# Utilizando kd-trees para particionar o ambiente virtual e balancear dinamicamente a carga sobre servidores de MMOGs

Carlos Eduardo B. Bezerra, João L. D. Comba, Cláudio F. R. Geyer
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Av. Bento Gonçalves, 9500, Porto Alegre

## Abstract

MMOGs (massively multiplayer online games) are applications that require high bandwidth connections to work properly. This demand for bandwidth is specially critical on the servers that host the game. This happens because the typical number of simultaneous participants in this kind of game varies from a few hundreds to several tens of thousands, and the server is the one responsible for mediating the interaction between every pair of players connected to it. To deal with this problem, decentralized architectures with multiple servers have been proposed, where each server manages a region of the virtual environment of the game. Each player, then, connects only to the server that manages the region where he is playing. However, to distribute the load among the servers, it is necessary to devise an algorithm for partitioning the virtual environment. In order to readjust the load distribution during the game, this algorithm must be dynamic. Some work has already been made in this direction, but using a geometric algorithm, more appropriate than those found in the literature, it should be possible to reduce the distribution granularity without compromising the rebalancing time, or even reducing it. In this work, we propose the use of a kd-tree for dividing the virtual environment of the game in regions, each of which being designated to one of the servers. The split coordinates of the regions are adjusted dynamically according to the distribution of avatars in the virtual environment. We compared our algorithm to some approaches found in the literature and the simulation results show that our algorithm performed better in most aspects we analysed.

**Keywords:** MMOGs, load balancing, distributed server, kd-trees.

**Author's Contact:**

{carlos.bezera, comba, geyer}@inf.ufrgs.br

## 1 Introduction

The main characteristic of MMOGs is the large number of players interacting simultaneously, reaching the number of tens of thousands [Schiele et al. 2007]. When using a client-server architecture for the players to communicate with one another, the server intermediates the communication between each pair of players.

To allow the interaction of players, each one of them sends his commands to the server, which calculates the resulting game state and sends it to all the players to whom the state change is relevant. We can see that the number of state update messages sent by the server may grow proportionally to the square of the number of players, if all players are interacting with one another. Obviously, depending on the number of players, the cost of maintaining a centralized infrastructure like this is too high, restricting the MMOG market to large companies with enough resources to pay the upkeep of the server.

In order to reduce this cost, several decentralized solutions have been proposed. Some of them use peer-to-peer networks, such as [Schiele et al. 2007; Rieche et al. 2007; Hampel et al. 2006; El Rhalibi and Merabti 2005; Iimura et al. 2004; Knutsson et al. 2004]. Others propose the use of a distributed server composed of low-cost nodes connected through the Internet, as in [Ng et al. 2002; Chertov and Fahmy 2006; Lee and Lee 2003; Assiotis and Tzanov 2006]. Anyway, in all these approaches, the "world", or virtual environment of the game is divided into regions and for each region is assigned a server – or a group of peers to manage it, when usign peer-to-peer networks. Each of these regions must have a content such that the load imposed on the corresponding server is not greater than its capacity.

**Figure 1:** *Division into cells and grouping into regions*

When an *avatar* (representation of the player in the virtual environment) is located in a region, the player controlling that avatar connects to the server associated to that region. That server, then, is responsible for receiving the input from that player and for sending, in response, the update messages. When a server becomes overloaded due to an excessive number of avatars in its region and, therefore, more players to be updated, the division of the virtual environment must be recalculated in order to alleviate the overloaded server.

Usually, the virtual environment is divided into relatively small cells, which are then grouped into regions and distributed among the servers. However, this approach has a severe limitation in its granularity, since the cells have fixed size and position. Using a more appropriate geometric algorithm, it should be possible to achieve a better player distribution among different servers, making use of traditional techniques that are generally used for computer graphics.

In this work, we propose the utilization of a kd-tree to perform the partitioning of the virtual environment. When a server is overloaded, it triggers the load balancing, readjusting the limits of its region by changing the split coordinates stored in the kd-tree. A prototype has been developed and used in simulations. The results found in these simulations have been compared to previous results from approaches which use the cell division technique.

The text is organized as follows: in section 2, some related works are described; in section **??**, the algorithm proposed here is presented in detail; in the sections **??** and **??**, we present, respectively, the simulation details and its results and, in section **??**, the conclusions of this work are presented.

## 2 Related Work

Different authors have attacked the problem of partitioning the virtual environment in MMOGs for distribution among multiple servers [Ahmed and Shirmohammadi 2008; Bezerra and Geyer 2009]. Generally, there is a static division into cells of fixed size and position. The cells are then grouped into regions (Figure 1), and each region is delegated to one of the servers. When one of them is overwhelmed, it seeks other servers, which can absorb part of the load. This is done by distributing one or more cells of the overloaded server to other servers.

[Ahmed and Shirmohammadi 2008] propõem um modelo de balanceamento de carga orientado a células. Para balancear a carga, seu algoritmo encontra, primeiro, todos os agrupamentos de células que são gerenciadas pelo servidor sobrecarregado. Seleciona-se o agrupamento que contiver o menor número de células e, deste agrupamento, é escolhida a célula que tiver menor interação com outras células do mesmo servidor – considerando que a interação entre duas células A e B é definida como o número de pares de avatares interagindo um com outro, estando um em A e o outro em B. A célula escolhida é, então, transferida para o servidor menos carregado, sendo que a "carga" é definida como o uso de largura de banda para enviar atualizações de estado aos avatares posicionados

**Figure 2:** *Representação do ambiente em um grafo*

**Figure 3:** *Particionamento do espaço com uma árvore BSP*

em células gerenciadas por aquele servidor. Esse processo se repete até que o servidor não esteja mais sobrecarregado ou que não haja mais servidores capazes de absorver a carga excedente – neste caso, uma opção seria diminuir a freqüência de envio das atualizações de estado [Bezerra et al. 2008].

Em [Bezerra and Geyer 2009], também é proposta a divisão em células. Para realizar a divisão, o ambiente é representado por um grafo (Figura 2), onde cada vértice representa uma célula. Cada aresta no grafo liga dois vértices que representam células vizinhas. O peso de um vértice equivale ao uso de largura de banda do servidor para enviar atualizações de estado aos jogadores cujos avatares estão na célula representada por aquele vértice. A interação entre cada duas células definirá o peso da aresta que liga os vértices correspondentes. Para formar as regiões, o grafo é particionado, utilizando um algoritmo guloso: começando do vértice mais pesado, a cada passo adiciona-se o vértice ligado pelas aresta mais pesada a algum dos vértices já selecionados, até que o peso total da partição do grafo (soma dos pesos dos vértices) atinja um determinado limite relacionado à capacidade total do servidor que receberá a região representada por aquela partição do grafo.

Embora essa abordagem funcione, há uma séria limitação na granularidade da distribuição que pode ser feita. Se for desejada uma granularidade fina, é necessário definir as células como sendo muito pequenas, aumentando o número de vértices no grafo que representa o ambiente virtual e, conseqüentemente, o tempo necessário para executar o balanceamento. Sendo assim, pode ser melhor utilizar uma outra abordagem para o particionamento do ambiente virtual que utilize uma estrutura de dados mais adequada, tal como a kd-tree [Bentley 1975].

Esse tipo de estrutura de dados costuma ser usado na computação gráfica. No entanto, como em MMOGs também há informação geométrica – como a posição de cada avatar no ambiente –, árvores de particionamento do espaço podem ser utilizadas. Além disso, já existem técnicas de distribuição de objetos no espaço, buscando manter o balanceamento entre as diferentes regiões definidas pela árvore. Em [Luque et al. 2005], por exemplo, busca-se reduzir o tempo necessário para calcular as colisões entre pares de objetos se movendo no espaço. Para isso, é utilizada uma árvore BSP (*binary space partitioning*, ou particionamento binário do espaço) para distribuir os objetos da cena (Figura 3). Obviamente, se cada objeto de um par está completamente contido em uma partição diferente, eles não colidem e não é necessário fazer um teste mais demorado para esse par. Partindo de uma divisão inicial, é proposto pelos autores que a árvore se ajuste dinamicamente à medida que os objetos se deslocam, balanceando a distribuição dos mesmos na árvore, evitando que o tempo necessário para o cálculo das colisões se eleve muito. Algumas das idéias propostas pelos autores podem ser utilizadas para o contexto de balanceamento de carga entre servidores de um MMOG.

# 3 General Recommendations

Avoid to leave a section or subsection head isolated in the previous column or page.

Only the first paragraph of a section starts with no tab; any other paragraph starts with a tab of 0.5 cm. Every paragraph and head has a blank line before them. You should avoid writing a subsection immediately after a section head; trying to have at least a sentence explaining the section.

## 3.1 Citations and References

SBGames citation format is the familiar 'author year' format (quite similar to the one used by the SIGGRAPH conferences), also called


**Figure 4:** *Example of image*

Harvard notation. Although the Harvard notation uses parentheses, SBGames citation format uses brackets. Detailed information about the Harvard notation can be found elsewhere [**?**].

The Harvard notation can be summarized as follows. The year is separated from the author by a single space. If the article has two authors, their last names are used, separated by the word *and*. If the article has three or more authors, the primary authorss name, followed by et al. are used. Multiple citations are separated by semicolons: [**?**; **?**].

The reference list must be unnumbered, alphabetized by primary author last name, with the author list set in small caps, followed by the year of publication, followed by other information. The page number, if any, appears last in the reference. The second and successive lines of each entry are indented 0.5cm. Journal, book, thesis, and conference proceeding titles are set in italic type. See examples in the section References below. File *sbgames.bst* refereced below, perform automatic bibliography formatting.

## 3.2 Figures, Images, and Tables

You may have figures crossing the columns. However, large images should be placed at the very end of the paper or poster. You should avoid framing the figure with a visible line (unless the border is part of the figure). Figure 4 illustrates this situation.

In order to know if an image has sufficient resolution to be faithfully reproduced, you should multiply the size in cm by the factor 120. For example, an image of 5 cm x 7.5 cm in your document should have a resolution of no less than 600 pixels x 900 pixels. Another example: a screenshot of your entire 1024 x 768 display monitor should be no larger than 8.5 cm x 6.4 cm when positioned in your document.

Table titles should be centered above the tables.

As specified in CFP, papers must be in PDF format. So, the best way to generate PDF from .tex is using *pdflatex* (Unix), directly. Then, figures include into the text must be in PDF format. EPS format is employed, when latex, and dvips are adopted. But, using these programs, a ps-to-pdf conversion is required and can introduce quality loss. Please: use *pdflatex* to achieve high quality.

# 4 Generating the paper in PDF file format

Use *template.tex* and *template.tex* to write your paper and enumerate bibliographic references. So, run:

pdflatex template

bibtex template

pdflatex template

pdflatex template

That's it. You can rename template.tex and .bib. So, keep *sbgames.cls* and *sbgames.bst* without modifications. They are required to format text and bibliography according to SBGAMES format.

## 5 How to submit

Each manuscript must be submitted electronically using the JEMS submission site at `https://submissoes.sbc.org.br/sbgames2007` ONLY PDF file format is accepted. An additional 10 MB will be available for the (optional) ZIP file with the supplementary material.

Every co-author of a manuscript must also be registered as a user of JEMS before the manuscript is submitted. Instructions on how to register new JEMS users and how to retrieve forgotten JEMS passwords are available at the same URL above. PLEASE MAKE SURE THAT EVERY CO-AUTHOR IS INCLUDED AT THE TIME OF SUBMISSION. WE CANNOT LATER ON ADD AND/OR REMOVE AUTHORS FROM SUBMITTED PAPERS.

Upon logging on at `https://submissoes.sbc.org.br/sbgames2007` select the icon "submit paper" for the track "Computing - Full Papers", in order to submit a full paper. You will then be taken to a page with the title "Submit a paper to SBGAMES 2007 - Computing Track". Fill in the paper registration information requested in that page (don't forget to chose the keywords for your paper at the bottom of the page) and then click on the "submit" icon, in order to complete your paper's registration. After doing so, you will view a page called "Registering Paper". The first line after the title of this page should say "Paper <5digits> created", where <5digits> is a five-digit number assigned by JEMS to your manuscript. Before you upload your manuscript, include this five-digit number in the place where you would normally put the author names (for instance, by including the command *author*{Manuscript number <5digits>} in your LaTeX source file).

After you have generated the final PDF file containing the manuscript number (which should be renamed as <5digits>.pdf), you can upload it immediately by following the "upload" link available on the page "Registering Paper", or log out of the JEMS site and return later to upload your paper. If you choose the latter option, an "Upload" icon for each registered manuscript will be accessible from your SBGAMES 2007 home page within JEMS. In any case, please upload the PDF file with your manuscript first and then, if desired, return to the manuscript upload page (either using your browser's "Back" button or through your SBGAMES 2007 JEMS home) and upload the optional ZIP file with the supplementary material. You should receive an e-mail confirmation every time you perform an upload.

## 6 Conclusion

The final sections of your work are: acknowledgements and references. These final sections are not numbered.

## Acknowledgements

To Robert, for all the bagels.

## References

AHMED, D., AND SHIRMOHAMMADI, S. 2008. A Microcell Oriented Load Balancing Model for Collaborative Virtual Environments. In *Proceedings of the IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, VECIMS*, Piscataway, NJ: IEEE, Istanbul, Turkey, 86–91.

ASSIOTIS, M., AND TZANOV, V. 2006. A distributed architecture for MMORPG. In *Proceedings of the ACM SIGCOMM workshop on Network and system support for games, NetGames, 5.*, New York: ACM, Singapore, 4.

BENTLEY, J. 1975. Multidimensional binary search trees used for associative searching.

BEZERRA, C. E. B., AND GEYER, C. F. R. 2009. A load balancing scheme for massively multiplayer online games. *Massively Multiuser Online Gaming Systems and Applications, Special Issue of Springer's Journal of Multimedia Tools and Applications*.

BEZERRA, C. E. B., CECIN, F. R., AND GEYER, C. F. R. 2008. A3: a novel interest management algorithm for distributed simulations of mmogs. In *Proceedings of the IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications, DS-RT, 12.*, Washington, DC: IEEE, Vancouver, Canada, 35–42.

CHERTOV, R., AND FAHMY, S. 2006. Optimistic Load Balancing in a Distributed Virtual Environment. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV, 16.*, New York: ACM, Newport, USA, 1–6.

EL RHALIBI, A., AND MERABTI, M. 2005. Agents-based modeling for a peer-to-peer MMOG architecture. *Computers in Entertainment (CIE) 3*, 2, 3–3.

HAMPEL, T., BOPP, T., AND HINN, R. 2006. A peer-to-peer architecture for massive multiplayer online games. In *Proceedings of the ACM SIGCOMM workshop on Network and system support for games, NetGames, 5.*, New York: ACM, Singapore, 48.

IIMURA, T., HAZEYAMA, H., AND KADOBAYASHI, Y. 2004. Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In *Proceedings of the ACM SIGCOMM workshop on Network and system support for games, NetGames, 3.*, New York: ACM, Portland, USA, 116–120.

KNUTSSON, B., ET AL. 2004. Peer-to-peer support for massively multiplayer games. In *Proceedings of the IEEE Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, 23.*, [S.l.]: IEEE, Hong Kong, 96–107.

LEE, K., AND LEE, D. 2003. A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution. In *Proceedings of the ACM symposium on Virtual reality software and technology*, New York: ACM, Osaka, Japan, 160–168.

LUQUE, R., COMBA, J., AND FREITAS, C. 2005. Broad-phase collision detection using semi-adjusting BSP-trees. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, ACM New York, NY, USA, 179–186.

NG, B., ET AL. 2002. A multi-server architecture for distributed virtual walkthrough. In *Proceedings of the ACM symposium on Virtual reality software and technology, VRST*, New York: ACM, Hong Kong, 163–170.

RIECHE, S., ET AL. 2007. Peer-to-Peer-based Infrastructure Support for Massively Multiplayer Online Games. In *Proceedings of the 4th IEEE Consumer Communications and Networking Conference, CCNC, 4.*, [S.l.]: IEEE, Las Vegas, NV, 763–767.

SCHIELE, G., ET AL. 2007. Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, CCGRID, 7.*, Washington, DC: IEEE, Rio de Janeiro, 773–782.