Diploma Thesis in audiovisual media

the guerilla guide to a virtual 3D-realtime puppet performance

by Friedrich Kirschner

Hochschule der Medien, Stuttgart

21st December 2007

First Examiner: Dr. Johannes Schaugg

Second Examiner: Michael Nitsche, Ph.D.

I hereby testify that I have written this thesis autonomously and unaided and that all my sources are referenced either in the text itself or in the reference list in Appendix III.

_____

Friedrich Kirschner, New York, the 21st of December 2007

Abstract:

A Puppet Play is an animated short film that is performed live for an audience. It was created using a modification of a first person shooter computer game. The complete performance setup can be downloaded and re-played by everyone that owns the computer game.

A Puppet Play is conceived as *Gesamtkunstwerk*.
The production process, the live performed narrative and its distribution are an integral part of the narration that defines it.

Contents

1 Concept

"An alternative conception of democracy is that the public must be barred from managing their own affairs and the means of information must be kept narrowly and rigidly controlled. That may sound like an odd conception of democracy, but it's important to understand that it is the prevailing conception..."

- Noam Chomsky, Massachusetts Institute of Technology, March 17, 1991[1]

Media Control is my main motif.
It is not only discussed as a narrative in the performance, or through its production method by re-purposing an already existing and commercially produced computer game, but most importantly through enabling the audience to replay the performance and experience the same creative process as the performers themselves, or change it to their liking and thus taking part in the subversion of the commercial product.
Mass Media is metaphorically represented in the photoreaistic advertising and live action video depicted in the second half of the live performance.

By releasing the original characters, environments and the technological framework to restage the performance, I am handing over the control of the performance to the audience and enable them to modify and interpret the content, narrative and production process.

1.1 Why make a guerilla guide?

"a person who engages in irregular warfare especially as a member of an independent unit carrying out harassment and sabotage"
Mirriam Webster online dictionary definition of the word "guerilla".

While the term warfare is certainly an exaggeration, the fact that individuals and small independent groups fight against the prevalence and dominance of a few major companies in the field of visual mass media exposure is not.
With cinemas reserved to movie productions that cost millions of dollars and television largely driven by advertising demands, the diversity of content lies solely in the hands of a few media empires.
But with digital content delivery and open rendering systems, the possibilities to take part in the conversion and modification of this situation is put back in the hands of the audience.
Already, people are modifying or replacing the content of the movies they share over peer-to-peer networks, as seen in a recently downloaded illegal copy of 28 weeks later, where every television and monitor in the film carried a custom subliminal message composited into it.
Or people create their own critical content altogether, as seen in the hilariously to-the-point web series Red vs. Blue by Texas based Rooster Teeth[6], that spawned

a whole new wave of self-awareness in computer game criticism.

The growing convergence of visual media started with sharing assets between computer games and movies[7], continued with creating music videos with computer game characters and will eventually lead to the reduction of content to the very definition of story and character, interpreted and played back by complex computer systems[72] that will be vulnerable to subversion just like every complex computer system is vulnerable to hacking attempts - be it the new generation of game consoles or the iPhone [8]. With the guerilla guide to a virtual 3D-realtime puppet performance, I aim to show new possibilities of engagement for the end user with his media consumption, and how an alternative media distribution and creation system could be established, both technically and conceptually, that would leave its very content open to modification and reinterpretation by anyone.

2 Narrative

The main motifs that lead to the performance were contrasts between the individual and the mass, the media and its intrusion on public space and the question of the virtual and the real. During the production process, the concept of Control was introduced through layering different control mechanisms throughout the virtual and the real performance space using tangible interfaces and a live dancer.

2.1 Influences

Conceptually, *A Puppet Play* is influenced by the works of Noam Chomsky, mainly his book "*Media Control"*[1] and Theodor W. Adorno and his criticism of the culture industry, mainly his essay *"Culture industry reconsidered"*[46] .

The basic ideas and images for the story arc have their roots in the Hans Christian Andersen Tale "*The Little Match Seller*" released in 1848 and its numerous recent adaptions, namely the Demo production "*Halla*"[47] and the Korean movie adaption *"The Resurrection of the Little Match Girl*" [48]. In addition, images from the movie *Metropolis* [49], mainly from the "*M-Machine*" had an impact in the writing process. So did the works of Brian Wood, especially his graphic novels *DEMO*[50] and *ChannelZero*[51].

The initial story is constructed around fairly human acting characters and their basic motivations, following traditional storytelling guidelines[52]. The final prodction concept shifts to a more constructed and abstracted approach that also incorporates the live performance aspect, the setting and the production method.

This shift evolves around the fact that the production method itself - using a violent computer game[1] for animated storytelling - is part of the subtext of the piece and is adressed during the presentation.

I decided to develop the story towards a more image and motif based piece that works mostly through visually citing the aforementioned sources. I added parts at the beginning and the end of the presentation that explain the production process and introduce its subtext to willingly break the concept of suspension of disbelief, not unlike some of the principles of Brecht's "*Epic Theatre*".

2.2 Story Process

The original script-sketch that I developed at the beginning of the production process was different from the final performance[2]. It was not intended to be the final script and was only used to give an idea of the setting and the characters involved in the environment for the production team.

The connections between the characters involved - both human and digital - were developed during the actual production process and lead to a significant shift in the

1 rated 16 and up by the Unterhaltungssoftware Selbstkontrolle, § 14 JuSchG USK Nr.: 10449/04

2 see Appendix I

emphasize of the story.

In the end, there was no final script but a sequence plan that consists of roughly three motifs and a basic dramatic arc that is defined by key events which happen during the performance. This sequence plan leaves is highly modular in storytelling and allows for live-improvisation without breaking the overall dramatic arc. The timing of the performance is thus not fixed but variable and certain parts can be extended or narrowed down.

Another product of the production process is the equalization in the characters' roles. Instead of having a clear protagonist, the character focus switches from the first character introduced, *the inside character*, to the second character introduced *the outside character*. You can see the performers controlling the first character and the second character is controlled by an actual live dancer whose movements are being mimicked by the puppet on screen. This reminds the audience to the fact that all of this is a controlled puppet play, in turn supporting the story concept of control.

The goal of the story development process is to tie the narrative closely to the performative system itself. Instead of reading and analyzing the story after the performance, you can explore it by downloading the performance software, look and perform the sequences and clearly become aware of all the layers that are involved in the story by going through the production process yourself.

3 Framework

I developed *Moviesandbox*[9], the software used to create *A Puppet Play*, as a modification for the computer game *Unreal Tournament 2004*[10] to transform the initial game into a toolset that I can use for creating animated movies.

(Fig. 1 - UT2004)

It was designed as a plattform for realtime filmmaking and animation production and reached version 1380Beta before the start of this project. Its design allows for the unique production experience that made *A Puppet Play* possible. I also developed a number of smaller applications in C# that help with data exchange between Moviesandbox and the outside world.

3.1 Related Work

There are other machinima applications based on game engines that fall into a similar category than *Moviesandbox*.

The ILL clan uses a custom setup for their show *"Trash Ta1k"*[11], implemented in the Torque game engine[12] that extends their multiplayer game setup. Every character is controlled using its own computer, that sends data to a server. Every puppeteer thus sees the environment from the perspective of their individual character, similar to a multiplayer first person shooter game setup like UT2004.

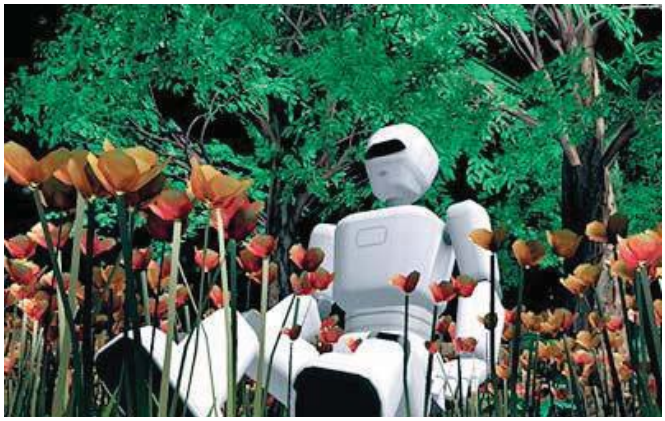
(Fig. 2 - TrashTa1k)

It relies on predefined animations that can be triggered using key inputs from a device called *Nostromo*[13]. Lipsyncing is performed manually by swapping the face textures on the character.

The art direction is very similar to Quake3[14], UT2004 and comparable games. Objects can be imported in a variety of different 3D data formats but canot be created in Torque. The system is proprietary and not open to the public.

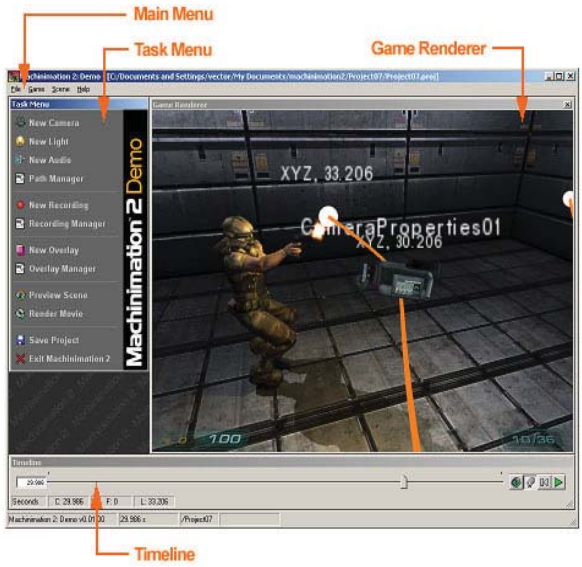
Machinimation[15] is a tool specifically designed for machinima filmmaking and was the first standalone application for machinima filmmaking. Its professional version has been used to create the short animation "*Anna*"[17] and the music video "*In the waiting*

line"[16] for British band *Zero7*, directed by *Tommy Palotta.*



(Fig. 3 - still from "In the Waiting Line")

The basic version of *Machinimation* is free and allows the puppeteering of game characters from the game Quake3 or, in version 2, Doom3[18]. The puppeteering capabilities are limited to the game characters' animations and actions and use the game's control scheme.



(Fig. 4 - Machinimation)

Performances can be created by recording performance "tracks". Additional camera "tracks" can be recorded seperately and modified using keyframes in a timeline. Depth of field and overlays are applicable as post-processing effects.

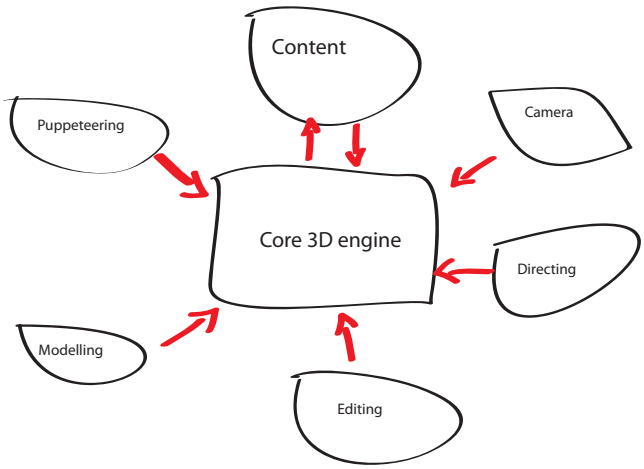
Additionally, tracks can be recorded using the application's network functionality, enabling multiple players to be recorded at once.

The art direction is directly tied to the game assets that are used. Environments can be created using the free Level Editor GtkRadiant[19]. Props and Characters can be imported using a proprietary 3D data format.

3.2 Underlying Concept

Moviesandbox was built as a system that allows for different tangible interfaces to be implemented for animation specific tasks. It was conceived as a prototype framework to test new workflows in realtime animation and is highly modular and extendable using the *UScript* scripting language and its own graphical user interface (GUI).

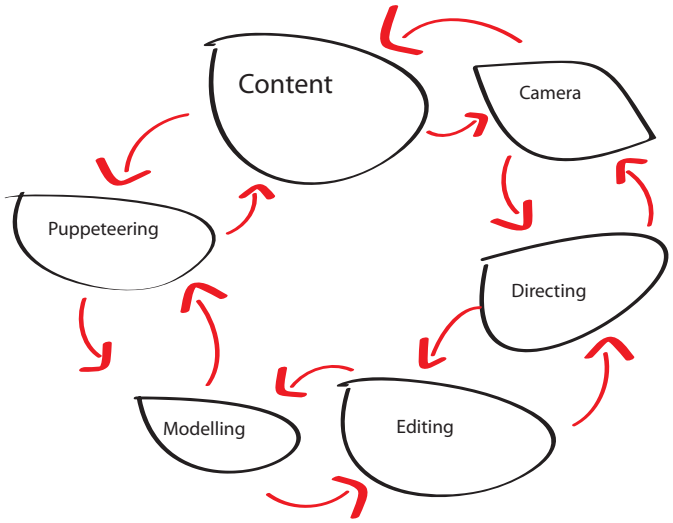
I developed *Moviesandbox* from October of 2005 until now with a variety of production workflows, visual concepts and features in mind.



(Fig. 5 - Moviesandbox concept)

3.3 The Workflow

Instead of a traditional animation production workflow consisting of pre-production, production and post-production[24], the workflow is open to an iterative process in which every production step has an influence on every aspect of the production.



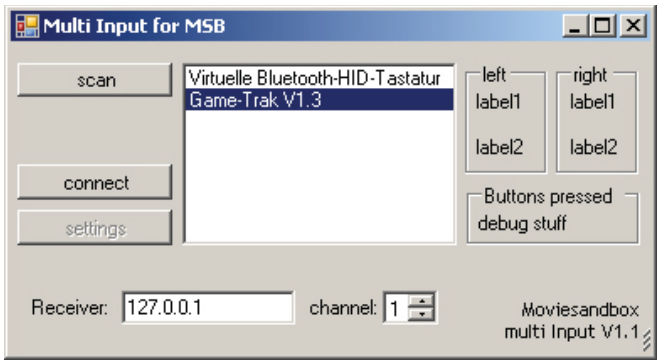
(Fig. 9 - Moviesandbox workflow)

This means that a new asset has an impact on the color palette, a new character or the implementation of new input devices has an impact on the narrative structure, which in turn has an impact on the character design. Due to the realtime nature and the fast implementation times of *Moviesandbox*, quite lot of different iterations can be tested and the different aspects of filmmaking, including editing, lighting and camerawork can influence each other more fluently, in the sense of Wagner's idea of *Gesamtkunstwerk*.

The initial narrative approach that was sent to the production team on October 28th and that marked the official beginning of the production was heavily modified during the production process.

3.4 Input devices

A wide range of Human Input Devices[20] are supported in *Moviesandbox*. This is achieved by using a standalone programm called "*Multi HID Input*" written in C# that communicates with the game using the network protocol UDP[21] and a simple data structure that reads coordinates and button states and sends them using either 1 or 2 unsigned bytes per value. The data structure depends on the device connected. Object parameters such as location and/or rotation or specified events in *Moviesandbox* can be manipulated in realtime reading and interpreting input data from these devices. There is no set maximum number of instances of *Multi HID Input* that can be open at any one time, theoretically allowing an unlimited amount of different devices to connect to *Moviesandbox*.



(Fig. 6 - Multi HID Input)

Multi HID Input is made available through the lesser GPL software license and can be extended to provide special functionality.

3.4.1 The WiiMote

The WiiMote is a wireless Input device created by Nintendo for use with the Wii Console and PC. It provides twelve Buttons, a 3-axis accelerometer and an IR sensor and transmits data using the Bluetooth protocol as a HID device[22] . The Accelerometer allows for two degrees of freedom (roll and pitch of the WiiMote) but can be coupled with the IR sensor for a wide variety of different setups[23].

(Fig. 7 - the WiiMote)

3.4.2 The GameTrak Device

The *GameTrak* is a USB Human Input Device created by In2Games in 2004 for the PC, Playstation2 and Xbox. It consists of two retractable strings that are measured in length and angle (ranging roughly from -30 to 30 degrees in two physical axis) using potentiometers. Each of the strings provides three degrees of freedom when converting the angle and length to absolute XYZ coordinates. If you connect both strings through a rigid object, you can get five degrees of freedom, XYZ and the Pitch and Yaw of the rigid object connecting them.

(Fig. 8 - the GameTrak)

3.5 Directing

Moviesandbox offers a mouse cursor and keyboard controlled-GUI for setting up scripted events and sequences that modify the parameters and states of objects placed in the virtual world over time. It uses a node based system similar to Apple's Shake[25] that is based on events rather than a timeline system like Adobe After Effects CS3[26].

A node can be connected to another node, visualising the order of execution for the individual nodes' code content.

(Fig. 10 - Moviesandbox GUI)

Moviesandbox has two main modes of operation, the *Setup Mode* where all the different events and nodes are set up, and the *Play Mode* where the setup is executed. There are a number of special nodes that can form the start of a node queue.

ScriptNodes can be used to start a list of commands that are executed by *Moviesandbox* when in *Play Mode.*

RootNodes are created every time a character is placed. They form the logical starting point for a queue of commands related to that specific character.

KeyInput nodes can be used to execute one specific node-command at the press of a specified button on the keyboard.

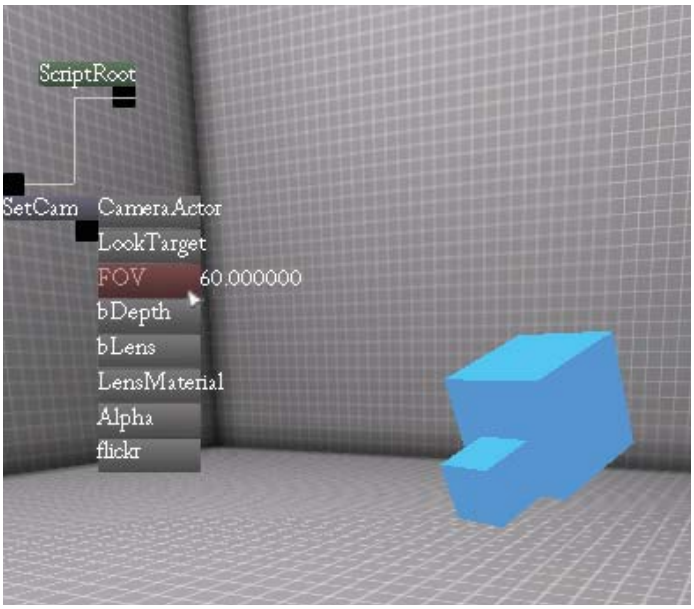
UDPInput nodes can be used to execute one specific command when data is sent to a specified network port.

Apart from the live aspect of *Moviesandbox*, it also allows for simple visual scripting of character actions. The aforementioned *RootNode* can be linked to a list of different actions that the character belonging to the node will then try and execute. These actions include a movement command, interpolating to a specified pose, rotating towards a certain point and more.

The movement and rotation commands use UT2004's internal functions for pathfinding and skeletal animation.

3.6 Cameras and Editing

Camera objects can be placed anywhere within the boundaries of the virtual environment using the GUI. Every camera object can be switched to using the *SetCam* Node that provides modifiable parameters which define the Camera's field of view, depth of field and potential image overlays. In addition, camera objects have a control implementation for the *GameTrak* device using the aforementioned *Multi HID Input* tool. This allows tangible control of both camera location and rotation.



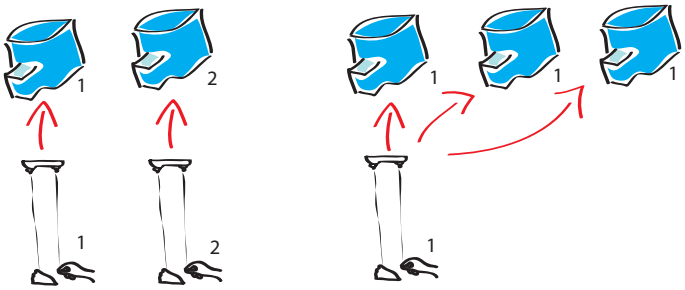
(Fig. 11 - Camera Options)

While it is be possible to use one *GameTrak* device for each camera to be controlled, I decided to implement a "channel" based system.

Every camera object can be assigned to an input channel, with multiple objects being able to share the same channel.

The *Multi HID Input* application sends the device values to a channel that corresponds to the ones selected in *Moviesandbox*.

This way, the user can choose to use individual devices for multiple cameras or use one device to control them all.

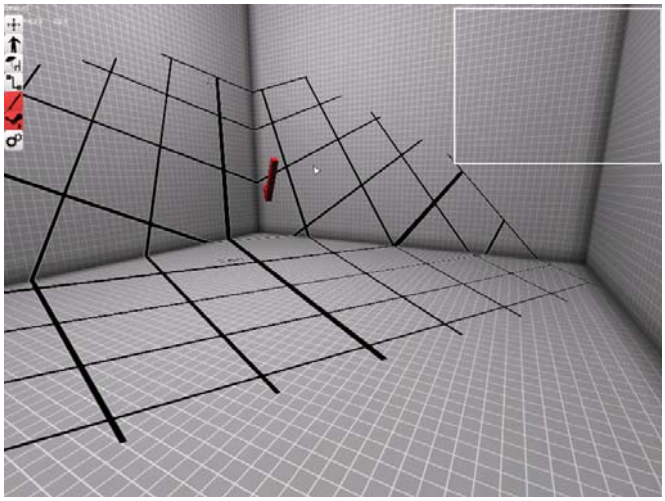


(Fig. 12 - input data distribution)

Moviesandbox can switch between individual camera objects and -settings through keyboard or data input or through scripted sequences.

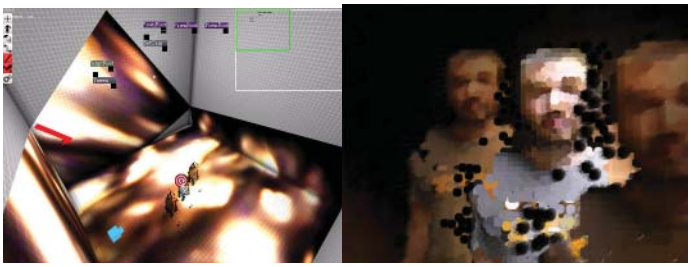
3.7 Lighting

The *Moviesandbox* GUI exposes *Projectors*[62] as a means of lighting your environments. Projectors work just like non-object specific projection mapped textures[63] that modify the color information of the texture map or particle they are projected on. Their effect is similar to a real life projector that shines on physical objects.



(Fig. 21 - projected texture)

They can be animated just like any other *Moviesandbox* object and their effect can be restricted to specific objects.



(Fig. 22 - projected light)

3.8 Animation

Every character type with the exception of the Sprite character can be animated in a number of different ways.

The first way is through skeletal animation[27], each hierarchically connected skeletal bone can be rotated to create a specific pose for the character. *Moviesandbox* can then interpolate the rotation values over time to make the character animate from one pose to another. It allows the user to create such poses with its own GUI. The interpolation is linear and the poses are a static set of data.

The second way is through physical animation, using constrained rigid body dynamics calculations[28]. In this case, a *Ragdoll* is used for the movement of the character's bones[29] that can be defined by a program called *Karma Authoring Tool* that is part of the UT2004 software distribution. The dynamics calculations are handled internally through UT2004's physics system called *Karma[30]* and are not exposed in *Moviesandbox*'s UScript code.

This approach grew out of early work on UT2004 for Kurt Hentschlaeger's contemporary dance visualisation N/8[31] and then later *KARMA*[32], where the rigid body dynamics were used to synthesize and react to sound[33].

The *Moviesandbox* implementation for animating characters using the *Karma* system adds constraints to key points of interest for the Animator, namely the hands, the spine, the head and the feet. The position of these points can be modified through user input. Given the dynamic calculation model, this form of animation creates unique movements for every position change.

The third way is through triggering animations that are included in the proprietary UT2004 character file. This method is not available through the GUI. It is used for the walking animations of non-Karma based and non-Sprite based characters. Technically it works just like the skeletal animation described at the beginning of the chapter.

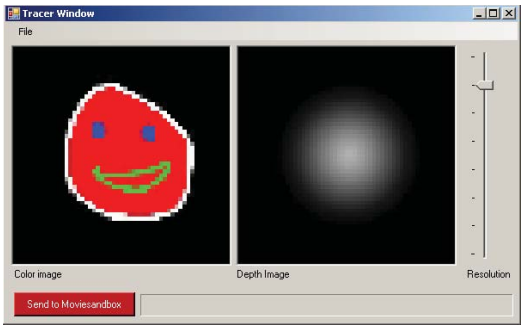
In addition to character animation, every object created in *Moviesandbox* can be animated through the GUI in both location and rotation through linear interpolation along a set amount of points defined in space or absolute data from a data source such as a human input device or a network data stream. Every object can have its own general data source controlling its location and rotation or can have a special control implementation defined in UScript.

3.9 Props and Environment creation

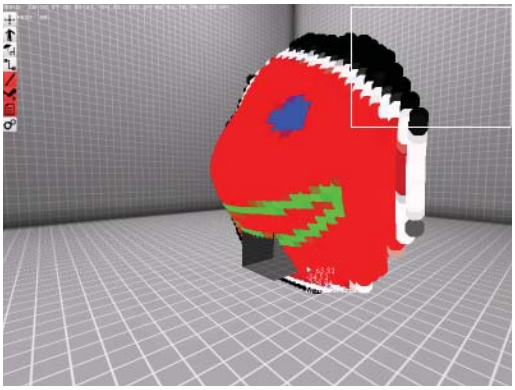
Props in *Moviesandbox* can be created using a particle system approach[59] where the particle system is being treated as a brush that can be used to paint the shape of an object directly into the virtual environment. Color and brush thickness can be chosen from a GUI, the brush shape itself can be changed through code.

(Fig. 18 - creating props in Moviesandbox)

Building on this same approach, images can be scanned into the virtual environment using a program called *Tracer*. It sends color and depth information of every single pixel of a 256x256 pixel dimension image to be redrawn in *Moviesandbox*. The color information is taken directly from the desired source image, while the depth information is taken from a separate greyscale image that works similar to a depth map[60], where the greyscale value of each pixel is converted to the relative Z-Coordinate of the particle drawn within *Moviesandbox*.



(Fig. 19 - my Tracer Application)



(Fig.20 - resulting particle object)

The size, position and rotation of the resulting particle system, as well as each individual particle, can then be modified in realtime through the GUI and a set of console commands.

In addition to the particle based approach, it is also possible to import predefined vertex based Meshes[61] through UT2004's Library of content and workflow options.

Finally, simple Images (sprites) can also be placed into the environment.

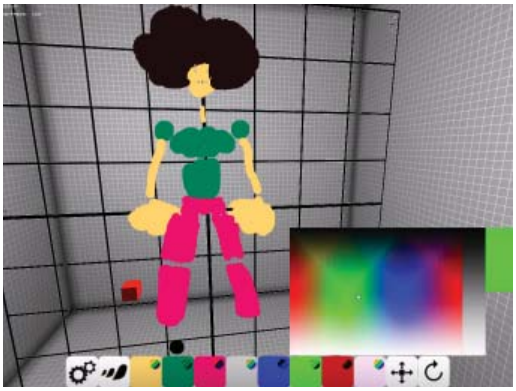
3.10 Character Creation

Characters in *Moviesandbox* are differentiated not only by their visual rendering style but also by the means of how they can be controlled. Similar to the different options of creating props, you can create vertex based, particle based or sprite based Characters.

Vertex based Characters can be imported through *UnrealEd* and can not be edited or created within *Moviesandbox*. All the game's original character models fall under this category, as well as characters downloaded from community websites such as

Skincity[1][67] or *Planet Unreal*[2][68] and following UT2004's character creation guidelines.

The particle system approach uses the basic UT2004 Character's skeletal structure and attaches a particle system as described in the *Props* section to each individual bone. This way, characters can be created and modified using a seperate GUI within *Moviesandbox* called the *VoxelEditor*.



(Fig. 35 - the VoxelEditor)

Sprite Characters can be created by also using a seperate GUI called the *2D Character Editor* and consist of a set of animated textures for every state (moving, standing or animating) the character is in. These animated textures are then displayed depending on the viewing position of the virtual camera and the state the character is in accordingly.

3.11 Organising the System

On larger projects with many cameras, events and scripted sequences, the screen can get cluttered with nodes fairly quickly and finding the connections between nodes and objects can be tedious at times.

Moviesandbox therefore not only allows all of the *RootNodes*, *ScriptRoots* and *KeyInputs* to be named individually, but also offers visual clues of connections between nodes and objects. Every node that is connected to an object will highlight the object through superimposing a red square around it when the mouse cursor moves over the node.

Furthermore, nodes can be grouped and hidden inside container nodes. Ultimately, *Moviesandbox* provides a minimap on the upper right hand corner of the GUI that allows for a work area that is 4 times as big as the current screen resolution.

These organisation functions are implemented to overlook large and complex setups and can eliminate confusion, especially with the idea of redistributing the source setup to the end user in mind.

3.12 Conclusion

The director can set up a system of commands and events to switch cameras, trigger animations or change other parameters in real time. The system is not static and predefined but can be dynamically orchestrated.

1 http://skincity.beyondunreal.com

2 http://planetunreal.gamespy.com/

4

Puppeteering

Puppets in general and Marionettes in particular are systems that appear intuitive and understandable because we are used to dealing with them[34].

Most people have probably been confronted with a marionette in their life and even though the initial mechanical construction seems complicated, the idea of pulling a string and seeing the puppet react to it in a physically comprehensive way is comforting and demystifying.

A marionette very far from the level of abstraction that we see in "*Making of*" videos, where the movements of actors in spandex suits are magically superimposed onto a highly detailed 3D model of the exact same actor[35] or the complex movement of characters that result through pressing a single button[36].

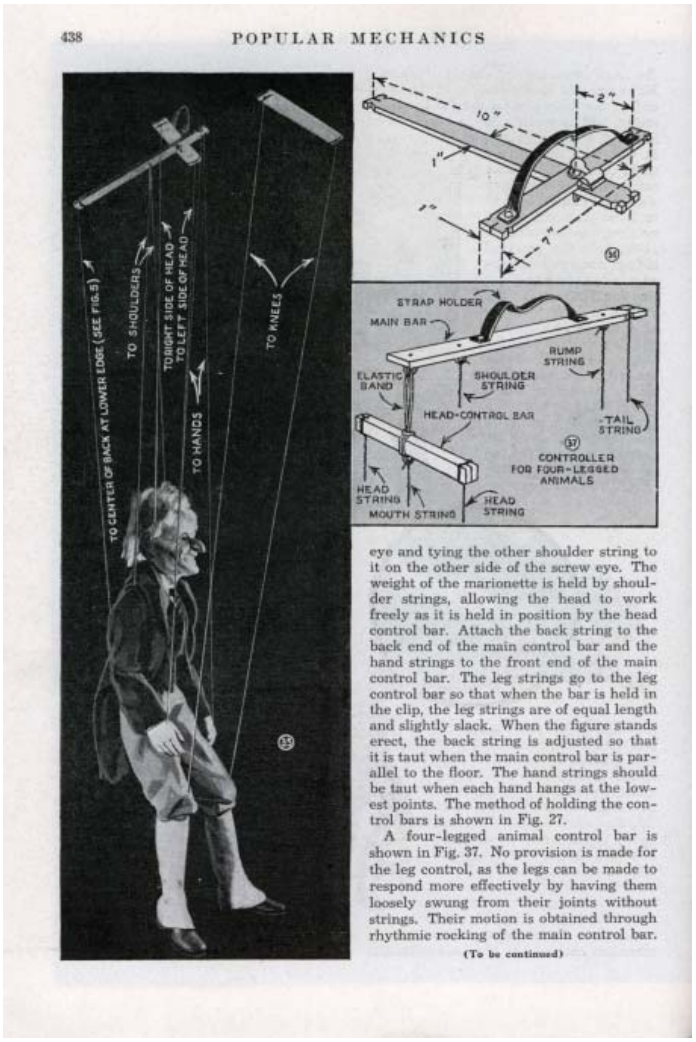
Using Marionettes as characters makes the whole process of animation easier to relate to for the audience.

4.1 The wood and wires of a digital marionette

Moviesandbox supports the *GameTrak* device as a means of converting two locations in the physical space to locations in the virtual space. It has a simple rigid body puppet implementation that allows a puppeteer to control the arms of a character.

For *A Puppet Play*, the digital marionette required a broader spectrum of movement, including foot and head movement than Moviesandbox provided in its previous state.

I implemented a second *GameTrak* device to allow for foot placement, loosely based on the string setup of traditional physical marionettes[37].



(Fig. 13 - Marionette schematics)

I also decided to mount the *GameTrak*s upside down in order to get a better range of values from the *GameTrak*. The device measures angles up to 30º from the center of each string. I needed to provide a certain amount of distance from the device to allow for a wider range of trackable motion.

Since my setup already requires both hands of the

puppeteer for moving the limbs, I decided to map the head rotation to a WiiMote's accelerometer data, allowing me to control the head's yaw and pitch by physically rotating my hand accordingly, without the need to release the strings that I hold at the same time.

(Fig. 14 - final controller stand)

4.2 How to puppeteer a digital marionette and not walk off the stage in the process.

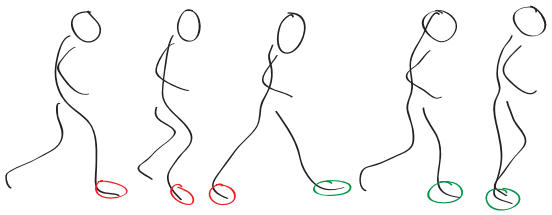
The problem is easily described. Imagine you have a wodden marionette and want to make it walk around. When you place the marionettes feet one after the other, your hands subsequently move in the same direction with the marionette causing you to change position in the physical world.

In a traditional puppet theatre, the amount of space for the puppet is usually manageable and the puppeteer has enough space to freely move around together with his puppet.

In a digital environment though, the screen or projection creates an infinite space for the virtual puppet to move in, the physical space the puppeteer resides in is limited. The movement can thus not be mapped directly.

Comparable realtime puppeteering systems, like GeorgiaTech's Cactus Jack (presented at the Machinima Film Festival 2006 by Michael Nitsche and Ali Mazalek) do not present an answer to the question of how to dynamically generate a walking animation without actually changing the position as a puppeteer using live data input devices.

To solve this problem, I investigated the anatomy of a human walking sequence and found out that there is always one foot touching the ground.

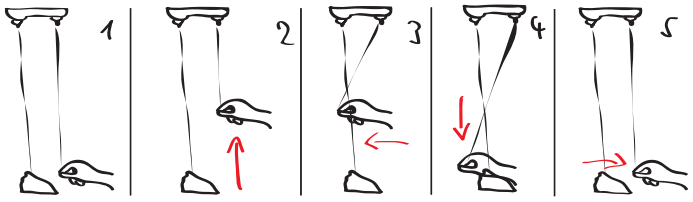


(Fig. 15 - passive foot touches ground sketch)

Thus, one could argue that a step consists of an active foot and a passive foot, the active foot being the foot that actually changes position, while the passive foot keeps its position through the duration of the step.

I identify the active foot as the one that is raised first.

It stays active as long as it doesn't move below a certain threshhold height. If it falls below, the active foot becomes passive and the next foot that is raised above the threshhold level becomes the active foot.
With this in mind, I created a simple algorithm that only reads the *GameTrak*'s input data on the active foot and ignores the input data for the other one.
This allows the Puppeteer to move the input cord for the passive foot back to a null-position.
After the step is finished and the active and passive feet swap, the next step starts from the null position.
This allows for a variety of movements, including sidestepping and walking backwards.



(Fig. 16 - gametrak movement sequence)

4.3 Grabbing the spoon

The sequence in the beginning of the performance in which the marionette grabs a spoon and starts eating cereal, requires the possibility to attach objects to the character's hands.
The objects have to be within a specific radius of the left or right hand and the A-Button (or B-Button for the right hand respectively) on the WiiMote needs to be pressed and held to "pick up" the object. The original position of the spoon is stored and as soon as the button is released, the spoon teleports back.
This behaviour is intentional, as tests have shown that it looks worse to have the spoon float in mid air or intersect with an object when being put down.
The slight visual jump to its original location can be minimised through practicing and reembering the correct initial location of the spoon.
This approach is intuitive enough to be extended to any object in the *Moviesandbox* system. However it can lead to the marionette picking up many different objects along the way, be it the bowl of cereals, the table or a complete wall of the set.
The solution to this was a simple parameter check to constrain the objects the character can pick up to those tagged "grabMe",
This further increased the range of expression of the digital marionette.

4.4 The MoCap Jacket

Another important part of the puppeteering was the production of the wearable human motion capture system that was created by Mika Satomi and Hannah Perner Wilson. Following a brief conversation about the possibilities of such a device that I had with Hannah Perner-Wilson in mid-October and a couple of tests, we decided to implement it into the performance by early November.


(Fig. 17 - wearable motion capture system)

Human Motion Capturing is usually referred to as "capturing the large scale body movements of a subject at some resolution"[38]. There are lots of different approaches using computer vision, ultrasound[39], inertia sensors[40] and so on.

Our wearable system uses the IO board Arduino[41] as the microcontroller to read the data from *Flexible Fabric Touchpad sensors*[42], that are "made from two layers of conductive fabric with a layer of ex-static in between"[43]. The way the sensor works is described as follows:

"When pressure is applied to the touchpad by pushing the layers together or by bending the sensor, the ex-static layer lowers its resistance. [...] Thus the current reaching the microcontroller input varies with the pressure applied to the sensor."

This information is then sent to *Moviesandbox* through an application written in Processing[44] using the UDP protocol. It is interpreted as positional data for the limbs of *Karma* driven characters.

The first working interpretation of the motion capture data of dancer Ivana Kalc was established on the 21st of November 2007, three days before the scheduled performance. In the days and weeks before, we communicated on a regular basis negotiating data structures and evaluating the overall quality of the captured data[45].

The wearable motion capure system had the biggest impact on the transformation of the narrative by enabling a physically represented actor to actively perform in *A Puppet Play*.

The motion capture control is not as versatile as the *GameTrak* controlled marionette. It does not provide an easy way of capturing or interpreting absolute position data for the person wearing the suit.

4.5 Puppeteering Conclusions

Using a dynamic rigid body marionette with the appropriate controller can produce original animation in realtime and gives lots of options for the animator to interact with its environment.

Mapping the complete range of character motion to the hands of a single puppeteer requires a lot of training though.

The overall quality of the movement is thus not only depending on the setup of the rigid body simulation but also on the skill, versatility and creatvity of the human puppeteer, adding an organic and human touch to this form of computer animation.

5 Production

The performance is divided into two main components - the "*Inner World*" and the "*Outer World*" which are meant to be contrasting yet at the same time not feel like they are inhomogenous. The characters that inhaibt these two places also needed to fit in as if it were their natural habitat. Thus, the environments were created before the characters were finalized, allowing continuous infuence from both aspects to the respective other.

5.1 Influences

Similar to the story, the Art Direction is influenced by Ridley Scotts movie "*Blade Runner*"[53] and Fritz Lang's movie "*Metropolis*"[49]. For the camera shots, and the visual composition, the animated series "*Serial Experiments Lain*"[54] was a big influence.

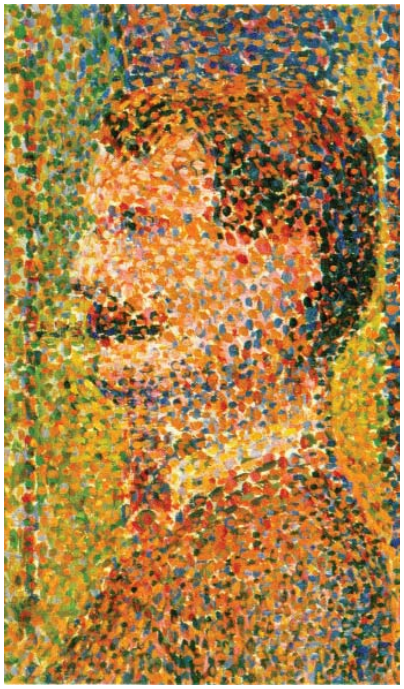
In terms of animation and movement, the performance relates to the pupeteering in the opening sequence of the movie *"Being John Malcovich"*[55] performed by Phillip Huber, the movie *"Peter & the Wolf"*[56] , and the Jim Henson hommage 3D computer animatied short film *"Overtime"*[57].

The process of applying color and contrast and the merging of two-dimensional elements in a three-dimensional environment, was inspired by the recently released teaser for the computer game *Street Fighter IV*[58].

5.2 Designing the Inner World

I chose the color palette to be restricted to earth-tone colors and neutral greys to create an organic and warm environment. Emphasizing the Inside feel, I used warm, confined light sources that cast obvious shadows and we framed the camera so that it seems obstructed trough the environment in most of the shots.

The visual style is strongly based on the impressionistic concept of Pointilism as seen in the paintings of Georges Seurat, stressing the metaphorical and surreal nature of the inside world.



(Fig. 23- Georges Seurat, La Parade)

In addition, the camera depth of field is extremely exaggerated to simulate a microscopic view of the environment and to subtly increase the perceived visual density. A visual flickering exposure effect is introduced and the overall animation timing is slightly randomized to simulate an old and unstable projection, accenting the reference to Lang's *Metropolis*.

5.2.1 Stress Testing and extending the Visual Repertoire

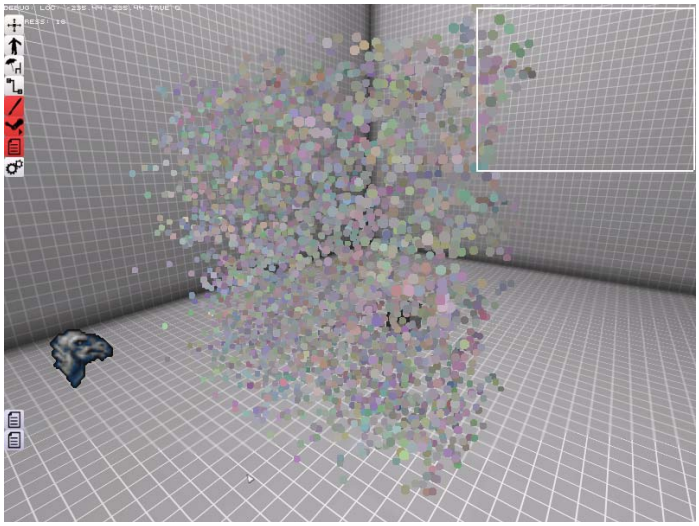
It was crucial for the improvisational process of the production to know exactly what our limits were in terms of visual fidelity. I performed a series of performance stress tests early on.

The first of these tests was aimed at providing a maximum number of particles that I could draw in a scene. The test began with a series of stacked, fully painted planes, by which I was able to create up to 200,000 particles while still keeping a reasonable framerate.

5.2.2 From Stress Test to Key Visual Element

Early conversations with the DoP Bianca Bodmer made it clear that we both want to convey a strong sense of depth and dimension in the visuals, as well as a very organic and non-computer-generated or uniform feel. Especially in the *Inner World* sequences.

Given the high particle count I achieved in my stress tests, my experiments became more focused on creating density in the environment using randomly placed particles to simulate indoor dust. I created noise images as both color maps and depth maps for use with the *Tracer* application to create a random, volumetric, dust like shape in *Moviesandbox*.

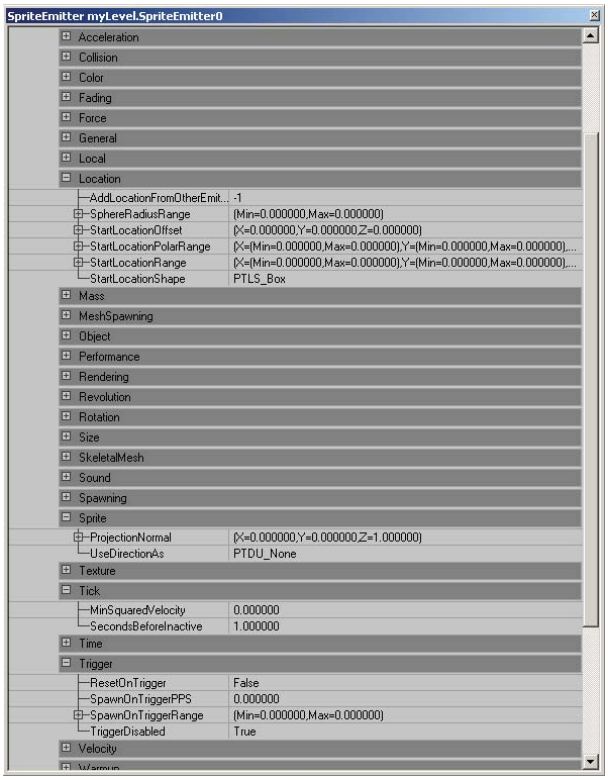


(Fig. 24 - uniform dust particles)

The volumetric shapes created the desired density in the test environments, but made them seem overly static. Knowing about the extensive amount of parameters of the UT2004 particle system[73], I started to experiment with different ways of animating the position of the particles over time, to become aware of possible visual bottlenecks and workflow issues.

I realised that the *Moviesandbox* GUI does not expose enough possibilities to quickly apply certain parameters to particle systems.

UT2004's particle system consists of a large amount of parameters to set up the development of a particle system over time. Only few of those options were exposed to the GUI, but exposing all of the parameters would clutter the GUI and would not help in fast prototyping during the production and improvisation process.

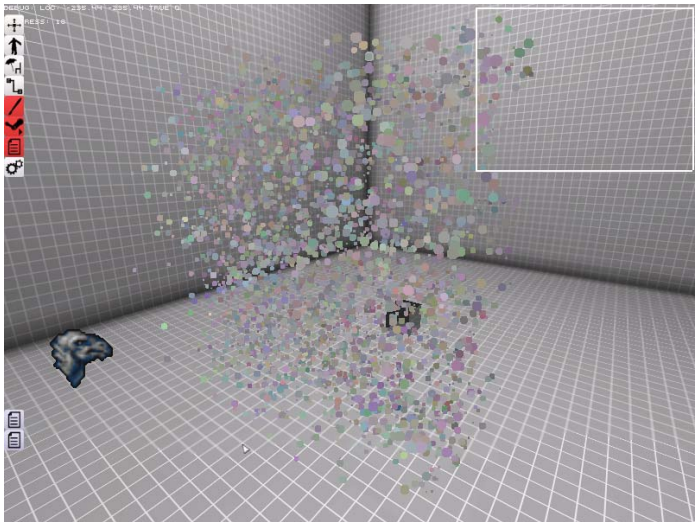


(Fig. 25 - particle system compexity)

I decided to implement a *Filters* tab that allows the placement of triggerable *Filter Objects* that execute UScript code when triggered, similar to a Node.
The difference is that filter objects also work when *Moviesandbox* is not in *Play Mode* and can thus be used to refine parameters of particle based objects.
These filter objects enable quick prototyping of particle effects and procedures very closely tied to this production.

5.2.3 The Road to Organic Pointilism

The particle shapes resulting from the stress test quickly became a conerstone of the visual art direction that we developed for the performance. The problem of uniformity was still prevalent though.
With the filter objects system in place, I created a randomizing filter, that randomly scales every particle individually in a *Moviesandbox* particle object. The parameters start from a user defined minimum fraction of the original size of the individual particle. Then they add a random value up to an also user defined maximum size.
I started to experiment with numbers.
By leaving the very first particle in every particle system unchanged, I could revert to the original uniform state by reapplying the size information of the unchanged particle. Both procedures can be applied using the same filter object. It has all necessary parameters exposed to the GUI and uses a simple boolean parameter to switch between randomized and uniform.


(Fig. 26 - non- uniform dust particles)

The desired, non-uniform effect was later not only applied to the dust shapes but to every single shape in the *Inner World* of the performance.

5.2.4 Simulating a lens with rendertargets

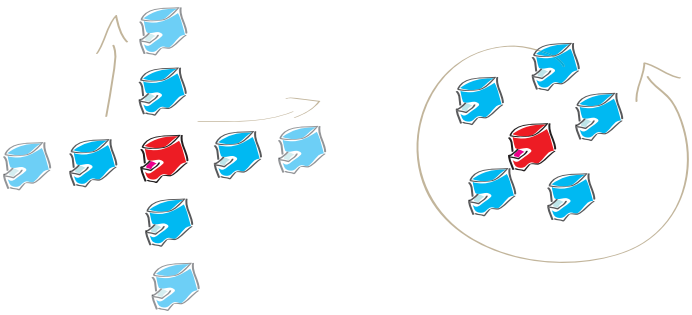
The depth of field simulation in *Moviesandbox* is achieved through a simple lens simulation. A user-definable amount of additional images are rendered

from slightly different Locations [65] but are still oriented towards the specified camera look target. They are then alpha blended over the actual camera position's view.

This generates a focal plane. Objects that are directly on the focal plane will be on the same position in all the rendered images. Objects off the focal plane will be shifted in every additional image. When all rendered images are combined, the slight shift in the different rendered images makes those objects look out-of focus.

(Fig. 27 - first depth of field implementation)

These additional images were first arranged in a shape around the camera's location that resembles a cross, resulting in unsatisfying visual results .



(Fig. 28 - depth of field implementation schematics)

Since depth of field was one of the visual elements I relied on heavily to create the aforementioned dimension and density, I decided to implement a different placement algorithm for determining the relative Location of the additional rendered images and the original camera position. I switched from a cross like arrangement to a circle like arrangement using simple sine and cosine functions to determine the relative position of each image, while leaving the maximum amount of additionally rendered images still user defineable.

Not only did the new placement algorithm enhance the visual results, it also required fewer additional renders to look satisfying, resulting in a performance gain.

(Fig. 28 - new depth of field implementation)

5.2.5 Adding the camera flickering effect

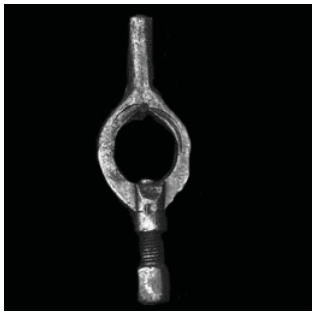
During the implementation process for the depth of field in *Moviesandbox*, I noticed a strange but appealing flickering effect that occured when the rendered images were not in sync with the overall frame update.

It introduced a seemingly organic layer of animation in

the contrast and exposure of the frames that reminded me of projection artefacts and old film material. This fit perfectly into the general mood of the *Inner World*.
I decided to try and reproduce it on purpose and implemented a flickering algorithm that would randomly skip the update function of the rendertargets, displaying a frame from the "past" and creating an overexposing artefact. I later added in a parameter with which I could adjust the amount of flickering. It represents the percentage of randomly missed updates on a scale from 0 to 1 and exposes it to the GUI as part of the *SetCam* Node settings.

5.2.6 The Machine Room

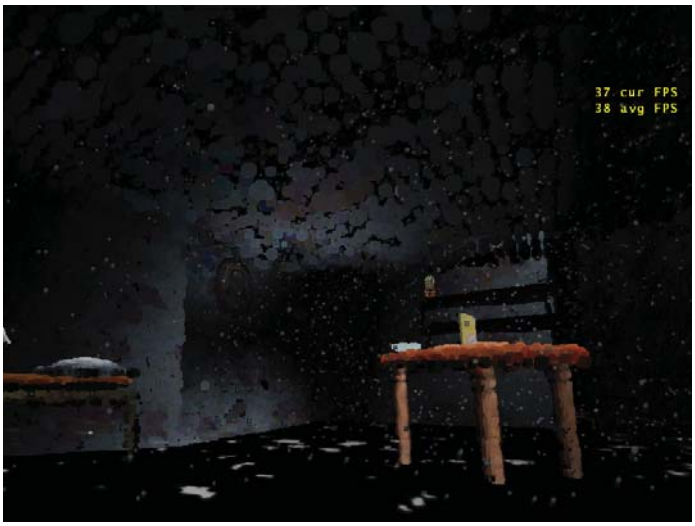
The "*Inner World*" very much evolved around the machine room. The initial idea for the machine was for it to look complex like a clockwork but at the same time big and dangerous.
Working closely with the DoP Bianca Bodmer,
I converted the pictures we produced in our first brainstorming for color palettes and shapes as machinery in the room but changed the size of the structures to create a surreal tone.



(Fig. 29/30 - machine room prop source and implementation)

5.2.7 The Living Room

The Living Room follows the basic guidelines of the Machine Room but is generally warmer and more playful. Only the most necessary pieces of furniture are introduced - a bed, a table and some shelves on the wall.
Knowing that it will be the first environment the audience would see, it contains some easily identifiable components of everyday life such as the cactus and the cereal box to introduce a bit of irony and lightheartedness to the performance and also emphasizes the fact that th character is tied to his environment. This does not break with the overall pointilistic and surreal style. To make them stand out and identifyable, these objects are not subject to the rooms lighting setup.



(Fig. 31 - the living room)

Building upon the concept of layering different aspects of real and virtual, a real life picture of the dancer hangs on the wall as though to suggest that the character actually is aware of his connection to the real world.

5.3 The Outer World

Establishing the contrast between the *Outer World* and the *Inner World* was achieved through moving away from the pointilism style and towards a comic-book like aesthetic that I already established in my former works. *Person2184* and *The Photographer* [66] both use strong black and white contrasts as backdrops and photorealistic advertisement images and video.

(Fig. 32 - still from Person2184)

The backdrops suggest densly populated living areas and forms of concrete to establish an unspecified urban environment.

The advertisement is the only naturalistic looking aspect of the whole performance and thus has a very prominent appeal. Given the abstracted and surreal nature of the other elements of the performance, the advertisement seems intrusive and uncanny.

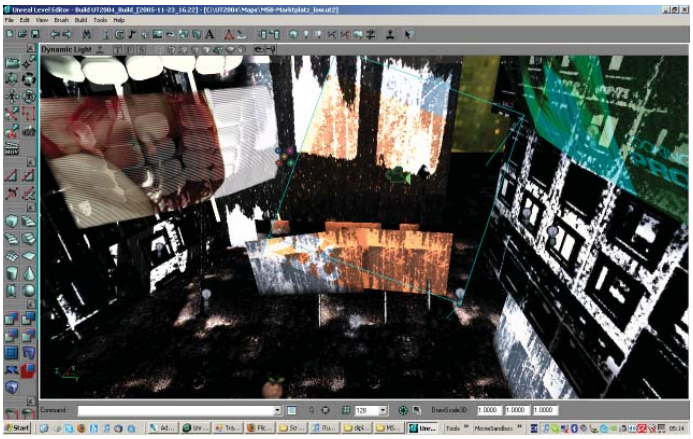
The Camera's exaggerated depth of field changes into an overlay that tones all the outside sequences in a bright and neon green to make them clearly and easily distinguishable. This emphasizes that the overall lighting situation seems created through advertisement billboards.


(Fig. 33 - outer world overlay)

The *Outer World* is constructed in UT2004's level editing program *UnrealEd* using textured planes and sprites. The textures are derived from modified photographs of physical places and buildings in Berlin,

Prague and New York.

Additionally, large scale textures are projected on top of the finished environment, in this case not to provide the lighting, but to add variety and color to the environmental texturing and help blend in the walls with the floor.



(Fig. 34 - UnrealEd and projected texture)

5.3.1 Managing the Masses

The *Outer World* has a lot of characters to manage.

I wanted to simulate a continuous stream of people that are moving through the scene and are not interactive. In order to save time during the setup of the scene, I implemented a simple system that makes it possible for characters to use another character's node tree as their own.

This is exposed to the GUI through the *RootNode* parameter "*myMaster*" pointing to the Character that the sequences should be adopted from.

This allows for a quick deployment of a large amount of characters that can all be driven through one simple script.

A problem I faced though, was the fact that characters get stuck, as Unreal's pathfinding does not take collision with other characters into account. It was thus necessary to implement a "switch" function for the characters' collision detection. Without collision detection, the characters would admittedly intersect, but any congestion in their movement paths would quickly resolve.

5.3.2 Creating the Overlay as an FX Lens

As opposed to the depth of field effect used for the *Inner World*, the *Outer World*'s overlay effect only requires one additional image to be rendered. The added image is rendered from the same position and orientation, but in a lower resolution. The resulting render is blurry in comparison to the original, due to the lowered resolution (in this case 1/4th the original resolution).

Using the overlay texture as a composite channel, the blurred textured is then added to the original rendered image. This adds visual complexity and keeps the illuson of real life camera optics by offering subtly blurred areas in the frame.

5.4 Character Creation

In order to bring characters from the *Tracer* application to *Moviesandbox*, I extended the *Tracer* application with a seperate color image, the so called *Bone Map*, to determine which parts of the character should be attached to which bone of the character template.

(Fig. 36 - the new Tracer application with Bone Map)

This was the last change that found its way to the 1380Beta release of *Moviesandbox* and already had the performance in mind.

The *Bone Map* uses one exact color to represent each bone. Each pixel of the color and depth maps can only be applied to one specific bone.

Early tests with this system revealed that due to the single-bone connection of the particles in contrast to the multi-bone weighted concept of the UT2004 vertex characters, a generic bone map will leave large gaps in areas of high movement such as the shoulders or the hip.

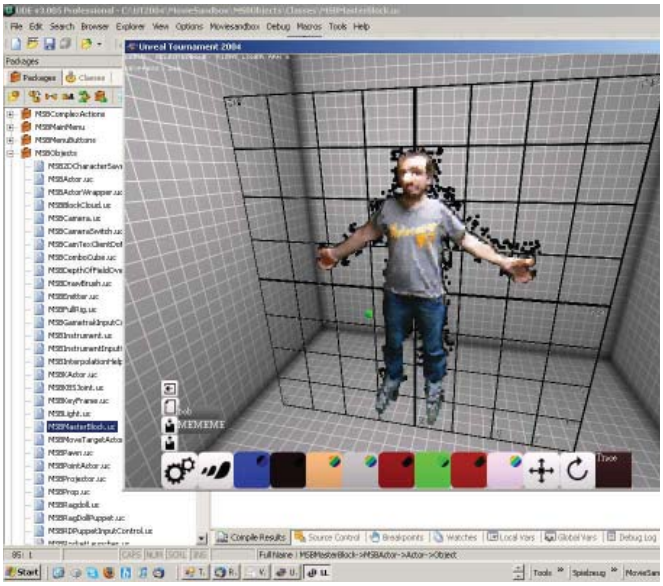
In order to simulate a more fluid distribution of the particles on the bones, small gradient areas can be introduced to the *Bone Map*.

(Fig. 37 - a Bone Map)

5.4.1 Looking into the Uncanny Valley

One of the tests for the character creation process application had me take a picture of myself and paint a rough depth map to see how realistic source material would look as a particle based character in *Moviesandbox*.

After a couple of tweaks to the depth map, the basic 3D image in the *Voxeleditor* looked surprisingly satisfying.



(Fig. 38 - me in Moviesandbox)

As soon as things started moving though, the visual appeal was gone.

The term *Uncanny Valley* was introduced as a mathematical relation between familiarity and human-likeness by Mashiro Mori in 1970 explaining that "as robots become more humanlike, their familiarity increases until we come to a valley. I call this relation the 'uncanny valley."[69] and has since then also been applied to CGI movies. Recently, even "Games have

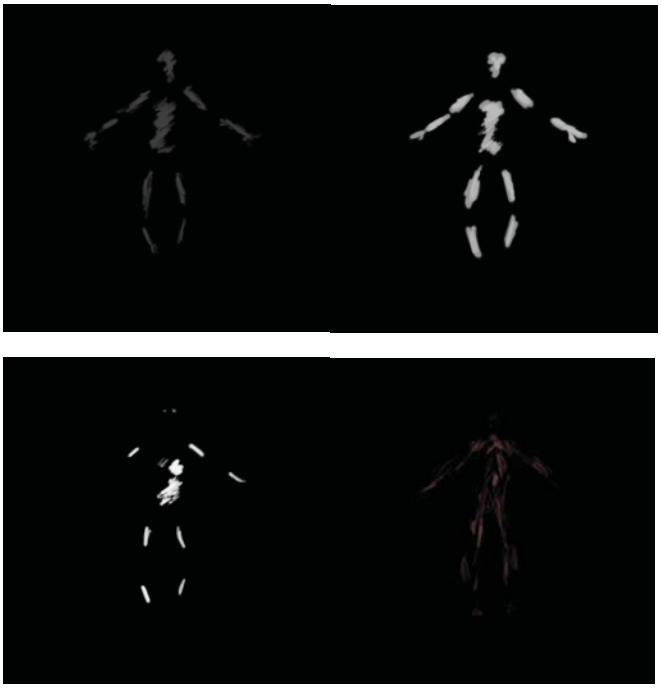
unexpectedly fallen into the Uncanny Valley"[70] .

I do not want to argue that the graphical fidelity of my virtual representation in roughly 8000 particles is on par with contemporary computer game visuals. I can report, however, that watching a representation of myself move as a character in *Moviesandbox* caused me as a healthy person to feel uneasy (Mori).

At that point, I lost all interest in creating an accurately human looking representation for either one of the characters in the performance.

5.4.2 The Inner Character

In order to fit in with the environment he lives in, the character of the *Inner World* is designed to blend into the background and is often only visible through his movement. His figure is drawn going from warm to cold colors from the inside to the outside but leaving a lot of open spaces to be able to look through him and inside him.



(Fig. 39 - layers of the Inner Character)

This also supports the contrast of him being a fragile character as supposed to the complex and dangerous machine he is operating.

5.4.3 The Outer Characters

To keep a certain amount of cohesion in the overall performance and to emphasize the organic forms of the characters, the outer characters are designed in the same pointilistic way than the "*Inner World*", but using much higher resolutions to give them more distinct forms.

Initially, there were two seperate *Outer World* characters - the dancer controlled puppet and the masses. The masses were first sketched as dark masses very similar to the opening scene of the aforementioned series *Serial Experiments Lain*.

During the production process, I decided to make the masses and the puppet become one and the same character, turning the puppet into someone more generic as it does not seperate itself from the rest of the outside world characters in shape or color.

Thus, the emphasize lies more on the movements and actions of the character and suggests them to be more universal and based on choice, rather than predetermination. It also makes the puppet more human and rooted in society, which is closer to the original intentions of the performance.

The puppet's coloring is largely chosen to suggest her as "blank" so that she does not represent anyone or anything in particular. Her figure though is directly taken from the performing dancer (Ivana Kalc) to add

to the direct virtual representation that the dancer establishes with the puppet.

5.4.5 Milkscanning

Milkscanning is a 3D scanning process that I developed to generate accurate depth maps for use with the *Tracer* application.

The name "milkscanning" derives from the initial process using milk as the contrast liquid.

The object to be scanned is put in a sufficiently large container which is then slowly filled with a liquid that creates a strong optical contrast with the object.

At periodical stages during the filling process, pictures are taken from an elevated position, preferably orthogonal to the liquid plane.

These pictures are then used to create the individual layers for the final depth map of the scanned object.

Finally, all the individual layers are shaded according to their depth and put together to create the final depth map.

In order to facilitate and automate the process to some extent, I created a small application that can be used to automatically generate the depth map with greatly reduced user input. The program is based on the managed DirectShow Library and uses a live camera input.

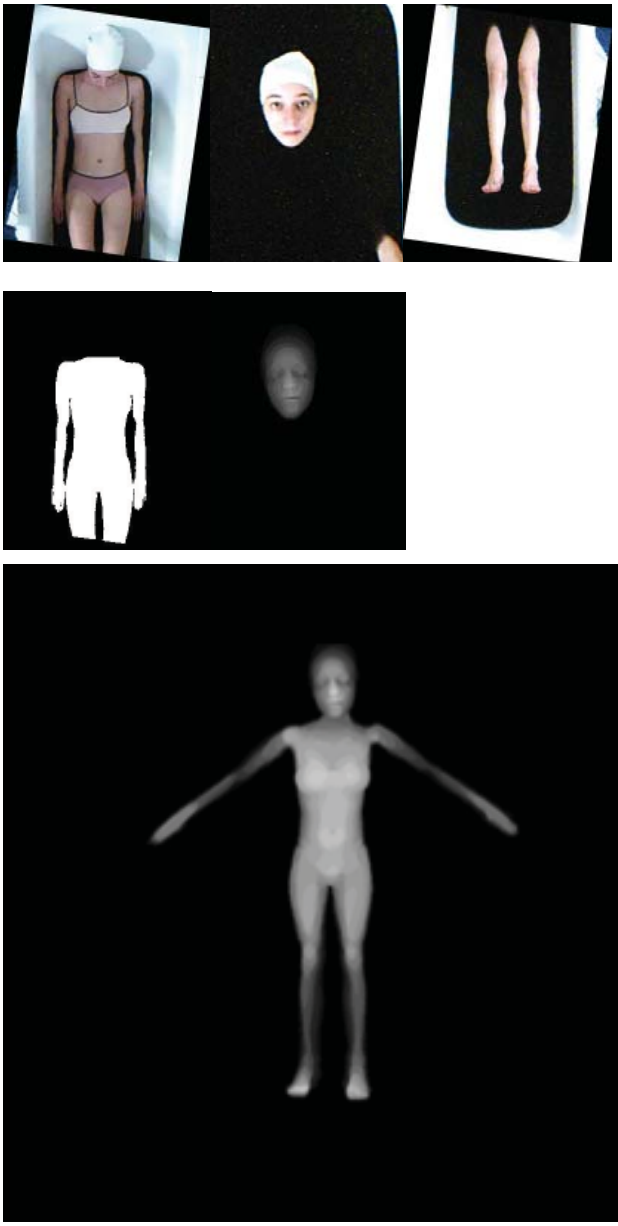
It visually marks contrast levels using a user definable threshhold number and automatically shades and stacks the individual depth layers.

5.4.6 Milkscanning in "A Puppet Play"

The only character in the piece that has an accurate human form is the O*uter World* puppet, used for the crowd, and the puppet controlled by the motion-capture jacket. To further emphasize the layering and connections between the real and the virtual in the narrative, I decided to base the puppet on the actual dancer that performs in the piece and thus use the milkscanning process to model the puppet after her figure.

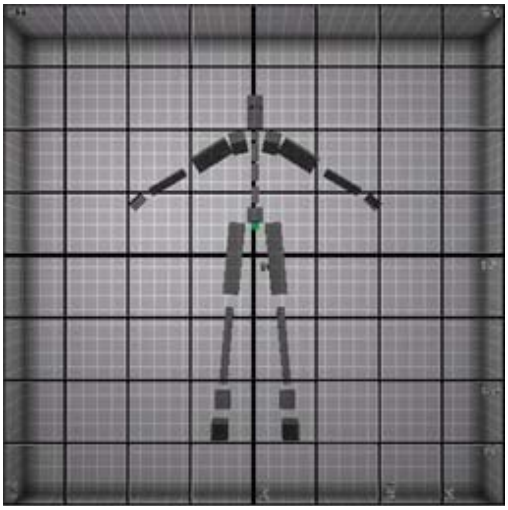
In order to generate the depth map for the outside world puppet, three different scanning passes are used. Since I was scanning in a human body, I chose ink as contrast liquid against skintone and light clothing.

The production did not have the necessary facilities to do the scan in one go (which means our bathtub was not big enough to completely fit one person lying flat). I decided to scan the upper body, the head and the legs in seperate passes and bring them together afterwards, only using the milkscanner application as a reference for the contrast values and assembling the final depth map in an image editing program.



(Fig. 40 - different stages of scanning Ivana's body)

The final composite of the depth map consists of the individually scanned parts and had to be altered to fit in the character template that *Moviesandbox* uses. Since the basic *Moviesandbox* character has its arms stretched out to a T-Pose and the body pass of my scan has the arms close to the body, they had to be rotated in the final composited depth map. Also, the size of the scan had to be altered to fit the template.



(Fig. 41 - the UT2004 character template)

5.4.7 Dissolving the inner world

The dissolving effect is a direct result of my experiments with the dust particles I created for the *Inner World*.

As mentioned before, the dust is subtly animated to make the environment look more dynamic. During experimenting with numbers, I found that the same effect can be applied to every object that consists of particles.

I implemented a hardcoded function that would test the visual impact of an effect like this on the overall environment and bound the function to a *KeyInput* using the *ConsoleCommand* Node.

The result completely restructured the performance, replacing the character-based structure of the narrative with a visual one, based on the dissolving effect.

(Fig. 42 - the dissolving)

5.5 Conclusion

The transition from vertex based computer game visuals to a more artistic and organic style was more of a conceptual task than a technical one. Today's computer game engines have all the technical prerequisites to offer a wide variety of art directions and visual styles, but use them in an all too traditional way.

A Puppet Play's distinct art direction consciously moved away from the traditional computer game look and feel, demonstrating alternatives not only to the naturalistic trend in computer game graphics, but also showing new useage scenarios for computer game engines that are closer to animated filmmaking and puppetry.

The different styles and cited motifs all played into creating different surreal environments that are linked not only to one another but also to the real world by visual clues just as much as through the controlling interfaces and the dancer on the stage.

6 Distribution

A Puppet Play will be distributed as a series of movie files including the project documentation and as a modification to the game UT2004.
The movie files will consist of the performance itself and all the explanatory videos found on the DVD accompanying this thesis and will be continuously updated when new documentation becomes available.

The modification will contain all the source material needed to modify and rearrange the performance to whatever the end user sees fit. It is included with the next release of *Moviesandbox* and will be fully documented including documentation on how to puppeteer and modify the performance to exclude puppeteering. This will allow people that do not have access to a *GameTrak* device to still take part in the modification process of the performance by rebuilding it using scripted sequences and custom animations.
Users that want to experience the modification will have to buy a copy of UT2004. Once they have UT2004 installed on their system, they can download the modification package from *www.moviesandbox.net* and follow the installation guide in the archive or on the website.

Users will have access not only to the complete system that was originally used for the performance, but also to the source files for all the characters and props and are free to use them in their own productions and modifications.
Since *Moviesandbox* is part of the distribution, all the tools that are needed to create new content or modify the existing are in the hands of the end user. Documentation on how to use these tools is available both in written text and video tutorials on the *Moviesandbox* website and will be continuously updated as new features become available.

Ideally, this will lead to sequels, adaptions and remixes of the performance by different people.
Furthermore, the complete UScript sourcecode for all the functionality created for *A Puppet Play* and also the complete UScript sourcecode of *Moviesandbox* and all the C# sources of its additional tools such as the *Tracer* and the *Multi HID Input* applications are available for download and use under the GPL or lesser GPL respectively.

The *Moviesandbox* Wiki currently provides a place for users to give feedback, ask questions and post their alterations to the performance and *Moviesandbox* to transform this project into an ongoing task.

7 Conclusions and future work

A Puppet Play demonstrates that a visually engaging virtual 3d-realtime puppet performance can be staged without spending large sums of money for proprietary equipment like motion capture systems, full body scanners or expensive computer animation software.

In this accompanying thesis, I explained the core mentalities and technological concepts that led to the performance, its narrative structure, its art direction and technical setup.

While this particular performance depends on the computer game UT2004 to work, its technical concepts and the algorithms described in this document can be applied to other computer game engines and virtual 3d-realtime systems in general.

The underlying philosophy for free content and free content creation tools will be pursued in a future project that aims to create an independent moviemaking solution implementing the ideas described in this text on the basis of the open source software library Open Frameworks[71], independent of any computer game system.

8 Credits

The production took place in the time between 28th of October and 24th of Novemeber 2007.
The performance premiered at the Festspielhaus Hellerau in Dresden, Germany as part of the CynetArt07_encounter festival for media art.

8.1 The Team

The people that came together to produce *A Puppet Play* consist of individuals that represent different artistic fields and practices, ranging from traditional filmmaking to contemporary dance.
It was important to me to have people on the team that would bring their unique perspective into the production process, often representing a different category of storytelling, and whose work i personally admire.

8.2 Acknowledgements

I would like to thank all the wonderful individuals and organisations whose support made this document and my ongoing work in this field possible:

My Examiners Joachim Schaugg and Michael Nitsche for their support and open-mindedness.
Paul Marino and the AMAS, Florian Berger, Andreas Jalsovec, Klaus Neumann, Alexander Scholz, Epic Games, Labor Interaktive Medien, Hochschule der Medien, Stuttgart, Ars Electronica Futurelab, Eyebeam Centre for Arts and Technology, LCC, Georgia Tech, Staatsministerium für Wissenschaft und Kunst, Sachsen, Trans Media Akademie Hellerau, Medienkulturzentrum Sachsen, Play08 Initiative fuer creative gaming.

I would like to thank my parents, brother and family for supporting me in my ideas, how weird they might seem.

And Hannah, for sheep.

Director of Photography and Live Editing:

Bianca Bodmer

Additional Puppet Performance:

Ivana Kalc

Music and Sound:

Sebastian Zangar

Wearable Interface:

Hannah Perner-Wilson, Mika Satomi

Director and Technical Development:

Friedrich Kirschner

performed on:

Shuttle XPC SG33G5

Intel Core2 Duo 3.0Ghz

4GB RAM

GeForce 8800GTX OC with 768MB RAM

Widows XP Professional with Service Pack 2

3x GameTrak controller

1x WiiMote

1x wearable MotionCapture device.

Appendix I

The original Script

The little man in the clock.

The little man in the clock gets up early. Every morning, after getting up, he stretches a bit, then slowly moves to the table, where a small bowl of cheerios is waiting for him. Half soaked in milk. He likes it that way. He finishes and usually drinks the last bits of milk from the bowl directly. He likes it that way, very much.
Walks to the board and makes a mark. Walks out of the bedroom, into the machineroom. A warm glowing from the kitchen, a shadow, maybe even moving. He barely notices.
It is warm in the machine room, misty, cloudy. His movement may be slow, but his hands are very precise.
He lowers the lever of the main gear. He looks up to the pressure indicator.
He slowly turns the knob for the steam release while watching.
Steps back a little. Waits a little. Looks to the handle on the big wheel to his left and starts to work it.
Then gets back and forth between the wheel and levers, in a strange rhythm
Later, he wipes his face with a piece of cloth.

Outside, a young, dirty looking girl tries to sell marionettes on the street. She's moving them around , a bit bored maybe. When noone watches, she kneels down and starts playing around with the puppet in her own way – warping it around and dancing – the strange rhythm comes back and finds its way into the movement of her puppet.

After a long and exhausting day, he lowers the lever again, then goes back to sleep.

The next day he slowly eats his cheerios and makes a mark on the board, just like the day before. He walks past the kitchen to the machine room. He lowers the lever of the main gear. He looks up to the pressure indicator.
He slowly turns the knob for the steam release while watching.
Steps back a little. Waits a little.

Something is different. The pressure indicator rises. The rhythm starts without him. He tries to move the wheel, tries to pull, push and work, but the machine is out of his control. The big wheel becomes imbalanced .
Noise, steam, the machine breaks under its own weight, the pressure finds its way through the tubes and suddenly.
Suddenly. A crack. A little one, a small one.
But with lots of light coming in and definitely big enough for him to pass through.

A sound from the kitchen. The shadow moves out of sight.

The little man awakes again. He was propelled to the far end of the machine room with rubbish and dust on top of him. His eyes can't focus on the beams of light projected through the crack. But it illuminates him. And it projects on him. The young, dirty girl is projected on the wall of the machine room. And her puppet still dances to the rhythm of the machine.

For a moment, that is.

Then she turns around and looks at him. He looks at the kitchen that is empty. He looks back at the crack. And then he waits a little. And then he decides to walk out.

It is a steep way down the clock and he falls half way out. The fall is long. The impact is hard.

It leaves him as a small puppet in front of the mechanical clock.

The dirty girl picks him up and puts strings on him. And finally, after dancing a little, he bows his thanks.

Appendix II

Interview with Bianca Bodmer, Director of Photography for A Puppet Play

Bianca Bodmer is a DoP who studied Camera at the Filmakademie Ludwigsburg and just finished her Diploma Project "Rounds", a 16mm live-action short movie directed by Stefan Buennig.

Did you work on 3D Animation before?

No, not really. I helped animation students with lighting issues and discussed lighting and camera concepts. I also sometimes use Frame Forge[1] as a previsualisation tool, but i never worked on a 3D project on my own.

What was the main difference between working on a real set and working in a virtual one?

The workflows are very different. The way I work in the physical world would not work here. In the virtual world you are completely open to set your cameras and lights wherever you want and you're not limited by impossble positions, like on real world sets. Especially in the lighting, you have complete freedom to get the characteristics you want.

How is the lighting different from working on a real set?

As I said, the workflow is very different.

In reality, I would cut my lamp with a flag or I would use filters and scrims to define it more precisely.

In *Moviesandbox*, I can just define my texture to be the light I want - it is much more intuitive. It also is a lot easier to set moving lights or special effects lights.

The *Moviesandbox* light works pretty much like a projector with the lighting images I create.

The results cannot be compared to real lighting, but I like that it is easier to control. Like, for example, the fact that you can define the objects that should be hit by the light and the ones that should not. Lighting in *Moviesandbox* is much more experimental, but in a good way. It is cool to experiment and find the effects you are looking for.

What about working with the virtual camera?

With the camera I feel that it is much closer to reality.It is very easy to find good positions and it gives you the freedom to go for the extreme angles, which is much easier to pull off in *Moviesandbox*.

On a real set though, if you want to make small adjustments to your camera angle you just do it. In *Moviesandbox* you have to define the Looktarget and set the camera to be controllable or fixed. It takes a lot longer to link up everything and get things organised, especially when you want to make minor adjustments.

The handcamera feature is an excellent way to create a more vivid camera though.

You have to make sure that you know where you're going before you start setting your cameras. It is pretty

1 (FrameForge 3D Studio 2003-2007

easy to get lost and set millions of different cameras and lights.

What aspect of the production was the most inspiring/ interesting for you?

I liked the combination of all the different layers of content coming together in the end - the environments, the puppeteering, using the *Milkscanner* for Ivanas character, the live-sound and Ivana's dancing.
From a technical point of view, I also really loved the hand-camera as you really had the feeling to have something in your hands and you can move it exactly the way you want it to.
The most inspiring part for me is the possibility to link input from all sorts of different actions to *Moviesandbox* in realtime. There are thousands of interesting possibilities to make something fresh and astonishing.

Do you see a future for this form of filmmaking?

Yes I do. I haven't been so much into Machinima and all, but this week of very intense work with *Moviesandbox* gave me an idea of the challenges and possibilities for both filmmakers and the audence. I could imagine doing similar stuff like *A Puppet Play* to show in theaters or as interactive installations open for everyone to try and play.

Appendix III

1 Chomsky, N., Media Control, 1997, MIT Press

2 Marino, P. 3D Game based Filmmaking: The Art of Machinima. Paraglyph Press, Scottsdale, AZ, 2004.

3 The ILL Clan. Larry and Lenny on the campaign trail [recording of live performance at Machinima filmfestival], 2003

4 Bong + Dern productions. This Spartan Life, Director: Chris Burke, 2004-2007 [video series]

5 Bungie Studios. Halo 2, 2004 [pc-game]

6 Rooster Teeth Productions. Red vs. Blue, Director: Burnie Burns, 2003-2007, http://www.redvsblue.com [video series]

7 Shea, Cam. Spider Man 3 Character Modelling Q&A, yahoo games, 27.Apr.07 [online], visited 21.Dec.07

8 Sadun Eric. iPhone Hacking 101: Jailbreaking, Aug 8th 07, [online] http://www.tuaw.com/2007/08/08/iphone-hacking-101-jailbreaking/, visited 20.Dec. 07

9 Kirschner Friedrich. Moviesandbox, - a machinima tool for the unreal engine, 2007, [online] http://moviesandbox.net, visited 20th Dec. 07

10 Epic Games. Unreal Tournament 2004, 2004 [pc game]

11 The ILL Clan. TrashTa1k with ILL W1ll, 2005 [video series]

12 GarageGames. Torque engine, 2000 - 2007 [pc-game]

13 Belkin International Inc. Nostromo SpeedPad, [online] http://catalog.belkin.com/IWCatProductPage.process?Product_Id=157024 visited 20. Dec. 2007

14 iD Software. Quake3, 1999 [pc-game]

15 Fountainhead Entertainment Inc. Machinimation, http://www.fountainheadent.com, 2004-2007 [pc-application]

16 Fountainhead/Ghost Robot. In The Waiting Line, Director: Tommy Palotta, 2004 [video]

17 Fountainhead Entertainment Inc. Anna, Director: Katherine Anna Kang, 2004 [video]

18 iD Software. Doom3, 2005 [pc-game]

19 iD Software, Loki Software. GtkRadiant [pc-application]

20 USB Implementers Forum, Inc. Device class definition for Human Interface Devices (HID), 1996-2001

21 J.Postel, User Datagram Protocol , August 1980

22 WiiLi Project, [online] http://www.wiili.org/index.php/Wiimote, visited 20.Dec. 07

23 Chung Lee, Johnny. Low-Cost Multi-point Interactive Whiteboards Using the Wiimote, 2007 [online] http://www.cs.cmu.edu/~johnny/projects/wii/, visited 20.Dec.07

24 Winder C., Dowlatabadi Z., Producing Animation, Focal Press, 2001, p 240ff

25 Apple Computer Inc. Shake, 2006 [pc application]

26 Adobe Systems Inc. After Effects CS3, 2007 [pc application]

27 Gosselin, David. Character Animation with Direct3D Vertex Shaders from: Shader X: Vertex and Pixel Shader Tips&Tricks, Wordware Publishing; Pap/Cdr edition (June 30, 2002)

28 Foley et. al, Computer Graphics Principles and Practices, Addison Wesley 1997, 21.3.7.ff

29 Gästrin Johan, Physically Based Character Simulation – Rag Doll Behaviour in Computer Games, Royal Institute of Technology, Sweden, 2004 [thesis]

30 Mathengine PLC. Karma Physics engine, 2003 [software library]

31 Preljocaj Angelin, Granular Synthesis. N/8, 2004 [ballet performance]

32 Hentschlaeger Kurt. KARMA, 2004-2006 [interactive installation for Ars Electronica Festival 2004]

33 Hentschlaeger Kurt. KARMA/live, Proceedings of the 7th international conference on New interfaces for musical expression, 2007

34 Chen, Tay et al. Marionette: From Traditional Manipulation to Robotic Manipulation, International Symposium on History of Machines and Mechanisms, 2004 p119-133

35 Wachowski, A. and L. Matrix Revolutions - Neo Realisms Featurette, 2003

36 as seen in UT2004

37 Rufus Rose. The Secrets of Making Marionettes, Popular Mechanics 9-1934

38 Moeslund, Granum. A Survey of Computer Vision-Based Human Motion Capture in Computer Vision and Image Understanding 81, Academic Press 2001, p232ff

39 Vlasic, Adelsberger et al. Practical Motion Capture in Everyday Surroundings, ACM Transactions on Graphics Vol.26/3, 2007

40 Moven, Xsens Motion Technologies. [Online] http://www.moven.com/Static/Documents/UserUpload/dl_20_leaflet_moven.pdf, visited 20.Dec.07

41 Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis, [online] http://www.arduino.cc/

42 Perner-Wilson, H. Satomi, M. Flexible Fabric touchpad, [online] http://www.instructables.com/id/Flexible-Fabric-Touch-Pad/ 2007, visited 21st Dec.07

43 Perner-Wilson, H. Puppeteer - a wearable costume for motion-capture, 2007, [online] http://massage-me.at/puppeteer/index.php, visited 21.Dec.07

44 C.Reas, B.Fry, Processing, 2001, [online] http://www.processing.org, , visited 21.Dec.07

45 Kirschner, F. Satomi M. data flow diagram, 2007, [online] http://zeitbrand.de/mediawiki/index.php?title=PuppetPlay, visited 21.Dec.07

46 Adorno, T. W. Culture industry reconsidered, New German critique, 6, 1975, 12-19

47 Moppi Productions, Halla, 2002 [video]

48 original title: “Sungnyangpali sonyeoui jaerim”, 2002, Director: Sung-Woo Jang [video]

49 Metropolis, Director: Fritz Lang, 1927 [video]

50 Wood, Brian. DEMO, AiT/PlanetLair, 2005

51 Wood, Brian. Channel Zero, AiT/PlanetLair, 2000

52 Field, Syd. *Screenplay: The Foundations Of Screenwriting*, 2005

53 Blade Runner. Director: Ridley Scott, Production Design: Syd Mead, 1982 [video]

54 Serial experiments Lain, Director: Ryutaro Nakamura, 1998 [video]

55 Being John Malkovich, Director: Spike Jonze, 1999 [video]

56 Peter & the Wolf, Director: Suzie Templeton, 2006 [video]

57 Overtime, Directors: Oury Atlan, Thibaut Berland, Damien Ferrie, 2004 [video]

58 Street Fighter IV Trailer, Capcom, 2007

59 Latta, L. Everything about Particle Effects, Game Developers Conference 2007, Tutorial Proceedings

60 Foley et. al, Computer Graphics Principles and Practices, Addison Wesley 1997, p752-753

61 Epic Games, Unreal Skeletal System, [online], http://udn.epicgames.com/Two/SkeletalSetup.html, visited 21.Dec.07

62 Epic Games, Projectors Table of Contents, [online] http://udn.epicgames.com/Two/ProjectorsTableOfContents.html

63 Segal et al. Fast Shadows and Lighting Effects Using Texture Mapping in Computer Graphics 26, July 1992

64 Foley et. al, Computer Graphics Principles and Practices, Addison Wesley 1997, p1031-1039

65 Riguer, G. Tatarchuk, N. Isidoro, J. Real-Time Depth of Field Simulation from ShaderX2 - Programming Tips and Tricks with DirectX9, Wordware Publishing, Inc., 2003

66 Kirschner F., Person2184/The Photographer, 2005 [video]

67 http://skincity.beyondunreal.com [community resource]

68 http://planetunreal.gamespy.com [community resource]

69 Mori, Masahiro. The Uncanny Valley in Energy, 7(4),p33, 1970 translated by Karl F.MacDorman and Takashi Minato

70 Thompson, Clive. The Undead Zone in Slate, June 9th, 2004, [online] http://www.slate.com/id/2102086 visited 20.Dec.07

71 Lieberman, Watson, http://www.openframeworks.cc

72 Nitsche, Michael. American Film Institute goes Machinima, 01.Nov.07, [online] http://gtmachinimablog.lcc.gatech.edu/?p=58, visited 21.Dec.07

73 Epic Games, Example Particle System, [online], http://udn.epicgames.com/Two/ExampleParticleSystems.html, visited 21.Dec.07