DEVELOPMENT TOOLS OF COMPLEX TECHNOLOGICAL OBJECTS COMPUTER-AIDED SIMULATION ENVIRONMENTS

M. Yu. Orekhov [1,a], D. V. Kalinin [2,b], A. V. Matrosov [3,c]

[1] *Alexandrov Research Institute of Technology, Russia*

[2] *Alexandrov Research Institute of Technology, Russia*

[3] *St. Petersburg State University, Russia*

E-mail: [a] genazvale2005@yandex.ru, [b] kdv2112@mail.ru, [c] avmatrosov@mail.ru

This article examines instrumental support issues of complex technological objects simulation on the level of specialized container and string libraries – tools, aimed for elaboration of real-time visualization systems. Such system, destined for the purposes of computation's visualization and control, displays vector object-oriented 2D open-text formatted schemes with large number of simulated object's equipment pieces. Numerous modifications made throughout design of complex technological object suggest treating scheme graphical objects and their dynamic behaviour as entities with flexible structure – associative string-indexed containers. This approach simplifies visualization system development and maintenance under simulated sub-objects' types and dynamic behaviour patterns indeterminacy condition. The paper emphasizes the significance of rapid string comparison and container's item lookup by given string-key for application's performance. Present article also stresses the importance of string and container libraries reliability as a vital condition of visualization system sustainability.

Keywords: flexible structures, vector graphics, dynamic displaying, associative containers, index table, string class, parsing

1. Introduction

Design of complex technological objects, such as Nuclear Power Plant Unit, requires verification by virtue of simulation. This requirement [1], [2] states the problem of modeling and training complexes elaboration taking place *within* simulated object design process. Specific tools of solving this problem, including real-time visualization system, have to meet certain conditions:

- support of models (vector object-oriented 2D open-text formatted schemes), that represent the complexity of simulated object with large number of interacting equipment pieces;
- support of indefiniteness in types and attributes of simulated sub-objects, caused by numerous models modifications made throughout design.

As a way to fulfill the first condition visualization system must ensure using significant number of graphical objects' types, editing of saturated schemes and operating in large namespaces of simulated parameters. Accordance to the second condition implies support of user-defined graphical objects' types with volatile set of attributes and control parameters.

Considering mentioned conditions suggests handling vector graphical objects as flexible structures with mutable set of attributes and rules of dynamic behaviour. "Flexible structure" here is a synonym to associative container of diverse-typed items. Comparing to use of fixed-structured graphical objects this approach simplifies elaboration of visualization system, eases its maintenance, reduces time and pecuniary expense.

In order to perform simulated object's parameters monitoring and its systems control *interactively* visualization environment must comply with requirements of high sustainability and efficiency. Environment's development tools have to meet the same demands. Hence, speed of parsing and string-key search becomes critically significant for efficient use of flexible structures, serializeable as a sequence of pairs "name=value". Safety of using flexible structures is ensured by protection of memory releasing errors, errors of using iterators and dataset indexing. These circumstances determine the relevance of specialized string and container libraries design.

2. Approach advantages

Let us enumerate major aspects of visualization environment functionality, provided by application of reliable and efficient flexible structures:

- dynamic visualization of graphical objects with high renewal frequency;
- small time of parsing and generating graphical objects' text definitions, quick performing of schemes reading and writing inter alia;
- implementation of various scheme editing operations;
- library storing of created schemes, objects and groups of their attributes.

At the same time, usage of flexible structures simplifies visualization system elaboration and maintenance, allowing to:

- modify graphical object's structure by adding, deleting and changing user-defined attributes;
- minimize the impact of graphical platform or its current version shift by encapsulating the interaction of abstract flexible-structured container with its platform-dependent graphical fixed-structured representation within methods of graphical object's driver;
- refuse from graphical object's attributes cashing, reckoning with high speed of obtaining desired value by its string key from the container;
- abstract dynamic exchange mechanism, that projects modeling parameter value onto value of graphical attribute, from distinctions in graphical objects' and primitives' types;

- create efficient items of graphical user interface without considering peculiarities of graphical objects' types.

3. Specialized string and container libraries

Speed of graphical objects' text definitions parsing and generating together with duration of graphical attribute's value lookup by its name appear to be key-parameters of flexible structures efficiency. Elimination of memory freeing errors along with errors of container items indexing and using iterators on them guarantees secure employment of flexible structures.

3.1. Stating problem

This section shortly describes major drawbacks of STL and Qt string and container systems. These drawbacks hinder those systems from being used in implementation of flexible structures.

In STL and Qt string systems "String"-type serves as general argument and return value type for their functions [3]. Passing literals and fragments of strings as function's argument leads to allocating memory block for temporary "String"-object's shared buffer. Furthermore, buffer construction includes copying of characters. These costs of string-types conversion significantly decrease efficiency of string processing.

Sequential and associative containers of STL and Qt libraries do not own their items. If container stores pointers, its destructor deletes only container's nodes, not the data pointers point at. This approach permits memory leaks and multiple freeing of memory blocks. With possibility of direct deletion container's items, i.e. without using container's methods, index tables and indexed dataset can be easily misaligned. Moreover, inserting *one* item requires allocating *two* memory blocks of different size: one for created item, another – for container's node. This technique fragments heap and decreases efficiency as a result.

Duration of associative data structure's item lookup depends both on its theoretical efficiency and on efficiency of software (possibility of storing the found key) and hardware (expense of nodes visiting) cashing [4]. Thereby, choice of compactly allocated data structure for implementation of container's index table would lessen lookup time in comparison with the results of balanced trees and skip-lists, used in STL and Qt maps. We do not consider hash-maps due to impossibility of presenting keys sorted and complexity of creating reliable hash-function for composite key.

Data structures of either container systems support employment of STL-style iterators. Such iterator stores a pointer to data structure's node. Implicit deletion of data structure's item makes iterator usage unpredictable and leads to application crash.

3.2. Specialized string library

Application of program class "SubString" as general argument and return value of string functions as well as implementation of to-"SubString" methods conversion for every string type – C-string, string and its fragments – radically lower mentioned costs of string-types conversion. "SubString"-object stores string position pointer and number of characters [3]. That ensures dramatic diminution of requests to dynamic memory in parsing procedures: processing of substring extraction and passing it to comparison operator needs no heap using. "SubString" construction comprises nothing but characters array's *address* (not characters) copying and counting of characters number (if it was not passed as an argument). Shared buffer of "SString"-type object (abbreviation for "Swift String" – name of the specialized library string class) inherits "SubString" (Figure 1). Thus, conversion of "SString"-object to "SubString" consists of buffer's reference passing, which terms in single assembler instruction.

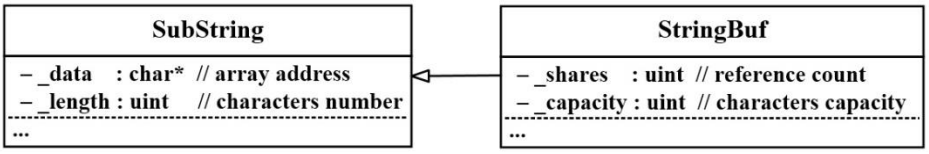

Figure 1. UML-class diagram of string shared buffer ancestry

Figure 2 illustrates the results of test, measuring string comparison duration. Test performed for low-level comparison standard library function (memcmp), specialized string library string-class (SString) and its analogues from STL and Qt libraries on gcc 4.9.2 compiler. StackAlloc-curve represents the result of STL-sring, equipped with LIFO-allocator.

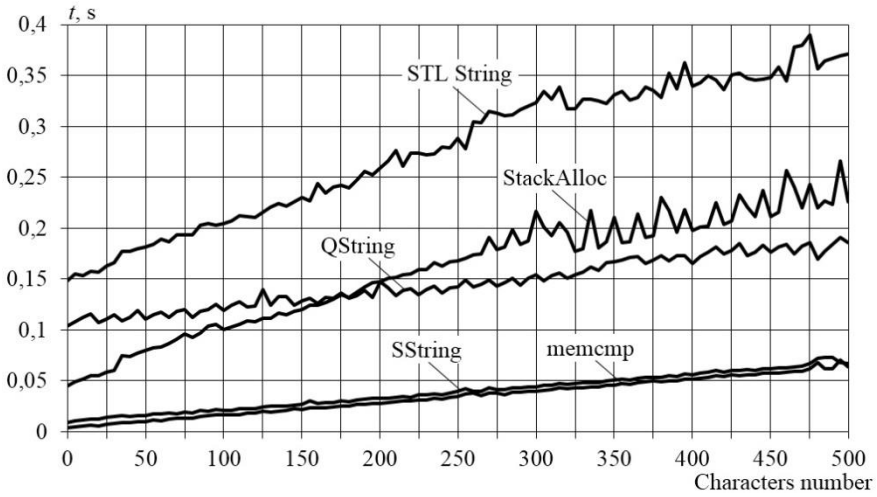

Figure 2. Measurement of string comparison duration

3.3. Specialized container library

Container class comprises bidirectional linked list, owning its items, and set of index tables. Establishing a container item ancestor – base class, encapsulating list node's functionality, permits to avoid usage of auxiliary list-node-type objects. Concept of items ownership secures against memory leaks and multiple memory freeing: list deletes its items in destructor; two containers cannot have common items. Passing items to container-receiver implies their removal from source-container [4].

Index tables form a circular list with bidirectional container list as a head item. This arrangement guarantees accordance between tables and list of data: list modifications incur sequential update of tables. Implementation of index table uses continuous array of sorted pointers to container's list. This data structure choice ensures high efficiency of map lookup through cashing of found item index along with localized allocation of array and list items within memory pages.

As a way to eliminate errors of using iterators container blocks its modifications for a constant iterator lifetime and automatically registers non-constant iterators in a circular list in order to correct their positions in case of container size would change.

Figure 3 represents the results of string-key lookup test, performed for developed container (AttrTab) and corresponding combinations of bidirectional lists with treelike data structures from STL and Qt libraries on gcc 4.4.0 compiler.

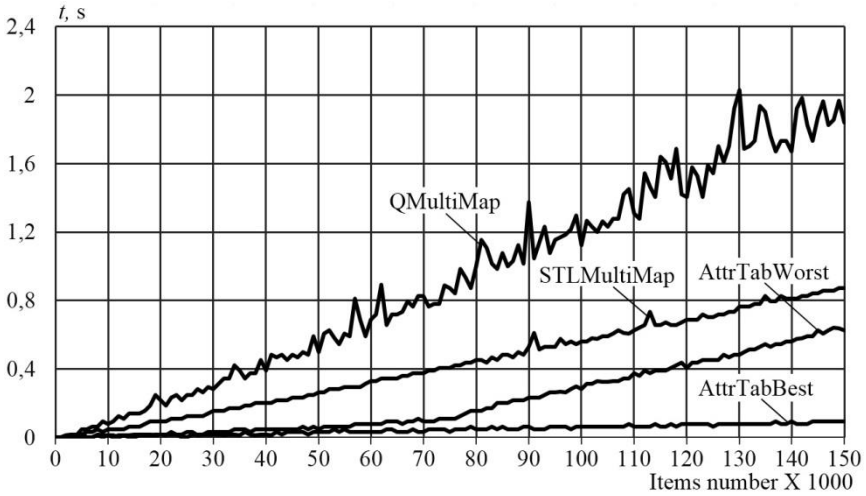

Figure 3. Measurement of container lookup duration

4. Applications

Specialized string and container libraries were designed within ”Virtual Power Unit of Nuclear Power Plant with Pressurized Water Reactor” (VPU) project under ”Supercomputers and Grid-Technologies Advancement” government program. One of its objectives – ”Visualization of Control Room Soft Panels” implied that soft panels (Figure 4) of projecting automated control system for technological process of LNPP-2 (Leningrad Nuclear Power Plant 2) power unit would operate under control of mathematical models within VPU simulation suite [1]. Libraries employment ensured elaboration of Multifunctional Graphical Editor (MGE – visualization environment) by a specified date. Editor’s rich functionality, high efficiency and sustainability guaranteed soft panels real-time dynamic displaying in SVG-quality with renewal frequency more than 50 FPS. MGE is an integral part of VPU that has a number of applications. Inter alia, VPU was used in development of a complete design package for Hanhikivi NPP-1 [5].

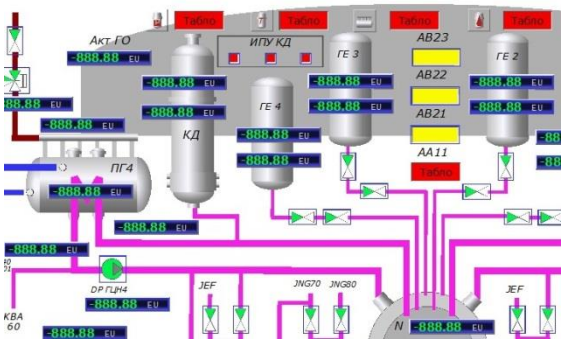

Figure 4. Fragment of imported LNPP-2 ACS soft panel

5. Conclusion

This article illustrates the issues of complex technological objects computer-aided simulation support in terms of visualization system (integral part of innovative design technology) development. It suggests application of flexible structures – string-indexed associative containers – for the purposes of simulated sub-objects graphical presentation. Paper outlines major peculiarities of specialized string and container libraries, vital for flexible structures efficiency and reliability. This approach allows to widen functionality, increase performance, reduce pecuniary and time expense of development.

References

[1] Bezlepkin V.V., Kukhtevich V.O., Obraztsov Ev.P. et al. ”Virtual Power Unit of Nuclear Power Plant with Pressurized Water Reactor” Instrumental Complex Employment for Verifying Design Solutions. Available at: http://www.gidropress.podolsk.ru/files/proceedings/mntk2013/documents/mntk2013-168.pdf. (accessed 21.09.2018)

[2] Semukhin M.V., Novizkiy V.V., Bezlepkin V.V., Obraztsov Ev.P. et al. Optimization Perspectives of Start-and-Adjustment Operations with Application of ”Virtual Power Unit of NPP” Simulation Suite. Available at: http://mntk.rosenergoatom.ru/mediafiles/u/files/2016/Book_TEZISY.pdf. (accessed 15.09.2018)

[3] Orekhov M.Yu., Substring Employment in C++ Quick-operating String System Implementation // Vestnik St. Petersburg University. Ser. 10., 2, 2015, pp. 134-149.

[4] Kalinin D.V., Orekhov M.Yu., Specialized Container Library for Purposes of Vector Graphics Dynamic Displaying // Vestnik St. Petersburg University. Ser. 10., 2, 2016, pp. 45-61.

[5] Obraztsov Ev.P. Innovative Technologies in Nuclear Modeling and Designing. http://globalforum.items-int.com/gf/gf-content/uploads/2017/03/GF-2015_Proceedings_01012016.pdf. (accessed 29.09.2018)