

# Project Tetropolis - Application of Grid Computing to Interactive Virtual 3D Worlds

M. Fakler, S. Frenz, R. Goeckelmann, M. Schoettner, P. Schulthess  
Distributed System Laboratory, Computer Science  
Ulm University, James Frank Ring 1, Ulm, D-89069, Germany  
Phone: (0049) +731 5024140, E-mail: schulthess@informatik.uni-ulm.de

**Abstract - The Tetropolis project will provide an interactive virtual 3D-world for CS students and faculty at the University of Ulm. Users are represented by avatars and perceive a rich multi- and hypermedia experience. This will include animated lecture material, meeting opportunities, games, fun and customized personal objects & spaces.**

Our prototype is implemented on a novel peer-to-peer grid-computing platform. The common scene graph is kept in a shared memory space and the participating grid stations individually prepare & render their avatars perspective. In the paper relevant details of the underlying grid computing platform will be discussed together with realistic performance data.

Tetropolis is developed as a students project - challenging skills in rather diverse fields such as 3D-modelling & -design, graphics hardware, computer communications, distributed programming, artificial intelligence ... An up-to-date perspective of the prototype will be presented at the conference.

## I. INTERACTIVE VIRTUAL 3D-WORLDS

Advances in personal computer graphics hardware and the advent of affordable high-speed networking have given rise to a large market for massive multiplayer games. Examples are the *Half-life Series*, *World of Warcraft*, or *Dark Ages of Camelot*. Unfortunately the content of computer games is often violent and it is currently focussed towards adolescent population segments.

Interactive virtual 3D-worlds like *There.com*, *Active Worlds*, *Second Life*, or *Project Entropia* are typically populated by adults seeking a virtual get-away and a



Figure 1: Group meeting in There.com

meeting place. Figure 1 shows a screen shot from a get-together in *There.com*.

Acceptance for more peaceful virtual 3D-worlds has been lagging behind in comparison to the action-loaded massive multi-player games. The diverse social aspects of virtual environments were discussed in [1]. Our *Tetropolis* world is an attempt to combine get-together, fun, education and teaching material. One of our primary goals is to mediate sophisticated human interaction, to create a strong perception of being present and of “really” meeting others in the “metaverse” even though the virtual presence is in fact conveyed by avatars. This vision of a life in virtuality has been masterfully presented in [2].

The construction of our virtual world represents a major challenge for our students and the teachers. Skills from many different areas of computer science and beyond are required to promote the project. We will continue to elaborate on these issues below.

## II. CREATE LANDSCAPE, EVENTS, MISSIONS

Providing interesting content is of primary importance and computer technology takes second place. We are experimenting with different settings: an ocean with floating tetraeder buildings, a winter landscape with little huts, or a metropolitan urban setting.

A detail view of the ocean scenario is shown in figure 2. The landscapes are navigated with a virtual surf- or snowboard or a bicycle [3] respectively. It is important to convey a natural feeling of movement to the visitor and this is best done by a gliding device of some kind. Simulation of a walking avatar will not create a credible illusion of being present in the virtual scene.

Meetings for leisure and coursework will be scheduled and missions are planned to complement regular course-

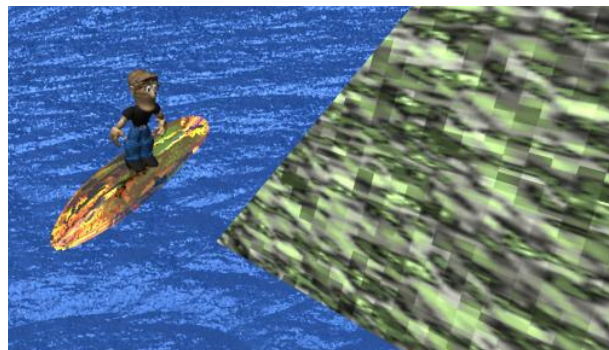


Figure 2: Surfer in the virtual ocean scenario

work. The “learnscape” will implement successive levels of achievement for the visiting students. The levels will represent the natural progression of the student through the topics of a course and through the curriculum.

### III. VIRTUAL 3D EXHIBITS FOR CS-TOPICS

The virtual buildings and sometimes also the open spaces will house special exhibits from current course material. Some topics are best conveyed through some interactive 3D-animation the student might play with. Typically we develop exhibits for those topics which are particularly difficult to convey. An example for an exhibit is a detailed animation of stack-based procedure invocation, or a visualisation of our proprietary Transactional Consistency model used in the grid-computing approach. A snapshot from a demonstration of the operation of the priority reservation scheme in an IEEE 802.5 token-ring is given in figure 3. Visitors will have the option to interact with the exhibit via push-buttons and other control gadgets – and possibly team up with colleagues or with their avatar representatives.

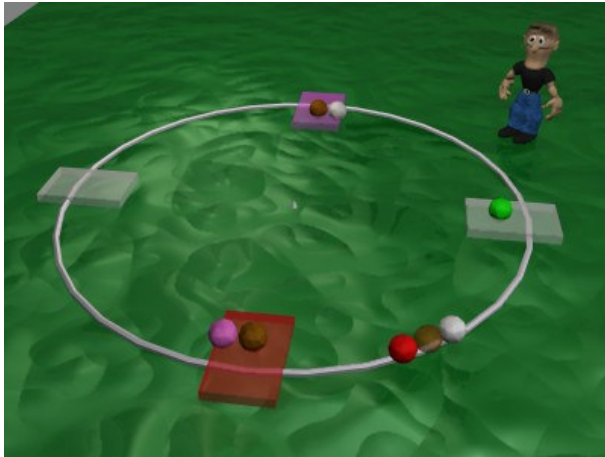


Figure 3: Animation of 802.5 priority reservation

### IV. CHARACTERS, OBJECTS & INVENTORY

*Tetropolis* will be populated by a large number of characters and their objects. Characters will be “real” avatars representing a human participant or they might be artificially intelligent non-player characters; participating in exhibits and events or performing service functions such as aiding the visitors to find his way, explaining the operation of exhibits, selling and buying of virtual objects and the like.

Each participant owns an easily distinguishable avatar and typically a multitude of virtual objects he/she is particularly fond of. The construction of virtual objects is currently done off-line using commercial 3D modelling programs such as *3D Studio Max*, *Maya*, *Gmax*, *TrueSpace* or similar. 3D-object construction is very time consuming

and requires special skills. To make an object unique, some type of digital signature might be added after they have been imported into *Tetropolis*.

Avatars will have a home of their own and an associated inventory of personal objects. The value of the objects created will depend on the effort required to build them, as well as on their artistic and functional merits. We expect that a sizeable market for virtual objects will eventually develop - together with a set of rules for socially acceptable behaviour (Non-player policemen might help enforcing those rules). Virtual economies are beginning to attract the attention of economic scientists [10].

### V. AI FOR BOTS, PHYSICS AND ANIMATION

Beyond the 3D representation of objects their physical and logical behaviour must be implemented. Softbots or non-player characters should interact sensibly with the avatars representing the visitors. We do not intend to implement speech synthesis and face animation but will be pleased to include it when it becomes available.

The physical characteristics of objects include gravity, inertia, and collision volumes. The collision volumes of objects are included in the shared world graph. The computation of progressively finer collision polyhedra is CPU intensive and might possibly be delegated to non-player machines in the grid.

Animation of objects is done by repeatedly invoking a control object attached to the virtual object. Each invocation might simply advance the object position, it might initiate complex reactions or it might take into account pending keyboard or mouse input for the object. The chairlift in the *snowscape* scenario is an example of an animated object (figure 4).



Figure 4: Chairlift and animation

### VI. MULTIMEDIA INTERCOMM FACILITY

We aim at providing a rich multi-media environment to the “inhabitants” of *Tetropolis*. In addition to real-time

animation and high-performance graphics sophisticated and finely tuned meeting scenarios between inhabitants should be possible. Initially we will provide chat and simple gestures from avatars. In due time voice communication will be possible between adjacent characters and it is planned to project a camera image on the face of avatars.

Voice communication will be based on the shared memory paradigm within the *Tetropolis* domain and it will be relayed to the public telephone system through a SIP gateway for IP-telephones. IP-telephony systems lend themselves easily to the implementation of graphically enhanced voice conferencing and to a large array of telephone user amenities as suggested for example in [4].

## VII. IMPORT / EXPORT OF VIRTUAL OBJECTS

*Tetropolis* objects and their behaviour are represented as a collection of Java objects and methods. Objects which have been modelled outside of *Tetropolis* are converted from their original format to an internal object-oriented world graph. Our filters import the widely accepted 3DS format and the OBJ graphics format. To keep the resulting world graph compatible with *OpenGL* some coordinate mappings are required and animated objects require special treatment. A study of respective formats has been prepared by [5] and additional information is available in the internet.

Export of virtual objects from *Tetropolis* is currently not supported. Exporting will be more important once we have integrated modelling facilities within *Tetropolis*. For export we plan to support serialized Java objects but will try to avoid additional XML-wrappers. The real challenge is to preserve the specific behaviour of an object after it has been exported to a different environment.

## VIII. RENDERING ENGINE AND WORLD GRAPH

The general idea of using distributed shared memory to store the virtual world of *Tetropolis* is outlined in Figure 5. All “worldly” objects are organized in a common world graph. Movements of avatars and animation of objects are directly reflected in shared memory. Unlike other game engines we bypass the sending and interpretation of control messages and object identifiers: the movement of an object is not effected by sending it a message “Object 4711: step forward 2 units” but simply by incrementing its x,y-position by the specified amount.

Whereas the world graph is global and shared by all participating stations the individual stations will assume their own camera position – typically representing the view of the associated avatar. All information specific to the perspective of an individual station is logically separated from the central graph and independently rendered by the local graphics card.

In order not to create an excessive number of access conflicts the stations proceed in lockstep. For a short period of time all pending animations are performed and recorded in the scene graph. Possible access conflicts are resolved by the DSM consistency protocol, which is

explained below.

Following the animation phase all stations extract their local perspective from the world graph. Typically this will only touch a small portion of the available objects. The world graph includes so called containers and inside a container all outside objects are irrelevant. Likewise if you are out in the open space you may be looking onto some containers but objects inside containers will be hidden.

In a third step each station renders the current frame from its previously extracted view graph. The rendering phase takes approximately 90 % of the total CPU time but no access conflicts will occur during the rendering phase. The detailed structure of the world graph and of the rendering process is described in [6].

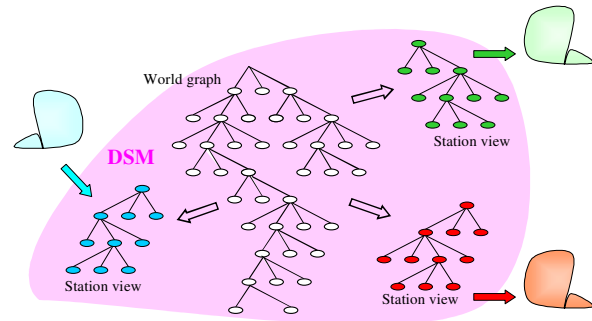


Fig. 5: World graph & station views in shared storage

## IX. OPENGL LIBRARY AND ATI GRAPHICS

Standard operating systems like *Linux*, *Microsoft Windows* and *MacOS* do not lend themselves easily to an efficient implementation of a distributed shared memory platform (see below). Our native grid-computing platform using distributed shared memory is preferred instead. As an unfortunate consequence the leading-edge 3D graphics drivers available under *Microsoft Windows* and partly under *Linux* are not usable and we were forced to write our own drivers for hardware accelerated 3D graphics.

Figure 6 describes the graphics architecture chosen. Between the rendering engine and the graphics driver we have interposed a partial implementation of *OpenGL* in order to reduce the learning effort for programmers with

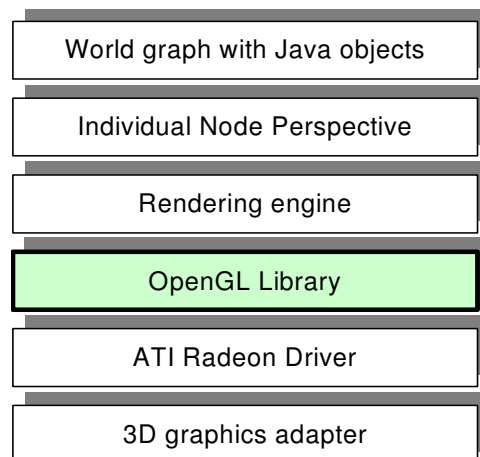


Fig. 6: Native 3D-graphics architecture

*OpenGL* experience. The *OpenGL* layer is a collection of *Java* objects which offer methods with an *OpenGL*-like signature. We do not compile *OpenGL* scripts into *Java* though but rather expect the programmers to invoke the methods of our *OpenGL* objects himself.

Due to the lack of adequate documentation the writing of graphics drivers has not been an easy task. Our current 3D graphics drivers will handle *ATI Radeon* cards using either the R100 GPU or the R200 GPU [7]. Cards using these GPUs are typically numbered *Radeon 7000* up to *Radeon 9200*. Unfortunately for newer *ATI* cards we have almost no information. The frame rates achieved for scenes of intermediate complexity are between 20 and 100 frames/sec - mainly depending on the complexity of lighting and textures and less on the number of vertices.

## X. MANAGING OF THE SHARED MEMORY GRID

The basic challenge in distributed systems building is to maintain a consistent state of all objects in a distributed application in spite of the network latencies. Traditional multiplayer scenarios maintain the system state in a central server or a server network and selectively propagate the changes to the clients. In such a scenario the central server is likely to present a bottleneck and network latency is doubled by the transit delay through the central server.

Our grid computing scenario dispenses of the central server and uses a peer-to-peer communication scheme. The distributed shared memory is used to store all objects and a special consistency protocol guarantees global consistency for all objects. This guarantee is made through the use of *Transactional Consistency*. All computations are encapsulated in short transactions which can easily be restarted in the event of an access collision. The results of a transaction are provisionally written to local memory but the original state of a page is preserved as a shadow copy. The shadow copies are restored in case of an abort and a subsequent automatic restart will occur. The basic Transactional Consistency scheme is shown in figure 7.

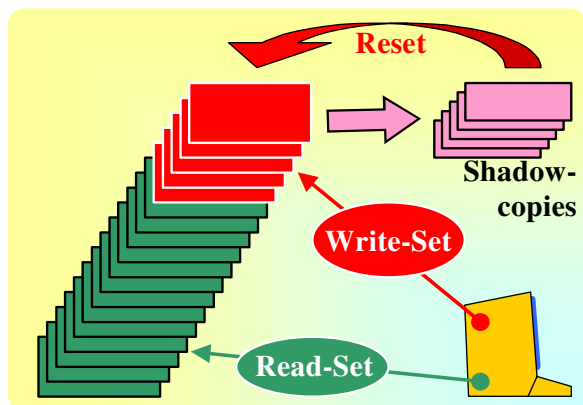


Figure 7: Transactional Consistency for one Grid-Station

Collisions between competing transactions are detected by multicasting to the grid the addresses of all pages written by a transaction. A concurrent transaction which has read one of the modified pages will realize that a conflict has

occurred and that it must restart itself. The set of all pages read or written is recorded by the local Memory Management Unit and requires little software action.

A transaction does not need to acquire locks for the objects to be written. Instead we use an optimistic synchronization scheme, where the transaction is allowed to proceed locally and validation of the transaction is postponed to the end of the transaction. This scheme avoids the network latencies which would appear when acquiring locks during normal execution.

As we have no sequential file system and keep all objects in grid storage it is easy to restart a transaction. Managing restartable computations in a traditional OS which includes files and a plethora of undocumented system state is very hard work, although it has been done in the *Kerrighed* project [8]. Unfortunately, the work has to be repeated for each new upcoming OS version.

Conceptually the shared grid storage is never switched off and objects persist until they are collected by the garbage collector. Serializing and deserializing objects between memory and disk is not required and this greatly simplifies the structure of operating system and programs. In order to shut down the system and as a safeguard against hardware failures and software errors a page server is provided. It stores a sequence of consistent images of the grid storage. A single station can join the grid or restart in less than 200 milliseconds. This period does not include the time required for BIOS processing as we refrain from writing our own BIOS for the individual motherboards.

## XI. OS KERNEL & DEVICES

The grid computing platform is implemented on top of a simple high-speed OS kernel. The kernel manages system startup, storage allocation, object naming, interrupts handlers and transaction scheduling. A clean separation between interrupt system state and transactional system state is achieved by so-called "Smart Buffers" which preserve input from devices in case of an aborted transaction and which delay the output from a transaction until it has actually committed (see figure 8).

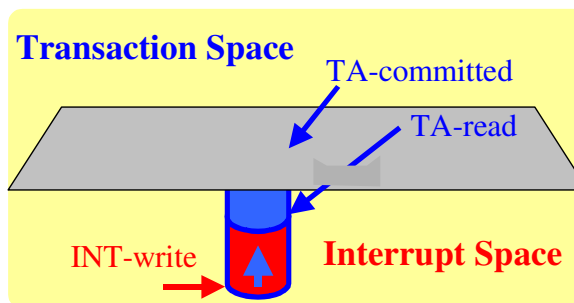


Figure 8: Smart buffers

The kernel is written in *Java* and compiled to native i386 code using our own proprietary *Java* compiler. As such the *Java* language offers no possibility to access device registers and raw memory resources. Our compiler therefore recognizes a special virtual class "MAGIC" which offers access to the hardware and which is only

available for system programmers.

Storage allocation is carefully crafted to minimize access collisions from memory allocation while still retaining the option to relocate objects at run-time and to perform incremental garbage collection in the heap.

Objects are typically accessible via references into shared grid storage but may optionally be registered and retrieved in the naming service. The naming service offers directory structures much like the naming structures in traditional file systems. Sequential file service, however, is not part of the kernel services.

Transactions are executed in a central operating system loop and run either to completion or are aborted in case of a conflict with another station. All transactions owned by a single station are collected in a station vector and are normally executed in a round-robin fashion. Optionally a station may insert a transaction into the transaction vector of another station and thus offload some of the work to the other station.

The codename of our operating system platform is Plurix and more information can be obtained from [9]. The Plurix platform is very stable and can run for days without losing pages and without losing the consistent state of the grid memory. Major drawbacks at the moment are the moderate quality of the code generation and the geographical restrictions on grid size and latency. Currently the shared storage grid is confined to a local area and extending it to a wide-area scenario would reduce the transaction rate from a theoretical 2000 transactions per second to approximately 100 transactions per second. Extending shared grid storage to a diameter of 3000 km and above is under study.

## XII. CONCLUSION

Implementing a virtual world scenario in a peer-to-peer scenario using a shared memory grid computing platform appears to be an interesting and easy alternative. In contrast to the traditional client server approach the programmer is relieved from observing global consistency of the data structures or from relaxing that consistency if necessary. Communication by means of shared storage is simpler than message passing because packing, unpacking and interpretation of messages & addresses is avoided.

A drawback of our approach is that leading-edge 3D graphics unavailable to us due to the lack of documentation for recent graphics chips. Nevertheless the achieved frame rate is acceptable and can exceed 100 frames per second in some situations. The fact that we require a homogeneous grid environment and restrict ourselves to regular PC hardware does not appear to be a disadvantage. Rather we believe that future compute environments will be grids of homogenous commodity PCs offering some kind of object-oriented shared memory platform.

In addition to exploring the potential of "life in cyberspace" the Tetropolis project serves as a test case for our local grid-computing platform. Figure 9 shows that the Plurix platform is also useful for parallel computation in general and that it scales well up to and beyond our current maximum cluster sizes of 12 machines.

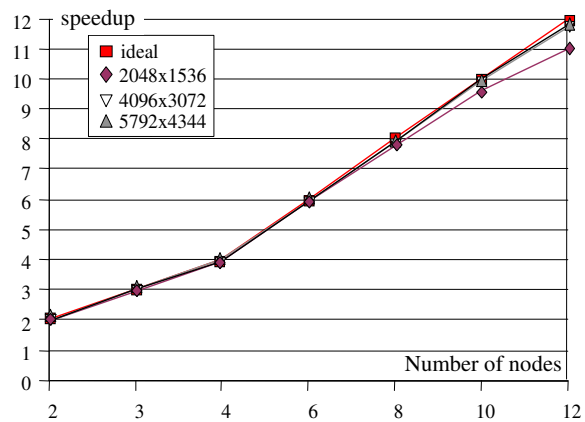


Figure 9. Raytracing using a Plurix Grid of 12 CPUs

## REFERENCES

- [1] R. Schroeder, *The social life of avatars*, 237 pages, Springer, Berlin, January 2002.
- [2] N. Stephenson, *Snowcrash*, 450 pages, Bantam Doubleday Dell publishers, June 1992.
- [3] Meißner, M., J. Orman and S.J. Braun: *Case Study on Real-Time Visualization of Virtual ...*. Proc.IEEE Visualization conference (2001).
- [4] Schulthess P., Froitzheim K., Sweeney St.: *Abstract Personal Communications Manager (APCM)*. ACM Computer Science Conference, Kansas City, 1992.
- [5] Keizer Liesa: *Analyse von Modellierungsprogrammen und 3D-Formaten*. Bachelorarbeit in Informatik, Universität Ulm, WS 2004/05.
- [6] Altmayer, Alexander: *Aufbau und Visualisierung virtueller Welten mit Mehrbenutzerszenarien in einem gemeinsamen verteilten Speicher*. Diplomarbeit in Informatik, Universität Ulm, WS 2003/04.
- [7] Fakler, Markus: *Direkte Ansteuerung von Grafikfunktionen einer Hochleistungsgrafikkarte am Beispiel der ATI Radeon*. Diplomarbeit Informatik, Universität Ulm, 2003.
- [8] Badrinath R., Morin Ch., Vallée G.: *Checkpointing and Recovery of Shared Memory Parallel Applications in a Cluster*. Proc. Intl. WS on Distributed Shared Memory on Clusters (DSM 2003), Tokyo, pages 471-477, May 2003. Held in conjunction with CCGrid 2003. IEEE TFCC.
- [9] [www.plurix.de](http://www.plurix.de)
- [10] Castranova, E., *Virtual Worlds: A First-Hand Account of Market and Society on the Cyberian Frontier*, The Gruter Institute Working Papers on Law, Economics, and Evolutionary Biology: Vol. 2: Article 1, 2001.