A Multi-Agent System Approach Applied to Light Raycasting

H. Andrade*, F. Ramos, Y. Kotsarenko

Tecnológico de Monterrey
Cuernavaca, Morelos, México
*humberto.andrade@gmail.com

ABSTRACT
Light and shadows caused by the interaction with objects are important features in computer graphics which are usually taken into account to achieve realistic images. In order to simulate them, some attempts have been carried out which are based on direct illumination classical approaches as shadow mapping and shadow volumes. However, classical approaches in their beginnings could not support semi-transparent objects, soft-shadows, light interactions inside objects and the possibility to update a scene based on previous information.

In this paper a novel shadow casting approach is proposed to solve the previously mentioned problem using an interactive cooperative multi agent system to provide a better understanding and easy customization of the rendered scenes; for instance, the scenes are represented with object agents that propagate rectilinear photon information through them causing several changes on photon properties such as wavelength, intensity, among others. This system uses a two-dimensional space represented by pixels.

Our multi-agent system (MAS) uses a blackboard architecture for storing and sharing data and the implicit invocation design pattern. The system was developed to calculate direct illumination in a two-dimensional space. In addition, the proposed system supports point light agents, opaque agents, semi-opaque agents and empty agents.

A comparison is presented between the classic approaches and the proposed one presented in this work in scenes composed of opaque and semi-opaque objects. The proposed approach, as opposed to the classical ones, allows the shadows to be casted by the light that passes through semi-opaque objects. The light is casted by one or many light agents producing hard and soft shadows.

Keywords: shadows, ray-casting, multi agent system, blackboard.

RESUMEN
Luz y sombras causadas por la interacción con los objetos son características importantes en gráficas computacionales para lograr realismo en las imágenes. Para simularlas, se han hecho intentos en enfoques de iluminación directa como lo son shadow mapping y volumes. Sin embargo, éstas no pueden soportar objetos semi-transparentes, sombras suaves, interacciones de la luz en el interior de los objetos y la posibilidad de actualizar una escena basada en información previa.

En éste trabajo se propone un novedoso enfoque de generación de sombras para resolver los problemas antes mencionados desarrollados bajo un enfoque de sistema multi-agente cooperativo para una mejor comprensión y una fácil personalización de escenas; como ejemplo, las escenas son representadas con agentes-objeto que propagan la información de los fotones de manera rectilínea entre ellos, causando varios cambios en las propiedades de los fotones como la frecuencia, intensidad, entre otros. Este sistema utiliza un espacio bidimensional representado por pixeles.

Nuestro sistema multi-agente (SMA) utiliza la arquitectura pizarrón (blackboard) para almacenar y compartir datos entre agentes y el patrón de diseño invocación implícita. El sistema fue desarrollado para calcular la iluminación local en un espacio discreto bidimensional. Adicionalmente, el sistema propuesto soporta agentes luz puntual, agente opaco, agente semi-opaco y agente vacío.

Se muestra una comparación entre los enfoques clásicos y la alternativa propuesta utilizando escenas con objetos opacos y semi-opacos. La alternativa propuesta en comparación con los enfoques clásicos permite sombras generadas por la luz al cruzar objetos semi-opacos. La luz es generada por uno o varios agentes luz puntual provocando sombras fuertes o suaves.

1. Introduction

Real environments consist of matter that can be opaque or semi-opaque allowing proportional amounts of light energy passing through them in depending on how opaque the objects are. In the computer graphics field, the classic direct illumination approaches handle opaque objects that are composed of superficial surfaces excluding semi-opaque objects letting a portion of the total incident light to cross several sub-surfaces of the object. Thus, the direct illumination effect is computed incorrectly. In addition, since the proposed approach in this work is based on simple concepts of light propagation it is easier to understand and develop than the classic direct illumination approaches which due to the specialized knowledge that required are harder to understand.

Modern rendering is all about shadows. Modeling light visibility is fundamental in the field of computer graphics. Shadow mapping [1] was introduced by Lance Williams in 1978 as a fast and efficient approach for computing shadows in image space based on the visibility depth of a Z-buffer [2]. The computational cost of shadow maps is to render the scene two times: 1) Render the scene from the viewpoint of the light source and stored it into a depth map (shadow map). 2) Render the scene from the viewpoint of the camera and for each pixel of the final render view point a mapped is performed into the shadow buffer transforming the camera coordinate system into the coordinate system of the light source. If the depth value is less or equal to that one stored in the shadow map then the point is lit otherwise it is in shadow. Figure 1 illustrates the original idea of shadow mapping, see Everitt [3] for details.

Although shadow maps are very popular, they are not perfect, and their limitations are unacceptable for some applications as it produces aliasing due to the texture-based nature of the approach. Some filtering has long been used to reduce the shadow map aliasing and to simulate the effect of area lights causing soft-shadows.

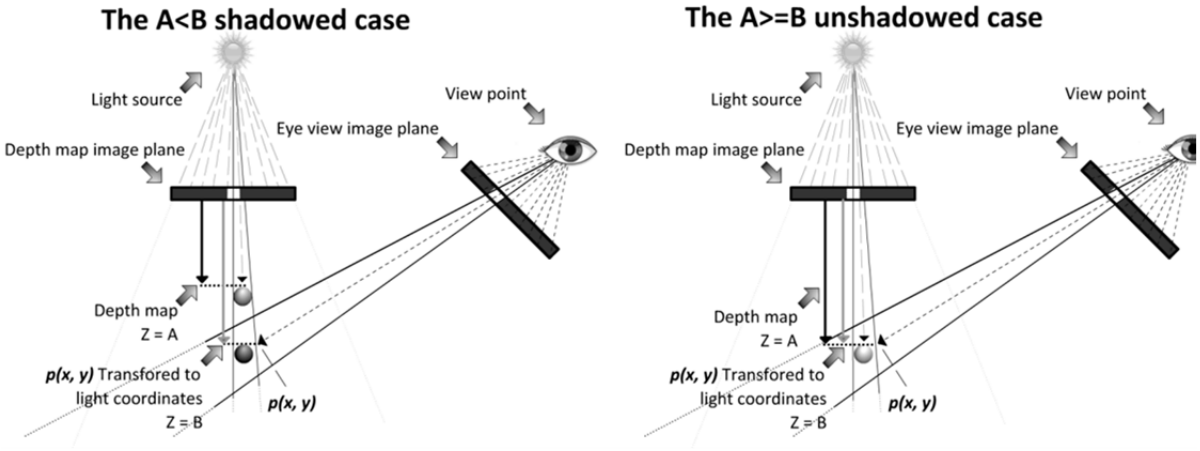

Figure 1. Determining whether a pixel in viewpoint or camera is lit or in shadow by shadow mapping depends on the comparison between the depth stored in the shadow map and the depth of the transformed point p(x, y) into light source coordinates.

The most widely used alternative to solve the aliasing problem is shadow volumes [4] introduced by Frankling Crow as off-line pre-processing approach based on the stencil buffer by adding invisible polygons that represent the boundary between an illuminated and a shadowed space with object precision. The invisible polygons are computed by finding all edges and extrude them to create geometry that divides the space in light or in shadow as shown in Figure 2.

Once the scene is rendered from the camera, each pixel depth being drawn is compared with the shadow geometry depth and counting operations to properly render the shadows. Shadow volumes, as shadow mapping, have been a topic of much research because the complexity of the shadow volumes is tied to the complexity of the shadowing geometry, thus robust and fast shadow volumes are required for real-time applications as the binary space partitioning (BSP trees [5]) latter used to develop video games such as Doom.

Classical mentioned approaches fail to simulate real scenes since they lack physical interaction of light as it propagates in the scene. This paper proposes a novel approach to compute hard and soft shadows based on a multi agent system (MAS) approach that is developed in a two-dimensional (2D) discrete space. This approach can be used to determine the light visibility and the intensity of light in the scene because of the direct illumination considering opaque and semi-opaque object agents and point light agents. The physical behavior of the light is modeled by light agents that initialize the energy traveling from light sources to object agents. The agents then propagate light information through the scene in order to capture light information which can be then used to apply shadows.

A comparison is presented between the mentioned classical shadow approaches and the proposed approach using scenes with only opaque objects and with both opaque and semi-opaque objects. Moreover, soft shadows cast by many light sources are presented illustrating the use of opaque and semi-opaque objects.

2. Direct illumination

This section discusses the direct illumination approaches most used in video games considering only the light that directly hits objects without reflections and refractions to generate the shading information required to render a discrete scene with shadows.

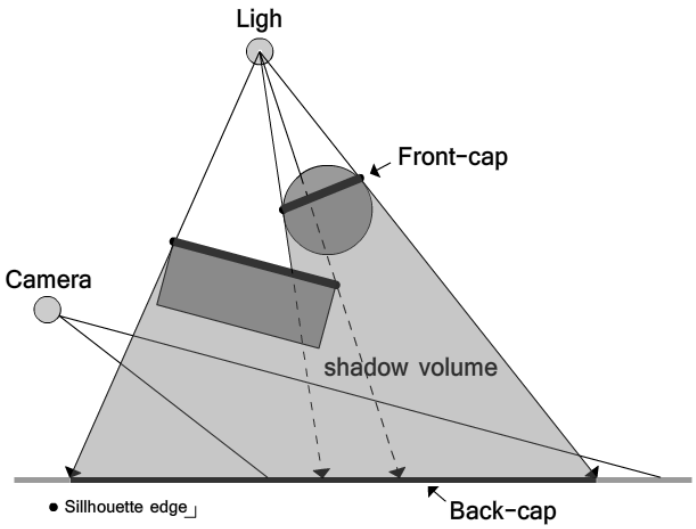

Figure 2. Light ray projection and shadow volume limited by a front-cap and back-cap to enclose the semi-infinite volume of space.

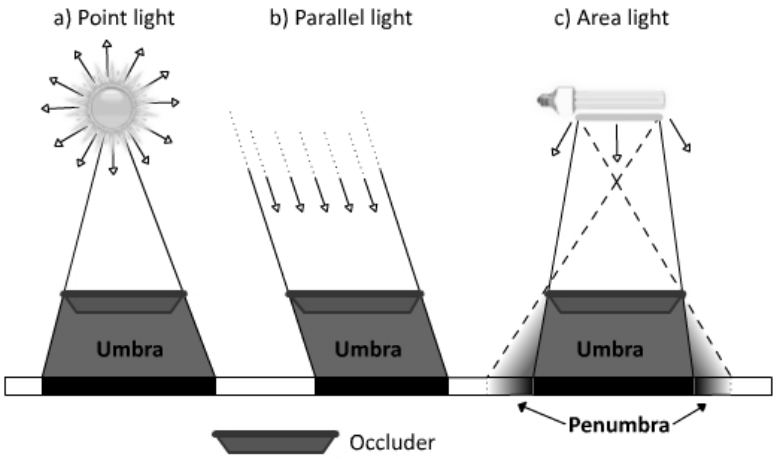

Figure 3. a) Hard shadow caused by point and spot light sources, b) hard shadow caused by parallel light source, c) soft shadow caused by line, area or volume light sources.

2.1 Hard shadows

Computing hard shadows involves only the determination of whether or not a point in the scene lies in shadow commented Woo in [6] (as a binary decision). This is the case of umbra section of the shadow as shown in Figure 3 (a) and (b). Perfect hard shadows do not exist physically but many shadow approaches use the simple supposition that any point of the scene is either lit or in shadow as in shadow volumes [4], shadow mapping [1] and fake shadows [7]. These approaches have been widely used in video games since the computational power required is considered low. In spite of their simple mathematical approach they are not easy to develop, they are limited to opaque objects and the last cannot cast self-shadows.

Williams shadow map suffers from anti-alias artifacts that occur when the shadow map resolution mismatch the final render resolution. The anti-alias is evident in the final render when many pixels are transformed into the same texel stored in the shadow map as shown in Figure 4. Another well-known problem of shadow maps is the bias problem that occurs when no enough depth precision can be stored in a shadow map texel (commonly the max depth value is 16-bits or 32-bits per texel). The most recent survey of shadow map modifications on hard shadows is found in [8].

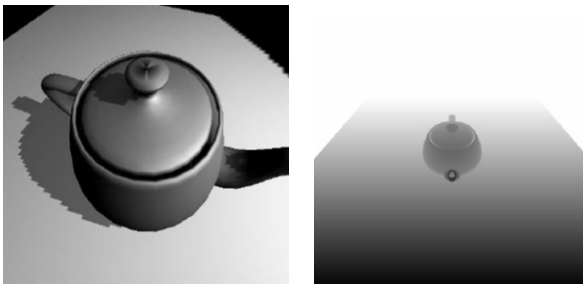

Figure 4. On the left, a final render is shown with a shadow anti-alias artifact due to the mismatch resolution. On the right, the corresponding shadow map is shown rendered from the viewpoint of the light source.

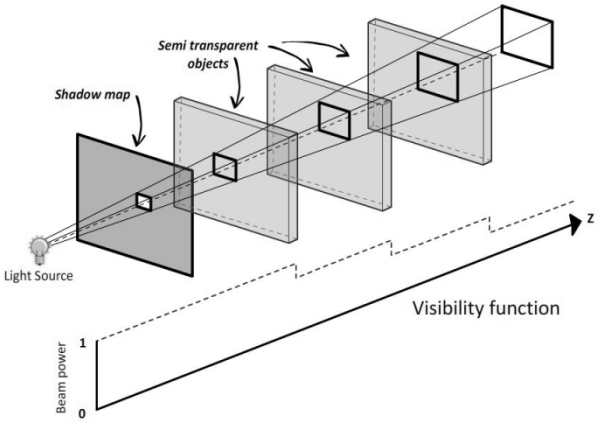

Figure 5. Visibility function of a deep shadow map.

Crow's shadow volumes were implemented with hardware acceleration by Funchs [10] but the widespread started with the Z-pass algorithm of Heidmann [11]. Semi-transparent shadows have been recently implemented with shadow volumes by Kim [12], where the stencil buffer is replaced with a floating-point buffer and the counting operations are replaced by adding or removing $log(1-\alpha)$ where α is the opacity of the object that generated the shadow volume. This implementation produces per-pixel accurate sharp shadows and can be extended to support colored shadow but requires the opacity to be constant per object. The limitations are that this approach cannot compute self-shadows and do not support textured shadows.
The proposed approach presented in this work allows the casting and self-casting of hard shadows over opaque and semi-opaque objects in a two-dimensional space by the use of a novel light ray-casting algorithm in order to prove the feasibility of the approach.

2.2 Soft shadows

Computing soft shadows involves the inclusion of the penumbra region along with the umbra for a higher level of visual quality. Full occlusion of the incoming light produces the umbra region, and partial occlusion of the incoming light produces the penumbra region. Backward ray tracing is one of the most popular and powerful approaches in the image synthesis that can be used to compute hard and soft shadows. In addition to the popular ray tracing approach, other classical approaches that produce soft shadows and commonly used in computer games use shadow volumes and shadow mapping with their corresponding modifications, as shown in [13,14,15,16,17,18,19] to mention some. A detail explanation and survey of those modifications focused on shadow mapping soft shadows is found in [20], a more general survey is found in [21] covering shadow mapping and a survey that covers shadow mapping and shadow volumes is found in [22].

These approaches generate better quality images than hard shadows but, as hard shadows, they are limited to opaque objects. Furthermore, the light interaction only occurs at object surfaces without taking into account the interactions of light before hitting the object and the light modification as it crosses the object layers. As in the case of hard shadows the proposed system allows soft shadows of semi-opaque objects along with opaque and transparent objects. The system handles the light interaction with the whole object and not only its surfaces, allowing objects of many layers to model the interior of them.

3. Multi agent system arquitecture

The traditional object-oriented paradigm states that objects are not autonomous and are not capable of deciding what to do in a specific situation such as that declining a request. In the present work, we used a cooperative multi-agent system (MAS) to facilitate the customization of light and object agents within the scene. It pretends also to simulate light propagation from light agents to object agents in all directions as is the case of point light sources.

According to Wooldrige [23], agents are capable of independent, autonomous action in order to meet their design objectives. In other words, they are capable of deciding what to do in any given situation by sensing their environment and performing actions in order to modify the environment. We pretend to assign propagation actions to object agents and initialization actions to light agents in order to find the appropriate shading of a scene as whether they were a group of experts working together trying to complete a large task. The form of cooperation is based on a result-sharing approach in which according to Davis et al., [24] agents assist each other by sharing particular results. In this case the result-sharing approach is based on the light propagation. The sharing of results drives the system to converge on shading information, answering the question of which areas are lit or in shadow.

The real opaque or semi-opaque objects allow or not passing some amounts of light energy through them. The proposed object agents use an absorption color to determine how much red, green and/or blue components absorb while light crosses the agent's position.

The process to get the proper shading information of the scene is composed of three steps: 1) scene initialization, 2) light propagation (ray-casting) and 3) shading computation.

In the first step, the object agent's properties are initialized according to their color absorption of light, as well as their position, color and intensity. In the light propagation step, light agents compute initial ray data containing the current color, direction of the light and light agent ID. Then this ray data is stored in the blackboard (BB) which acts as a central repository of ray data. In this approach, the object agents are treated as knowledge source agents (KS) and their activation depend on control policies stored in the blackboard. The propagation of ray data is done by object agents and it ends once all rays leave the scene.
In the shading computation step, the BB containing one or more ray data for each object agent's position is processed to leave only one ray data for each object agent's position. Doing this, the multiple contributions of KS and the ray data are summarized that remain in the BB, corresponding to the searched shading information.

For the development of this large multi-agent system, design properties as modularity, expandability, reusability, maintainability, robustness and performance of the system are to be addressed. To deal with these properties, the multi-agent system architecture is based on the idea of integration of two popular architectural patterns: the blackboard and the implicit invocation pattern allowing the independent but cooperative propagation of light in order to find the proper shading information of the given scene. Because the system is highly modular it also offers the advantage of conceptual clarity and simplicity of design.

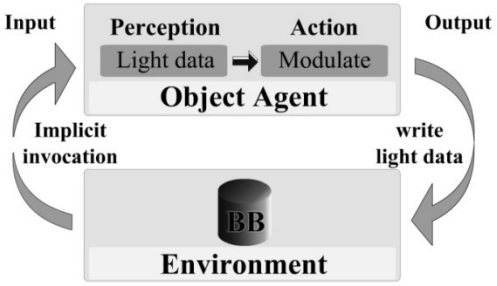

Figure 6. The object agent and the environment. This agent takes input light data from the environment located in the BB, modulates it and writes it back to the BB as the action output of the agent.

The object-agent architecture of the proposed system consists of a basic reactive architecture whose main process is the cycle of perception-action as shown in Figure . In the system, the perception or the input sensor is handled by implicit invocation explained later in this paper and the actions of the object agents are the proper writing of incoming light ray data on the BB preserving the light ray direction. The object agent has internal properties such as light color absorption, intensity, ray direction, Bresenham variables and Id's.

3.1 Ray-casting

Taking into account that photons follow a linear path from light sources to objects and then to the observer, the hard and soft shadows in a two-dimensional discrete space are computed by the use of a novel approach which consists in casting light rays on a scene composed of cells of the same size as shown by the right image in Figure 7, where all cells represent the whole scene and each cell represents the individual piece of the objects inside it. In the system, these pieces of transparent, semi-opaque and opaque objects are agents used as a description of the objects inside the scene (how much light color absorbs) to propagate light ray-casting data through the scene in order to find the shading information.

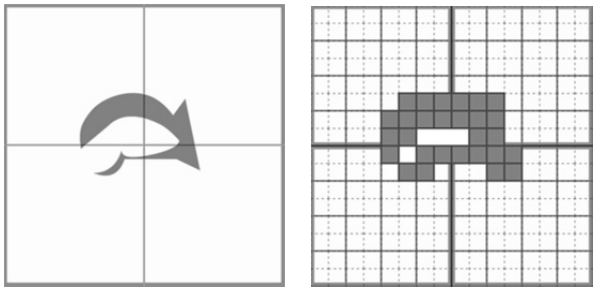

Figure 7. A scene representation consisting of geometric entities can be turned into a two-dimensional regular spaced grid. Each cell of the grid encodes specific information about the scene as the light absorption color.

The light ray-casting process is initialized by the light agents that are located inside the scene and passes light ray data to object agents from cell to cell following a linear path using the Bresenham algorithm et al., [25]. The proposed system uses

this algorithm because it computes lines by changing the question from “where to paint the line?” to “where is the next cell of the line to be painted?” This is particularly useful since only a few operations are needed to know the next direction of the light ray every time an object agent propagates light data.

As light ray data is propagated in the discrete space of the object agents, these agents modify ray data properties according to the object’s light absorption and store their results in a common repository to accumulate the shading information.

Figure 8 illustrates this ray-casting process with a 16x16 discrete scene that contains one light at the top left corner of the scene, one semi-opaque object (darker cells) and transparent objects (white cells). Dark grey lines correspond to those light rays that hit at least one semi-opaque object and the brighter ones correspond to light rays that only cross empty objects. The left and right image in Figure 8 shows the rays' directions and the resultant shading information of the scene by taking out the objects.

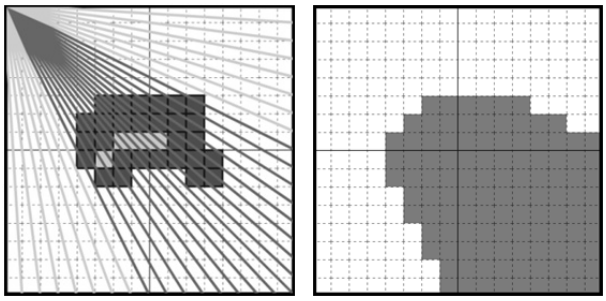

Figure 8. The light trajectories required for raycasting for a light source and the shading information produced after the light propagation.

3.2 Blackboard

There are three main components in this design pattern: 1) the knowledge sources, 2) the blackboard, and 3) the control component.

In the blackboard metaphor, the knowledge sources (KS´s) are independent specialists that can contribute to solve a part of the problem described in a blackboard (BB). In this case, the KS´s are the object agents that propagate the light ray data. According to Velthuijsen [26], the KS´s are capable of monitoring the blackboard (BB) anytime, deciding themselves when and how to contribute to the overall problem solving process. However, the system does not need constant monitoring by BB.

Blackboard is a centralized global data structure, used to allow indirect communication between knowledge source agents and acts as a shared memory visible to all of them.

The control proposed in [27] is a manager that considers each KS’s request to approach the blackboard in terms of what the KS can contribute in the proposed system. This task is assigned to the BB in order to handle and store policies in the same place. The control implemented in the system is a global interface between agents and the user to modify their properties and starting the ray-casting process.

According to Jing Dong [27], knowledge sources have two sub-components: conditions and actions. Conditions specify when the condition is met, it invokes appropriate action. Action includes the modification or placement of new facts on BB.

The proposed system uses blackboard as shared memory to store all photon information and after the photon propagation has been performed the information stored in the BB can be applied to the scene adding shadows.

3.3 Implicit invocation

Also known as Publish/Subscribe, this design pattern can be understood by paraphrasing Jing Dong [27]:

“In this pattern, each agent registers interest in particular event that may be announced by other agents. When the events are announced, all agents that are interested in the events are notified. They are implicitly invoked. In implicit invocation architecture, the announcers of events do not know which agents will be affected by those events. There are several advantages of the implicit invocation architectural pattern. First, the agents are more independent from each other than those with explicit invocation. Second, the interaction policy can be separated from interacting agents. Third, static name

dependencies are not troubled. On the other hand, there are some disadvantages of this pattern. First, there is no control over the sequences of implicit invocations. Second, a central management is needed to keep track of events, registration and dispatch policies. Third, event handling may interact badly with other run time mechanisms".
Instead of explicit invocations of procedures, a component advertises one or more events and other components register interest in an event involving a procedure with that event. The occurrence of an event causes the invocation of "implicit" procedures in other modules.

Other advantages of applying this design pattern to the MAS are simplicity in the conception of the problem, evolution by considering that knowledge source agents can be changed, and that the efficiency can be improved eliminating the need for polling by occurrence of events.

3.4 Mas design

The composition used by Jing Dong [27] uses blackboard to store knowledge data and control policies to activate agents. The MAS described in this work uses a similar architecture as the described in [27] as shown in Figure 9, where the implicit invocation is called directly from the BB instead of using the control agent to complete this task. The data in the BB may consist of knowledge data and control policies. The control policies can be accessed only by the BB, while knowledge data (light data), can be accessed by object agents. The solid lines with an arrow head represent the access of knowledge data and the dotted lines with an arrow head denote implicit invocation. In this architecture, knowledge source agents are invoked implicitly. Consequently, they are independent and can be changed dynamically.

In addition to light and object agents, the system uses one control agent that serves as a user interface to modify properties of the light agents and object agents as well as to setup the BB size and agent count as shown at the top of the image in Figure 10.

The BB used in our approach is divided into zones of the same size as a grid where each zone is a piece of the whole scene. Once each object agent is created, it is logged into the control policy located in the BB based on its position. The system uses the policy registry to know the allowed information of the object agents. When new information is added to the BB, it notifies the registered object agent in the control policy as it has been suggested to do with the control agent to avoid a polling architecture by [27] and shown at the bottom of the image in Figure 10.

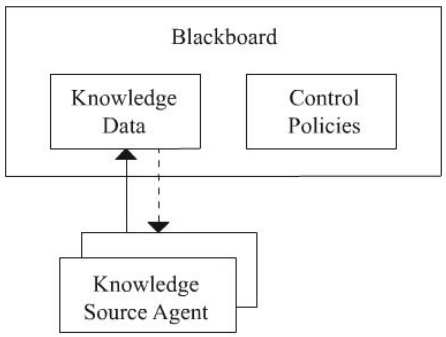

Figure 9. The architecture diagram of the event-based blackboard system with knowledge source agents, control policies and blackboard.

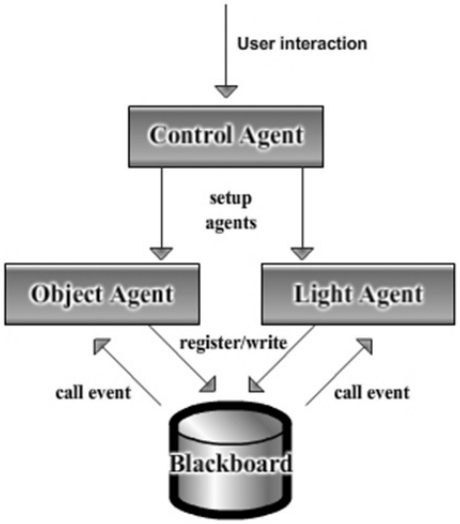

Figure 10. General structure of agents and the blackboard.

The light propagation process starts because of the user's interaction with the control agent that indicates when the ray-casting process is initialized.

As explained in Section 3.1, the ray-casting process starts with the initialization of all light agents computing all necessary light ray data to propagate light in the whole scene. Figure 11 shows three different scene sizes (4x4, 5x5 and 10x10), the dotted lines correspond to the imaginary ray data direction starting at the light source position and ending in the center of the last agent intersecting that direction. This direction is used to build a ray data which contains the ID, position and color of the light source agent as well as the position of the destination object agent and a current position of the ray for propagation purposes.

Once the light ray data is calculated, every light agent writes this data in the BB and is the BB itself who calls the previously registered object agents depending on the destination location of the data and the control policy related to that position.

The propagation continues from one object agent to another until the light ray data leaves the scene or until the intensity drops to zero. Once there is not active light rays in the system, the control agent takes the final version of the BB and creates an image for each light source to represent the final hard shading of the scene from a particular light source. In order to compute the soft shading of the same scene, it is required to get an average color from all hard shading images. It is important to mention that a white color indicates a lit area, gray color indicates a penumbra area, black color indicates the umbra and blue or any other color indicates the components present in the light.

Once the light propagation process has ended, the BB contains all the light shading information including the intensity, color and direction of the light at any point in the scene. In contrast to classical approaches, this information is important in animations and whenever the light source position remains static, since the changes of object's positions from frame to frame during an animation is relatively small and due to the fact that the BB can continue the propagation from any position, the previous BB shading information can be used to compute the new frame, thus reducing the BB access and CPU processing by just adding the new shading information in the BB. Since this feature increases the amount of memory of each frame, it is recommended to do a full computation of the BB shading information from scratch once the BB reaches a maximum memory size in order to control the memory usage.

3.5 Results

The classic shadow approaches such as shadow mapping and shadow volumes were developed to handle hard shadows. Figure 12 shows the hard shadow generated by opaque objects with shadow mapping at the left, shadow volume in the middle and our approach light ray-casting at the right. As it can be noticed, the three approaches compute direct illumination hard shadows. Shadow volumes have the best quality while shadow mapping quality depends on the resolution of the light map and the proposed approach quality depends on the amount of object agents that contains the scene.

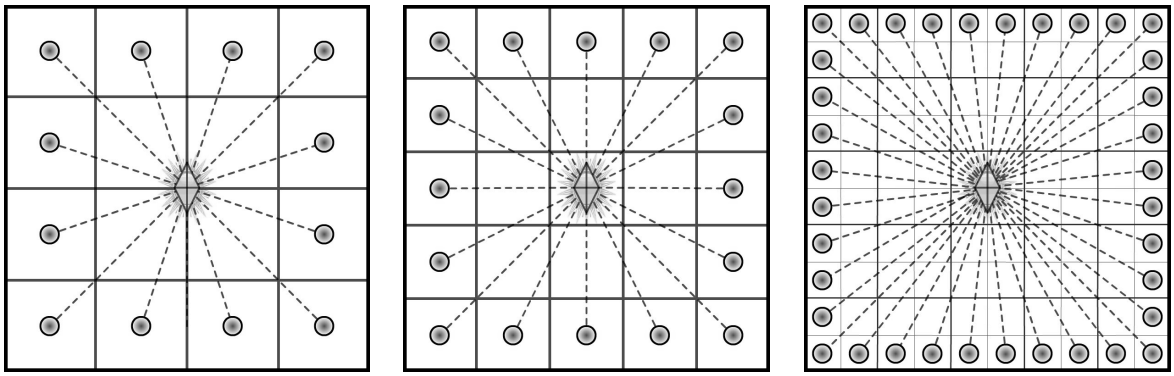

Figure 11. The necessary initial light rays depend on the size of the scene.

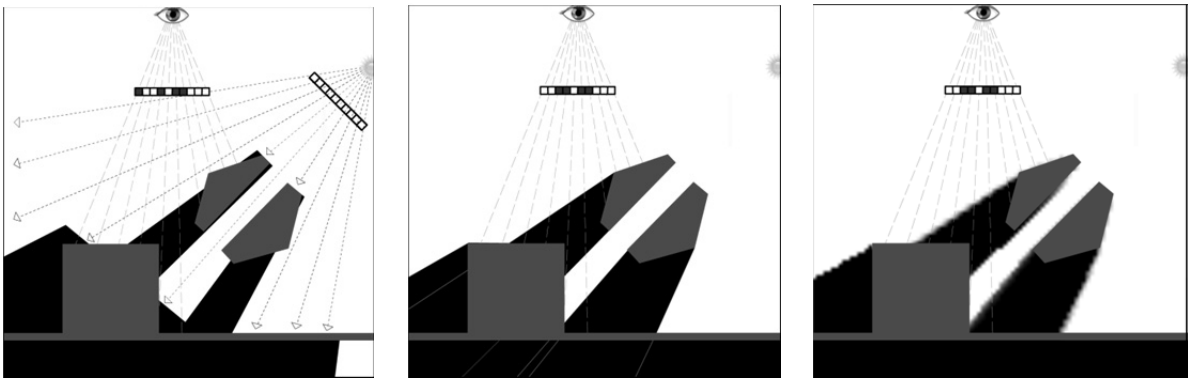

Figure 12. A comparison between shadow mapping, shadow volumes and light ray-casting with opaque objects.

When the scene contains semi-opaque objects, like a glass or a plastic, it is not surprising that classic algorithms fail to compute these shadows since they only support opaque objects. Figure 13 shows the same scene used in Figure 12, with the addition of a semi-opaque object. As we can see, the classic approaches handle semi-opaque objects as opaque objects avoiding areas where light is present but not completely.

In the case where it is needed to simulate the light effect produced when light crosses a semi-transparent object as is the case of the light source inside a colored bulb, classic algorithms will fail because the light energy does not cross the semi-opaque object since they only handle opaque objects. The first image in Figure 14 shows a transversal sample scene of two chairs, one glass table with a glass vase in the center and three colored bulbs in the ceiling of the scene. The left bulb in Figure 14 absorbs most of the green component of the light therefore when a light ray crosses it loses its green component. A similar effect occurs with the bulb to the right that absorbs most of the blue and red components allowing just the green component to cross and illuminating with green light the rest of the scene. In the same scene in Figure 14, the light sources in the same positions try to cast shadows by the use of classic algorithms, it is clear that light sources will only illuminate the interior of the bulb leaving the rest of the scene in complete umbra.

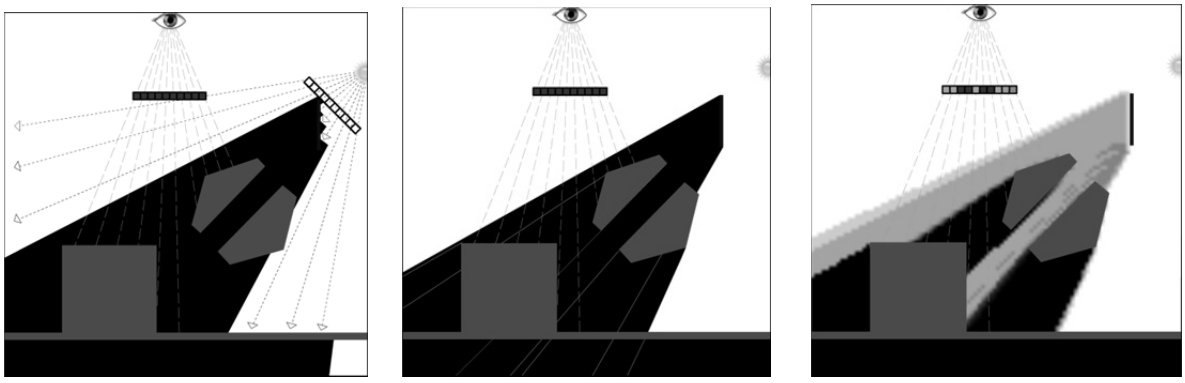

Figure 13. A comparison between shadow mapping, shadow volumes and light ray-casting with opaque (green) and semi-opaque (blue) objects.

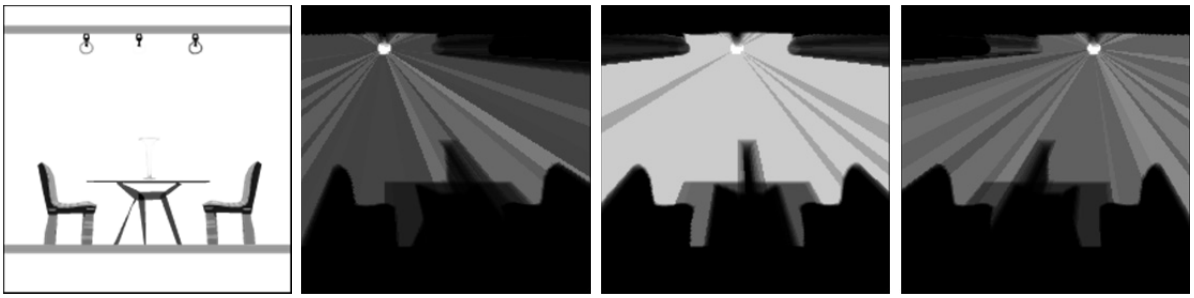

Figure 14. Basic scene with opaque and semi-opaque objects and three different light positions in the scene.

The proposed MAS can compute hard shadows of semi-opaque objects allowing the transmission of some amount of light. Moreover, this approach allows semi-opaque objects to be able to absorb different light frequency components allowing colored shadows as shown in Figure 15.

The current state of the MAS allows the light propagation through empty, opaque and semi-opaque object agents where empty object agents leave light unchanged, semi-opaque object agents absorb part of the incoming light and opaque object agents absorb all incoming light.

So far, it has been shown that the proposed approach not only allows the computation of opaque hard shadows but also contributes with semi-opaque hard shadows generating tinted shadows. Additionally, the proposed approach allows the straight forward soft shadow computation of opaque and semi-opaque objects by the use of many light hard shadow composition. With this composition, soft shadows can be generated by adding the penumbra region which is inexistent in hard shadows as introduced in Figure 3(c). This composition is created by the computation of the average hard shadow of all light sources inside the scene.

Figure 16 illustrates a scene containing opaque objects and seventeen point light sources forming a diagonal line. The first seventeen images are the light source's hard shadow computation of every light; note that only the last image shows the penumbra area since it is the average composition of all previous seventeen hard shadows.

Figure 17 shows a similar scene as Figure 14, without the ceiling and bulbs, but this time with 50 point light sources located at the top of the scene in order to cast soft shadows.

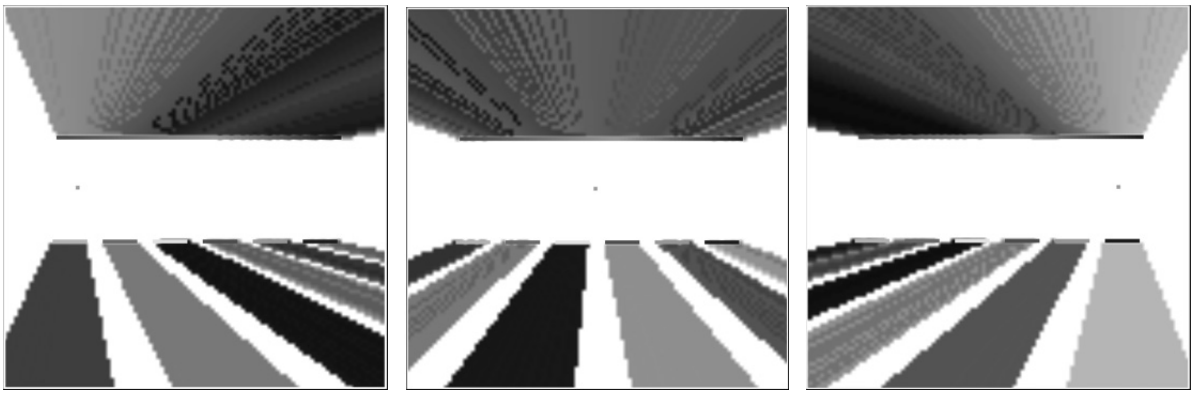

Figure 15. Shading information samples of the same scene containing semi-opaque objects. They show the colored hard shadow propagation generated with the proposed approach from three different light source positions.

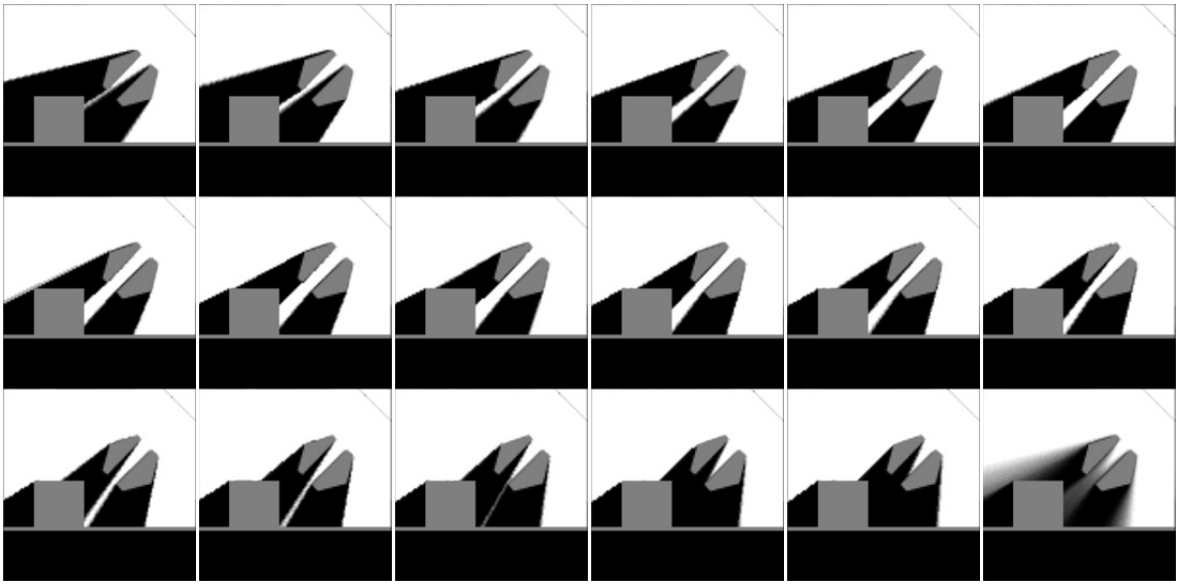

Figure 16. Seventeen opaque hard shadows with one soft shadow composition at the end.

Figure 17. Soft shadows on a basic scene with opaque and semi-opaque objects.

An interesting part of the proposed approach is the colored soft shadows casted by semi-opaque objects. These kinds of shadows are also caused by the composition of many hard shadows with object agents that do not absorb light components uniformly (red, green and blue). Once a ray light is created, it has the color of the parent light source as could be white. When this ray hits an opaque object its energy drops to zero immediately causing black shadows. Something similar occurs with semi-opaque objects that absorb just part of the three components RGB (Red, Green and Blue). The first three images of Figure 18 show a sample scene with semi-opaque objects represented that absorb different amounts of this RGB components.

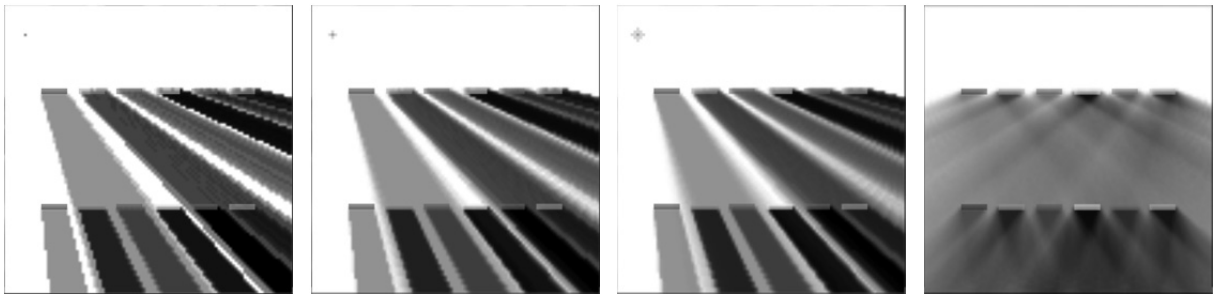

Figure 18. Colored soft shadows generated by using 1, 5, 13 and 100 semi-opaque objects. The first three images show different sizes of light sources as a circle and the last have a line of light sources at the top of the scene.

4. Conclusion

In the field of computer graphics the shadow volumes approach is used for better per-pixel precision while the shadow mapping approach is used because it is popularly supported in hardware. Shadow mapping has been expanded by Lokovic [9] to support colored shadows however they are done off-line. Shadow volume has been recently expanded to support semi-opaque objects by Kim [12] but it cannot cast self-shadows and works only for hard shadows.

Moreover, since classic algorithms just work with surfaces, they are not well suited for voxel scenes commonly used to display volumetric scenes. Since the proposed approach uses the agents to modify the light at particular places and those can be as small as we require, we can create as much agents to simulate not only the surface of object but also the interior of them.

The cooperative multi-agent system presented in this paper proves to be a practical and applicative alternative approach to handle hard and soft shadows from opaque and semi-opaque objects; it calculates light visibility from light sources to objects as in real life, leaving the opportunity to treat light as it is desired in its traveling. Because of the modularity of the system, it is simple to understand and easy to maintain. The event-based blackboard architecture made this approach suitable for multithreading and multi-core platforms increasing the frames per second on a dynamic application and making this approach ready for future multi-core processors. Shadow mapping and volumes compute hard shadows and have been expanded to support colored hard shadows recently. Our approach can compute colored hard and soft shadows by using many light sources and changing the absorption parameter of the object agents to absorb some a part of the RGB, as shown in Figure 18.

The final result stored in the BB is used as the shading information of the scene and this must be recalculated only if the light agent or any other object agent changes its position or type, then this approach is highly recommended for environments where only the observer changes and light and objects remain static. Scenes such as simulations of buildings in virtual architecture as well as virtual reality and video games are potential applications of shadow information since once the propagation ends no further computation is needed which accelerates the rendering process.

In the last years, graphic hardware has evolved and became a scientific tool for medical or pedagogical purposes allowing volume rendering in real-time as shown by Markus [28]. However, this technique and other similar volume rendering techniques use shadow mapping or shadow volumes to compute shadows. The future work includes the computation of shadows in three-dimensional voxel space and the hardware implementation of the light propagation approach.
As classical approaches, the alternative solution presented in this paper is subject to further research by taking into consideration the main following issues: 1) reduce the continuous increment of memory by using the feature

proposed for animated scenes, 2) include the diffuse interreflection of objects by adding light propagation based on the diffuse reflection of the object agents being illuminated, 3) reduce aliasing caused by discrete space of objects, 4) the hardware implementation of this approach, and 5) expand the current approach to three dimensions by replacing the classic two dimensional Bresenham algorithm [25] for its three-dimensional version [29] in order to compute light visibility in the three-dimensional space.

References

[1] Williams, Lance. "Casting curved shadows on curved surfaces." (Computer Graphics (Proceedings of SIGGRAPH 78)) 12 (August 1978): 270–274.

[2] E., CATMULL. *A Subdivision Algorithm for Computer Display of Curved Surfaces. PhD thesis*. University of Utah, 1974.

[3] C., Everitt. "Hardware Shadow Mapping." *NVIDIA Corporation. andrewd.ces.clemson.edu*, 2000.

[4] Crow, Franklin C. "Shadow algorithms for computer graphics." (ACM Computer graphics) 11 (July 1977): 242–248.

[5] FUCHS HENR, Z. M. KEDEM AND B. F. NAYLOR. "On visible surface generation by a priori tree structures." In Computer Graphics (SIGGRAPH '80 Proceedings) 14(3) (July 1980): 124-133.

[6] Woo, Andrew, Pierre Poulin, and Alain Fournier. "A Survey of Shadow Algorithms." (IEEE Computer Graphics & Applications) November 1990: 13–32.

[7] James, Binn. "Me and my (fake) shadow." (IEEE Computer Graphics) 8 (January 1988): 82–86.

[8] Daniel Scherzer, Michael Wimmer and Werner Purgathofer. "A Survey of Real-Time Hard Shadow Mapping Methods." *EUROGRAPHICS, State of The Art Report*, 2010.

[9] Lokovic Tom, Veach. "Deep shadow maps." (Proceedings of ACM SIGGRAPH 2000, ACM Press) 385–392.

[10] FUCHS, H., GOLDFEATHER, J., HULTQUIST, J. P., SPACH, S.,AUSTIN, J. D., BROOKS, JR., F. P., EYLES, J. G., AND POULTON, J. "Fast spheres, shadows, textures, transparencies, and image enhancements in pixel-planes." *SIGGRAPH Comput. Graph*., July 1985: 111–120.

[11] HEIDMANN, T. "Real shadows, real time." Edited by Inc Silicon Graphics. *Iris Universe 18*, 1991: 28–31.

[12] KIM, B., KIM, K., AND TURK, G. "A shadow-volume algorithm for opaque and transparent nonmanifold casters." *journal of graphics, gpu, and game tools*, 2008: 1–14.

[13] Randima, Fernando. "Adaptive Shadow Maps." (Computer Graphics (Proceedings of SIGGRAPH 2001), ACM Press) 2001: 387–390.

[14] Assarsson, Tomas Akenine-Möller & Ulf. "Approximate Soft Shadows on Arbitrary Surfaces using Penumbra Wedges." (Proceedings of the 13th Eurographics Workshop on Rendering) June 2002: 297–306.

[15] Arie Kaufman, Daniel Cohen & Roni Yagel. "Volume Graphics." (IEEE Computer Society Press) 26 (July 1993): 51-64.

[16] Drettakis, Marc Stamminger & George. "Perspective Shadow Maps." (ACM Transaction on Graphics) 2002: 557–562.

[17] Arvo, Jukka. "Tiled Shadow Maps." (Computer Graphics International, IEEE Computer Society) 2004: 204–247.

[18] Michael Wimmer, Daniel Scherzer & Werner Purgathofer. "Light Space Perspective Shadow Maps." (In proceedings of the Eurographics Symposium on Rendering) 2004: 143–151.

[19] Tiow-seng, Tobias Martin &. "Anti-aliasing and Continuity with Trapezoidal Shadow Maps." (In proceedings of the Eurographics Symposium on Rendering, Eurographics Association) 2004.

[20] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F. "A survey of real-time soft shadows algorithms." *In Eurographics* , 2003: 753-774.

[21] Nan LIU, and Ming-Yong PANG. "Shadow Mapping Algorithms: A Complete Survey." *Computer Network and Multimedia Technology*, January 2009 : 1-5.

[22] ANDREW WOO, PIERRE POULIN AND ALAIN FOURNIER. "A Survey of Shadow Algorithms." *IEEE Computer Graphics & Applications*, November 1990: 13–32.

[23] Michael, Wooldrige. *Reasoning about rational agents.* Cambridge Massachusetts London, England: The MIT Press, 2000.

[24] DAVIS, REID G. SMITH AND RANDALL. "Frameworks for Cooperation in Distributed Problem Solving." *IEEE TRANSACTIONS* ON SYSTEMS, MAN, AND CYBERNETICS, January 1981.

[25] E., Bresenham J. "Algorithm for computer control of a digital plotter." (IBM Systems Journal) 4 (January 1965): 25-30.

[26] Velthuijsen, H. "The Nature and Applicability of the Blackboard Architecture." (Leidschendam: PTT Research) 1992.

[27] Dong J., Chen S. & Jeng J. "Event-based blackboard architecture for multi-agent systems Information Technology: Coding and Computing." (ITCC 2005) 2 (April 2005): 379-384.

[28] Markus H, r, Patric L., Christof R. S. and Timo R. "Advanced illumination techniques for GPU-based volume raycasting." *Proceeding SIGGRAPH '09 ACM SIGGRAPH 2009 Courses*, 2009.

[29] CHENG, X-W LIU AND K. "Three-dimensional extension of Bresenham's algorithm and its application in straight-line." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 2002.