TerraME Observer: An extensible real-time visualization pipeline for dynamic spatial models

Antônio José C. Rodrigues¹, Tiago G. S. Carneiro¹
Pedro R. Andrade²

¹ TerraLAB – Earth System Modeling and Simulation Laboratory, Computer Science Department, Federal University of Ouro Preto (UFOP)
² Earth System Science Center (CCST), National Institute for Space Research (INPE)

GeoInfo - Brazilian Symposium on Geoinformatic
Campos do Jordão, São Paulo, Brasil – Nov/2012
Applications of Scientific Visualizations

Medicine, Physics, Environmental Modeling, etc.
Petabytes Environmental Simulations

Dedicated high-performance hardware and remote graphical workstations
Visualization software architecture

Simulation

Observer 1
Observer 2
...
Observer N

Visualization

Process 1

Process 2
Modeling languages: Compiled or Interpreted

Model in high-level modeling language

Model in general purpose programming language (C++, Java, etc.)

Model in general purpose programming language COMPILER

Model application

model state GUI
simulation engine

general programming language COMPILER

modeling platform COMPILER

modeling platform INTERPRETER

model state

simulation engine

GUI
Visualization pipeline overview

Adapted from (Wood et al, 2005)
Justification and Objectives

- **Justification**
  - Support of model development
    - Verification and interpretation
    - Synthesizing and analyzing simulation outcomes

- **Objectives**
  - Build a high performance visualization pipeline
    - High-level language primitives for visualization (definition and updating)
    - Monitoring mechanism
    - Serialization protocol
    - Several graphical user interfaces (GUI)
Hypothesis

- Software design patterns Observer and BlackBoard (Gamma et al, 1995; Buschmann et al, 1996) and multithreading are good strategies to improve visualization response times.
Solution

To combine software design patterns
TerraME platform

- Have been built jointly by UFOP in TerraLab and INPE
- Written in C++ and provides a high-level modeling language (interpreted platform)
- Development in multiscale
  - Spatial, Behavioral and Temporal scales
- Multiple modeling paradigms
  - Multi-agents, discrete-event simulation, the general systems theory, and theory of cellular automata
- Support to geographical databases through TerraLib
## Related Works

- Comparing the most popular simulation platforms

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Paradigm</th>
<th>GUI definition</th>
<th>New GUI’s</th>
<th>GUI updating</th>
<th>Way of updating</th>
<th>Modeling Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swarm</td>
<td>Multi-agents</td>
<td>Primitives</td>
<td>-</td>
<td>Implicitly via scheduler</td>
<td>Asynchrony</td>
<td>General purpose programming language</td>
</tr>
<tr>
<td>Repast</td>
<td>Multi-agents</td>
<td>Primitives and Wizards</td>
<td>Inheritance via general purpose programming language</td>
<td>Implicitly via scheduler</td>
<td>Asynchrony</td>
<td>General purpose programming language or high-level programming language</td>
</tr>
<tr>
<td>NetLogo</td>
<td>Multi-agents</td>
<td>Primitives via GUI</td>
<td>-</td>
<td>Implicitly via scheduler</td>
<td>Asynchrony</td>
<td>High-level programming language</td>
</tr>
</tbody>
</table>
Architecture and Implementation

- **Functional requirements:**
  - Graphically present the dynamics of continuous, discrete and spatial state variables;
  - Provide visualizations to temporal, spatial and behavioral dimensions of an environmental model;
  - Graphically exhibit the co-evolution of continuous, discrete and spatial state variables so that patterns can be identified and understood.

- **Non-functional requirements:**
  - Present real-time changes in state variables with as little as possible impact on the simulation performance;
  - Enable the monitoring mechanism to be extensible so that new visualizations can be easily developed by the user;
  - Keep compatibility with models previously written without visualizations.
Monitoring mechanism outline

Visualization pipeline (Adapted from [Wood et al., 2005])

TerraME platform architecture (Adapted from [Carneiro, 2006])
Serialization protocol

- Message format is described using the Backus-Naur formalism

```
<subject> ::= <subject identifier> <subject type> <number of attributes> <number of internal subjects> [*<attribute>] [*<subject>]

<attribute> ::= <attribute name> <attribute type> <attribute value>
```
Class diagram of Observer and BlackBoard design patterns

```java
// Subject
+ attach() : void
+ detach() : void
+ notify() : void

// ConcreteSubject
- subjectState : int
+ getState() : void

// Observer
+ update() : void

// BlackBoard
- data : int
+ getState(subjectID : int) : void
+ setDirtyBit(subjectID : int) : void

// ConcreteObserver
- observerState : int
+ update() : void
+ setDirtyBit() : void
+ draw() : void

notify()
| blackBoard->setDirtyBit(subjectID)
for all o in observers
| o->setDirtyBit()

getState(subject)
| if (not data[subjectID is valid])
| data[subjectID] = subject->getState()
| return data[subjectID]

setDirtyBit()
| update()
| update()
| observerState = blackBoard->getState(subject)
```
Sequence diagram of Observer and BlackBoard design patterns

Sequence diagram:
- ConcreteSubject
- Subject
- BlackBoard
- Observer
- ConcreteObserver

1. notifyObserver()
2. getState(subjectID:int) : void

Methods:
1. setDirtyBit()
2. setDirtyBit()
2.1. setDirtyBit()
Several types of TerraME Observers
Monitoring mechanism programming interface

...

-- Definition
obsMap = Observer{
    subject = cellSpace, type = "map",
    attributes = {"soilWater"}, legends = {soilWaterLeg}
}

...

-- Updating
cellSpace:notify()

...
Performance analysis

End-user model

Kernel TerraME

TerraME Observer

Recovery

Decoder

Rendering

Draw
Performance analysis

- Workload of the performance analysis

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Subject</th>
<th>Attributes</th>
<th>Observer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cell</td>
<td>2</td>
<td>2 charts</td>
</tr>
<tr>
<td>2</td>
<td>Cell</td>
<td>12</td>
<td>12 chart</td>
</tr>
<tr>
<td>3</td>
<td>100 x 100 Cellular Space</td>
<td>3</td>
<td>2 maps</td>
</tr>
<tr>
<td>4</td>
<td>100 x 100 Cellular Space</td>
<td>13</td>
<td>12 maps</td>
</tr>
</tbody>
</table>

- Test environment
  - Single machine: Xeon 64 bits, 32GBytes RAM and Windows 7
  - Time: 100 simulation steps
Amount of bytes serialized

Exp 1 and 2 - Sum serialized bytes

<table>
<thead>
<tr>
<th></th>
<th>Test 1</th>
<th>Test 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>with BB</td>
<td>121.818</td>
<td>415.818</td>
</tr>
<tr>
<td>without BB</td>
<td>179.636</td>
<td>992.000</td>
</tr>
</tbody>
</table>
Amount of bytes serialized

Exp 3 and 4 - Sum serialized bytes

<table>
<thead>
<tr>
<th>Test</th>
<th>with BB</th>
<th>without BB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>1.205</td>
<td>1.785</td>
</tr>
<tr>
<td>Test 2</td>
<td>4.076</td>
<td>9.857</td>
</tr>
</tbody>
</table>
Response time for the type Cell

Exp 1 - Average response time

<table>
<thead>
<tr>
<th>Time (ms)</th>
<th>With BB</th>
<th>Without BB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.625</td>
<td>3.687</td>
<td>3.641</td>
</tr>
<tr>
<td>0.697</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.416</td>
<td></td>
<td>4.432</td>
</tr>
</tbody>
</table>

Graph showing average response time with and without BB.
Response time for the type Cell

Exp 2 - Average response time

<table>
<thead>
<tr>
<th></th>
<th>with BB</th>
<th>without BB</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (ms)</td>
<td>21.159</td>
<td>21.467</td>
</tr>
<tr>
<td></td>
<td>2.183</td>
<td>2.276</td>
</tr>
<tr>
<td></td>
<td>18.740</td>
<td>18.850</td>
</tr>
<tr>
<td>time (ms)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Legend:
- Decoder
- Rendering
- wait
- Handle-Draw
- Recovery
- Handle
- Total
Response time for the type Cellular Space

Exp 3 - Average response time

<table>
<thead>
<tr>
<th></th>
<th>with BB</th>
<th>without BB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (ms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>48.565</td>
<td>74.757</td>
</tr>
<tr>
<td>100.0</td>
<td>153.630</td>
<td>152.580</td>
</tr>
<tr>
<td>200.0</td>
<td>133.392</td>
<td>233.091</td>
</tr>
<tr>
<td>300.0</td>
<td>337.917</td>
<td>467.659</td>
</tr>
</tbody>
</table>

Decoders: ##Decoder## | ##Rendering## | Wait | Handle-Draw | Recovery | Handle | Total

Graph showing the average response time with and without BB.
Response time for the type Cellular Space

**Exp 4 - Average response time**

<table>
<thead>
<tr>
<th>Time (ms)</th>
<th>with BB</th>
<th>without BB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>138.996</td>
<td>408.358</td>
</tr>
<tr>
<td>500.0</td>
<td>849.503</td>
<td>833.238</td>
</tr>
<tr>
<td>1000.0</td>
<td>1283.341</td>
<td>1278.025</td>
</tr>
<tr>
<td>1500.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2500.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3000.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- Decoder
- Rendering
- Wait
- Handle-Draw
- Recovery
- Handle
- Total
Performance analysis

Exp 1 and 2 - Average memory consumption

<table>
<thead>
<tr>
<th></th>
<th>Exp 1</th>
<th>Exp 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>with BB</td>
<td>156.328 ± 0.0</td>
<td>160.701 ± 0.0</td>
</tr>
<tr>
<td>without BB</td>
<td>156.275 ± 0.0</td>
<td>160.604 ± 0.0</td>
</tr>
</tbody>
</table>
Performance analysis

Exp 3 and 4 - Average memory consumption

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Memory Consumption (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp 3 with BB</td>
<td>234.237</td>
</tr>
<tr>
<td>Exp 3 without BB</td>
<td>231.763</td>
</tr>
<tr>
<td>Exp 4 with BB</td>
<td>550.510</td>
</tr>
<tr>
<td>Exp 4 without BB</td>
<td>536.289</td>
</tr>
</tbody>
</table>
Final Remarks

- **Hypothesis**
  - Combining multithreading and blackboard improves visualization performance
    - 50% reduction in response time and 3% increase in memory consumption
  - All software requirements was achieved
  - Robustness and Flexibility
Final Remarks

- **Future works:**
  - Adding a synthesis stage into the visualization pipeline
  - To implement change control algorithms
  - Do several experiments
References


