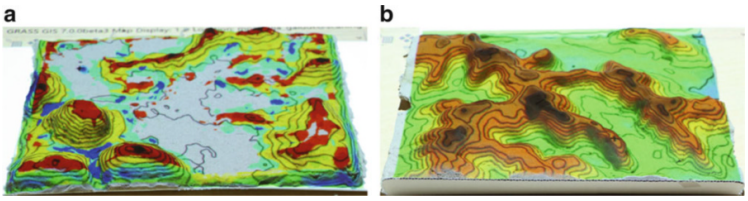


Fig. A.5 Reconstructing a paleolake and a terrain model used for cell tower planning. (**a**) A reconstruction of a paleolake in the Galuut Valley, Mongolia showing the landforms computed by *r.geomorphon*. (**b**) A physical model used to site cell towers in Athens county, Ohio

Reconstructing a Paleolake in Mongolia

Gantulga Bayasgalan used a sand model to reconstruct a paleolake in the Galuut Valley in central southwest Mongolia. He modeled alternative reconstructions of the landscape in sand and used Tangible Landscape to compute the flow accumulation pattern, geomorphological features, and several other characteristics for each landscape configuration (Fig. A.5a).

Cell Tower Planning in Athens County

Jesse Boyd used Tangible Landscape to site cell towers in Athens county, Ohio. To find good sites for cell towers, Boyd tested different spatial configurations by placing markers representing cell towers on a physical model of the landscape (Fig. A.5b). Since topography blocks cellular signals he used viewshed analysis to estimate cellular coverage. He experimented with different cell tower heights by changing the observer elevation parameter of the module *r.viewshed*.

Monitoring Coastal Erosion

Tristan Dyer used terrestrial lidar data to monitor coastal erosion on a beach near the US Army Corps of Engineers' Field Research Facility in Duck on the Outer Banks of North Carolina. Coastal erosion can be visually and tactilely explored by building a sand model for each lidar survey. Dyer scanned a transect of the beach with a terrestrial lidar system, interpolated the lidar point cloud as a DEM, and then built a sand model with the aid of Tangible Landscape to quickly prototype and visualize the landform (Fig. A.6a). A sequence of these models would show the morphological evolution of the beach.

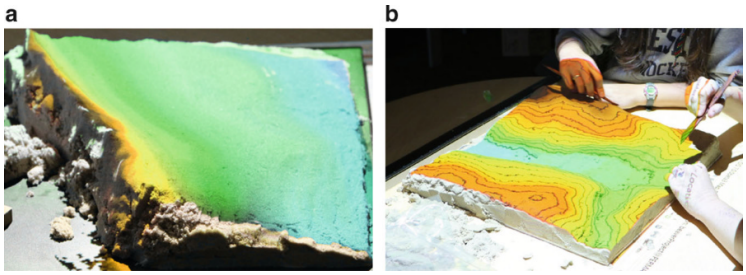


Fig. A.6 Sand models used to study beaches and streams. (**a**) A sand model of a beach in Duck, North Carolina used to study beach erosion. (**b**) A sand model of a valley in North Carolina Piedmont used to study how beaver dams influence stream flow and morphology

Exploring Impacts of a Beaver Dam

Karl W. Wegmann and his students used Tangible Landscape to study how beaver dams affect streams in North Carolina. They used Tangible Landscape to model the morphology and flow of a stream with and without beaver dams. First they built a sand model of a segment of the existing stream with beaver dams and studied its topographical parameters and hydrological processes. Then they sculpted the sand model and computed different water flow scenarios to study how the stream could have been without beaver dams (Fig. A.6b).

In this application we merged the scanned elevation model with a DEM of the surrounding landscape to put the modeled area into its broader context and account for water flowing from the entire contributing watershed. The water flow simulation was then run over the entire area so that the water from the contributing watersheds would fill the stream.

Modeling the Potential Impacts of a Coal Ash Pond Spill

Matthew Horvath (2014) studied the potential impacts of spills from the Cape Fear Plant coal ash ponds near Moncure, North Carolina. Horvath built a sand model for a study area near the Cape Fear river and used Tangible Landscape with the add-on module *r.damflood* to simulate a breach in a coal ash pond (Fig. A.7). Artificial landforms kept the simulated spread of contaminated surface water from reaching the nearby Cape Fear river.

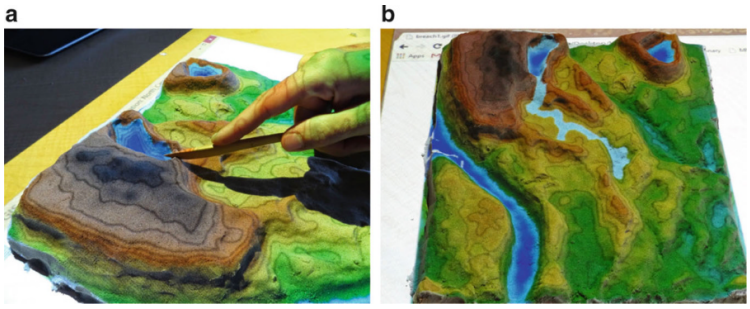


Fig. A.7 Making a breach in a coal ash pond (**a**) and simulating the spread of contaminated water (**b**)

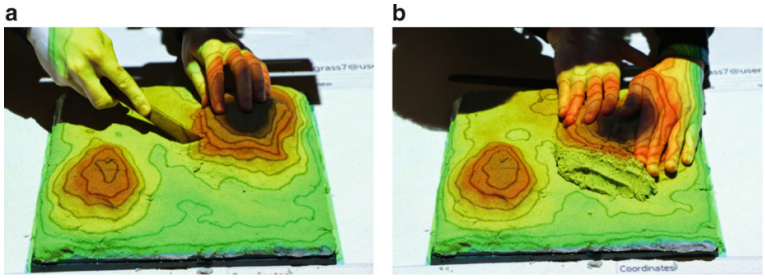


Fig. A.8 Iteratively sculpting a landform in sand to generate time-series data: (**a**) the initial state followed by (**b**) changing the position and shape of the landform

Testing a Landform Migration Algorithm

Petras et al. (2015) used Tangible Landscape to test the behavior of a newly developed algorithm for describing landform migration. The algorithm maps the horizontal migration of complex landforms as gradient fields based on the analysis of contour evolution. The gradient field is a vector field representing the movement and deformation of contours. It can be used to quantify the rate and direction of landform migration at any point in space and time. Data for testing algorithms is usually computer generated. In this case, however we wanted to create simplified, yet realistic abstractions of landforms that would allow us to clearly model and visualize landform migration. It was easier to make these abstract landforms by hand than on the computer. To create test data in a controlled environment we built sand models by hand to represent a moving landform. Using Tangible Landscape we sculpted and scanned a model of the initial terrain and then sculpted and scanned a model of the terrain after it had migrated to create an elevation time-series (Fig. A.8).

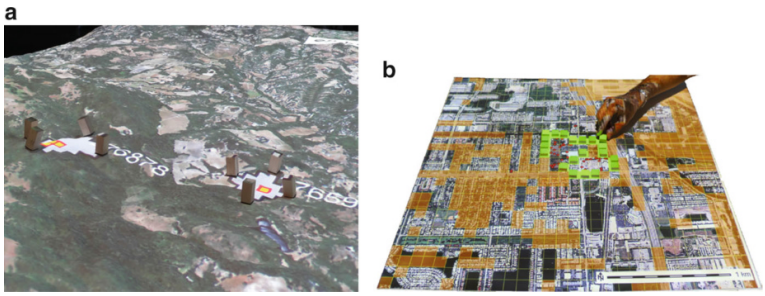


Fig. A.9 Using object recognition to digitize polygons and pixels representing treatment areas in order to computationally steer (**a**) Sudden Oak Death in Sonoma Valley, California, and (**b**) termites spread simulation in Fort Lauderdale, Florida

Managing the Spread of Sudden Oak Death in Sonoma Valley

In collaboration with Francesco Tonini and Douglas A. Shoemaker we used Tangible Landscape to test the effectiveness of different treatment scenarios for controlling the spread of Sudden Oak Death in Sonoma Valley, California. We used a stochastic model to simulate the spread of Sudden Oak Death and Tangible Landscape to visualize the spread and designate new treatment areas. We CNC routed a large (1 m × 1 m) MDF model of the terrain and coated it with magnetic paint. To make markers that would hold in place on the complex topography we attached adhesive magnetic strips to the base of basswood markers. We used the markers to collaboratively draw polygons representing treatment areas with Tangible Landscape (Fig. A.9a). The polygons reduced the host species of the disease on aligned pixels thus influencing the next run of the simulation. Each treatment area was automatically assigned a label with its area. The sum of all of the treatment areas was dynamically displayed on a simple dashboard. The disease spread simulation was written in the R (R Core Team 2013) statistical programing language and environment.

Managing the Spread of Termites in Fort Lauderdale

In collaboration with Francesco Tonini we used Tangible Landscape as a spatial decision support tool for collaboratively managing the spread of termites in Dania Beach, Fort Lauderdale, Florida. We used pixel-based object recognition to interact with a stochastic simulation of the spread of termites. Placing a marker on a flat surface with a projected grid changed a value on the corresponding pixel in the simulation input—a raster index of termite habitat (Fig. A.9b).

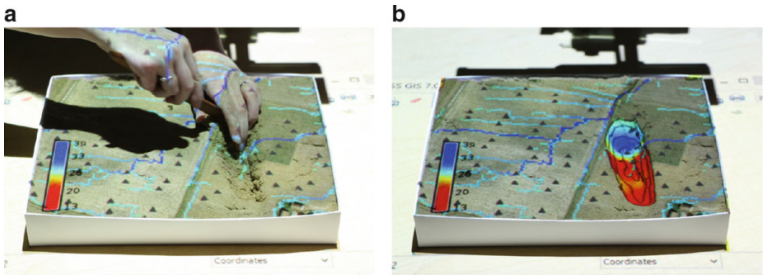


Fig. A.10 Exploring subsurface moisture by removing sand and projecting the cross-section of a scanned surface with the 3D raster of moisture

Tangible Exploration of Subsurface Data

Using Tangible Landscape we explored the 3D distribution of the percentage of subsurface moisture (Petrasova et al. 2014) measured in Kinston, North Carolina and represented as a GRASS GIS 3D raster (GRASS Development Team 2015). By digging into the sand model with our hands we could explore underground as if we were at an excavation site. As we dug into the sand the cross-section of the scanned surface with the 3D raster representing soil properties was projected in real-time (Fig. A.10). We also projected additional GIS layers (such as an orthoimagery, elevation contours, and flow accumulation) over the model to provide spatial context and gain further insights into soil moisture distribution and its relation to the landscape.

Additionally, in combination with marker detection from Sect. 6.2.2 we can create and visualize vertical profiles or soil core samples. This method is an intuitive and natural way of exploring subsurface data and it represents an alternative to more abstract 3D computer visualization tools.

A.2 Data Sources

Throughout the book we use various data sources. Here is a list of the most useful data sources for the study areas presented in the book.

Sample Data for This Book

NCSU OSGeo Research and Education Laboratory

http://geospatial.ncsu.edu/osgeorel/data.html

US Lidar Data

United States Interagency Elevation Inventory

http://coast.noaa.gov/inventory/

Earth Explorer

http://earthexplorer.usgs.gov/

Digital Coast

http://coast.noaa.gov/dataviewer/

Open Topography

http://www.opentopography.org/

US Digital Elevation Models

National Elevation Dataset on the National Map Viewer

http://viewer.nationalmap.gov/viewer/

US Orthoimagery

USGS EROS Orthoimagery web mapping service

http://raster.nationalmap.gov/arcgis/services/Orthoimagery/USGS_EROS_Ortho_1Foot/ImageServer/WMSServer?

US Soil Surveys

USDA Web Soil Survey

http://websoilsurvey.sc.egov.usda.gov

US Fire Modeling Data

LANDFIRE

http://www.landfire.gov/

Global Datasets

GRASS GIS community list of global datasets
http://grasswiki.osgeo.org/wiki/Global_datasets

NC Climate Data

State Climate Office of North Carolina
http://www.nc-climate.ncsu.edu/

A.3 Starting with GRASS GIS

To run examples from this book we provide a dataset which can be downloaded from the NCSU OSGeo Research and Education Laboratory website.[1] Once the downloaded dataset is unzipped we create a directory `grassdata` in our home directory and place the dataset there. Then we start GRASS GIS and a start up dialog in Fig. A.11 appears on screen. In the upper text field we specify the path to our `grassdata` directory. Then we should be able to see Location `nc_spm_tl` containing several Mapsets. We select mapset `practice1` and press the Start button.

After opening the Layer Manager and Map Display windows we display the raster map `elevation` as shown in Fig. A.12. To execute commands from this book presented in boxes with green background we paste the commands into the *Command console* tab as shown in Fig. A.13a and press Enter. The Python code in boxes with blue background can be executed in the *Python shell* tab (Fig. A.13b) or in any Python shell running in a GRASS GIS session.

More instructions how to use GRASS GIS GUI are available on the official GRASS GIS introduction page[2] and in the GRASS GIS GUI manual.[3]

[1] http://geospatial.ncsu.edu/osgeorel/data.html
[2] http://grass.osgeo.org/grass70/manuals/helptext.html
[3] http://grass.osgeo.org/grass70/manuals/wxGUI.html

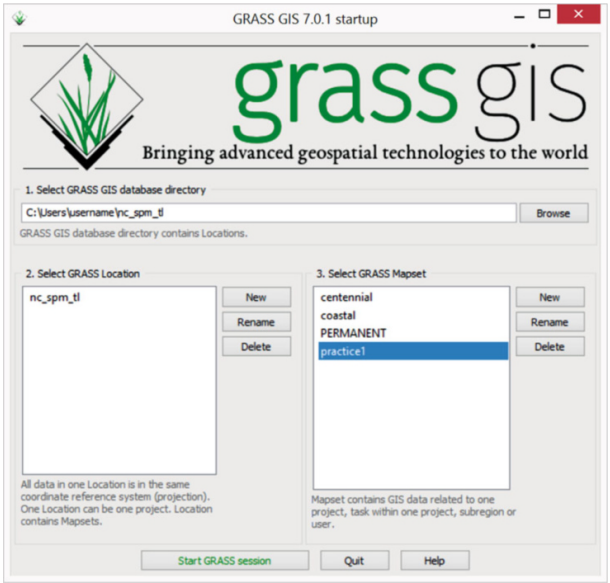


Fig. A.11 Starting GRASS GIS

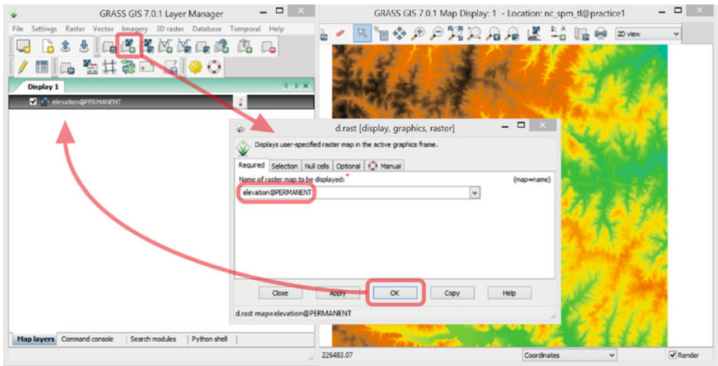


Fig. A.12 Display raster map elevation: click on *Add raster map layer* button in toolbar, select `elevation` from drop-down list, and press OK

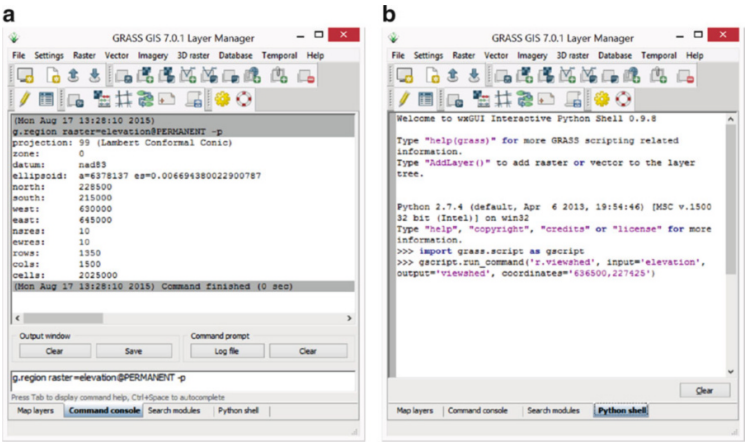


Fig. A.13 GRASS GIS modules can be run from (**a**) the *Command console* tab or (**b**) Python shell using Python syntax

References

GRASS Development Team. 3D raster data in GRASS GIS [online]. (2015) [Accessed 2015-08-13]. http://grass.osgeo.org/grass70/manuals/raster3dintro.html.

Horvath, M. (2014). Spill impacts from coal ash pond using GRASS GIS. Upublished poster.

Meentemeyer, R. K., Tang, W., Dorning, M. A., Vogler, J. B., Cunniffe, N. J., and Shoemaker, D. A. (2013). FUTURES: multilevel simulations of emerging urban-rural landscape structure using a stochastic patch-growing algorithm. *Annals of the Association of American Geographers*, 103(4):785–807.

Petras, V., Mitasova, H., and Petrasova, A. (2015). *Mapping gradient fields of landform migration*, pages 173–176. Bogucki Wydawnictwo Naukowe, Adam Mickiewicz University in Poznań - Institute of Geoecology and Geoinformation, Poznań, Poland.

Petrasova, A., Harmon, B., Mitasova, H., and White, J. (2014). Tangible Exploration of Subsurface Data. Poster presented at 2014 Fall Meeting, AGU, San Francisco, Calif., 15–19 Dec.

Petrasova, A., Petras, V., Shoemaker, D. A., Dorning, M. A., and K. Meentemeyer, R. (2015). The integration of land change modeling framework FUTURES into GRASS GIS 7. In *Geomatics Workbooks n 12 – "FOSS4G Europe Como 2015"*.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/.

Stopková, E. (2007). Modelovanie oblastí vzniku lavín s využitím GIS. In *Sborník studentské konference GISáček*.

Stopková, E. (2008). Predikcia lavínového nebezpečenstva s využitím GIS. Bachelor's thesis, Slovak University of Technology in Bratislava.

Index

Symbols
3D print, 10
3D printing, 31
3D raster, 24, 128

A
actuated shape display, 4
Aegis Hyposurface, 4
animation, 17, 70, 100
aspect, 55, 68, 71, 85, 92, 98, 108
Augmented Reality Sandbox, 8

B
bathymetry, 73
BEHAVE model, 106
binning, 29, 54

C
CAD, 1, 3, 36, 39
calibration, 27, 28, 53
CAM, 35, 38, 39
casting, 87, 107
change detection, 12
clearcutting, 108
CLI, 24
CNC, 10, 31
color ramp, 61, 101, 109
construction site grading, 69
continuous shape display, 7, 8, 11, 12, 31
contour models, 31, 36, 37
cumulative cost surface, 84, 85
curvature, 55, 61

D
dam break, 67, 73
declination angle, 98
Delaunay triangulation, 35
DeltaSand, 33
DEM, 7, 8, 27, 34, 54, 60, 73, 79, 87
deposition, 68
diffusion coefficient, 66
diffusive wave, 66
divergence, 68
DSM, 34, 47, 79, 87
dynamic shape display, 4, 6, 7, 11

E
elliptically anisotropic spread, 107
erosion, 68
exaggeration, 44

F
FEELEX, 4
finite volume method, 67
fire, 129
fire spread, 105
firebreaks, 107
flooding, 72
flow accumulation, 69
flow tracing, 69
flow velocity, 66
friction, 85
fuel, 105

G
g.extension, 63

© The Author(s) 2015
A. Petrasova et al., *Tangible Modeling with Open Source GIS*,
DOI 10.1007/978-3-319-25775-4

g.gui.animation, 75, 107
geographic information system, 24
geomorphons, 55, 57, 78
GIS, 1, 8
GRASS GIS, 8
Green's function, 66
Green's theorem, 67
GUI, 1, 2

H
human-computer interaction, 1, 6
hydrologic conditions, 69

I
ignition, 106
Illuminating Clay, 7
inFORM, 5
interpolation, 29, 54

K
Kinect, 5, 8, 9, 31
Kinect 3D Video Capture Project, 8
Kinect for Windows SDK, 24, 26
Kinetic Sand, 33

L
LANDFIRE, 109, 129
landform, 63, 122, 124–126
landforms, 58
least cost path, 69, 83, 84
lidar, 18, 34, 35, 43, 124, 129
line-of-sight, 77
linear regression, 28, 61

M
Manning's equation, 66
Manning's roughness coefficient, 74
map overlay analysis, 85
medium density fiberboard, 38
momentum equation, 67

N
Naismith's rule, 84
Non-uniform rational B-splines, 35
Northrop Grumman Terrain Table, 4

O
object recognition, 8, 12, 31
object tracking, 5
overland water flow, 65

P
partial derivatives, 57
partial differential equations, 67
particles, 66
path sampling method, 66
polynomial approximation, 56
profile curvature, 55
Projection Augmented Relief Models, 7

Q
quadtree, 55

R
R, 127
r.damflood, 68, 72, 74, 125
r.divergence, 71
r.fire.spread, 107, 111
r.futures, 123
r.geomorphon, 58, 63, 122, 124
r.info, 43
r.lake, 74, 115, 117, 123
r.lake.series, 116
r.mapcalc, 26, 88, 91
r.null, 74
r.param.scale, 57, 61
r.ros, 106–110
r.series, 87, 88
r.sim.water, 67, 69–71
r.skyview, 78
r.slope.aspect, 61, 69, 91
r.spread, 106, 107, 110
r.sun, 99
r.sun.daily, 99, 101
r.sun.hourly, 99–101
r.surf.contour, 34
r.surf.gauss, 109
r.to.vect, 80
r.viewshed, 78, 124
r.walk, 83, 90
r.watershed, 69, 70
random effect, 109
rate of spread, 106
Recompose, 5
Relief, 4
ridges, 63

S
Sandscape, 8
SandyStation, 8
scale, 43
sediment transport, 68
seed point, 74
shallow water continuity equation, 65
shape changing display, 4
sky-view factor, 78
slope, 55, 71, 98, 108
smoothing, 55
solar incident angle, 99
solar irradiance, 99
solar irradiation, 99
solar radiation, 97
spline, 54
spotting, 106
summer solstice, 100

T
tangential curvature, 56
TanGeoMS, 8
Tangible CityScape, 5
Tangible Landscape, 1–3, 8, 11
thermoforming, 42
topographic parameters, 55
transport capacity, 68
traveling salesman problem, 84, 85, 90
tree canopy, 107
triangulated irregular networks, 35
TUI, 1, 2, 4, 5, 11

U
Universal Soil Loss Equation, 68

V
v.generalize, 34
v.in.lidar, 46
v.info, 43
v.net, 90
v.net.salesman, 85, 90
v.surf.rst, 34, 54, 55
v.to.points, 80
v.to.rast, 91, 109
vacuuming forming, 42
valleys, 63
vertical shift, 61
visibility, 77
Vrui VR Toolkit, 8

W
water level, 74
watershed, 69
waypoints, 83
wildfire, 105
wind, 106
wind direction, 108
winter solstice, 100

X
XenoVision Mark III Dynamic Sand Table, 4

APPENDICES

A GIS-based environmental modeling with tangible interaction

Reprint

Anna Petrasova, **Brendan A. Harmon**, Vaclav Petras, and Helena Mitasova. 2014. GIS-based environmental modeling with tangible interaction and dynamic visualization. In D. P. Ames & N. Quinn, eds. *Proceedings of the 7th International Congress on Environmental Modelling and Software*. San Diego, California, USA: International Environmental Modelling and Software Society.

Attribution

Anna Petrasova, the lead author, co-designed the 1st generation Tangible Landscape, developed the code, co-developed the case studies, and wrote this paper.

I, Vaclav Petras, and Helena Mitasova also co-designed the 1st generation Tangible Landscape, co-developed the case studies, wrote part of the paper, and edited the paper.

International Environmental Modelling and Software Society (iEMSs)
7th International Congress on Environmental Modelling and Software
San Diego, California, USA, D.P. Ames, N. Quinn (Eds.)
http://www.iemss.org/society/index.php/iemss-2014-proceedings

GIS-based environmental modeling with tangible interaction and dynamic visualization

A. Petrasova [a], **B. Harmon**[b], **V. Petras**[a] **and H. Mitasova**[a]

[a] *Department of Marine, Earth, and Atmospheric Sciences, North Carolina State University, Campus Box 8208, Raleigh, North Carolina, USA (akratoc@ncsu.edu, vpetras@ncsu.edu, hmitaso@ncsu.edu)*

[b] *College of Design, North Carolina State University, Raleigh, North Carolina, USA (baharmon@ncsu.edu)*

Abstract: We present a new, affordable version of TanGeoMS, a tangible geospatial modeling and visualization system designed for collaboratively exploring how terrain change impacts landscape processes. It couples a physical, three-dimensional model of a landscape with geospatial modeling and analysis through a cycle of scanning and projection. Multiple users can modify the physical model by hand while it is being scanned; by sculpting the model they generate input for modeling of geophysical processes. The modeling results are then visualized by projecting images or animations back on the physical model. This feedback loop is an intuitive way to evaluate the impacts of different scenarios including anthropogenic and natural landscape change. Integration with GRASS GIS, a free and open source geographic information system, provides TanGeoMS with a variety of easily accessible geospatial analysis and modeling tools. To demonstrate the environmental modeling applications of TanGeoMS, we will demonstrate how development can be planned based on feedback from landscape processes such as hydrologic simulation and wildfire modeling with variable fuel distribution.

Keywords: tangible user interface; landscape process modeling; visualization; GRASS GIS; Kinect

1 Introduction

Landscape has a major influence on physical processes such as water flow and sediment transport. It affects, both directly and indirectly, how we make decisions about development, transportation infrastructure and agriculture. It is, therefore, crucial – not only for scientists, but also for decision-makers who may not have a strong background in the natural sciences – to understand the often unexpected impacts of changes to the landscape. Complex computational models are used to study the impact of landscape change; it can be relatively unintuitive to work with these models because we are separated from our data by multiple layers of abstraction when using a mouse, keyboard and digital display. With tangible user interfaces (TUIs), however, we can directly, haptically interact with a physical representation of digital data. By coupling representation and input, we can more directly, more intuitively interact with our data [Ishii and Ullmer, 1997]. By linking physical models with digital models, TUIs can make environmental modeling more intuitive and scientific representations more engaging. A TUI can be used, for example, to couple a physical model of terrain with its digital elevation model (DEM) [Piper et al., 2002; Kreylos, 2013].

The Tangible Geospatial Modeling System [Mitasova et al., 2006; Tateosian et al., 2010] extends the idea of a TUI that links a physical, malleable terrain model with its digital elevation model by integrating a GIS to broaden the capabilities of the system [Ratti et al., 2004]. GIS offers a set of ready-to-use tools for different types of geospatial analyses and simulations, as well as an interface for visualization. By modifying a physical model instead of its digital representation, users can experiment with different scenarios more intuitively and with much greater ease. Once the modified model is scanned and imported into GIS, and the analysis is performed, the result is projected back on the model to be evaluated. Users are encouraged to iteratively explore a variety of scenarios to reveal the impacts of modifying terrain and then decide upon the most suitable design. Due to the simplicity of the tangible interface, users do not need to have any prior experience with GIS.

In this article, we present a new, improved version of TanGeoMS. Drawbacks of the first TanGeoMS prototype include the cost and weight of the equipment, the difficulties in setting up the system, and the lengthy response time. The substantial lag time between interaction and visualization detracted from the intuitive nature of the process and impacted the number of scenarios users would test. Our goal is to enhance intuition and learning, encourage and stimulate creativity and make the system more accessible. Therefore, we developed a new, more flexible, affordable version of TanGeoMS capable of real-time response and dynamic visualization. Since the feedback loop of modifying, scanning, computing and projecting is the most important feature of TanGeoMS, we focused on automating the whole process so that it could run in real-time. With an automated workflow, users would be able to change the physical model and get feedback quickly without having to manually run any processing. We hoped this fast feedback loop would encourage people to experiment more as it would be very simple to test numerous scenarios.

2 METHODS

In this section we first describe the physical setup for TanGeoMS, including different types of physical models. Then we discuss the software components and the workflow for modeling with real-time response.

2.1 Physical setup

The new setup consists of 4 primary components: *(a)* a physical model that can be modified by a user, *(b)* a 3D scanning device, *(c)* a projector and *(d)* a computer with GIS software and additional software that connects all the components together. The physical model, laid on a table, is scanned by the scanning device mounted above. When the scan is imported into the GIS software and processed, the projector projects the resulting image or animation directly onto the modified physical model so that the results are put into the context of the modifications to the model.

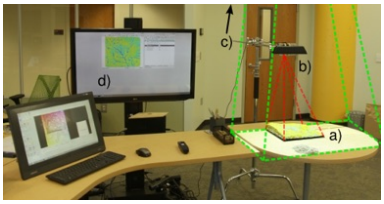


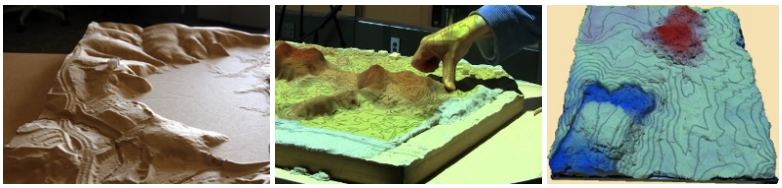
Figure 1. Physical setup: a) physical model; b) 3D scanner; c) ceiling mounted projector; d) computer with GIS software.

We use the Microsoft Kinect for Windows scanner to acquire a 3D scan of the physical model. Our new, updated TanGeoMS is now affordable since the Kinect scanner is much

less expensive than the 3D laser scanner that we had previously used. Furthermore, this small, lightweight scanner is highly portable and can be easily mounted on a stand. Given the scanner's limited angular field of view – 43° vertical by 57° horizontal – the height of the scanner above the physical model depends upon the size of the model. For smaller models (∼40 cm diagonally), the scanner should be mounted ∼0.5 m above the model to get a reasonable resolution. Scanning resolution deteriorates with increasing distance from the model. Therefore, the size of the physical model is limited. We have found that a model size of ∼70 cm diagonally is ideal for larger models. The Kinect scanner is connected to the computer as well as the projector. While we use a ceiling-mounted projector, the projector could also be mounted on a stand.

Initially, we built models by stacking layers of foam board cut along projected contours and then covered them with a layer of modeling clay [Mitasova et al., 2006; Tateosian et al., 2010]. Since this method required a lot of manual work, we experimented with different approaches and materials. When building models manually, we now use a sand enriched with a polymer since it is easy to sculpt, sticks together and holds its form and is unaffected by humidity. We can also build models using automated processes such as computer numerical control (CNC) routing (Fig. 2a), laser cutting, and 3D printing. Since a model created from a solid material (such as a CNC routed model made of medium density fiberboard) is not malleable, we build a second model that is an inverse of the first and then mold a layer of sand between the two models, casting a precise yet malleable surface (Fig. 2b). This way we benefit from the precision of computer-aided manufacturing and the flexibility of sand.

We typically work with models that represent real, geographic places. When manually building a model of a real place from polymeric sand, we overlay and project various raster and vector layers of the place – such as the DEM, slope, and contours – as a guide. In addition to projecting the contours and elevation of the real area, we can continuously scan the model and then compute and project the difference between the DEM and the scanned elevation, thus highlighting places where we need to add or remove sand in order to match the DEM (Fig. 2c). This way we can always rebuild or reset the model after modifying it.



(a) **(b)** **(c)**

Figure 2. Different methods of creating models: (a) a model carved with a CNC router; (b) a carved model combined with a layer of molded sand; (c) a sand model sculpted manually using projected contours and a color map representing the difference between the current and desired states. Blue shows that sand should be added; whereas, red indicates that sand should be removed.

2.2 Software tools

There are several different software components that we use with TanGeoMS: the Kinect for Windows SDK for scanning, GRASS GIS for processing scans, and our custom library that automates the entire process.

We wrote a simple application in C++ with the Kinect SDK to continuously scan the model and output a file with the point cloud. To connect TanGeoMS's components, we developed a small library and a set of scripts in Python which are able to process the scan and interface with GRASS GIS. The library contains methods for handling different types of input data (e.g. georeferenced versus ungeoreferenced inputs and models with different shapes).

GIS is a crucial component of TanGeoMS as it is used to create a DEM from scanned data, to run analyses or simulations, and to visualize data. GRASS GIS [Neteler et al., 2012], a free and open source GIS, offers a variety of tools for topographic analysis, hydrologic and erosion modeling, or wildfire simulation. Moreover, many highly specialized algorithms are implemented and accessible through the add-on repository. It is very simple to combine different tools in a script using its Python API. While the previous version of TanGeoMS used GRASS GIS, we have now fully integrated this GIS with TanGeoMS in order to easily automate the process and connect all of TanGeoMS's components, thus enabling a more effective workflow.

2.3 Workflow

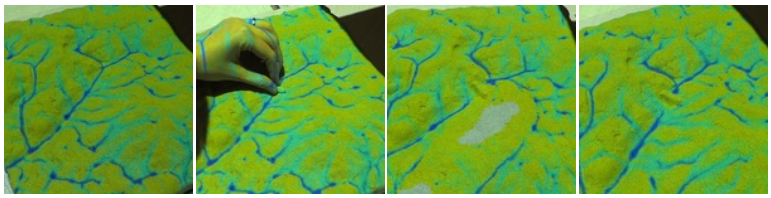
The TanGeoMS workflow is an interactive process in which users modify a physical model that is then scanned, the scan is imported into GIS software, analysis is performed on the modified input, and then the result is projected back on the model, allowing the users to revise their modifications.

Before we start scanning, we first calibrate TanGeoMS to geometrically correct for any inaccuracy in the angle or direction of the scanner. This is needed because the scanner is very difficult to align precisely perpendicularly to the model and uncorrected misalignments would result in a tilted scan. Next, we georeference the point cloud. Since we know the geographic extent of the area represented by the model and the model orientation is aligned with the scanner, we only need to translate and scale the point cloud. Then, we import the scan as 3D points into the GIS and create a DEM, either by binning or by interpolation. Once we have the DEM, we can run the desired GIS analyses and project the results as color maps back onto the physical model. Then we can evaluate the impact of the changes from the previous run, modify the physical model and repeat the entire process again, except for the calibration process.

We automated this feedback loop so that the impact of changes to the physical model can be projected back onto the model in real-time. Immediate feedback facilitates the testing of different scenarios, potentially resulting in better designs. The time between the updates is usually a few seconds, depending on the size of the model and, most importantly, the analyses run on the scanned DEM. Since interpolating the DEM would slow down each run significantly, we use binning (the per cell point average of z-values) combined with smoothing to create the DEM. This way we can compute standard topographic analyses including slope, aspect, and flow accumulation relatively quickly. Approximate solutions can be used for more complex analyses and computationally intensive models to produce results that are less accurate, but still reasonable and relevant.

To demonstrate a design process with real-time response we modeled overland water flow [Mitasova et al., 2005] using a sand model based on a DEM of an existing study site. We have tried to capture storm water and reduce runoff by building a small check dam on the model. Fig. 3 shows several snapshots which include Fig. 3a, the initial state, Fig. 3b, a user adding sand and shaping the check dam and Fig. 3d, the change in the water flow filling the dam. In Fig. 3c we see water flow computed on a scan captured when the user's hand was still modifying the model. This happens because we scan continuously;

therefore we always have to wait a few seconds after a change has been made to get a clean scan before we can evaluate the modifications.



(a) (b) (c) (d)

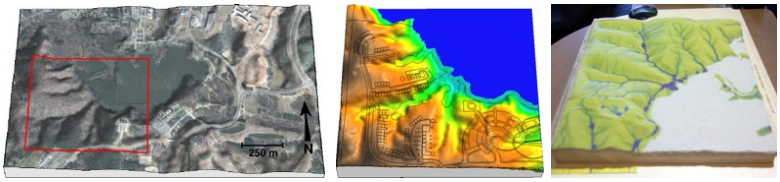
Figure 3. Real-time water flow modeling: (a) initial state; (b) adding a check dam; (c) water flow when hand was captured by scanner; (d) check dam filling with water.

3 Applications

TanGeoMS has been used for a broad range of applications supported by the extensive set of geospatial analysis and modeling tools available in GRASS GIS. We have explored how dune breaches affect the extent of coastal flooding, the impact of different building configurations on viewsheds, cast shadows and solar energy potential, and the effectiveness of various landscape designs for controlling runoff and erosion [Tateosian et al., 2010]. Here we present two new applications. We evaluated the environmental impact of a development scenario and then designed and tested alternatives. We also modeled the spread of a fire and then designed and tested fire-control measures.

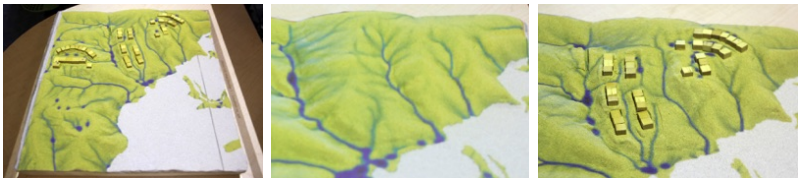
3.1 Scenario planning with hydrology and sediment transport modeling

TanGeoMS can be used to iteratively design and test scenarios. We used the system to study the environmental impact of building a residential development in Lake Raleigh Woods on North Carolina State University's Centennial Campus, a site that has been experiencing rapid urban development. Our study area is approximately 750 m in both dimensions. The mostly forested terrain is hilly with steep slopes, especially near the lake (Fig. 4a). We began by carving a medium-sized base model (50 by 50 cm) of the current landscape with a vertical exaggeration of 2. Then we carved an inverted model of landscape that we used to mold a 2 cm thick layer of sand on top of the base model. We modeled overland hydrologic flow, erosion and deposition using a particle sampling method [Mitasova et al., 2005] for the current state (Fig. 4c). By manually modifying the sand layer and adding 'buildings' on top, we modeled the impact of a development scenario based on the southwest part of a 1999 masterplan for Centennial Campus (Fig. 4b) that would require extensive excavation and grading. This development scenario changed the local hydrologic conditions significantly (Fig. 5b, c). To design a more sustainable development, we sculpted drainage and retention ponds (Fig. 6). Next we designed a second, alternative development scenario in which we attempted to minimize excavation and grading by building in low slope areas (Fig. 7a-c). Again, we studied the modeled impacts of the scenario and sculpted drainage and retention ponds to reduce erosion (Fig. 7d-f). The erosion modeling part was based on scanned DEM but was computed separately since it requires additional spatially variable inputs and is more process-intensive as well. Through this iterative process of experimentation and modeling, we were able to test and evaluate the impacts of our designs.



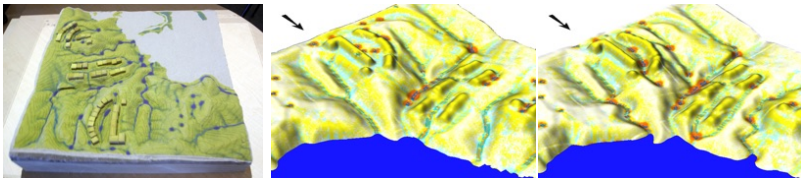
(a) (b) (c)

Figure 4. The current condition of Lake Raleigh Woods: (a) orthophoto with highlighted study area; (b); DEM with overlaid masterplan (looking north) (c) overland hydrologic flow (looking west).



(a) (b) (c)

Figure 5. First development scenario (a) and comparison of overland hydrologic flow before (b) and after development (c), looking west. Note that new stream channels draining developed areas were created.



(a) (b) (c)

Figure 6. Modified first development scenario (a) and comparison of erosion/deposition before (b) and after (c) applying best management practices. Results suggest that stream banks should be protected more by planting grass or trees.

3.2 Wildfire modeling

TanGeoMS can also be used to model wildfire spread. Here the physical model represents the terrain with the tree canopy. The tree canopy surface was acquired by lidar mapping and the sand layer of the physical model was molded using the inverted carved model of canopy. The simulated spread of wildfire is determined by the wind, the topography, and spatially variable fuel availability which is represented here by the tree canopy. Thus by removing sand representing the canopy from a certain area, we can easily create firebreaks. As we can intuitively create new scenarios, we can rapidly test how different sizes, shapes, and placements of firebreaks affect the spread of a fire. The wildfire simulation itself is computed using GRASS GIS's wildfire modeling tools [Xu, 1994] and the result is projected as an animation on the model. As an example, we have simulated the spread of a fire across our study area and observed that fire reached the Chancellor's House quickly due the abundance of fuel and the prevailing south-east wind direction (Fig. 8). We attempted to protect the house by building a fire line, perpendicular

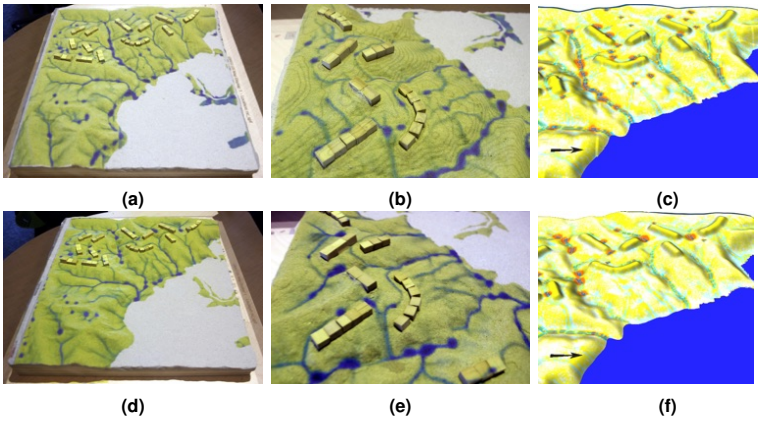


Figure 7. Second development scenario before (upper row) and after (bottom row) applying best management practices: (a) and (d) overland hydrologic flow (looking west); (b) and (e) detail of the western part (looking north); (c) and (f) erosion/deposition (looking west). Retention ponds successfully reduced downstream erosion.

to the wind direction, that stretched from the lake to the road. However, the fire eventually spread beyond our fire line due to the presence of low lying vegetation beside the road. Nevertheless the firebreak slowed the spread of the fire significantly, potentially giving firefighters more time to act.

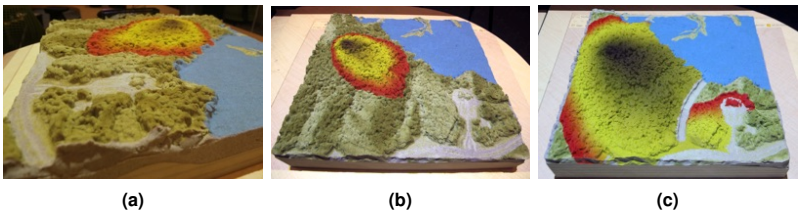


Figure 8. (a) Tree canopy in detail; (b) and (c) wildfire spread simulation before and after adding a firebreak.

4 Discussion

There were two major challenges in developing TanGeoMS: the accuracy of representation and the speed of real-time response. The inaccuracy arises as the physical model is a simplification of reality which is further modified manually and then scanned. We are trying to address these issues by using computer aided manufacturing to generate more precise physical models. However, we consider TanGeoMS to be primarily a 3D sketching tool, which enables users to rapidly explore different scenarios. To save time and effort, we only need to compute precise models for the most promising of scenarios. Modeling with real-time response is limited by the current scanning technology, modeling algorithms and process automation. While the Kinect scanner enables fast scanning, certain TanGeoMS applications would benefit from higher precision scanning. It is necessary therefore to continue exploring the potential of other scanning technologies. The

speed of real-time response is obviously limited by the processing time of the analyses selected. The automation of the process is complicated by usage of different types and shapes of models and variations in physical setup. However, due to the benefits of the fast feedback loop, we think it is worth trying to overcome these obstacles. Being able to see the impact of a change immediately highlights the cause and effect relationship. Moreover, users are able to test more scenarios in a shorter time.

5 Conclusion

We have presented an enhanced, affordable version of the Tangible Geospatial Modeling System capable of modeling and visualization with both real-time and near real-time response. The updated TanGeoMS has an automated real-time feedback loop that transforms changes to a physical model into a digital form, computes analyses, and projects the results back onto the model. This automated real-time feedback loop is a powerful method for testing multiple landscape-altering scenarios without having to interact directly with a GIS. By using the inexpensive Kinect scanner and GRASS GIS, a free, open source GIS, we made TanGeoMS financially affordable. Furthermore, we have made TanGeoMS even more intuitive to use by creating easily malleable models built with innovative technologies and materials. We have demonstrated how TanGeoMS can be used for urban planning and fire crisis management. These examples represent just a few of TanGeoMS's many potential applications.

References

Ishii, H. and Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, pages 234–241, New York, New York, USA. ACM Press.

Kreylos, O. (2013). Augmented Reality Sandbox.

Mitasova, H., Mitas, L., Ratti, C., Ishii, H., Alonso, J., and Harmon, R. (2006). Real-time landscape model interaction using a tangible geospatial modeling environment. *IEEE Computer Graphics and Applications*, 26(4):55–63.

Mitasova, H., Thaxton, C., Hofierka, J., McLaughlin, R., Moore, A., and Mitas, L. (2005). Path sampling method for modeling overland water flow, sediment transport, and short term terrain evolution in Open Source GIS. In Miller, C., Farthing, M., Gray, V., and Pinder, G., editors, *Proc., XVth International Conf. on Computational Methods in Water Resources (CMWR XV)*, pages 1479–1490. Elsevier.

Neteler, M., Bowman, M., Landa, M., and Metz, M. (2012). GRASS GIS: a multi-purpose Open Source GIS. *Environmental Modelling & Software*, 31:124130.

Piper, B., Ratti, C., and Ishii, H. (2002). Illuminating clay: a 3-D tangible interface for landscape analysis. In *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI '02*, page 355, New York, New York, USA. ACM Press.

Ratti, C., Wang, Y., Ishii, H., Piper, B., and Frenchman, D. (2004). Tangible user interfaces (tuis): a novel paradigm for gis. *Transactions in GIS*, 8(4):407–421.

Tateosian, L., Mitasova, H., Harmon, B. A., Fogleman, B., Weaver, K., and Harmon, R. S. (2010). TanGeoMS: tangible geospatial modeling system. *IEEE transactions on visualization and computer graphics*, 16(6):1605–12.

Xu, J. (1994). *Simulating the Spread of Wildfires Using a Geographic Information System and Remote Sensing*. PhD thesis, New Brunswick, NJ, USA.

B Integrating FOSS into Geospatial Science Education

Reprint

Vaclav Petras, Anna Petrasova, **Brendan Harmon**, Ross Meentemeyer, and Helena Mitasova. 2015. Integrating Free and Open Source Solutions into Geospatial Science Education. *ISPRS Int. J. Geo-Information* 4, 2 (2015), 942–956. DOI:urlhttp://dx.doi.org/10.3390/ijgi4020942

Attribution

Vaclav Petras and Anna Petrasova developed this approach, implemented methods for building course web pages, and wrote this paper.

I developed and taught the course *LAR582: GIS for Designers*, wrote the section about this course, and edited the paper.

Ross Meentemeyer directs the MGIST program at the Center for Geospatial Analytics at North Carolina State University. He contributed to this paper with advice, guidance, support, and editing.

Helena Mitasova pioneered this approach, developed and taught the courses *GIS/MEA582: Geospatial Modeling and Analysis* and *GIS595/MEA592: Multidimensional Geospatial Modeling*, wrote the sections about these courses, and edited the paper.

ISPRS Int. J. Geo-Inf. **2015**, *4*, 942-956; doi:10.3390/ijgi4020942

OPEN ACCESS

ISPRS International Journal of
Geo-Information
ISSN 2220-9964
www.mdpi.com/journal/ijgi

Article

Integrating Free and Open Source Solutions into Geospatial Science Education

Vaclav Petras [1,2,*], Anna Petrasova [1,2], Brendan Harmon [2,3], Ross K. Meentemeyer [2,4] and Helena Mitasova [1,2]

[1] Department of Marine, Earth, and Atmospheric Sciences, North Carolina State University, 2800 Faucette Drive, Raleigh, NC 27695, USA; E-Mails: akratoc@ncsu.edu (A.P.); hmitaso@ncsu.edu (H.M.)

[2] Center for Geospatial Analytics, North Carolina State University, 2800 Faucette Drive, Raleigh, NC 27695, USA; E-Mails: baharmon@ncsu.edu (B.H.); rkmeente@ncsu.edu (R.K.M.)

[3] Department of Landscape Architecture, North Carolina State University, 50 Pullen Road, Raleigh, NC 27695, USA

[4] Department of Forestry and Environmental Resources, North Carolina State University, 2800 Faucette Drive, Raleigh, NC 27695, USA

* Author to whom correspondence should be addressed; E-Mail: vpetras@ncsu.edu and wenzeslaus@gmail.com; Tel.: +1-919-578-4957

Academic Editors: Barend Kobben, Serena Coetzee and Wolfgang Kainz

Received: 12 February 2015 / Accepted: 21 May 2015 / Published: 1 June 2015

Abstract: While free and open source software becomes increasingly important in geospatial research and industry, open science perspectives are generally less reflected in universities' educational programs. We present an example of how free and open source software can be incorporated into geospatial education to promote open and reproducible science. Since 2008 graduate students at North Carolina State University have the opportunity to take a course on geospatial modeling and analysis that is taught with both proprietary and free and open source software. In this course, students perform geospatial tasks simultaneously in the proprietary package ArcGIS and the free and open source package GRASS GIS. By ensuring that students learn to distinguish between geospatial concepts and software specifics, students become more flexible and stronger spatial thinkers when choosing solutions for their independent work in the future. We also discuss ways to continually update and improve our publicly available teaching materials for reuse by teachers, self-learners and other members of the GIS community. Only when free and

open source software is fully integrated into geospatial education, we will be able to encourage a culture of openness and, thus, enable greater reproducibility in research and development applications.

Keywords: online course; GRASS GIS; ArcGIS; FOSS; geospatial modeling; visualization

1. Introduction

Reproducibility is a fundamental requirement of science. Scientists need access to all tools used in published research to fully reproduce and verify results and conclusions. If the software used in a research study is a "black box" and the code and algorithms cannot be scrutinized, then the results will not be verifiable and reproducible in the long run. Therefore, in science, the use of free and open source software has been considered a high priority (and often stated as mandatory by funding agencies), as it is fully transparent [1,2] and more accessible for institutions, individual students and scientists than proprietary software [3].

The emerging field of open science calls for greater transparency and free sharing of knowledge, methods, data and code, so that science is accessible to all. By requiring open source software, data and access [4], open science approaches ensure that scientists have access to the same methods, data and code used in a published study. Open science perspectives, such as open access and open data, are being increasingly adopted in research, but free and open source software is surprisingly less reflected in universities' educational programs, despite its wide adoption in industry [5,6].

Open software, open data, open standards and open education are the key components of the open GIS framework proposed by Sui [7]. There is great demand for open geospatial education, as spatial thinking and GIS concepts can be applied across disciplines. The popularity of massive open online courses (MOOC) is growing [8], including those focused on open geospatial education [9] and free and open source geospatial software [10]. Open education concepts, such as accessible teaching materials and the usage of free and open source software, are not limited to online courses and have also been applied to traditional on-campus education [11]. Several educators have suggested that open education concepts should be applied throughout the geospatial curriculum [12].

The application of geospatial concepts should be emphasized in education more than software-specific tasks. If students understand not only the implementation, but also the underlying science and technology, then they will be able to do better research and develop more flexible solutions. However, students who learn and use particular software packages as part of their classes inevitably mix general concepts with software-specific implementation. Teaching a diverse set of software packages may help to untangle the general concepts of geospatial analytics from software-specific features and issues. Free and open source software readily allows students to compare how scientific tools are implemented in different software packages; it is harder to make such comparisons when using proprietary software, because the source code and algorithms are concealed [13]. Moreover, easy access enables students to explore new technologies without requiring the university or their instructors to provide the software.

This is especially beneficial for project-based classes and independent studies where students solve real-world problems in which unique or custom capabilities are required.

Throughout this paper, we use the term free and open source software (FOSS) in which "free" stands for software freedom as defined by Stallman [14] and "open source" stands for a type of development model [15]. Proprietary software refers to software developed and licensed by a particular software vendor. It is sometimes referred to as commercial or enterprise software, which is incorrect, because both proprietary software and FOSS can be used commercially and in enterprise settings.

In this paper, we begin with an overview of newly-developed educational concepts and the associated online resources for courses in geospatial analytics at North Carolina State University (NCSU). These courses combine FOSS GIS and proprietary GIS to emphasize general concepts and techniques in geospatial science [16]. The NCSU OSGeo Research and Education Laboratory (OSGeoREL), established in 2012 as the founding member of the world-wide GeoForAll initiative in North America [17], provides the infrastructure to update and redesign teaching materials, expand courses and incorporate the latest visualization and collaborative technologies in classes.

In Section 2, we describe the design of our courses, including the technical solution for our teaching materials used by both on-campus and distance education students. Section 3 describes several examples of these courses. We conclude with a discussion of our experience offering open science education, and we outline future directions for moving the field forward.

2. Approach to Course Design and Implementation

The general concept for integrating FOSS into geospatial information science and technology (GIST) education was inspired by common real-world situations when GIST professionals need to combine various software tools and techniques to solve complex problems. This approach was used to develop graduate level geospatial science courses serving Master of Science (MS) and PhD students in the NCSU GIST program, as well as the students from a broad range of disciplines, such as natural resources management, Earth systems (marine, Earth and atmospheric sciences), landscape architecture, civil engineering and anthropology. The courses are designed for students with very diverse sets of skills and backgrounds and can be taken on-campus or through the NCSU distance education program.

2.1. General Course Design

The courses have three standard components: lectures, in-class assignments with homework and a project. The lectures are software independent (to the extent possible) and focus on concepts, methods and algorithms. Each lecture is accompanied by a hands-on assignment to demonstrate the implementation of the concepts presented in the lectures in two software packages: the FOSS package GRASS GIS and the proprietary package ArcGIS. This teaching model gives students the opportunity to improve their knowledge of software that they already know (as most students taking the courses are already familiar with ArcGIS) and, at the same time, challenges them with different software that uses a similar, but not identical approach and solves some of the tasks using different algorithms. Our goal is to ensure that students can more easily distinguish between GIS concepts and the specifics of GIS software,

become true experts in GIS, rather than a particular software, and become more flexible when choosing solutions for their independent work in the future.

One of the important assignment features is a flipped approach. Rather than giving the students a task for which they are supposed to find a workflow, the students are given step-by-step instructions for a workflow (see Figure 1), and their task is to explain what this workflow does, which methods or algorithms are used and then to interpret the results. Small experiments that require the design of independent workflows are also included. The assignments are then submitted in the form of short papers. Each paper includes a software-independent section describing the methods and techniques used in the analysis and results in the form of graphical output with relevant interpretation and evaluation. The software-specific issues are addressed in a special subsection of the discussion, so software is not ignored, but rather clearly distinguished from the data and methods.

Apply neighborhood operators

Use neighborhood operator to compute land use diversity map.
How diverse is land use in NCSU area? Would you change it?

```
g.region raster=landuse96_28m -p
r.info -g landuse96_28m
r.neighbors landuse96_28m output=lu_divers method=diversity size=7
r.report lu_divers unit=p
```

Figure 1. Instructions for GRASS GIS part of the assignment typically consist of general description and modules with parameters to be used in the form of commands.

The midterm exam includes the design of independent workflows for given geospatial problems; students can implement their workflows in either GRASS GIS or ArcGIS. It is not unusual for students to do at least part of the exam in both. Most of the independent work in the courses is performed in the course project, which starts early in the semester. The students can use any suitable software, as along as they are working on a project related to the course material or their thesis.

In the more advanced courses with smaller numbers of students, we experiment with emerging visualization and human-computer interaction technologies with special focus on dynamic landscape processes. Examples of the technologies explored by students in their projects include Tangible Landscape [18], panoramic visualization and eye tracking.

2.2. Educational Content Management

Currently, there is no standardized system for publishing and sharing course material in geospatial science and technology. One of the few examples of a sustained effort is GIS e-learning project GITTA [9], which is largely limited to the contributors from the Swiss universities. The newest MOOC effort by GeoAcademy [19], described in this issue, is one of the most promising current developments. Based on the past efforts and discussions in the OSGeo education community [20], it appears that it would be difficult to fulfill the needs of the very diverse worldwide community of GIST program educators using any single technical solution. Repositories of FOSS geospatial education content, such

as ELOGeo [21] or the OSGeo Educational Content Inventory [22], provide many valuable contributions, but without collaborative and versioning infrastructure, the material becomes obsolete relatively quickly.

Thus, in order to share and collaboratively develop educational content, standardized formats and collaborative environments are important. Often, more than one instructor and their teaching assistants work on the course material. Thus, managing the content in a revision control system, a practice widely adopted for software development by FOSS projects, offers several advantages. It facilitates collaboration, includes a full history of changes and inherently supports the reuse of the materials, either by the author (for example, for a different course with customized modifications) or by others in the community if the repository is public.

The easiest way to share source material for courses with others is through revision control hosting services such as GitHub, GitLab or Bitbucket. These web-based tools together with desktop graphical tools make revision control systems accessible for users who are not familiar with using revision control tools in the command line. We use Git as a revision control system, because it allows us to work offline and to easily switch hosting providers for remote repositories. We use a remote repository hosted at GitHub because collaboration is a priority for us and GitHub is well established in the FOSS community. It is important to say that with Git vendor lock to any particular hosting service is minimal due to the fact that everything is also stored locally.

2.3. Teaching Material for Assignments

The common revision control systems are designed to work best with text files, and thus, the formats should be text-based, not binary. The text files are independent from specific text editors; therefore, they can be changed by anyone, anytime, even if the original software used to create them no longer exists. Text files are also easy to modify using scripts.

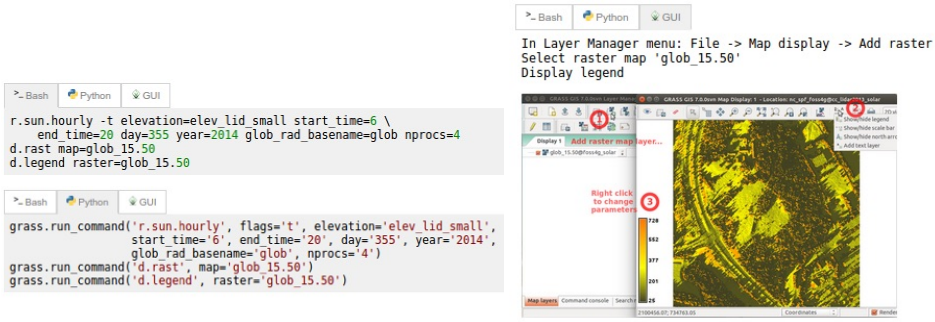
We have considered several formats for text documents, *i.e.*, markup languages, including LaTeX, DocBook, reStructuredText, Markdown and Textile (see the Appendix for a more complete discussion on format selection). After careful evaluation of these options, we decided to keep our teaching materials as a set of HTML files. This allows us to publish our material as a website and to take full advantage of HTML capabilities, such as CSS classes. The fact that our old course materials were already in HTML also contributed to this decision and made the transition easier. Eventually, our goal is to use the HTML5 standard, although not all of our pages have the correct HTML5 semantics yet.

The assignments are managed in individual files, which include only the assignment text component of the HTML documents. The sections of the HTML documents that are common to all pages (e.g., navigation) are kept separately, and we use scripts in Bash and Python to create a complete HTML document. The script builds a static website, which does not require any server-side application, so it can be published anywhere. We use a dedicated university server for courses, but the static pages can be published also using solutions, such as GitHub pages or websites on Bitbucket.

Our solution with custom scripts requires several steps to produce the web pages, but for the simple pages for which we are aiming, the amount of work and complexity is very low. More importantly, the scripting allows us to add new features. For example, we have developed a script that extracts examples from GRASS GIS assignments, which are then converted into a script executable in GRASS GIS. The

script allows us to efficiently test all examples in assignments for each new release of GRASS GIS. This proved helpful when several module parameter names were changed just before the release of GRASS GIS 7. We also use the scripts to ensure that everything published on the website is also stored in a repository, *i.e.*, the script for publishing checks that proper revision control commands were executed. Custom scripting allows us to accommodate different formats, such as reStructuredText, in the future.

Using HTML for both source and published material gives us the opportunity to use more sophisticated HTML formatting and JavaScript for interactivity. For example, in one of our tutorials, we show students how to do the same operation in the GRASS GIS command line, using the Python API or the graphical user interface. The page then consists of a general text description followed by an example box with tabs to switch between the different methods (Figure 2). This shows students several possible ways to interact with the software and may help them to transition between these approaches. For example, students can learn how to develop a Python script from a series of commands. The same method could be applied to provide instructions for different software packages.



(a) **(b)**

Figure 2. GRASS GIS tutorial using tabs with instructions for (**a**) scripting languages and (**b**) a graphical user interface. Examples are available at http://ncsu-osgeorel.github.io/grass-temporal-workshop/.

2.4. Teaching Material for Lectures

The choice of format and technology for lecture materials is not straightforward; it depends on the focus of the lectures, as well as personal preferences. Besides the standard desktop presentation tools, HTML5-based presentations and various cloud-based presentation services are being used more frequently. Our lectures were originally prepared in the PPTX format using PowerPoint and are available as presentation slides, as well as a screencast of the presentation accompanied by audio. The current format for slides has serious limitations in terms of updates and collaborative development, and thus, several options have been considered for redesigning the lecture material (see the discussion of different formats and tools, such as LibreOffice, LaTeX and HTML5, in Appendix 5).

We decided to use HTML5-based presentations, since they are plain text files and HTML5 supports dynamic and interactive content, including animated pictures, video files, automatically highlighted code, equations (through MathJax) and even embedding websites. HTML5 presentations, such as

Reveal.js, lack the ability to freely position elements on a slide or add decorations, such as arrows connecting text and pictures, in a simple way. Today, however, lecturers can take advantage of touch screens and use a stylus to draw directly on the slides to make their presentations more engaging for students. Based on these considerations, we are currently experimenting with the open source presentation framework Reveal.js to see whether it is suitable for our lectures. We created a set of templates for different slide layouts, so that only a basic knowledge of HTML is needed to create presentations with images and text in different positions.

2.5. Licensing

When sharing teaching materials, the choice of license is critical, since choosing an inappropriate license or not stating the license at all prevents others from using the material. There are standardized licenses that support sharing; Creative Commons (CC) licenses are the most used ones. Among the CC licenses, the best for sharing open education teaching materials are the Creative Commons Attribution-ShareAlike (CC BY-SA) and the Creative Commons Attribution (CC BY). The other CC licenses, such as the Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND), do not allow modification or commercial use and, thus, conflict with the values of FOSS, such as the freedom of reuse. We share both the material and the source files under the CC BY-SA license.

3. Course Implementation

The core components of the presented approach have been used to develop three courses focused on geospatial analysis, modeling and design.

3.1. GIS/MEA582: Geospatial Modeling and Analysis

The first course based on this concept, Geospatial Analysis and Modeling, was offered in 2008 and started as an on-campus special topics course with a small enrollment of 10 students. The distance education section was added in 2009, and today, the number of students has grown to around 50 per semester. It is one of two choices for a mandatory, core course in the GIST MS program and serves as an elective in several MS and PhD programs across NCSU colleges.

While the course topics are relatively stable, the teaching material is very dynamic, with assignments updated every semester based on the latest software releases. The topics covered include the digital representation and analysis of geospatial phenomena represented by continuous fields with a special focus on cost surfaces, terrain modeling, geomorphometry, watershed analysis and introductory GIS-based modeling of landscape processes. Perspective 3D visualizations and working with LiDAR data are an integral part of the course material. Figures 3 and 4 show examples of students' course work.

The take-home midterm exam consists of theoretical and practical sections. In the practical part, students develop their own workflow using ArcGIS, GRASS GIS or both to solve a given task. Project topics are typically chosen by students depending on their interests or masters theses research and, include, for example, the estimation of solar radiation and energy potential, coastal applications related

to beach erosion or sea level rise, hazard mapping, designing trails using cost surfaces and, also, occasionally, free and open source software development or testing.

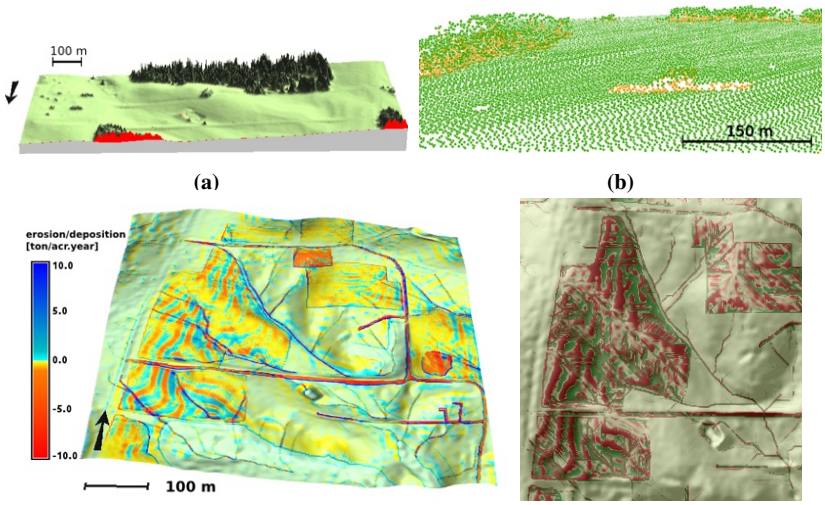


Figure 3. Images from the Geospatial Modeling and Analysis course assignment reports. Working with LiDAR data: (**a**) bare ground and first return surface cross-section in GRASS GIS; (**b**) multiple return point cloud visualized in ArcGIS. Modeling landscape processes: (**c**) erosion-deposition map draped over an elevation model visualized in GRASS GIS; (**d**) erosion-deposition map displayed over shaded relief in ArcGIS.

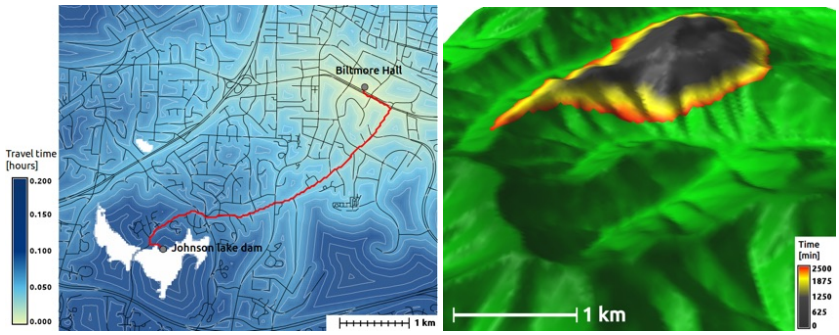


Figure 4. Images from the Geospatial Modeling and Analysis course exam and a project paper: (**a**) least cost path analysis done in GRASS GIS for the midterm exam; (**b**) fire spread modeling done in GRASS GIS for a final project.

On-campus sections are offered in a collaborative classroom with groups of students clustered around tables equipped with a computer with a large screen. Most students, however, use their own laptops,

running GRASS GIS on different operating systems, including Microsoft Windows, Mac OS X and various Linux distributions. FOSS enables students to continue using the software installed on their computers, even after they have finished their university studies.

The online course material includes lecture slides with audio and associated assignment workflows used by both distance education and on-campus students. The course website with all of the assignments was redesigned to improve navigation, readability and management of content. Students can interact with the instructor and teaching assistants through a private forum set up for the course, and they get feedback on their assignment reports and their project progress. The course is available at http://courses.ncsu.edu/gis582/common/.

3.2. GIS595/MEA592: Multidimensional Geospatial Modeling

The second, more advanced course is aimed at PhD students and highly skilled MS students. The focus is on working with 3D rasters, time series, dynamic models and simulations and multiple return LiDAR data. This course has a stronger focus on the advanced applications of free and open source GIS and GRASS GIS in particular. Programming (scripting) skills are useful, but not required. The first half of the course is led by the instructor, and the second half is led by students who have to prepare both a lecture and a working assignment on a topic of their choice while making sure that the other students in the class can successfully complete the assignment. Based on the students' interests, the topics change each year. In Figures 5 and 6a, students present their data and analysis in the Hunt Library Teaching and Visualization Lab and develop applications for the Tangible Landscape system. The course is available at http://courses.ncsu.edu/mea592/common/GIST_MEA592004.html.

Figure 5. Students in the Multidimensional Geospatial Modeling course visualizing a digital elevation model of an area affected by mountaintop mining in the Hunt Library Teaching and Visualization Lab.

3.3. LAR582: GIS for Designers

This course, GIS for Designers, aims to teach landscape architecture students how to use GIS as an integral part of the creative design process, rather than simply as an analytical tool. Students learn how to generate and develop the seed of a design in GIS as part of an iterative design process. They use conceptual and logical models to structure and generate designs, linking creative abstraction, computational exploration, structured decision-making and expression. As part of an iterative design

process, students learn workflows linking 3D GIS, 3D modeling and rapid prototyping, *i.e.*, 3D printing, computer numeric controlled (CNC) fabrication and laser cutting.

Topics for this course include terrain modeling, hydrological modeling, map overlay analysis, trail routing using least cost paths, 3D visualization and rapid prototyping. All topics are taught in both ArcGIS and GRASS GIS.

In a series of charrettes (*i.e.*, design workshops), students plan and design a trail network for Lake Raleigh Woods, a landscape on NCSU's Centennial Campus, using GIS and 3D modeling (Figure 6b,c). In the charrettes, student teams model environmental impacts and risks, identify potential trail routes and then develop a series of trail scenarios. Each team designs scenarios for differing and potentially conflicting stakeholder objectives, such as accessibility, aesthetics and conservation. For each scenario, they compute a trail network using map overlay analysis, iterative least cost path analysis and network analysis using either ArcGIS, GRASS GIS or the Tangible Landscape system, which uses GRASS GIS in the background. For their final review, students present their designs to a jury in the Teaching and Visualization Lab in Hunt Library using blended projectors, stereoscopic 3D and CNC routed models (Figure 7). The course is available at http://courses.ncsu.edu/lar582/common/.

(a) **(b)** **(c)**

Figure 6. Students working on projects with Tangible Landscape: (**a**) (**top**) sculpting a sand model; (**a**) (**bottom**) simulating storm surge on the Outer Banks of NC; (**b**) designing a park on the North Carolina State University (NCSU) campus; (**c**) and exploring various options for the design of a trail.

Figure 7. Landscape architecture students presenting their design for a trail system in the Hunt Library Teaching and Visualization Lab.

4. Future Directions

The integration of FOSS into geospatial education is largely driven by its suitability for science and its future potential in geospatial disciplines [12], following the success of FOSS in software engineering and business [5,23].

Students enrolling in our courses have very diverse backgrounds, often with limited programming skills when entering our GIST programs. Although the MS GIST curriculum includes the Geospatial Programming course, more programming experience is needed to introduce students to more advanced learning opportunities, including dynamic modeling and analysis techniques and to benefit from FOSS by examining or modifying source code [12,24]. The opportunity to study source code is an advantage often overlooked outside the computer science domain, yet important for geospatial science.

In our courses, we plan to introduce more opportunities for students to take advantage of the "open source" aspect of FOSS. Geospatial science, like any other science, requires an understanding of underlying processes, as well as the ability to verify results. To understand underlying processes, we continue to emphasize the importance of learning algorithms and how they are implemented in software; only with FOSS will this frontier in geospatial science education be reached. For example, to help students gain an understanding of the manner in which geospatial algorithms are implemented, we will ask students to find and study the segments of code in GRASS GIS that underlie the geospatial analyses discussed in a lecture, such as computing slope and aspect or bilinear interpolation between raster cells. We are also experimenting with ways to encourage students to connect equations and algorithms with the code, another benefit to education that is only possible with open source systems.

We are exploring the possibility of using a web-based tool that would allow students to interactively explore algorithm behavior as they study the teaching materials provided. The interactive tool would be linked with our teaching materials, standard user manual and, most importantly, the source code. With GRASS GIS as the back-end of analyses, online analyses and those done on students' computers would match the linked source code. If we limit the functionality of the tool, there will be no need to manage access rights for students, and we can keep the access open for everybody.

Another advantage of using open science and FOSS in education is the opportunity to interact with members of the community and to influence the FOSS development. Although we would like to see individual students getting involved in the community, we think that the direct involvement of the instructors is more important. The instructors can provide students with insights into the open source development process, how the community supports users and encourage student participation in the community later on. We, as instructors, have benefited from interaction with GRASS GIS developers, and the GRASS GIS project has been enhanced by our feedback and contributions.

Finally, we are planning a new course solely dedicated to the important role FOSS plays in science and how open science perspectives can lead to new geospatial solutions in research and decision making. Such training is essential for preparing next generation students for successful careers in open science and leaders across many fields.

5. Conclusions

We have demonstrated how free and open source software can be integrated into geospatial education to encourage a culture of openness, clearly differentiate between science and software and enable greater reproducibility in science. By using multiple software packages with different computational approaches, students can learn to decouple science and software and, thus, come to understand both the underlying concepts and their implementation. By understanding the concepts and their implementation,

students will become better scientists, who are able to generate, reproduce, critique and improve research and analysis. Furthermore, by learning to switch between computational tools, students may learn to be more flexible in developing strategies for research or analysis.

The method and tools we are using for the teaching materials are available to all, as all the dependencies are free and open source and the overall requirements are kept to minimum. The materials can be, if desired, easily combined with other more complicated solutions based on similar principles. Teaching materials are shared under a CC BY-SA license at https://github.com/ncsu-osgeorel.

Acknowledgments

We acknowledge the GRASS GIS developers and users, as well as the communities for the free and open source software used in the work presented in this paper, including, but not limited to, Git, Reveal.js, Python and Ubuntu. This paper was written using LaTeX and Overleaf.

Author Contributions

Helena Mitasova developed the courses GIS/MEA582 Geospatial Modeling and Analysis and MEA592 Multidimensional Geospatial Modeling in the Masters of Geospatial Information Science and Technology (MGIST) program at NC State University. Brendan Harmon developed the course LAR582 GIS for Designers. Vaclav Petras and Anna Petrasova developed the system to build the course web pages, created the layout and style and updated GIS/MEA582 course material to the new solution. Ross K. Meentemeyer directs the MGIST program at the Center for Geospatial Analytics at NC State. All authors participated in writing the paper.

Appendix

Comparison of File Formats

In this section we compare different file formats and tools we have considered when selecting solution for our teaching materials. First we discuss the assignment instructions and then the presentation slides. We evaluated the options keeping in mind that we want the assignment instructions to be managed in revision control system and to be published as web pages. We require the solution to depend only on FOSS, to be vendor-independent and accessible to anybody thanks to zero license cost.

1. Document Format for Assignments

HTML is one popular option, mainly because you can directly publish teaching materials online without any conversion needed. The known disadvantage of HTML is that it is not as easy to write by hand as lightweight markup languages. However, at least basic knowledge of HTML is typically expected from anyone in geospatial community. Unfortunately, for creating a well designed web site more advanced knowledge of HTML, CSS and also perhaps JavaScript is necessary, unless using graphical HTML editors which is not applicable because it usually excludes the possibility of editing files by hand without any particular software.

LaTeX is excellent, often used tool for desktop publishing and high-quality typography with the typical output format being PDF and PS. LaTeX provides means to be customized, to manipulate separate files, to typeset math expressions and to ensure consistent styles. However LaTeX is rarely used for creating HTML web pages.

XML formats such as DocBook provide a highly structured and flexible way of storing documents but writing XML by hand is too cumbersome. This applies also to eLML used by GITTA project which unlike DocBook is specialized for education purposes.

Many projects, including geospatial free and open source ones, write documentation in lightweight markup languages, such as reStructuredText in case of QGIS, OSGeoLive and GeoAcademy. reStructuredText is a feature rich language which is being standardized by Python community. Sphinx and Pandoc tools are often used to create websites from reStructuredText documents. Markdown is another widely used and supported lightweight markup, used for example by GitHub, Stack Exchange website or Doxygen documentation tool. Comparing to reStructuredText it is often considered simpler but it is not standardized because each implementation adds some additional features, specific to a tool or use case; there is for example feature-rich Pandoc's Markdown or GitHub flavored Markdown. Other lightweight markups, for example AsciiDoc or Textile, lack the wide acceptance which both reStructuredText and Markdown have.

Wiki systems such as MediaWiki use lightweight markup languages and store different revisions of a document. However, they require the same setup as other content management systems and the source documents are inside the system, so they are more difficult to share. On the other hand, there are some wiki systems which do not have such disadvantages. Zim, a desktop wiki, stores the documents as text files which can be easily managed using revision control system. One can edit the files graphically in Zim wiki application or as plain text files in any text editor and Zim wiki can export the content as a website.

2. Slides Format for Lecture Presentations

Although students in our courses take advantage of cloud-based solutions like Google Sites for managing their homework, we excluded Google Slides and similar services, since we would have minimal control over the files especially in the long run. Using free and open source desktop presentation software packages such as LibreOffice and OpenOffice in combination with OpenDocument format is often a good choice, because they support multimedia and they are simple to use in the sense that their users do not have to know any markup language. However, as discussed earlier, the binary format is not suitable for revision control systems.

Therefore we are more interested in presentations designed in plain text, for example in LaTeX or HTML5. LaTeX PDF presentations are easy and fast to prepare thanks to specialized packages such as Beamer and are especially useful for materials with many equations or source code. Unfortunately, the missing support for GIF animations and other multimedia is a big obstacle for using it in our courses. Students especially in the geospatial field are used to a lot of visuals and dynamic content in the lectures which LaTeX and PDF can't easily provide.

We decided to explore more HTML5-based presentations such as Reveal.js framework since they are almost as easy to prepare as LaTeX ones and although they typically lack sophisticated typography of LaTeX presentations.

Using lightweight markup language for presentations does not seems suitable for us because of number of visualizations we want to include and this usually requires finer control over inclusion of images and videos than lightweight markup languages usually offer. However, HTML5 presentation framework Reveal.js offers the possibility to write text in Markdown.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Rocchini, D.; Neteler, M. Let the four freedoms paradigm apply to ecology. *Trends Ecol. Evol.* **2012**, *27*, 310–311.
2. Steiniger, S.; Hay, G.J. Free and open source geographic information tools for landscape ecology. *Ecol. Inform.* **2009**, *4*, 183–195.
3. Lees, J.M. Open and free: Software and scientific reproducibility. *Seismol. Res. Lett.* **2012**, *83*, 751–752.
4. Rey, S.J. Open regional science. *Ann. Reg. Sci.* **2014**, *52*, 825–837.
5. Noyes, K. Open Source Software is Now a Norm in Businesses. Available online: http://www.pcworld.com/article/228136/open_source_software_now_a_norm_in_businesses.html (accessed on 15 January 2015).
6. Yves de Montcheuil, T. Open Source has Become Mainstream but Still Drives Innovation. Available online: http://www.zdnet.com/article/open-source-has-become-mainstream-but-still-drives-innovation (accessed on 30 January 2015).
7. Sui, D. Opportunities and impediments for open GIS. *Trans. GIS* **2014**, *18*, 1–24.
8. Pappano, L. The Year of the MOOC. Available online: www.nytimes.com/2012/11/04/education/edlife/massive-open-online-courses-are-multiplying-at-a-rapid-pace.html (accessed on 15 January 2015).
9. Weibel, R.; Bleisch, S.; Fisler, J. How open source can help achieving sustainability of e-learning content: The GITTA experience. In Proceedings of the 2012 Open Source Geospatial Research & Education Symposium, Yverdon-les-Bains, Switzerland, 24–26 October 2012.
10. Davis, P. Establishing a US National FOSS4G academy. In Proceedings of the 2012 Open Source Geospatial Research & Education Symposium, Yverdon-les-Bains, Switzerland, 24–26 October 2012.
11. Rey, S.J. Open source spatial analysis: Lessons for research and education from PySAL. In Proceedings of the 2012 Open Source Geospatial Research & Education Symposium, Yverdon-les-Bains, Switzerland, 24–26 October 2012.
12. Ertz, O.; Rey, S.J.; Joost, S. The open source dynamics in geospatial research and education. *J. Spat. Inf. Sci.* **2014**, *8*, 67–71.

13. Branger, F.; Jankowfsky, S.; Vannier, O.; Viallet, P.; Debionne, S.; Braud, I. Use of open-source GIS for the pre-processing of distributed hydrological models. In *Geospatial Free and Open Source Software in the 21st Century*; Springer: Berlin, Heidelberg, Germany, 2012; pp. 35–48.
14. Stallman, R. The GNU manifesto. *Dr. Dobb's J. Softw. Tools* **1985**, *10*, 30–35.
15. Raymond, E. The cathedral and the bazaar. *Knowl. Technol. Policy* **1999**, *12*, 23–49.
16. Mitasova, H.; Landa, M.; Shukunobe, M. Building open source geospatial education at research universities: Where we are and what is holding us back. In Proceedings of the 2012 Open Source Geospatial Research & Education Symposium, Yverdon-les-Bains, Switzerland, 24–26 October 2012.
17. Mitasova, H.; Schweik, C. Open source geospatial education and research. *GIM Int.* **2013**, *27*, 25–29.
18. Petrasova, A.; Harmon, B.; Petras, V.; Mitasova, H. GIS-Based environmental modeling with tangible interaction and dynamic visualization. In Proceedings of the 7th International Congress on Environmental Modelling and Software, San Diego, CA, USA, 15–19 June 2014.
19. Davis, P. Educating 21st century geospatial technology industry workers with open source software. *ISPRS Int. J. Geo-Inf.* **2015**, under review.
20. OSGeo-Edu Mailing List Subscribers. Our Proposed Educational Content and Management Standard. Available online: http://lists.osgeo.org/pipermail/edu_discuss/2008-January/thread.html (accessed on 30 January 2015).
21. E-Learning for the Open Geospatial Community (ELOGeo). Available online: http://elogeo.nottingham.ac.uk/xmlui/ (accessed on 30 December 2013).
22. OSGeo Community. OSGeo Educational Content Inventory. Open Source Geospatial Foundation. Available online: http://www.osgeo.org/educational_content (accessed on 30 January 2015).
23. Deutscher, M. Vast Majority of Software Developers Now Use Open Source, Forrester Reports. Available online: http://siliconangle.com/blog/2014/10/30/vast-majority-of-software-developers-now-use-open-source-forrester-reports/ (accessed on 16 January 2015).
24. Morin, A.; Urban, J.; Adams, P.; Foster, I.; Sali, A.; Baker, D.; Sliz, P. Shining light into black boxes. *Science* **2012**, *336*, 159–160.

© 2015 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).

C Open source approach to urban growth simulation

Reprint

Anna Petrasova, Vaclav Petras, Derek Van Berkel, **Brendan A. Harmon**, Helena Mitasova, and Ross K. Meentemeyer. 2016. Open source approach to urban growth simulation. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Prague: International Society of Photogrammetry and Remote Sensing. DOI:`http://dx.doi.org/10.5194/isprs-archives-XLI-B7-953-2016`

Attribution

Anna Petrasova, the lead author, developed the code, prepared the data and model parameters, designed, implemented, and documented the case study, and wrote the paper.

Vaclav Petras helped to develop the code and help to implement and document the case study.

Derek Van Berkel advised on the model, helped to prepare the data and model parameters, and edited the paper.

I helped document the case study and edited the paper.

Helena Mitasova and Ross Meentemeyer advised on the concept and model and edited the paper.

The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLI-B7, 2016
XXIII ISPRS Congress, 12–19 July 2016, Prague, Czech Republic

OPEN SOURCE APPROACH TO URBAN GROWTH SIMULATION

A. Petrasova[a,b,*], V. Petras[a,b], D. Van Berkel[a], B. A. Harmon[a,d], H. Mitasova[a,b], R. K. Meentemeyer[a,c]

[a] Center for Geospatial Analytics, North Carolina State University, USA - dbvanber@ncsu.edu
[b] Department of Marine, Earth, and Atmospheric Sciences, North Carolina State University, USA - (vpetras, akratoc, hmitaso)@ncsu.edu
[c] Department of Forestry and Environmental Resources, North Carolina State University, USA - rkmeente@ncsu.edu
[d] Department of Landscape Architecture, North Carolina State University, USA - baharmon@ncsu.edu

Commission VII, SpS10 - FOSS4G: FOSS4G Session (coorganized with OSGeo)

KEY WORDS: GRASS GIS, FUTURES, urbanization, land change, open science, simulation

ABSTRACT:

Spatial patterns of land use change due to urbanization and its impact on the landscape are the subject of ongoing research. Urban growth scenario simulation is a powerful tool for exploring these impacts and empowering planners to make informed decisions. We present FUTURES (FUTure Urban – Regional Environment Simulation) – a patch-based, stochastic, multi-level land change modeling framework as a case showing how what was once a closed and inaccessible model benefited from integration with open source GIS. We will describe our motivation for releasing this project as open source and the advantages of integrating it with GRASS GIS, a free, libre and open source GIS and research platform for the geospatial domain. GRASS GIS provides efficient libraries for FUTURES model development as well as standard GIS tools and graphical user interface for model users. Releasing FUTURES as a GRASS GIS add-on simplifies the distribution of FUTURES across all main operating systems and ensures the maintainability of our project in the future. We will describe FUTURES integration into GRASS GIS and demonstrate its usage on a case study in Asheville, North Carolina. The developed dataset and tutorial for this case study enable researchers to experiment with the model, explore its potential or even modify the model for their applications.

1. INTRODUCTION

Population growth in cities worldwide drives changes in land use often negatively impacting the environments in which people live and undermining the resilience of local ecosystems. The need to understand the trade-offs urban planners are facing gave rise to a number of different land change simulation models, which proved to be powerful tools for exploring alternative scenarios and their impacts on various aspects of human-environmental systems (Chaudhuri and Clarke, 2013, Verburg et al., 2002, Sohl et al., 2007, Waddell, 2002). Despite the influence of the spatial structure and connectivity of urbanizing landscapes on biodiversity, water quality, or flood risks (Alberti, 2005), most urban growth models are based on cell-level conversions and have not focused on generating realistic spatial structures across scales (Jantz and Goetz, 2005). To bridge the gap between cell- and object-based representation, we developed FUTURES (FUTure Urban-Regional Environment Simulation), a patch-based, multi-level modeling framework for simulating the emergence of landscape spatial structure in urbanizing regions (Meentemeyer et al., 2013). The FUTURES model was successfully applied in several cases including a study of land development dynamics in the rapidly expanding metropolitan region of Charlotte, North Carolina (Meentemeyer et al., 2013) and an analysis of the impacts of urbanization on natural resources under different conservation strategies (Dorning et al., 2015). Most recently, FUTURES was coupled with ecosystem services models to examine the impacts of projected urbanization and urban pattern on several ecosystem services and their trade-offs (Shoemaker, 2016, Pickard et al., in prep.).

In order to study the complex interactions between human and natural systems, interdisciplinary researchers are coupling existing simulation models. Land change modeling plays often a crucial role in these coupled models. Previous case studies with FUTURES have demonstrated that the model can be applied to a wide range of cases with different study systems and aims. The initial implementation of model, however, was a prototype that was not ready to be shared with scientific community. The model accumulated too much "technical depth" (Easterbrook, 2014) during its initial development, making it difficult to add new features or run the simulation at larger scales. In order to continue adding new capabilities to FUTURES and to promote its usage both inside and outside of the land use community, we decided to revise the implementation of the FUTURES model and develop a new version which would be (a) more efficient and scalable, (b) as easy to use as possible for a wider audience and (c) fully open source and maintainable in the long run. To achieve these goals we decided that instead of keeping FUTURES as a standalone application, we would take advantage of existing geospatial software and integrate FUTURES into open source GRASS GIS (Neteler and Mitasova, 2008). By using GRASS GIS' efficient geospatial libraries we can develop better and higher-level code. Providing open source software to the scientific community entails more than just releasing the actual code – documentation, tutorials, installation instructions, binaries and support are also needed and require considerable effort. Without this effort, models cannot be practically used by other researchers. By using GRASS GIS' existing infrastructure we could focus on developing the actual materials instead of managing our own server infrastructure.

In this article we present a new version of the FUTURES urban growth model that is available as the *r.futures* module set from the GRASS GIS add-on repository. This new version of FUTURES streamlines data processing, provides opportunities to study urbanization on mega-regional scales, and allows for more reproducible research in the land change community. We demonstrate this new version of FUTURES with a case study of the Asheville metropolitan area in North Carolina, USA.

*Corresponding author

2. FUTURES MODEL

FUTure Urban-Regional Environment Simulation is a stochastic, patched-based model for projecting landscape patterns of urban growth (Meentemeyer et al., 2013). FUTURES has a modular structure consisting of 3 main submodels: DEMAND, POTENTIAL and PGA (patch-growing algorithm), see Figure 1. Land conversion is driven by projected population demand computed by the DEMAND submodel, and is spatially defined by a probability surface derived by the POTENTIAL submodel from multiple environmental and socio-economic predictors. The population demand and the effects of land change drivers can vary in space by subregions, such as jurisdictional units, allowing projections across heterogeneous landscape. FUTURES main strength lies in realistically modeling the spatial structure of urban change by growing patches that are parameterized by size and compactness and calibrated using historical data. For a detailed explanation of FUTURES' components, please refer to Meentemeyer et al. (2013).

Figure 1: Simplified schema of FUTURES conceptual model with inputs and outputs in gray and submodels in yellow

The original implementation of FUTURES consisted mainly of the patch-growing algorithm, a standalone program written in a mixture of C and C++. The PGA program itself utilized inefficient algorithms and required raster data in ASCII format as input leading to very slow initialization. The DEMAND submodel was computed in a spreadsheet and POTENTIAL coefficents were derived using R statistical software. No official implementation of these submodels existed so each researcher developed a different workflow. This made it difficult for peers to verify each other's work. Several scripts for calibrating patch characteristics derived with FRAGSTATS (McGarigal et al., 2012) existed, however these tools were written in an unnecessarily low-level language for a specific case using the author's directory layout.

When revising the original implementation of FUTURES we identified several issues which needed to be addressed. First, it is important to follow best practices for scientific computing (Wilson et al., 2014) including use of a versioning system, writing documentation and testing. We also wanted to minimize tasks that had been done manually in order to make the process more efficient and avoid errors that are often difficult to detect. When automating tasks we had to compromise between the flexibility and simplicity of the workflow. We also focused on making FUTURES scalable so that it can run large scale applications at a relatively fine spatial resolution. Finally, we designed FUTURES to be more user-friendly and easy to test so that anyone can confidently apply it to their research.

3. INTEGRATION IN GRASS GIS

GRASS GIS has had a long history as a platform for scientific models (Chemin et al., 2015). As an open source GIS used by researchers worldwide and one of the founding projects of OSGeo, GRASS GIS provides a stable environment for the development of scientific models for studying problems from various domains including geomorphology, hydrology, planetary science, landscape ecology, hazard mapping, archaeology, renewable energy and transportation. Thanks to the numerous scientist and developers who have been involved, GRASS GIS today provides a large spectrum of geospatial modules ranging from basic GIS functionality to highly specialized models. Most of the specialized tools are not part of standard GRASS GIS installations, but are easily accessible from the add-on repository.

There were multiple reasons for our decision to integrate FUTURES into GRASS GIS as an add-on. Some of these reasons were specific to FUTURES, but others apply to any spatial, scientific model. Integrating a model into GIS gives both users and developers a wide array of standard geospatial tools that simplify the implementation of a model, and streamline pre- and post-processing and visualization. GRASS GIS provides model developers a raster library for highly efficient data reading and writing. This means that FUTURES no longer has to read in ASCII files, significantly reducing time needed for initialization. Furthermore, raster data from FUTURES simulations are efficiently compressed. Despite ever increasing disk space, it is still quite important to reduce the file size, especially for stochastic spatio-temporal simulations, which typically generate huge datasets. In order to achieve the best speed performance, most GRASS GIS functionality is implemented in C. Because of this, we could easily integrate FUTURES code, written in a mix of C and C++, without major rewriting. For portability reasons we later decided to use the C99 standard. While C and C++ are the preferred languages for computationally expensive algorithms, GRASS GIS also supports Python as the primary scripting language. This is crucial because FUTURES had many steps of data preparation that we were easily able to automate using Python scripting. Model developers can appreciate GRASS GIS' automatic generation of command line interfaces, Python interfaces and graphical user interfaces (GUI). Simply by defining options in C or Python modules we can call the same module from a GUI dialog, a Python script or a Bash script. A graphical interface makes FUTURES easy to use, especially for users on the Windows platform. A Python or Bash interface, however, is needed for more advanced applications such as running FUTURES in parallel on a high performance computer. GRASS GIS provides infrastructure for publishing and distributing models to users on all major platforms. Models and tools in GRASS GIS's Add-on repository[1] can be easily browsed and installed with their documentation, relieving researchers of the burden of maintaining such infrastructure.

3.1 Implementation

We implemented FUTURES as a set of GRASS GIS modules starting with a common prefix *r.futures*:

- *r.futures.demand* extrapolates the area of developed land from population trends and projections.
- *r.futures.devpressure* computes the development pressure predictor.
- *r.futures.potential* models the development probability surface through multi-level logistic regression.
- *r.futures.calib* calibrates patch sizes and shapes.
- *r.futures.pga* simulates urban development using the patch growing algorithm.

[1] `https://grass.osgeo.org/download/addons/`

In addition, we implemented the add-on *r.sample.category* needed for the workflow. Since its functionality is not specific to FUTURES, we kept it separate. All of these add-ons can be conveniently installed from GRASS GIS using the GUI or command line[2]. Each individual add-on has a manual page accessible both online and offline. Figure 2 shows FUTURES workflow and the inputs needed for each tool. In the following sections we describe the developed tools, their functionality and implementation in GRASS GIS.

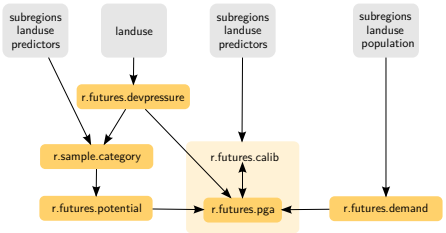


Figure 2: Diagram of FUTURES workflow showing how are *r.futures* modules (yellow boxes) chained and what are their input data (grey boxes). As indicated by the light yellow box, module *r.futures.calib* calls *r.futures.pga*.

3.1.1 r.futures.demand Based on historical land development and population growth, the DEMAND submodel (implemented as *r.futures.demand*) projects the rate of per capita land consumption for each year of the simulation and each subregion. This Python module uses GRASS GIS Python Scripting Library and the NumPy, SciPy and matplotlib libraries for scientific computing to approximate the relation between population and land consumption with a statistical model described by a linear, logarithmic or exponential curve. For example, a logarithmic relation means that a growing population requires less developed land per person over time. With enough data points, the module can select the best curve for each subregion based on residuals. The primary outputs are plain text files with tab-separated values representing the number of cells to be converted to developed land each year for each subregion. The module plots the resulting curves and projected points for each subregion (Figure 3) so that the results can be visually inspected. The module *r.futures.demand* provides a fast way to estimate the land demand given a large number of subregions with diverse population trends and thus allows us to quickly explore different population scenarios.

3.1.2 r.futures.devpressure Development pressure is one of the most important predictors of where development is likely to happen. For each cell it is computed as a distance decay function of neighboring developed cells (Meentemeyer et al., 2013). Compared to the tool previously used for computing development pressure, the new Python module *r.futures.devpressure* provides a faster and more efficient implementation by taking advantage of the existing GRASS GIS module *r.mfilter* written in C for moving window analysis with custom designed matrix filters. By precomputing the matrix of distances we avoid repeated distance computations resulting in faster processing. Because the new implementation is less memory intensive it can be used for larger regions than the previous tool.

3.1.3 r.futures.potential uses multilevel logistic regression to model development suitability based on environmental, infrastructural, and socio-economic predictors such as distance to roads or topographic slope. We randomly sample these predictors and

[2] `g.extension r.futures`

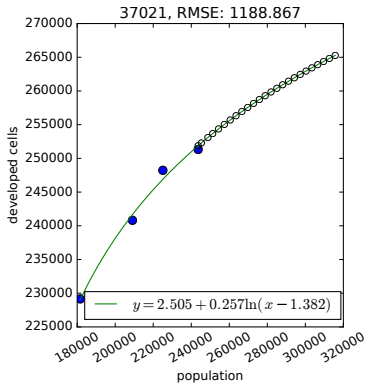


Figure 3: An example of *r.futures.demand* output plot showing the logarithmic relation between population and land consumption for the county with ID 37021. Observed data are showed as blue dots and predicted data as circles.

the observed change from undeveloped to developed cells to estimate the coefficients of the multilevel logistic regression. The core of this module is a script in the R language (R Development Core Team, 2008), which uses the package lme4 (Bates et al., 2015) for fitting generalized linear mixed-effects models and the package MuMIn (Barto, 2015) for automatic model selection. The output file is a plain text file with tab-separated regression coefficients. This script is wrapped in Python for more seamless processing and chaining of modules. The coupling between R, Python and GRASS GIS is intentionally very loose to make the workflow possible in the Windows environment where some of the other, more elegant, options such as rpy2[3] are complicated to use. We performed stratified sampling of observed new development and predictors using GRASS GIS add-on *r.sample.category*. Although we developed this add-on for urban growth modeling with FUTURES, its application is much broader. In order to encourage its use in other applications we made it a general module rather than making it part of the *r.futures* tool set.

3.1.4 r.futures.pga is the main engine of FUTURES – it simulates urban growth using inputs from the DEMAND and POTENTIAL submodels. The patch growing algorithm (PGA) stochastically allocates a seed for new development across the development suitability surface, challenges the seed by comparing it with a random number between 0 and 1, and then grows a discrete patch from the seed if it survives (Meentemeyer et al., 2013). This process repeats until the number of converted cells specified by DEMAND is met. The development pressure predictor and then the development suitability values are updated based on the newly developed cells. (The development suitability is computed internally from predictors and regression coefficients supplied by POTENTIAL.) We kept the original patch growing algorithm, but significantly improved its implementation to make it faster, more memory efficient and simpler to use. We replaced a custom, undocumented configuration file with a standard module interface usable from GUI or the command line, and restructured the input and output parameters and their names so that they are easy for users to understand. We used efficient GRASS GIS libraries for reading and writing raster data, which minimized the

[3] rpy2 is a Python package for using R from Python (`http://rpy2.bitbucket.org/`)

time needed to initialize the simulation. FUTURES now reads rasters in GRASS's native format instead of ASCII files. This decreased the time needed for model initialization from several minutes to several seconds for a region with tens of millions of cells. Furthermore, we replaced the static allocation of internal structures with dynamic allocation and reduced the overall memory requirements so that FUTURES could run on large regions with tens or hundreds of counties as well as smaller areas like our case study. Finally, through the use of appropriate programming techniques, such as binary search, we significantly increased the speed of the algorithm.

3.1.5 r.futures.calib We developed a dedicated Python module for calibrating patch sizes and shapes that runs the module *r.futures.pga* with different combinations of patch parameters and outputs a table with scores for each combination of patch parameters. The simulation is run multiple times for each combination to account for the stochasticity of the model. To speed up the calibration process *r.futures.calib* can take advantage of multiple computer cores.

4. CASE STUDY

To demonstrate how the new FUTURES framework can be used to simulate urban growth, we present a case study for Asheville metropolitan area located in the Blue Ridge Mountains in the west of North Carolina, USA. The region consists of five counties with total area of 6,271 km^2 and around 477,000 people based on 2014 population estimates. It is characterized by rapid population growth around Asheville, the largest city of the region. New development is constrained by the steep mountainous terrain and large national and state parks. We simulate urban growth from 2012 to 2030 using publicly available data, including the USGS's National Land Cover Database (NLCD) (Homer et al., 2015, Fry et al., 2011, Homer et al., 2007, Fry et al., 2009), past estimates and future projections of county populations (NCOSBM, 2015), boundaries and roads provided by the United States Census Bureau's database (TIGER) and a digital elevation model from the National Elevation Dataset (NED) distributed by the USGS.

4.1 Approach

There are several steps required to run the FUTURES simulation:

- Preprocess the data.
- Estimate per capita land consumption controlling the total area of converted land.
- Derive the development suitability statistical model to control where the new development happens.
- Calibrate patch size and shape.
- Run the urban growth simulation.

4.1.1 Data preparation The core input data for urban growth modeling with FUTURES is a timeseries of land cover maps, which can be derived by various methods from satellite imagery. In this study we used the 2001, 2006 and 2011 NLCD Land Cover products and 1992/2001 Retrofit Land Cover Change product to derive a 30-meter binary representation of developed areas. We excluded national and state parks, water bodies and wetlands from further analysis. We used NLCD products that are available for the contiguous USA so that this study and its workflow would be easier to reproduce and apply to other study areas. We obtained population statistics from the North Carolina Office of State Budget and Management, which are based on 2000 and 2010 censuses and include past as well as future estimates of population per county for each year up to 2035. Data for the 5 counties studied were extracted and formatted as a comma-separated values (CSV) file.

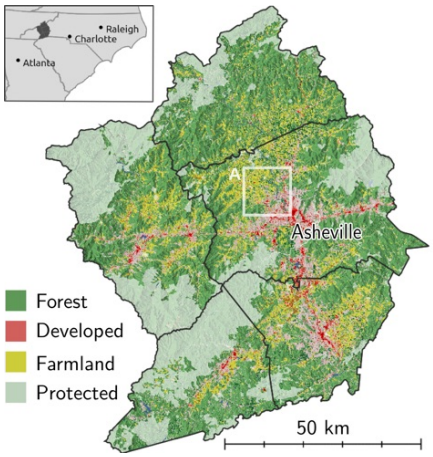


Figure 4: 2011 land cover (Homer et al., 2015) and protected areas (Anderson and Sheldon, 2011) in the Asheville metropolitan area in the west of North Carolina, USA. Inset A is used in Figure 5.

4.1.2 DEMAND We derived the relation between population and land consumption from the series of binary rasters of developed areas and population statistics to model how much land will be developed each year of the simulation. Using the module *r.futures.demand* we explored different curve fitting methods and derived the per capita land consumption from period 1992 through 2011, which was characterized by population growth with decreasing demand for land per person over time. We expect similarly low rates of per capita land consumption in the following years because development is restricted by the mountainous terrain and large protected areas. Based on RMSE and visual inspection of the plots created by *r.futures.demand* we selected either linear or logarithmic relations for each county, where the function coefficients were automatically determined using linear regression and non-linear least squares optimization in *r.futures.demand* (Figure 3).

4.1.3 POTENTIAL We used multilevel logistic regression to predict where new development happens based on environmental, infrastructural and socio-economic site suitability predictors. Using *r.sample.category* we sampled predictors on 8000 randomly selected locations and estimated the model coefficients using the R package lme4 integrated into the module *r.futures.potential*. The sample points were stratified by the response variable where new sites developed since 1992 have a value of 1 and sites that are undeveloped in 2011 have a value of 0. We included counties as the group level indicator in the multilevel model to account for differences across jurisdictional boundaries. From the initial list of hypothesized predictors (slope, distance to water, protected areas, interchanges, travel time to cities and towns, forest occurrence and road density) we identified a set of predictors (Table 1) resulting in a model with the lowest AIC (Akaike information criterion) score. We verified the robustness of the selected predictors by repeating the random sampling and model selection process multiple times. In addition to these predictors, we included also development pressure, a special, dynamic predictor that is updated during the simulation based on new simulated development to enable positive feedback. We computed the initial development pressure raster with *r.futures.devpressure*; its subse-

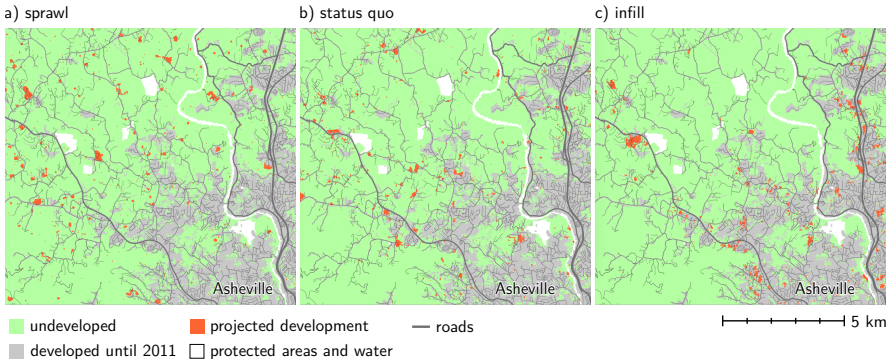


Figure 5: Results of three realizations of multiple stochastic runs with different scenarios. Depending on the scenario, simulated development is more diffuse (a) or more compact (c).

quent updates are performed in memory during the simulation.

Predictors	Estimate*	Std. Error
Intercept (varies by county)	-2.593	0.269
Development pressure	0.058	0.005
Road density	0.118	0.007
Percentage of forest	-0.013	0.002
Distance to protected areas	-0.140	0.039
Distance to water bodies	-0.148	0.022

* all P-values < 0.001

Table 1: List of selected predictors and estimated coefficients for site suitability model

4.1.4 Patch calibration Prior to running the urban growth simulation implemented in *r.futures.pga* we calibrated the input patch compactness and size to match the simulated patterns with the observed patterns from 1992 to 2011. Since calibration is a time consuming process, we ran the module *r.futures.calib* for Buncombe county and applied the results to the rest of our study region. We choose Buncombe County which includes the city of Asheville because it is where most new development will occur. For each combination of patch parameters we compared the patch characteristics averaged from 20 runs of the urban growth simulation with the known patches. Based on the score we selected patch parameters resulting in high compactness which is expected for mountainous regions.

4.1.5 Urban growth simulation Having collected all necessary input data, we ran *r.futures.pga* with a 1 year time step until 2035 for the entire study region at 30 m resolution. To account for different future policies regarding new development, we explored scenarios altering the site suitability to encourage infill or sprawl by changing `incentive_power` parameter of *r.futures.pga*. This value transforms the probability p a cell is developed to p^x where $x = 1$ represents status quo, higher values of x result in infill and lower values in sprawl. In addition to the status quo we simulated scenarios with x equals 0.25, 0.5, 2 and 4. We repeated each scenario 50 times to account for the model's stochastic behavior.

4.2 Results

The resulting development patterns of three realizations of the random runs are visible in Figure 5 for the status quo, infill scenario ($x = 4$) and sprawl scenario ($x = 0.25$). The simulated patches realistically mimic the current patches of development in shape and size and are mostly, but not exclusively adjacent to roads as expected. Furthermore, we post-processed the results to study how different urban growth policies influence the loss of forest and agricultural land in the Asheville area (Figure 6) by averaging the loss of both land use categories over the 50 runs. In all scenarios, forest is more affected by future development than farmland. The extreme case of urban sprawl results in twice as much forest as farmland being developed. Status quo scenario leads to the smallest difference between the areas converted to forest and farmland. Interestingly, infill scenario develops the forested area in a similar way as sprawl does, which is not surprising considering developed areas are largely surrounded by forest patches.

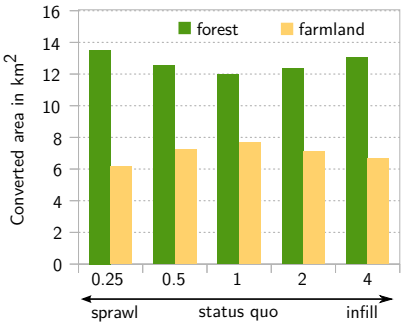


Figure 6: Area in km^2 of converted land from forest (green) and farmland (yellow) to urban differs for urban sprawl and infill scenarios. Numbers 0.25 to 4 represent the exponent x which transforms development probability p to p^x.

Table 2 shows the computational resources necessary for running this case study and compares the time and memory requirements with the original implementation of FUTURES. Note that our study area is fairly small (12 million cells) and when applied to larger regions with more projected development, the expected speed gain is even more significant as we changed the complexity of one of the core algorithms from linear to logarithmic. Because the individual stochastic runs are independent of each other this simulation is an "embarrassingly parallel" problem (Herlihy and

Shavit, 2012) in which the computation can easily be distributed across multiple computer cores.

FUTURES version	memory	1 run	all runs (250)
original	1.7 GB	60 s	4 h 10 min
r.futures	0.86 GB	19 s	1 h 20 min

Table 2: Time and memory needed to run the simulations with the old version of FUTURES and the new *r.futures* implemented in GRASS GIS on a laptop with 64-bit Ubuntu 14.04 LTS, Intel Core i7-4760HQ @ 2.10GHz using 1 CPU and running on external hard drive.

The input data and instructions to run the model are available as part of material developed for the US-IALE 2016 Annual Meeting workshop on FUTURES[4].

5. DISCUSSION

The new FUTURES framework is split into independent GRASS GIS modules so that the modeling workflow is flexible and extendable. By using standardized inputs and outputs (raster layers and CSV files) and described interface we allow FUTURES' users to replace DEMAND and POTENTIAL implementations by their own tools, which may be better suited to the characteristics and datasets available for their study systems. We ran all previous studies on county level at 30 m resolution. FUTURES, however, can be applied to larger or smaller scales as long as there is data available and the patch characteristics are properly calibrated. Future research will explore nested scales in order to address the different scales of the population data and the spatial drivers of land change.

6. CONCLUSION

We presented a new, open source version of the FUTURES urban growth model that is integrated into GRASS GIS, opening new possibilities for environmental scientists and urban planners to project and understand the impacts of urbanization at relevant ecological and decision-making scales. Integration into GRASS GIS allowed us to make FUTURES more efficient, simple to use and transparent. With documented code running on all platforms, FUTURES can now be easily tested and applied to study sites at local to megaregional scales. We illustrated how FUTURES can be used in a small case study of the Asheville metropolitan area. We also provided the instructions and data needed to reproduce this study as a step towards more reproducible research in land change science.

ACKNOWLEDGEMENTS

We would like to thank Monica Dorning and Douglas Shoemaker for discussing with us the original model implementation, and Brian Pickard and Georgina Sanchez for testing new FUTURES implementation.

REFERENCES

Alberti, M., 2005. The effects of urban patterns on ecosystem function. *International regional science review* 28(2), pp. 168–192.

[4] `https://grasswiki.osgeo.org/wiki/Workshop_on_urban_growth_modeling_with_FUTURES`

Anderson, M. and Sheldon, A. O., 2011. Conservation Status of Fish, Wildlife, and Natural Habitats in the Northeast Landscape: Implementation of the Northeast Monitoring Framework. Technical report, The Nature Conservancy, Eastern Conservation Science.

Barto, K., 2015. MuMIn: Multi-Model Inference. R package version 1.15.1.

Bates, D., Maechler, M., Bolker, B. and Walker, S., 2015. lme4: Linear mixed-effects models using Eigen and S4. R package version 1.1-9.

Chaudhuri, G. and Clarke, K. C., 2013. The sleuth land use change model: A review. *International Journal of Environmental Resources Research* 1(1), pp. 88–104.

Chemin, Y., Petras, V., Petrasova, A., Landa, M., Gebbert, S., Zambelli, P., Neteler, M., Löwe, P. and Di Leo, M., 2015. GRASS GIS: a peer-reviewed scientific platform and future research repository. In: *Geophysical Research Abstracts*, Vol. 17, p. 8314.

Dorning, M. A., Koch, J., Shoemaker, D. A. and Meentemeyer, R. K., 2015. Simulating urbanization scenarios reveals tradeoffs between conservation planning strategies. *Landscape and Urban Planning* 136, pp. 28–39.

Easterbrook, S. M., 2014. Open code for open science? *Nature Geoscience* 7(11), pp. 779–781.

Fry, J., Coan, M., Homer, C., Meyer, D. and Wickham, J., 2009. Completion of the National Land Cover Database (NLCD) 1992–2001 Land Cover Change Retrofit Product. Technical report, U.S. Geological Survey Open-File Report 20081379.

Fry, J., Xian, G., Jin, S., Dewitz, J., Homer, C., Yang, L., Barnes, C., Herold, N. and Wickham, J., 2011. Completion of the 2006 National Land Cover Database for the Conterminous United States. *PE&RS* 77(9), pp. 858–864.

Herlihy, M. and Shavit, N., 2012. *The Art of Multiprocessor Programming, Revised Reprint*. Elsevier.

Homer, C., Dewitz, J., Fry, J., Coan, M., Hossain, N., Larson, C., Herold, N., McKerrow, A., VanDriel, J. and Wickham, J., 2007. Completion of the 2001 National Land Cover Database for the Conterminous United States. *Photogrammetric Engineering and Remote Sensing* 73(4), pp. 337–341.

Homer, C., Dewitz, J., Yang, L., Jin, S., Danielson, P., Xian, G., Coulston, J., Herold, N., Wickham, J., and Megown, K., 2015. Completion of the 2011 National Land Cover Database for the conterminous United States – Representing a decade of land cover change information. *Photogrammetric Engineering and Remote Sensing* 81(5), pp. 345–354.

Jantz, C. A. and Goetz, S. J., 2005. Analysis of scale dependencies in an urban land-use-change model. *International Journal of Geographical Information Science* 19(2), pp. 217–241.

McGarigal, K., Cushman, S. and Ene, E., 2012. FRAGSTATS v4: spatial pattern analysis program for categorical and continuous maps. Computer software program produced by the authors at the University of Massachusetts, Amherst.

Meentemeyer, R. K., Tang, W., Dorning, M. A., Vogler, J. B., Cunniffe, N. J. and Shoemaker, D. A., 2013. FUTURES: Multilevel Simulations of Emerging UrbanRural Landscape Structure Using a Stochastic Patch-Growing Algorithm. *Annals of the Association of American Geographers* 103(4), pp. 785–807.

NCOSBM, 2015. County/State Population Projections.

Neteler, M. and Mitasova, H., 2008. *Open Source GIS: A GRASS GIS Approach*. Vol. 773, Springer, New York.

Pickard, B. R., Van Berkel, D., Petrasova, A. and Meentemeyer, R. K., in prep. Future patterns of urbanization reveal trade-offs among ecosystem services.

R Development Core Team, 2008. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Shoemaker, D. A., 2016. The role of spatial heterogeneity and urban pattern in modulating ecosystem services. Doctoral dissertation, North Carolina State University. Retrieved from http://www.lib.ncsu.edu/resolver/1840.16/11033.

Sohl, T. L., Sayler, K. L., Drummond, M. A. and Loveland, T. R., 2007. The FORE-SCE model: a practical approach for projecting land cover change using scenario-based modeling. *Journal of Land Use Science* 2(2), pp. 103–126.

Verburg, P. H., Soepboer, W., Veldkamp, A., Limpiada, R., Espaldon, V. and Mastura, S. S., 2002. Modeling the spatial dynamics of regional land use: the CLUE-S model. *Environmental management* 30(3), pp. 391–405.

Waddell, P., 2002. UrbanSim: Modeling urban development for land use, transportation, and environmental planning. *Journal of the American Planning Association* 68(3), pp. 297–314.

Wilson, G., Aruliah, D., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., Haddock, S. H., Huff, K. D., Mitchell, I. M., Plumbley, M. D. et al., 2014. Best practices for scientific computing. *PLoS Biol* 12(1), pp. e1001745.

D Immersive Tangible Geospatial Modeling

Reprint

Payam Tabrizian, Anna Petrasova, **Brendan Harmon**, Vaclav Petras, Helena Mitasova, and Ross Meentemeyer. 2016. Immersive Tangible Geospatial Modeling. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. GIS '16*. San Francisco, CA, USA: ACM, 88:1–88:4. DOI:http://dx.doi.org/10.1145/2996913.2996950

Attribution

Payam Tabrizian, the lead author, developed the concept, developed the code, designed, implemented, and documented the case study, and wrote the paper.

Anna Petrasova developed part of the code, co-designed Tangible Landscape, helped to design, implement, and document the case study, wrote part of the paper, and edited the paper.

I co-designed Tangible Landscape, helped design its integration with immersive virtual environments, helped to design, implement, and document the case study, wrote part of the paper, and edited the paper.

Vaclav Petras co-designed Tangible Landscape, helped to implement and document the case study, and edited the paper.

Helena Mitasova and Ross Meentemeyer contributed with advice, guidance, support, and editing.

Immersive Tangible Geospatial Modeling

Payam Tabrizian
ptabriz@ncsu.edu

Anna Petrasova
akratoc@ncsu.edu

Brendan Harmon
baharmon@ncsu.edu

Vaclav Petras
vpetras@ncsu.edu

Helena Mitasova
hmitaso@ncsu.edu

Ross Meentemeyer
rkmeente@ncsu.edu

Center for Geospatial Analytics
North Carolina State University
Raleigh, North Carolina 27695

ABSTRACT

Tangible Landscape is a tangible interface for geographic information systems (GIS). It interactively couples physical and digital models of a landscape so that users can intuitively explore, model, and analyze geospatial data in a collaborative environment. Conceptually Tangible Landscape lets users hold a GIS in their hands so that they can feel the shape of the topography, naturally sculpt new landforms, and interact with simulations like water flow. Since it only affords a bird's-eye view of the landscape, we coupled it with an immersive virtual environment so that users can virtually walk around the modeled landscape and visualize it at a human-scale. Now as users shape topography, draw trees, define viewpoints, or route a walkthrough, they can see the results on the projection-augmented model, rendered on a display, or rendered on a head-mounted display. In this paper we present the Tangible Landscape Immersive Extension, describe its physical setup and software architecture, and demonstrate its features with a case study.

CCS Concepts

•**Human-centered computing → Human computer interaction (HCI); Geographic visualization;** *Virtual reality;* Mixed / augmented reality;

Keywords

immersive virtual environments; augmented reality; tangible user interfaces; tangible interaction; landscape modeling; head mounted display; Oculus Rift

1. INTRODUCTION

Geographic information systems (GIS) are used to model, analyze, simulate, and visualize spatial patterns and processes. While GIS provide powerful computational tools for studying spatial phenomena and solving spatial problems, they can be challenging to use – requiring training and expertise. GIS's complicated interfaces and steep learning curve make it hard to rapidly test ideas, collaborate, or engage the public. Tangible interfaces for GIS enable multiple users to naturally, directly interact with geospatial data and analyses with their bodies affording an intuitive, collaborative environment that does not require training to use [7]. See Petrasova et al. and Amburn et al. for reviews of tangible interfaces for spatial modeling [5, 1].

Tangible Landscape – a tangible interface for GRASS GIS – couples a physical and digital model of a landscape through a real-time cycle of 3D scanning, geospatial computation, and projection [5]. Multiple users can interact with geospatial data, models, analyses, and simulations by manipulating a physical model of the landscape. Users' changes to the physical model are 3D scanned in real-time and imported into GRASS GIS, an open source GIS [4], for modeling, analysis, or simulation. The result is then projected back on the physical model so that users can evaluate their modifications. As a collaborative environment for exploring geographic data and solving spatial problems Tangible Landscape enables experts and non-experts alike to test how changes will impact the landscape.

Because Tangible Landscape represents landscape with projection-augmented, scale models that are seen from a bird's-eye view at a geographic scale, users do not experience the landscape at a human-scale. While early tangible interfaces for geospatial modeling like Illuminating Clay [6] used renderings to visualize human-scale views, we implemented an immersive virtual environment for a more engaging human-scale experience. We developed the Tangible Landscape Immersive Extension[1] that couples Tangible Landscape with an immersive virtual environment so that human-scale views can be rendered on devices like head-mounted displays (HMD). Users can define viewpoints and routes for walkthroughs with the tangible interface, see the view rendered on a display, and then experience the view with a HMD. For digital 3D modeling and animation we use Blender [2], a free and open source program for 3D modeling, rendering, simulation, animation, and game design. Blender has add-ons for working with georeferenced data and immersive virtual environments such as HMDs.

2. SYSTEM OVERVIEW

Figure 1 shows the physical setup of Tangible Landscape and the proposed extension. Tangible Landscape is com-

[1] http://tangible-landscape.github.io/immersive.html

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGSPATIAL'16, October 31 - November 03, 2016, Burlingame, CA, USA

© 2016 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4589-7/16/10.

DOI: http://dx.doi.org/10.1145/2996913.2996950

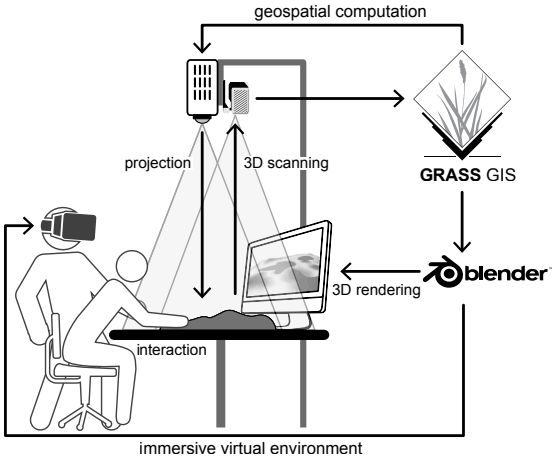


Figure 1: Physical setup of Tangible Landscape and the proposed extension

prised of 4 primary components: a malleable physical model, a 3D scanning device (Kinect for Windows v2), a projector, and a computer with GRASS GIS and software we developed for scanning and synchronizing all of the system components[2,3]. The scanner captures the shape of the physical model as the user modifies it and the point cloud is then georeferenced and transformed into a digital elevation model in GRASS GIS where all the subsequent geospatial analyses are performed. The resulting GIS layers are then projected back on the physical model to provide users with feedback. This extension adds two components to Tangible Landscape: 3D modeling software and an immersive virtual reality headset.

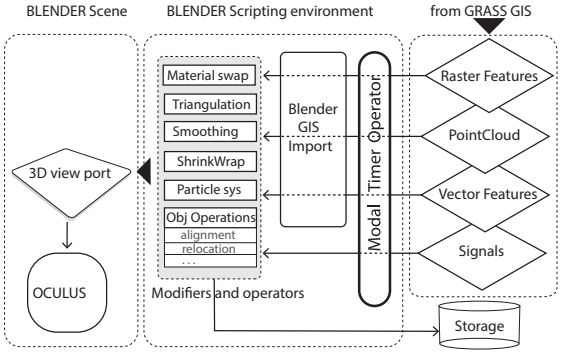


Figure 2: The application process flow diagram

Figure 2 illustrates the architecture of the Tangible Landscape Immersive Extension. GRASS GIS and Blender are loosely coupled through file-based communication. GRASS GIS (either continuously or when prompted) generates files with geospatial information in a specified system directory. Blender constantly monitors the incoming data and imports it for further processing using the *BlenderGIS* add-on module[4]. The history of the user's modifications can easily be preserved by storing the time-stamped, generated files in GIS formats. We setup and synchronized the monitored directory on a network drive to distribute the processing between two computers.

The system supports several modes of interaction (Figure 3) that generate four types of data: i) a point cloud in binary PLY format representing topography, ii) vector data in shapefile format including user defined tree patches, water features, walking routes, camera position, etc., iii) raster data in PNG format for texturing such as elevation maps, water flow, cost surfaces, orthophotos, etc., and iv) empty files signaling actions such as clearing the scene.

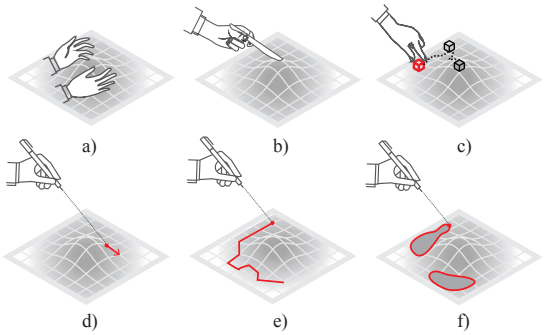


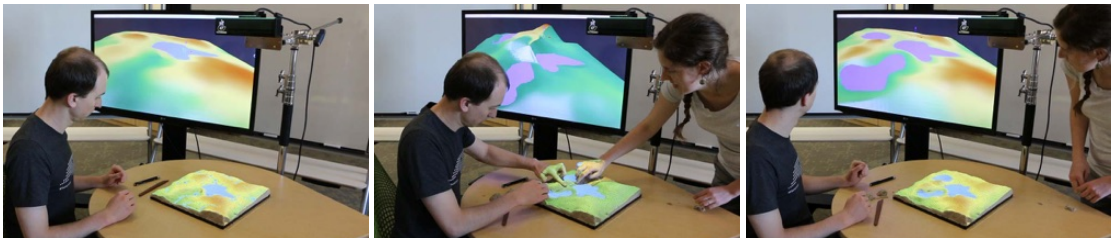
Figure 3: Interaction modes supported by Tangible Landscape: a) Sculpting topography by hand or b) with a tool. c) Placing markers to designate waypoints. d) Drawing viewpoints, e) walking routes, or f) patches with a laser pointer

Point cloud data. The core of the virtual 3D model is the terrain mesh, which is processed from a point cloud. As the user reshapes the physical model by hand (Figure 3a) or with a sculpting tool (Figure 3b), the depth data are acquired by the 3D scanner, processed by the GRASS GIS add-on *r.in.kinect*[3], and exported as a georeferenced point cloud in binary PLY format. Once the point cloud file has been detected in Blender, a *Delaunay triangulation* function is applied to generate a 3D mesh, which is then smoothed with the *Smooth* modifier. All scene objects that were dependent or attached to the previous mesh are aligned to the new mesh using Blender's *ShrinkWrap* modifier.

Raster data. Existing raster data such as orthophotos or raster-based geospatial analyses are projected on the physical model and can be imported into Blender and applied to the 3D terrain as a surface texture. For example, when modeling surface water flow, the raster projections represent the evolving spatial pattern of water depth. In coastal storm surge modeling the raster data represent water depth in flooded areas.

Vector features. Vector features are generated when a user draws a path (Figure 3d) or a boundary (Figure 3e) on the physical model using a laser pointer or markers (Figure 3c). The 3D scanner senses color information as RGB

[2]https://github.com/tangible-landscape/grass-tangible-landscape
[3]https://github.com/tangible-landscape/r.in.kinect
[4]https://github.com/domlysz/BlenderGIS



a) b) c)

Figure 4: Sculpting topography to form ponds

values for each point in the scanned point cloud. While the user is drawing with the laser pointer, *r.in.kinect* iterates over all points in each scan and searches for the brightest point. If the brightness, computed as sum of RGB values, is above certain threshold, *r.in.kinect* starts to record the positions of the brightest points. When the user stops drawing, no point above the threshold is found and the shape is written in a vector file as a line or area. By placing markers we can designate waypoints for a trail across the landscape. Their position is detected from the difference between the scan before and after placing the markers. Once their positions are identified, they are connected by a least cost path. Cost can be defined, for example, as the slope of the terrain. The designed features are then projected back on the physical model and simultaneously imported and rendered in Blender. Once features have been imported into Blender and aligned, they can be enhanced with Blender modifiers and operators. A water texture, for example, can be applied to polygons representing water bodies. Blender's *Particle System* modifier can be applied to polygons representing tree patches with given species, density, and diversity to generate trees on the 3D terrain model.

Viewpoints. Users can designate a viewpoint by drawing a line on the physical model. The starting point represents the view-origin and the end point determines the direction of the view (Figure 3d). After importing the line, the camera is aligned 6 feet above the vector's first vertex to keep the view-origin at the eye level.

Flythroughs and walkthroughs. Users can draw a curve on the physical model using markers or a laser pointer to design a flythrough or walkthrough animation in Blender. The curve represents the route the camera will follow during the animation. Users select either a perspective view or bird's-eye view and the camera is aligned to the first vertex of imported vector at the given height. The scene is animated to simulate a walkthrough or flythrough. Since the model will continue to update while the animation is playing, the user can continuously observe how landscape changes as it is being modified.

Immersive experience. We used the Blender add-on module *virtual_reality_viewport* to render the camera view as a separate viewport for the immersive HMD[5]. Since this add-on supports multiple modes of visualization in separate viewports, a group of users can simultaneously experience different views of the same scene. For example while one user interacts with the physical model and sees the scene rendered on a display, another user can explore the scene from another view with the HMD.

[5]https://github.com/dfelinto/virtual_reality_viewport/

3. DEMONSTRATION

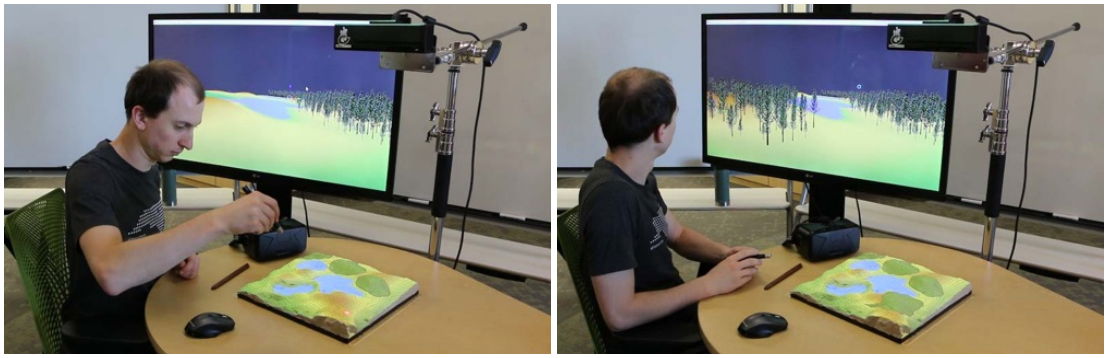
We demonstrate the coupling of Tangible Landscape with an immersive virtual environment with a case study in which users design a small park with water features, planting, and a recreational trail using the technology. Their goals were to control storm water runoff, create an aesthetic landscape, and plan a comfortable walking route with good views. They used polymer-enriched sand to build a malleable physical model that was easy to sculpt, yet would hold its form. They sculpted a scale model of a hypothetical, 50 ha landscape with several hills and a valley opening to the southwest.

Step 1: Ponding and landform design. Retention ponds are used by landscape planners and civil engineers to control and remediate storm-water runoff. Well designed retention ponds with planting can have aesthetic and recreational value. The users' first task was to design ponds for storm water management and aesthetics. The users modified the topography to create ponds and store runoff. Any soil excavated to create a pond had to be placed on site. Water flow and accumulation simulations were continuously projected onto the physical model to provide real-time feedback. The scanned and simulated data were also continuously imported into Blender for visualization. To simulate water flow during a storm event we used the module *r.sim.water*[6] to solve the shallow water flow continuity equation using a path sampling technique [3]. Water accumulation in the retention ponds was simulated by filling depressions in the terrain using the *r.fill.dir* module[7]. The Blender script processed the terrain point cloud, surface texture, and water boundary polygon and rendered the scene in 3D on a display. One of the users built a dam to form a large pond in the middle of the landscape (Figure 4a). Then both users collaboratively excavated two more ponds and built two small hills with the excavated soil (Figure 4b). They explored different views with the display to appraise the aesthetics of their design (Figure 4c).

Step 2: Vegetation design. In the second step one of the users designed patches of trees to frame views of the ponds. As the user drew patches with laser pointer the polygons were rendered on the physical model and filled with clusters of individual trees in the Blender scene (Figure 5).

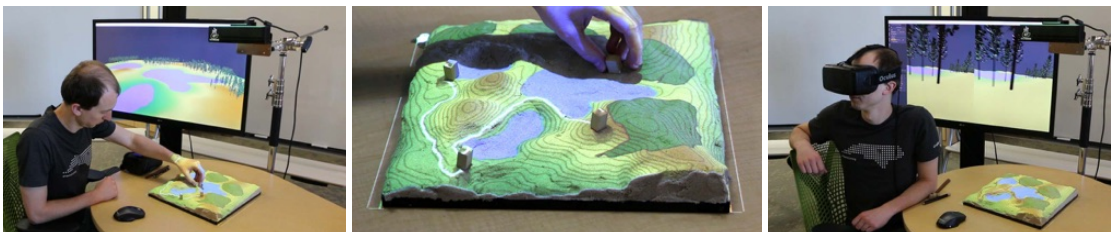
Step 3: Trail design. In the third step the users' goal was to design a cost-effective, easy to hike trail with good views. First the users placed markers on the physical model to identify waypoints along the new trail (Figure 6a, b). The optimal route between the waypoints was automatically

[6]https://grass.osgeo.org/grass72/manuals/r.sim.water.html
[7]https://grass.osgeo.org/grass72/manuals/r.fill.dir.html



a) b)

Figure 5: Drawing patches of trees with a laser pointer



a) b) c)

Figure 6: Designing an optimized trail

computed using the *r.walk*[8] algorithm to find the least cost walking path. *r.walk* can be parameterized to avoid high slopes and obstacles such as water bodies. The optimal trail route was continuously recomputed and projected onto the model to provide users feedback during the design process. The trails were also continuously updated in the Blender scene for visualization and evaluation.

Then one of the users adjusted the trail manually, drawing a new route with a laser pointer. The user adjusted the route of the computationally optimized trail to pass through the new forested patches and cross the saddle between the two new small hills. Finally the user put on the HMD, began the walkthrough simulation along the trail, and immersively experienced the landscape (Figure 6c).

4. CONCLUSION

Our application is a blueprint for an emerging model of geospatial education, collaborative decision making, and participatory design that synthesizes geographic scale analytics and human-scale perception. The hardware and software is inexpensive, open-source, and can be easily replicated. In the future we intend to enhance the fidelity of the rendering with improved texture mapping and a library of landscape elements such as different tree species. We also intend to evaluate user interaction and experience by assessing cognitive and meta-cognitive processes in landscape modeling and design problems.

[8]https://grass.osgeo.org/grass72/manuals/r.walk.html

5. REFERENCES

[1] C. R. Amburn, N. L. Vey, M. W. Boyce, and M. J. R. Mize. The Augmented REality Sandtable (ARES). Technical report, US Army Research Laboratory, 2015.

[2] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2016.

[3] H. Mitasova, C. Thaxton, J. Hofierka, R. McLaughlin, A. Moore, and L. Mitas. Path sampling method for modeling overland water flow, sediment transport, and short term terrain evolution in Open Source GIS. *Developments in Water Science*, 55:1479–1490, 2004.

[4] M. Neteler, M. Bowman, M. Landa, and M. Metz. GRASS GIS: a multi-purpose Open Source GIS. *Environmental Modelling & Software*, 31:124–130, 2012.

[5] A. Petrasova, B. Harmon, V. Petras, and H. Mitasova. *Tangible Modeling with Open Source GIS*. Springer International Publishing, 2015.

[6] B. Piper, C. Ratti, and H. Ishii. Illuminating clay: a 3-D tangible interface for landscape analysis. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '02*, Minneapolis, 2002. ACM Press.

[7] C. Ratti, Y. Wang, H. Ishii, B. Piper, D. Frenchman, J. P. Wilson, A. S. Fotheringham, and G. J. Hunter. Tangible User Interfaces (TUIs): A Novel Paradigm for GIS. *Transactions in GIS*, 8(4):407–421, 2004.

E Conference talks

Brendan A. Harmon, Helena Mitasova, and Anna Petrasova. 2014. Tangible geospatial modeling for landscape architects. In *2014 Geodesign Summit*. Redlands, California: Esri. http://video.esri.com/watch/3170/tangible-geospatial-modeling-for-landscape-architects

Brendan Harmon, Anna Petrasova, and Vaclav Petras. 2016. Serious gaming with Tangible Landscape. In *Coffee & Viz*. Raleigh, NC. https://baharmon.github.io/tei-2016/

Brendan A. Harmon. 2016. Embodied spatial thinking in tangible computing. In *TEI '16: Tangible, Embedded, and Embodied Interaction 2016*. Eindhoven, Netherlands. http://ncsu-geoforall-lab.github.io/coffee-and-viz/hunt.html#/9

Brendan A. Harmon, Anna Petrasova, Vaclav Petras, Helena Mitasova, and Ross K. Meentemeyer. 2016. Creative spatial thinking with Tangible Landscape. In *American Association of Geographers Annual Meeting 2016*. San Francisco, CA. http://baharmon.github.io/aag-2016/

Brendan A. Harmon, Anna Petrasova, Vaclav Petras, Helena Mitasova, and Ross K. Meentemeyer. 2016. Tangible interaction for GIS. In *FOSS4G NA 2016*. Raleigh, NC. http://baharmon.github.io/foss4g-na-2016/

Brendan A. Harmon, Anna Petrasova, Vaclav Petras, Helena Mitasova, and Ross K. Meentemeyer. 2016. Tangible Landscape: cognitively grasping the flow of water. In *International Society of Photogrammetry and Remote Sensing 2016*. Prague. https://baharmon.github.io/tangible-water-flow-presentation/

Brendan A. Harmon, Anna Petrasova, Vaclav Petras, Helena Mitasova, and Ross K. Meentemeyer. 2016. Tangible geographies. In *Royal Geographical Society Annual International Conference 2016*. London. https://baharmon.github.io/rgs-2016/