A Dimension-Independent and Extensible Framework for Huge Geometric Models

David CANINO - canino@disi.unige.it

Abstract

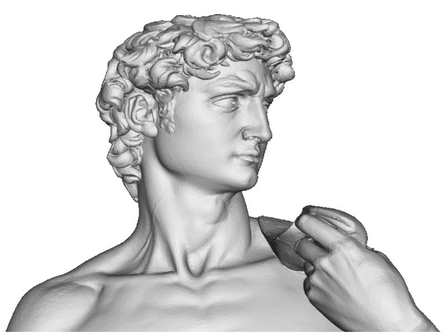
Nowadays, *huge* geometric models can be produced in many applications and their dimension can often *exceed* the RAM size in a common PC workstation. Thus, using an *external memory (EM)* technique is mandatory.

Here, we introduce a dimension-independent and extensible framework, called *Objects Management in Secondary Memory* (*OMSM)* framework, able to manage huge models.

The OMSM framework can be easily adapted to the users needs through *dynamic plugins,* integrating many techniques in a storing architecture.

Objectives

- Designing a framework for *rapidly designing* any *storing architectures* able to manage a *huge* set of independent *spatial objects*.
- This framework must *allow*:
  - to *manage* many *types* of spatial objects;
  - to *customize* each aspect of such architectures.
- We must be able to *adapt* this framework to any *user's needs* through *dynamic plugins*.
- We must be able to load a *plugin* without messing with the existing structures (even at run-time).
- We must be able to *operate* on the *spatial objects* in *different contexts*, as in [CMRS03].



Key idea

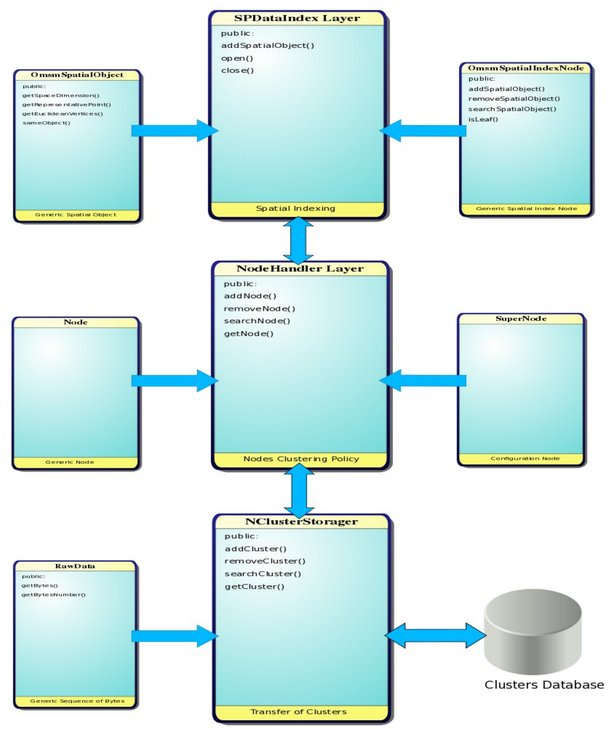
- The workflow of a *storing architecture* dedicated to spatial objects can be expressed through:
  a) the *spatial data indexing* [Sam06];
  b) the *subdivision* of spatial index nodes into a set of *clusters* [DSS96];
  c) the *dynamic transfer of clusters* between the RAM and a storage support.
- These aspects are *independent* of each other, i.e. we can group indexing nodes, by discarding which type of index we are using or we can write a cluster, by discarding its internal structure.
- A lot of *techniques* have been *developed* for each of the above aspects.
- Combining a *technique* for each aspect allows us to *satisfy* any *user's needs.*

Solution

- Abstracting a *component* with a *dynamic interface*, satisfying the *Interface* design pattern [GHJV95].
- Each *interface* controls all the *functionalities* and the *services* offered by a component.
- Each *component* communicates in the *same fashion,* regardless its *implementation*.
- This *design* allows to *replace* a component with an other one having the *same interface*, satifsying also the *Dynamic Loading* design pattern [SB98].
- Thus, a component is described by a *plugin* with an *independent data model*, discarding not interesting information (*dataview*).
- This *solution* is applied also to the *spatial objects,* by obtaining the same advantages.
- *Plugins* can be easily *shared* and *distributed* into the research community, by promoting *code reuse* and *development* about these topics.
- We must provide a *XML description* for a plugin.

Implementation

- We propose a *layered organization* with a layer for each of the above aspects (*OMSM Layer*)
- Each *layer* is described by a dynamic *interface* and an entity implements its *services.*
- An *entity* interacts only with the *beneath layer* and it offers *services* only to the *above layer*.
- We define an *interface* to describe all the objects to be made *persistent* through a custom *bytes sequence* (*RawData*).
- We *decouple* all the *persistence* aspects (*RawData*) from the *spatial* ones (*OmsmSpatialObject*) for all the geometric objects.
- A set of independent *spatial objects* becomes a *cloud of points*, by computing their *representative points* to be inserted and stored into a generic node of a spatial index (*OmsmSpatialIndexNode*).



Conclusions

- The OMSM framework can be used to perform any *operations* on a huge set of spatial objects *no matter* what modeling primitives are used.
- Our framework can be *extended* towards many directions through new *plugins*.
- A plugin for *simplicial complexes* [Ago05] is an important extension of the OMSM framework. We could obtain a generic EM *topological data structure,* decoupling all the *topological* aspects from the *storage* ones*,* as in [DFFMD08].
- Performances of storing architectures with many *repeated insertions* are not satisfactory, thus we must design a general *bulk construction*.
- A solution based on the *streamability* [IL05] can resolve this problem.

References

[Ago05] AGOSTON M.: Computer Graphics and Geometric Modeling, Springer, 2005

[CMRS03] CIGNONI P., MONTANI C., ROCCHINI C., SCOPIGNO R.: *External memory management and simplification of huge meshes*, In IEEE Trans. on Vis. and Comp. Graph., (2003)

[DFFMD08] DE FLORIANI L., FACINOLI M., MAGILLO P., DIMITRI D.: A hierarchical spatial index for triangulated surfaces, *Proc. of the GRAPP Conf.,* (2008)

[DSS96] DIWAN A., SESHADRI S., SUDHARSHANAN S.: Clustering techniques for minimizing the external path length, *Proc. of the VLDB Conf.*, (1996)

[GHJV95] GAMMA E., HELM R., JOHNSON, R., VLISSIDES J.: *Design patterns: the elements of reusable object-oriented software*, Addison-Wesley, 1995

[IL05] ISENBURG S., LINDSTROM. P.: Streaming meshes, In the *Proc. of IEEE Visualization*, 2005

[Sam06] SAMET H.: *Foundations of multidimensional and metric data structures*, Morgan Kaufmann, 2006

[SB98] SMARAGDAKIS Y, BATORY D.: Implementing reusable object-oriented components, *Proc. of the ICSR Conf.*, (1998)