Experiments in the Use of Game Technology for Pre-Visualization

Michael Nitsche
Georgia Institute of Technology
michael.nitsche@gatech.edu

Abstract

This overview paper outlines the value of real-time 3D engines for pre-visualization. Pre-visualization is a standard tool during pre-production of many modern film productions. First, the parallels between the two increasingly digitized technologies are discussed. Second, the paper outlines the special needs and problems posed by pre-visualization, and argues that animation control and camera control are the two main areas that need to be addressed. Finally, it presents a range of experiments that provide different practical approaches to these two core questions and utilize available game technology. The approach of these tests was to keep the rendering real-time – "liquid" – as long as possible. This follows original machinima-like production pipelines. Ultimately, the prototypes presented here illustrate the value of real-time game engines for pre-visualization as well as still prevailing limitations.

Author Keywords

Real-time 3D; pre-visualization; game; machinima; film.

Games and Moving Images

Like film and television, video games use moving images and sound as their dominant forms of expression. However, unlike more linear media formats, games allow for interactive access and manipulation of the events themselves as they are presented. Visualization in film is exclusively applied to tell, while in a game world it is geared to support the play. In-between these two approaches one can find media formats that involve both instant interactive access *and* a focus on story-telling. These combinations of "play" and "tell" in real-time environments opened up new game-based production methods for Computer Generated Imagery (CGI). In this setup, players not only control the performing virtual hero but they also become storytellers and producers that use game engines to stage their artistic visions. Game worlds become virtual sound stages for a new kind of movie production. This new production paradigm has led to the growing field of machinima, which has been defined as "*animated filmmaking within a real-time virtual 3D environment.*" (Marino, 2004, .p1) There are a number of new features that become available through such a shift in the production but in this essay we will concentrate on one specific sub category in this emerging hybrid form of game and film media: the value of video games for pre-visualization.

Pre-visualization is a widespread pre-production technique used in many film productions to plan camera work before the actual shoot. Planning during pre-production is important to optimize the excessively more expensive production stage. It allows a preparation of often very complicated shots that have to be clear not only to the director but also to the director of cinematography, the set designer, the lighting crew, and other members of the film team including the special effects and visual effects units. Thanks to continuous improvements of graphics, but also to ever-more accessible and refined game editors, video games have become a valuable new tool for pre-visualization. The essay will outline these values of games for pre-visualization and present a number of projects realized at the Digital World & Image Group at Georgia Tech that deal with still prevalent limitations of game platforms as tools for this special task.

Related Work

A range of tools originally designed for high-end graphics are used for digital pre-visualization: from *Maya* and *Motionbuilder*, to *3D Studio Max*, *Softimage XSI*, and *Poser*. In addition, a number of packages have emerged that are marketed in part for pre-visualization and in part for machinima production. These include *Poser*, *Antics3D*, *iClone*, and *Moviestorm*. Each of these programs has its own strengths and weaknesses but all of them allow for some kind of staging of events on virtual sets as well as a definition of virtual camera angles toward the resulting scenes. None of them allows the user to *play* the virtual character. Instead, they mix scripting tools for event staging with the concepts of the aforementioned animation packages including a final rendering of the constructed movie that is not necessarily real-time. This essay will concentrate on the real-time pre-visualization generated in a play environment. Thanks to the flexible visualization and the immediate access to the event space, game technology was envisioned as early as 1986 to help CGI film production. Smith's *The Colony*, an extremely early first-person-shooter title, attracted the attention of Cameron during the pre-production phase of *The Abyss* (USA 1989)--even though its practical us remains unclear (Katz, 2005). Today, game engines have been used in actual pre-visualization stages, such as a modification of the Unreal engine for pre-visualization work on Spielberg's *A.I.* (2001) [10] or the use of flight simulators for Cohen's *Stealth* (2005). The question remains whether today's tools can be adjusted to support game engine use in pre-visualization on a regular basis. In addition, the question is whether these tools will be useful for movies that might be less driven by futuristic technology and thus demand more traditional camera and actor control.

Early on, the academic community recognized the value of virtual environments for such a new real-time and virtual-based production. Based on earlier work done by Zeltzer, Drucker, Galyean (1992), Higgins developed the prototype for a 3D/ video pre-visualization tool that included camera and editing controls in a composited image output (Higgins, 1994). Modern versions of this approach often use Augmented Reality interfaces like Ichikari et al. did in their *Mr. Pre-Viz* project (Ichikari et al, 2006). Augmented Reality projects like these still include live video recording as integral part of the image unlike game visualizations that most often concentrate on the representation of the virtual 3D world alone.

A growing number of these real-time game engines started to provide their own editors, scripting tools, and exporters for content. These tools made them useful for pre-visualization experiments in their own right. The various installments of the *Matinee* tool and the Kismet environment for Epic's *Unreal* engine and game modifications such as *Gary's mod* for Valve's *Source* engine directly plug into features provided by the commercial game and provide increasingly sophisticated options for cinematic film execution in game worlds. Kirschner's *MovieSandBox* tool for the *Unreal* engine provides an astonishing package for machinima production that covers many crucial sections including character generation and animation control.

A number of research projects looked into the use of existing game engines for the creation and planning of cinematic sequences. One of the new options available in these real-time 3D engines is the use of Artificial Intelligence (AI) techniques to suggest appropriate camera angles for certain conditions in real-time. Elson and Riedl (2007) suggest *CamBot*, a real-time camera AI to quickly create visualizations based on a database of standardized shots. It was not aimed at pre-visualization tasks but its approach to first seek out blocking issues and then apply a possible camera perspective to a given scene in real-time does address typical pre-visualization problems. Jhala et al.(2008) concentrate more on the problem of pre-visualization itself and their *Longboard* project provides a 2D interface for planning and adjustment of 3D scenes processed in the *Unreal* engine. Jhala and Riedl's work might be the most applicable to the projects presented below as they specifically include machinima and – in the case of Jhala – also pre-visualization in their design.

Game Worlds as Virtual Stage

The use of game engines for cinematic production is based on a fundamental paradigm shift. Machinima producers use the game world not as a challenge to overcome but as a stage to deliver a form of expression. This describes the transition from pursuing a defined goal set by the game (e.g. to beat the high score) to a self-defined goal of artistic expression (e.g. perform a certain dramatic event). Although this is a shift in the basic approach of users to virtual worlds, content in the form of sets, characters, sounds, animations, and visualizations float freely from non-linear games to linear video and film and back. Kinder has termed the resulting intermedia network an entertainment supersystem here boundaries of media are constantly crossed in the traversal of elements from media to media (1991). A prime example for such a supersystem is the development of the *Star Wars* franchise from film to most other media (Jenkins, 2003). Machinima is one platform that thrives in these transitions. It builds bridges between film, theater, game, and performance and is inherently transmedial by definition (Nitsche, 2007).

From the earliest days of game-based machinima production to the current use of recent game engines, machinima artists as well as game and level designers often blend ideas of film set design and game world design. The video game *Stunt Island* (Stephens and Fortier, 1992) allows the player to not only perform stunts but also edit the recordings and design the sets. Playing the game and making the movie about the play-performance are combined. More recently, *The Movies* (Molyneux, 2005) put the player into the director's seat, influencing various aspects of film production. At the same time, many in-game features such as instant replay or the option of recording a whole game session – so called 'demo recording' – often improved technically and expanded the use of game worlds as virtual stages.

Machinima artists were actively involved in this blending of media, for example in the form of custom built game environments that often mirror TV studio setups or film sets. The ILL Clan's *Larry and Lenny Lumberjack* (2003-2005) and *Tra5hTa1k* (2005) machinima series are performed in customized game environments that are modeled after traditional TV studios. Massively Multiplayer Online (MMO) game worlds from *Activeworlds* to *Second Life* include areas that are recreations of film sets not designed to replicate the diegetic world of the film but that of the film production. The question, then, is not whether the merger of film production and game worlds is happening, but how to realize the emerging possibilities. This essay will concentrate on the area of pre-visualization as one example for this development.

Developing Tools for Game-Based Pre-Visualization

Although the shift to virtual stages opens up a lot of options for game technology to support cinematic work, this does not necessarily mean that it fits the needs and special conditions of pre-visualization in commercial film production. These have often developed from historic practices and present their own technical challenges that have to be met by the game system.

Demands of Pre-Visualization

The practice of pre-visualization can be traced back to the much older tradition of storyboarding. Both provide means to plan a certain shot or a whole sequence during the pre-production of a movie and help to develop the visual story. Over time, the storyboards were filmed, animations were added, and occasional model shots or other footage was included. This resulted in countless forms of so-called "animatics." To this day, animatics remain a key pre-production technique and serve often as basis for more detailed pre-visualization. However, pre-visualization itself has become increasingly digital and the more accessible computer graphics became, the more they began to replace these more traditional tools in the industry (Dickreuter, 2007). Production studios like the Pixel Liberation Front have specialized in this niche market and continue to blur the borderlines of the cinematic media.

The main task for all of these tools is to assist the director and the production team in the planning of the specific film shoots. Other tools provide comparable help: e.g. concept art helps to define the graphical look, color palettes, and artistic style. Pre-visualization helps to plan the setup of shots, movements of the camera, avoid blocking problems, and inform different members of the production team about specifics of the individual shot. Although pre-visualization is often done in fast and low quality renderings, it has to be precise enough to provide the necessary information about framing, movement, and staging of the scene at hand. Because pre-visualization is not only a tool for technical planning but also one that supports communication between different members of the production team, it has to be expressive and at the same time flexible enough to allow simple changes. Is should allow for visual experiments which itself should be easy to control and implement.

Problems and Parallels

Manovich discussed specifics of the digital visual media and his defining principles (numerical transcoding, modularity, automation, variability, cultural transcoding) apply to CGI film as well as to video games (2001). Conceptually, both games and digital films are based on the idea of the computer as 'media processor' (Manovich, 2001). Technically though, the difference between a multi-purpose Central Processing Unit (CPU) at work in traditional CGI render farms and specialized Graphics Processing Units (GPU) used on graphic cards usually needed for cutting edge real-time rendering in game consoles and home PCs often still remain. It is up to developments like *Gelato*, programmable GPUs (e.g. using NVidia's *Cg*) and graphic card hardware improvements like *SLI* to shrink these differences and improve a direct integration of real-time into high-end CGI. The shared concept of a digital production remains. It is no surprise then that video games and film production increasingly share comparable production pipelines. For example, Industrial Light and Magic's (ILM) proprietary production pipeline *Zeno* is connected to their real-time tool *Zed*. A continuation of this gradual merger seems only too logical.

However, in order to develop game-based pre-visualization tools one has to be aware of issues of compatibility and technical differences. Compared to the typical workflow in a CGI movie production, games still differ in a number of important ways. Even though engines have become more powerful, a clear difference in render quality remains as any use of real-time technology always includes a lower level of detail, simplified lighting setups, simpler 3D models, a simpler skeleton and rigging conditions, among others. Although none of these differences plays too much into visual quality of pre-visualization these differences include major discrepancies between the systems. Two fundamental differences are found in the animation and camera control.

A classic CGI animator can re-use animation circles, alter them, or use procedural techniques to change them. However, the main movements of the central characters are often unique and directly controlled using the production's 3D animation package. In video games the animations are generally pre-produced. Animators provide them during the game production process – often using exactly the same 3D package one would use for a CGI movie – and import them into the game engine. Games can change a given animation in dependency to collision, physics, or procedural techniques (Perlin, 2004), but direct control over the animation is not available. Players are usually not allowed to create a new animation for the given character but they have to use the ones that were provided by the game.

3D video games can offer extensive virtual camera work but they have to concentrate these efforts on the playability. Cameras are not only dramatic narrative devices but also functional viewpoints into the virtual performance space. This leads to camera control mechanics that are optimized for gameplay but that are often too rough for professional standards which need careful adjustment of framing, focus, and image assembly. Instead of standardized perspectives, like the ubiquitous First-Person perspective or a single following camera the tools need to be adjusted to provide more detailed access to camera control to design and test individual shots. Position, orientation, focus, field of view might be all available in next-gen game engines but need to be accessible through feasible interfaces; means to edit the results in real-time, and access to the powers of modern game engines need to be developed and combined to create better control mechanisms for the virtual camera work.

Sample Projects

The following projects were experimental approaches to cover different needs for more expressive control in real-time virtual environments. They were conducted over the last years to investigate machinima and real-time image control. The short descriptions of this work below are not meant to provide full technical specifications (some of the work has been published in greater detail in other venues) but to outline a range of tools that each address different problems of pre-visualization and try to solve them in an operating prototype.

Approach: Stay Liquid

Between the real-time game controls and the traditional high-end content creation packages a spectrum opens up. On the one hand we might identify ever more sophisticated tools provided by real-time game engines that often miss some basic tools for detailed control or creation; on the other side one might envision a plethora of 3D modeling and animation packages of varying complexity and limited real-time capabilities. The approach at the Digital World and Image Group at Georgia Tech follows principles of early machinima production. It is driven by the commitment to keep the presentation of the dramatic scene real-time (like in a game engine) while gradually improving the interactive access for any artist (as usually optimized by high end packages). In that way, the hope is to provide flexibility and expression.

At the same time, it is acknowledged that different tasks during the creative process might need their own approaches. While a pre-visualization system might need to deal with camera, light, stage design, and animation control – these specific tasks might not necessarily be answered by a single input system. A useful camera control and AI system conceptually differs from a useful animation system and might need its own solution. Not all challenges will be solved immediately in a single tool; instead the aim is to investigate the various problems individually with the intent to learn more about them and attempt for a more holistic solution at a later stage.

One principle concept was followed, namely to keep the format of the real-time rendered image as long as possible, keeping the content "liquid" as long as possible into the production process to allow for maximum flexibility in every stage. It is the notion of the technically real-time rendered and therefore still flexible image that opened up new production (and possibly also new delivery) methods. Thus, this paper argues that any system should try to continuously push this option as far as possible and not succumb to a single render too early. In addition, all of the following projects utilize consumer level hard- and software to remain accessible for machinima production and stay mobile and easy to use.

The New User Camera Control Interface (NUCCI) project offered an early test of possible player control of the camera work (Shapiro, 2006). The project re-staged parts of Fincher's *Panic Room* (2002) in a real-time 3D game environment. All animations were reproductions of the actual movements of the characters in the film scene and fixed. However, the camera offered two options: either players could follow the carefully re-imagined scene in the viewpoints chosen by Fincher for the final movie, or they could interrupt the scene at any moment and create their own camera viewpoints and visualization strategy.

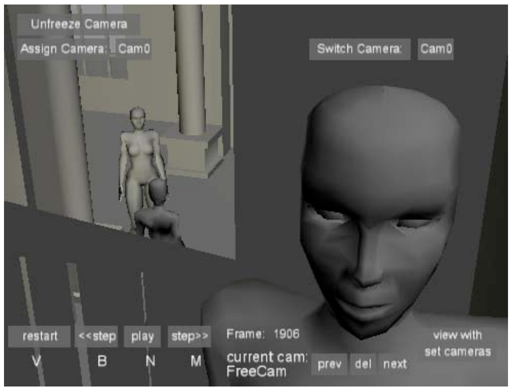

Fig 1: NUCCI: free camera control of a cinematic scene in a virtual world

NUCCI offered basic camera controls via the keyboard and its models and animations were directly imported from *Maya*. It proved that a free camera can indeed open up a playful interaction with the dramatic scene and encourages users to experiment with the material at hand. At the same time it became clear that the interfaces for the camera control were too limited and that an import of animations from a traditional package (like *Maya*) does not utilize the real-time possibilities of the game engine.

Ultimately, the tasks can be broken down into two main categories: control over the image and control over the action. As described above, game engines have shown considerable value especially in the control of the image and specifically camera control. On the other hand, traditional or non-game-based packages are more powerful in the control and staging of the action and especially in the generation of elaborate and unique animations. The aim was to develop different tools to support both areas but conceptually keep them compatible. That is why all the experiments outlined below use the same 3D real-time game engine, namely the *Unreal* engine as it is in use for the *Unreal Tournament 2004* (Bleszinski and Morris 2004) game. The question was how feasible real-time engines are to support the needs for pre-visualization. At the same time, these projects were undertaken with the area of machinima in mind. That means that expensive technology was actively avoided (e.g. the motion capture facilities at Georgia Tech were not used) but instead a game-based and inexpensive approach that continues the hacker mentality of many machinima pioneers was privileged.

Controlling the Action

Controlling the action performed by the avatar in a virtual environment is a puppeteering task. Standard video games allow players to navigate digital avatars and control a limited amount of pre-fabricated local actions and animations. The problem is that games usually optimize the animation systems and the pre-fabricated movements for a selective gameplay experience. They usually simplify controls to streamline player control toward this goal. Although it is possible to import new animations into the *Unreal* engine and the game itself blends between different animations during runtime, it does not provide for any control comparable to a real world puppetry setup. Players cannot easily control a single arm or a leg to create new poses and animations that might be necessary for a useful pre-visualization of a certain scene.

To address this issue the Tangible User Interfaces for 3D Virtual Environments (TUI3D) project was conducted in collaboration with Ali Mazalek (Mazalek & Nitsche, 2007). The project experimented with new, tangible interfaces to control single bones and joints in real-time. The goal was to improve the expressive range available to a machinima artist as s/he controls the animations of a virtual character in more detail than originally provided by the *Unreal* engine. In a first step the tangible puppet controller was implemented which directly mapped onto a virtual character and controlled the movements of the specific virtual puppet via Kirschner's *MovieSandBox* modification of the *Unreal* engine.

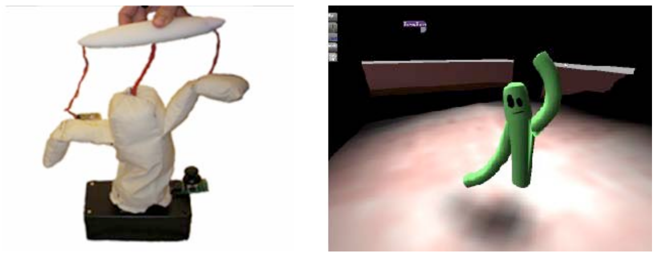

Fig 2: TUI3D: tangible puppet interface (left) controls the virtual character in the Unreal engine (right)

The first puppet was limited to accelerometers and a joystick input and its design mirrored the appearance of the virtual character – "Cactus Jack" – that was modeled for the game world.
In the next step a more modular interface was developed, and the Uniball allowed for the connecting of multiple limbs and sensors to a central interface and map different sensors to according bones and joint structures. This modular approach allowed a more abstract design of the interface and granted more freedom into the way the input was mapped onto a number of different virtual skeletons. Both interfaces use a U-HID board that is recognized as a game controller by the operating system. This way the setup remains portable and accessible.

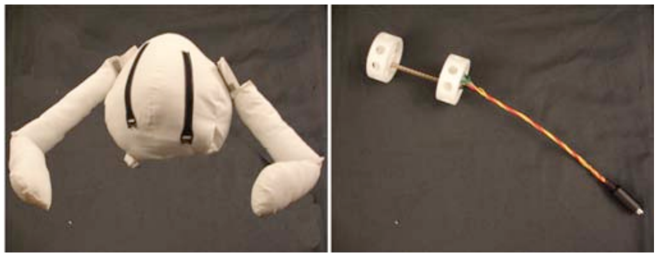

Fig 3: TUI3D: Uniball interface (left); various limbs with own sensors (see right) can be plugged into a central interface controls the virtual character in the Unreal engine (right)

Existing game interfaces were also re-used, such as the Wii-controller, the more elusive Gametrak controller, and the Xbox 360 controller as input devices and the puppeteering controls were hooked up to those devices. These interfaces were presented at various occasions and the concept of virtual puppeteering instead of game-playing was easily understood and the initial feedback from the machinima community was very positive.

Controlling the Action

The second main task for a game-based pre-visualization system is the control of the camera. On the one hand, camera controls were mapped on simple interfaces, like the Wii controller, but first feedback from professional producers pointed out that this approach lacked the necessary precision. Thus, ways to provide a more precise camera system that would still focus on easy access to those cameras to a director/ cinematographer were investigated.

Shotbox

Two main camera directives were implemented in the ShotBox project: focus on the character/ avatar and views defined by the spatial setup of the virtual set. The character-focused camera approach generates a range of pre-defined camera perspectives around any given avatar active in a Matinee scene in *Unreal Tournament 2004*. Matinee is the default tool shipped with the Unreal Editor to set up in-game cut scenes. Wherever the virtual actor moves, the Shotbox camera network follows. That means the director can activate, for example, a pre-defined close up for any character's face at any given moment in the scene. With two button presses the director can activate a specific camera angle such as a close up of the face, an over-the-should shot, a following camera, or a low angle shot for any avatar directly. At the same time, director and cinematographer are free to add new cameras to this network, save them, and re-use them like the other given perspectives. This not only allows the use of an existing camera network but also provides the initial means to generate a new customized one.

Instead of concentrating the camera behavior around the characters, the second approach allows users to define camera viewpoints freely on the virtual film set. While the first version character-based, this second approach is based on the space of the main stage. These cameras stay fixed and do not move in dependency to the actors. However, they remain accessible to the editor and camera operator. To make access to these perspectives easier, the camera control was connected to a tablet PC on which users see the various cameras currently available on the virtual stage. They can then use the stylus to drag these cameras into new positions and rotate them to new directions. At the same time they can control and create new cameras in the 3D view on a second computer running a simultaneous multiplayer session of *Unreal Tournament 2004*. This simplifies the camera selections considerably and offers a working approach to the question of editing in the game world. Although the tablet PC offered one solution, the question of editing control of camera work in a real-time 3D environment remained a mayor challenge.

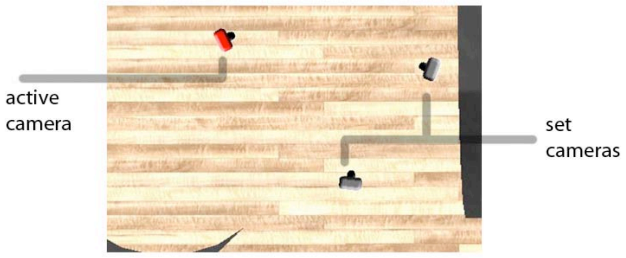

Fig 4: ShotBox: control of multiple cameras positioned on the virtual film set using a 2D tablet PC interface

PlayViz

While the ShotBox camera controls allow for an editing on the fly, they do not support the more traditional approach of first creating a range of cameras and camera behaviors and then editing between the different perspectives in an offline tool. This was the task of the PlayViz project. Recording camera positions and orientations in a parallel running Java application, the PlayViz project addressed the editing problem in virtual environments. PlayViz allows users to record different camera behaviors, copy and paste recorded sections, and cut between those cameras to create a useful camera sequence.

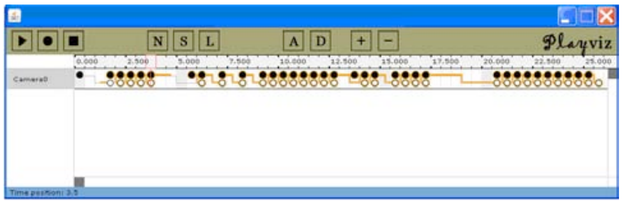

Fig 5: PlayViz: recording of a single camera behavior in the editor window

During this process, the image remains a real-time rendered perspective into an *Unreal* game world. Rendering the camera out not as a linear image but using the Unreal engine and as a live camera behavior in a virtual environment proved challenging and the problems faced were mainly due to the inaccessible code of the underlying game engine. As with the other projects, the commercial *Unreal* game engine was modified and re-used, but due to the proprietary nature of this commercial engine, there were problems optimizing code and debugging the connection to the game world as there is no access to the source. That also meant that Playviz operated independently from the action control and concentrated entirely on the editing options.

Challenges

The various projects were presented to machinima artists as well as professional film and TV producers and the overall feedback was very positive. As far as can be assessed, there seems to be a great interest in using game technology for more traditional moving image productions. Within such an interest, there are a number of arguments that support the use of machinima for pre-visualization and the above mentioned experimental prototypes provide some practical means to improve this development. Camera control in game worlds seems to be one of the most versatile features of this technology, for example. However, finding the right interface for the new features is not particularly easy as they have to respond to traditional film and TV production methods to remain accessible instead of confusing. Even more challenging might be the development of new editing tools that truly embrace the new features of the game engine.

Keeping the image "liquid" and rendered in real-time is the revolutionary ability of any game engine in regard to the moving image production. Keeping the content and image liquid as long as possible in the production pipeline is a core philosophy that supports machinima as pre-visualization tool. In its real-time format machinima can provide a new quality to pre-visualization because it allows for changes in the action and visualization during runtime and at any stage in the process. The task for the development of pre-visualization via machinima, then, is to support the real-time aspects and keep the access to those changes as simple as possible by continuing the accessibility and functionality of the underlying game. It is this ability that differentiates it from commercial 3D modeling packages. Yet, keeping the image real-time aspect is also the source of the most technical and conceptual challenges.

Acknowledgements

The research was generously supported by Turner Broadcasting; Ali Mazalek and her Synlab collaborated on the TUI3D project which was supported by the GVU center at Georgia Tech; special thanks has to go to my students Cameron Aubuchon, Carl Goodson, Mike Lee, Evan Mandel, and Matthias Shapiro, who worked on the respective projects.

References

Dickreuter, R. (2007). Interview with Colin Green (posted April 1st 2007) available at: http://www.xsibase.com/articles.php?detail=133 <accessed July 3, 2008>

Drucker, S.M., Tinsley A. G., and Zeltzer, D. (1992). Cinema: A System for Procedural Camera Movements. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, Cambridge, MA: ACM Press. 67-70.

Elson, D. K., and Riedl, M. (2007). A Lightweight Intelligent Virtual Cinematography System for Machinima Production." In *AIIDE '07*, Stanford, CA: AAAI.

Higgins, S. (1994). The Moviemaker's Workspace: Towards a 3d Environment for Pre-Visualization. MIT Master thesis.

Ichikari, R., K. Kawano, A. Kimura, F. Shibata, and Tamura, H. (2006). Mixed Reality Pre-Visualization and Camera-Work Authoring in Filmmaking. In *Mixed and Augmented Reality, 2006. ISMAR 2006. IEEE/ACM International Symposium,* IEEE. 239-40.

Jenkins, H. (2003). Quentin Tarantino's Star Wars?: Digital Cinema, Media Convergence and Participatory Culture. In David Thorburn and Henry Jenkins (eds.) *Rethinking Media Change,* Cambridge: MIT Press, 281-314.

Jhala, A., Rawls, C., Munilla, S. and Young, M.R. (2008). Longboard: A Sketch Based Intelligent Storyboarding Tool for Creating Machinima. Presented at *Florida AI Research Society Conference - Special Track on Games and Entertainment*, ACM, Coconut Grove, FL.

Katz, S. D. (2005). Is Realtime Real? Part 2. (posted April 1st 2005) available at http://digitalcontentproducer.com/dcc/revfeat/video_realtime_real_part_2/ <accessed June 12, 2008>

Kinder, M. (1991). Playing With Power in Movies, Television, and Video Games: From Muppet Babies to Teenage Mutant Ninja Turtles. Berkeley: University of California Press.

Lehane, S. (2001). Unrealcity. ILM Creates Artificial Cities for Artificial Intelligence. I *Film and Video*, 18 (7).

Marino, P. (2004). *3D Game-Based Filmmaking: The Art of Machinima*. Scottsdale, AZ: Paraglyph Press.

Manovich, L. (2001). *The Language of New Media*. Cambridge, MA; London: MIT Press.

Mazalek, A. and Nitsche, M. (2007). Tangible Interfaces for Real-Time 3D Virtual Environments. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology ACE 2007* (Salzburg, AU June 13-15, 2007) (New York: ACM Press, 2007), 155-162.

Nitsche, M. (2007, December). Claiming Its Space: Machinima. In *Dichtung Digital: New Perspectives on Digital Literature: Criticism and Analysis,* ed. by Astrid Ensslin and Alice Bell, Vol 37.

Perlin, K. (2004). Can There Be a Form between a Game and a Story? In *First Person: New Media as Story, Performance, and Game*. Ed. by Noah Wardrip-Fruin and Pat Harrigan, Cambridge, MA: MIT Press, 12-18.

Shapiro, M. (2006). *The Novice User's Camera Control Interface (NUCCI): A Real-Time Cinematic Solution to Previsualization*. Georgia Institute of Technology M.Sci. thesis.