Program Visualization through Visual Metaphors

Bernhard Reitinger
Institute for
Computer Graphics and Vision
Technical University Graz
Inffeldgasse 16/II
A–8010 Graz, Austria
reitinger@icg.tu-graz.ac.at

Dieter Kranzlmüller
GUP (Dept. for
Graphics and Parallel Processing)
Joh. Kepler University Linz
Altenbergerstrasse 69
A–4040 Linz
dk@gup.uni-linz.ac.at

Andrej Ferko
Institute for
Computer Graphics and Vision
Technical University Graz
Inffeldgasse 16/II
A–8010 Graz, Austria
ferko@icg.tu-graz.ac.at

ABSTRACT

Program understanding is of major importance for software developers. This defines the demand for on-the-fly inspection and analysis tools to understand the behavior of software. This paper describes an approach based on Virtual Reality (VR) for visualization of program activity. The solution shows, that VR can substantially support the users' understanding by providing suitable graphical representations. The main difference of this approach compared to traditional program analysis tools is the usage of metaphors instead of common textual or 2D graphical displays.

Keywords

program visualization, virtual reality, program activity, behavioral analysis

1 INTRODUCTION

Visualization is a central point of todays computer science applications. As large amounts of data are not perceivable by humans, graphical representations are utilized to bring the processed data into a comprehensible form.

The word *visualization* itself is often misused [SDBP98]. Since *visual* and means "sight" (in Latin), many people believe that visualization is restricted to making pictures. However, a more appropriate definition related to the field of visualization uses the term "*mental image*", which is not necessarily related to something in one's visual field [Gal95].

WSCG POSTERS proceedings
WSCG'2003, February 3–7, 2003, Plzen, Czech Republic.

Definition 1: ***Visualization [SDBP98]***

> The power or process of forming a mental picture or vision of something not actually present to the sight.

Visualization, if properly used, quickly cuts the essentials [Mil93]. On contrary, improper usage brings colorful but disastrous results. Thus, evaluation of visualization tools must focus on the difference between *guiding* and *rationalizing*. To guide means, that the user will discover things, which were unknown before. To rationalize means, that visualization does not provide a new experience. It just lets you know things, which you are already aware of.

This paper focuses on ***program visualization***, where aspects of a program or it's run-time execution behavior are illustrated [Mye90]. The research has been conducted in the frame of the *MoSt* environment[1], a monitoring and steering tool for interactive investigation of large-scale long-running parallel programs. MOST supports the user during various stages of the software lifecycle such as performance tuning and error detection.

The paper is organized as follows: Section 2 de-

[1] MOST [KRV00] [RKV01] is an ongoing research project at the GUP, Joh. Kepler University Linz.

scribes the activity formula in theory. Corresponding visual metaphors including some results are given in Section 3, before a discussion about future work and a conclusion summarizes this paper.

2 PROGRAM ACTIVITY

One source of information exploited in the *MoSt* environment is the activity of a program, which is computed for time interval $[t_{min}, t_{max}]$ and delivers a floating-point value between 0 and 1 (indicating low or high activity, respectively). Each single memory cell can have two states, accessed (1) and not accessed (0). The activity $a(t)$ of a certain memory cell at a time t is 0, if the memory cell has not been accessed, or 1, if it has been accessed since the last observation. This activity is determined over a given discrete time interval $[t_{min}, t_{max}]$, so that memory locations with many accesses will yield higher activity values. Since an arbitrary area of a program consists of n such memory cells, its normalized level of activity can be computed as follows:

Definition 2: ***Accumulated activity***

Let $t_{min}, t_{max} \in \mathbf{N}$, $t_{min} < t_{max}$, and $a_i(t) \in \{0, 1\}$ for some $i \in \mathbf{N}$.

$$A_i(t_{min}, t_{max}) = \frac{\sum_{t_{min}}^{t_{max}} a_i(t)}{t_{max} - t_{min}}$$

is called *accumulated activity* of a single memory cell i within $[t_{min}, t_{max}]$.

Summing up the activity values of all single memory cells of one process leads to the following definition:

Definition 3: ***Process activity***

Let $t_{min}, t_{max}, n, i \in \mathbf{N}$.

$$PA(t_{min}, t_{max}) = \frac{\sum_{i=0}^{n-1} A_i(t_{min}, t_{max})}{n}$$

is called *process activity.*

The formula of Definition 2 is illustrated in Figure 1 with a single memory cell, whose activity is monitored over 10 time steps. In Figure 2 the corresponding function A is shown, where $t_{min} = 0$ and t_{max} takes values between 0 and 9.

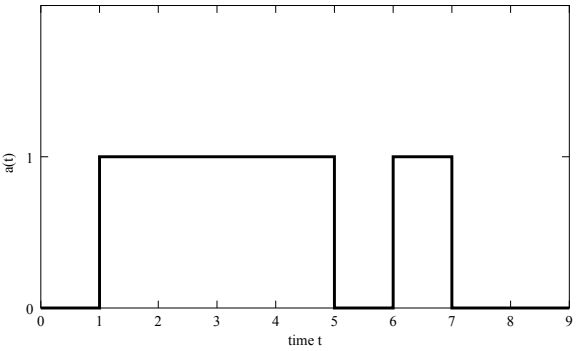

Figure 1: Activity of a memory cell

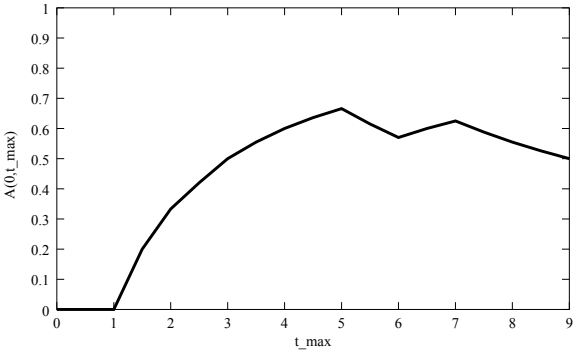

Figure 2: Accumulated activity

3 ACTIVITY VISUALIZATION

A very challenging tasks of *MoSt* was to find a suitable representation for displaying the activity information. After some investigations, we implemented the activity landscape and the immersive stripe tunnel.

The *activity landscape* approach describes the program's activity with attributes like color, object size, and animation. The accumulated activity of a memory cell is expressed by hills or flat regions intensified by some colors as shown in Figure 3. The higher (and more red) the hills are, the more activity is determined in this part. Flat (and blue) regions indicate no activity measured over a period of time.

Due to the fact, that the memory content is one-dimensional and the landscape has two dimensions, the content must be mapped in a certain kind of way. Each "row" of the landscape represents a block of the memory content. This means

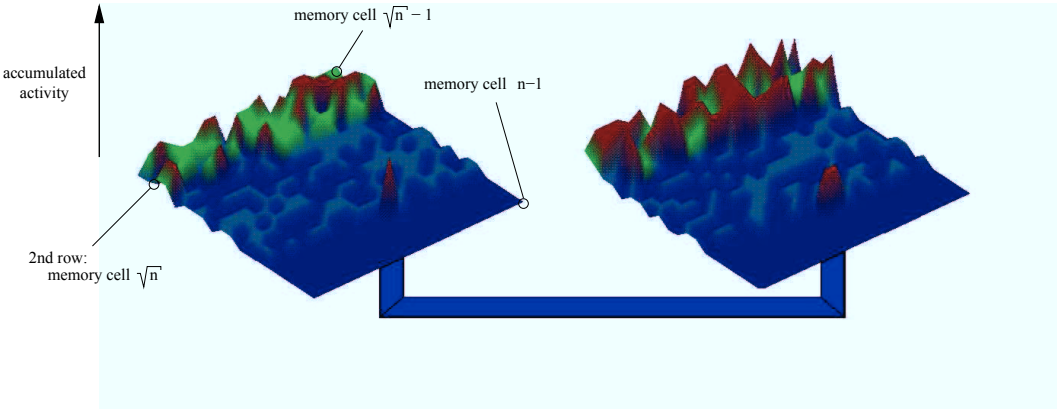

Figure 3: Landscape approach [KRV00]

that the conglomeration of all rows results in the whole memory content again. Consequently, it establishes relations with neighboring cells that actually do not exist in memory. Another drawback is the visualization of communication between processes, which is displayed with a communication tube. With more than two processes, the visualization of communication may become very complex.

The *immersive stripe tunnel* consists of n stripes each representing the memory contents of one process of a parallel program. Similar to the landscape, the color indicates the amount of activity in a particular region.

In the initial state an inactive stripe tunnel is displayed, where all stripes have the same blue color. As soon as memory activity information is delivered by the monitoring module, the tunnel changes its colors according to the activity map. An example is shown in Figure 4, where regions are colored according to their amount of activity.

The problem of communication visualization is solved by cylinders, which connect the sender's and receiver's respective memory block. A fictitious communication channel in the middle of the tunnel represents the physical media between the sender and receiver node. All communication is transferred over this main channel.

Considering a send or receive event, several parameters are used to specify the message: identifier of source/destination process, message type, message length, and message buffer. The first parameter specifies the identification of either the source or the destination process. The second one describes the type of the message and the third one indicates the message size of the transmitted message buffer in bytes. Besides these parameters, the memory location of the message buffer is also needed, to display the cylinder at the correct position in the tunnel.

Figure 4: Immersive stripe tunnel with active regions [RKV01]

Each send event is represented by a green cylinder, drawn from the memory location of the corresponding buffer straight to the communication channel. On the contrary, a receive event is displayed in yellow as a cylinder from the memory location of the message buffer to the corresponding sending event in the message channel.

Almost all of the parameters above can be found in the graphical representation. The identification is used to localize the destination (source) process, and the message length is expressed by an elliptic cylinder.

Figure 5 shows a parallel program with 8 processes, where process 0 broadcasts a message, which is displayed by a green cylinder. All other processes receive the message which is indicated

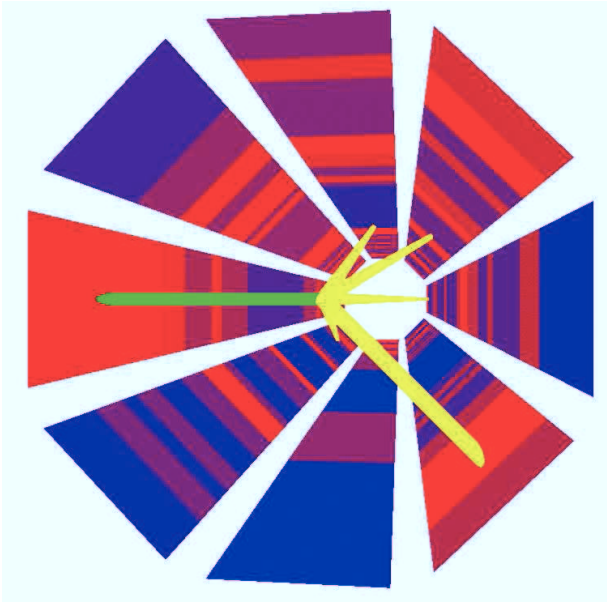

Figure 5: Immersive stripe tunnel including communication channels

by the yellow cylinders to the communication channel.

4 CONCLUSION AND FUTURE WORK

This paper discusses on-line program visualization in the *MoSt* environment. Analysis activities concentrate on the activity information, which supports scientists during identification of critical points in their applications. The VR program visualization of activity data may seem a little bit controversial at first. However, leading the user to an suitable starting place for in-depth investigations is very important, and there is only limited support from related work in this area.

An idea for future work in this project is sound mapping, which has originally been inspired by a story about the Whirlwind computer (1950) [Fre91]: *"You even had audio output in the sense that you could hear the program because there was an audio amplifier on one of the bits of one of registers - so each program had a signature. You could hear the tempo of how your program was running. You could sense when it was running well, or when it was doing something surprising."* (A similar idea is presented in [FJ93].) By integrating sound output, the activity in the region of the tunnel, that is currently inspected by the user, can further be emphasized. This includes sonification of hardware performance counters, the program counter, and similar state data.

5 ACKNOWLEDGMENTS

We are most thankful to Christian Glasner, Roland Hügl, and Prof. Dr. Jens Volkert who have made essential contributions to the MoSt project.

References

[FJ93] J.M. Francioni and J.A. Jackson. Breaking the silence: Auralization of parallel program behavior. *Academic Press, Inc.*, pages 181–194, 1993.

[Fre91] K.A. Frenkel. An interview with Fernando Jose Corbató. volume 34, pages 83–90, September 1991.

[Gal95] R.S. Gallagher. *Computer visualization: graphics techniques for scientific and engineering analysis.* CRC Press, Inc., USA, 1995.

[KRV00] D. Kranzlmüller, B. Reitinger, and J. Volkert. Experiencing a program´s execution in the CAVE. In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, volume 1, pages 259–264, November 2000.

[Mil93] B.P. Miller. What to draw? when to draw? an essay on parallel program visualization. *Journal of Parallel and Distributed Computing*, 18:265–269, 1993.

[Mye90] B.A. Myers. Taxonomies of visual programming and program visualisation. *Journal of Visual Languages and Computing*, 1, 1990.

[RKV01] B. Reitinger, D. Kranzlmüller, and J. Volkert. The MoSt Immersive Approach for Parallel and Distributed Program Analysis. In *Proc. of the 5th International Conference on Information Visualisation (IV 2001)*, July 2001.

[SDBP98] J. Stasko, J. Dominigue, M.H. Brown, and B.A. Price. *Software Visualization.* MIT Press, 1998. ISBN 0262193957.