Curriculum Vitae

Name:	Walter
First name(s):	**Jan** Douglas Bert
Birthplace:	Munich, Germany
Date of birth:	01/17/1967
Address:	Baerwaldstr. 17, D-10961 Berlin, Germany
Cell Phone:	+49-151-70152344
Citizenship:	German
Languages:	German, English
Certification:	Dipl. Inf.

Education

From 10/08/1992 to 10/24/1996	Technische Universität Berlin
From 10/14/1988 to 09/30/1992	Friedrich–Alexander–Universität Erlangen–Nürnberg

Employment

11/01/2011 – 09/30/2020	The Mill, London, UK
01/01/2009 – 10/31/2011	mental images, Berlin, Germany
09/25/2006 – 12/31/2008	mental images, Marina del Rey, CA, USA
08/27/2005 – 09/20/2006	Digital Domain, Venice, CA, USA
2003 – 2005	Filmakademie, Ludwigsburg
12/29/2002 – 08/26/2005	The Mill, London, UK
10/08/2001 – 12/28/2002	Mill Film, London, UK
05/29/2001 – 10/07/2001	Moving Picture Company, London, UK
07/07/2000 – 04/05/2001	Not a Number, Amsterdam
11/01/1997 – 06/30/2000	Q–bus Mediatektur GmbH, Berlin
09/09/1996 – 10/31/1997	Artemedia productions GmbH, Berlin
11/01/1995 – 09/08/1996	Fraunhofer Institute, Berlin Production Systems and Design Technology
03/31/1993 – 10/31/1995	CAS Peter Klose GbR, Berlin CAD Animation Software

The Mill

In 2011 I helped with the transition from The Mill's render pipeline away from mental ray towards Arnold. Back then Softimage XSI was still widely used, so we had to support, MtoA (for Maya to Arnold), SItoA (for XSI), and Frederic Servant (at that time a colleague at The Mill) developed HtoA (for Houdini). Later Frederic left The Mill to work full time on HtoA before Autodesk acquired Solid Angle and he became the Arnold Development Manager. I took care of XSI and Maya and compiled all available 3rd party Arnold shaders. In contrast to mental ray a lot of the shaders were developed open source (or it took not too much convincing to get access to the source code). Solid Angle also granted access to all their DCC plugins, which helped finding bugs and also allowed to slightly modify the plugins to meet local pipeline needs.

Later XSI was slowly retired and we used more and more Houdini and Maya. For HtoA we decided to use the plugin as it came from Solid Angle/Autodesk, for Maya we kept some local adjustments, so that I was mainly maintaining the MtoA (Maya to Arnold) pipeline and slowly reduced the usage of 3rd party shaders, which either got abandoned over time or replaced by shaders which shipped with Arnold.

Some side projects included HLSL shaders for Flame (e.g. some denoising shaders) or the investigation of the Halide programming language for e.g. the Kuwahara filter, which we had already for Nuke.

The biggest side project became for me learning the Rust programming language for highly concurrent and highly safe programming. And is there a better way to learn such a language and at the same time learn more about rendering algorithms than translating some already existing (and well documented) C++ code into a new language? Originally I just wanted to implement bits and pieces, but I ended up converting the entire C++ code base to Rust, which was about 100.000 lines of code. For most example scenes the resulting images matched the C++ counter part 100%. The source code is available on GitHub and Codeberg.

mental images

During my time in Los Angeles I was responsible to help customers with their current movie productions. I was, for example, working several weeks on site for Digital Domain during Speedracer, while being payed by mental images. I helped them with basic mental ray shader development (there was a library with several hundred, very basic, shaders which could be connected for more complex tasks) and e.g. with a method to do motion vector based motion blur for shadows (they had already shaders for the motion blur of objects based on motion vectors).

Beside this production support tasks I was mainly hired to convert some of

mental images' most complex shaders (the architectural shaders) from a C/C++ implementation to a new shader language called **MetaSL**. While I was working on converting more C/C++ shaders to MetaSL for Autodesk and their 3DS Max application, I did an internal transfer to Berlin, Germany, to be closer to the developers of mental ray and the new MetaSL language.

In January 2010 I had finished the MetaSL conversion projects and moved to a new group called CCG[1] where we basically took content (3DS Max and Maya scenes) from customers and helped them to prepare those scenes to be used with mental images' product **RealityServer**, where you can interactively navigate through a scene while it's rendered in real–time. One of the renderers you can use is called **iray**, which allows you to use global illumination in conjunction with a daylight system, while still having interactive frame rates during navigation. This renderer is using heavily parallel programming provided by **Nvidia**'s **CUDA** technology[2]. For internal usage I developed several exporters for DCC applications like Maya, 3DS Max, and Blender, which allowed me to export the scenes in a way that we could use them directly with **RealityServer**. It is planned to release those exporters to some customers in the future. Therefore I wrote a specification how we would like MI files to be structured for the usage with **RealityServer** and **RealityPlayer** and modified the already existing exporter plug–ins to follow this new guidelines.

Digital Domain

Movies

- Flags of Our Fathers
- Zoom
- Speedracer

Technology

- Data exchange between Houdini, Maya, Lightwave, and 3DS Max
- Mental ray pipeline for Zoom (re-creating whole HyperGraphs outside of Maya while overwriting certain shader parameters on a per object basis)
- Integrating Python into a Houdini ROP via Boost
- Various importers/exporters from/to several file formats

[1] Content Creation Group

[2] Nvidia acquired mental images in 2007.

- Python GUI communicates via port with Houdini/Maya and via sockets with Asset Management System/Database

The Mill

Commercials/Video Clips:

- Mercedes — Movement
- Radiohead — Go to Sleep
- Levi's — Train
- Playstation — Mountain
- Nike — The Other Game
- Dyson — Motion
- Pontiac — New Worlds

Technology

- Mental ray output shader
- CFD integration, mental ray volume shader (raymarcher)
- Mesh reduction algorithms, massive crowd system
- RenderMan shaders, massive integration
- XtoR — RenderMan exporter for XSI (RenderMan and mental ray shader library)
- Mental Ray geometry shaders
- XSI and Maya plugins
- Voxelizer
- Real–time lighting in Houdini's compositing system
- Maxwell rendering tests
- Teaching shader writing and Houdini at the Filmakademie Ludwigsburg

Mill Film

Movies

- Harry Potter and the Chamber of Secrets
- Black Hawk Down

Technology

- Prototype for walking spiders
- Mental ray volume shader (raymarcher)
- Maya to mental ray
- Maya to RenderMan
- Solving motion blur and eyesplit problems (PRMan)
- Jig, Air (occlusion), Radiance, baking radiosity
- HDK (voxel field to I3D)

Moving Picture Company

Movies

- Harry Potter and the Sorcerer's Stone

Technology

- Lighting tools
- Using Alfred dependencies on renderfarm (Maya/PRMan/Shake/Maya)
- Deep shadow (MPC's — not Pixar's) integration
- RenderMan and Maya communicate via sockets

Not a Number

Today Blender is an open–source animation and rendering system but there was a time when Not a Number distributed the software for free and tried to be a commercial company at the same time. The software is multi–platform and can be used for three purposes:

- Animation and rendering
- Game Engine
- Compositing

I was responsible for the programming API of a language called **Python** which was used for the modeling and animation part[3] as well as for the game engine. I wrote import and export scripts for file formats like OpenInventor, VRML 2.0, RenderMan, Povray, Radiance, Panorama, Lightflow Rendering Tools, Lightwave, Wavefront Object Files (OBJ), and 3DS.

Q–bus

Most of the time at Q–bus I worked for a project called VIN for the International Net Management Center (INMC) of Deutsche Telekom in Frankfurt. The heart of this system was a huge wall (72 square meter) consisting of 96 screens (each with a resolution of 800x600 pixels). The screens were in a 16x6 arrangement and therefore the wall had a 12800x4800 resolution which is about 46 Million pixels. Each updated in realtime by an Onyx 2 machine from SGI. The wall was mainly used for communication purposes between different people watching the international net infrastructure of Deutsche Telekom. There were a lot of net management tools on various platforms to integrate.

Other projects I was involved in include more computer graphics related problems like mixing a real person into a virtual environment in realtime. We installed a bluescreening room with two cameras at CeBIT 1998 and mixed a TV presenter with the inside of a virtual house with three floors. We had to decide in realtime if the person is in front or inside an elevator and mix the camera picture with the virtual stuff. The resulting movie was shown on a big wall which was raised at the end of the show to present more details of the technology being used to the audience.[4]

[3] We called this the **Creator**.

[4] The technical stuff and the bluescreen room was indeed behind the wall.

For CeBIT 1999 we used a slanting glass with some special layers to project a picture from behind. It was not obvious where the picture came from. A moderator was holding a new 3D input device consisting of a sphere. The movement of the sphere was tracked in 3D and used for interaction with a kind of 3D PowerPoint presentation. All pictures were calculated in realtime and the moderator could present the content in any order, stop whenever he needs to explain some things in more detail or skip some details. The presentation allowed to mix the presentation with video material and animated the "flying in" of new content.

The last project I was involved was the Expo 2000 job for Deutsche Telekom and their T–Digit. Inside the T–Digit there were several floors showing new techniques and one floor was designed by Q–Bus.

Up to twelve people were able to sit in a half circle wearing stereo glasses from Sony and to look into a virtual scene. Two virtual and one human presenters were explaining the future of communication. The twelve people all had a different view in the same scene and their head movement was tracked so that they had the feeling of being part of the virtual scene. The human presenter and live TV programs were mixed with the virtual environment and gave a much more immersive feeling.

Artemedia

For Artemedia we implemented a virtual flight through Berlin. The software was installed in a so–called InfoBox at Potsdamer Platz in Berlin, Germany. The tourists were allowed to use a joystick to fly (or walk) through the virtual Berlin before the actual Potsdamer Platz was rebuilt. The Sony Center wasn't finished at that time but you were able to see how it would look like. We implemented a new collision system and moving trains were part of an animated scene. There were automatically BSplines built from any point you navigated to some sightseeing points like the Brandenburger Tor. This means you can fly through the city and when you feel lost you can select a place you want to visit from a touch panel on the joystick and you automatically fly there on a smooth path. The German Reichstag was shown with the new dome before it was begun to build up.

Fraunhofer

Fraunhofer is a research center which works closely together with universities like the Technical University Berlin. The ESPRIT project I was working for was called CAESAR. It was about semi–automatic repair of free form surfaces built up from NURBS. In contrast to pure computer graphics applications you have to handle topology and know about neighboring surfaces. Sometimes surface–surface inter-

sections have to be calculated and the user has to provide the information which surface part to drop and which part to keep.

We had industrial partners like DASA (German Aerospace) and British Airways (UK) and cooperated with the University of Swansea (UK).

CAS

CAS was the first company I was working for as a computer scientist. During my studies I focused on NURBS curves and surfaces. I just was beginning to understand the basic concepts but my former boss convinced me to invest into a working prototype of a NURBS based modeler. 1994 we showed an early version at CeBIT and we had a plugin for 3D Studio (at that time DOS based). In fact it was a Bézier patch modeler but the source code was already prepared for NURBS. Later I worked as a freelancer for the same company and implemented a NURBS library with surfaces of revolution, interpolation, sweeping and other useful functions. Unfortunately the NURBS products were never commercially successful.