Boletim Técnico da Escola Politécnica da USP
Departamento de Engenharia de Sistemas Eletrônicos

ISSN 1517-3542

BT/PSI/0011

InterFace: A Real Time Facial Animation System

José Daniel Ramos Wey
Marcelo Knorich Zuffo

São Paulo – 2000

O presente trabalho é um resumo da dissertação de mestrado apresentada por José Daniel Ramos Wey sob orientação do Prof. Dr. Marcelo Knorich Zuffo.: "InterFace: A Real Time Facial Animation System", defendida em 16/12/99, na Escola Politécnica.

A íntegra da dissertação encontra-se à disposição com o autor e na Biblioteca de Engenharia de Eletricidade da Escola Politécnica/USP.

FICHA CATALOGRÁFICA

Wey, José Daniel Ramos
InterFace : a real time facial animation system / J.D.R. Wey, M.K. Zuffo. -- São Paulo : EPUSP, 2000.
7 p. -- (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia de Sistemas Eletrônicos, BT/PSI/0011)

1. Computação gráfica 2. Operação em tempo real (Computadores) 3. Animação por computador I. Zuffo, Marcelo Knorich II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos III. Título IV. Série

ISSN 1517-3542

CDD 006.6
004.33
778.5

InterFace: a Real Time Facial Animation System

JOSÉ DANIEL RAMOS WEY

LSI - Laboratório de Sistemas Integráveis
Escola Politecnica de Universidade de São Paulo
Av. Prof. Luciano Gualberto, 158 trav 3 Cidade Universitária, SP, Brazil
wey@lsi.usp.br

Abstract. This paper describes **InterFace**, a real-time facial animation system. The system uses a simple set of pre-modeled facial expressions to create a wide range of emotions, mouth positions and complete head movements. The animation is done through layered groups of actions that combined gives the virtual actor complete freedom to perform different actions independently and simultaneously, without the intervention of the animator. Due to its software structure, InterFace can be used as a "render plug-in" for an artificial intelligence program or a script-based animation system. The system was successfully implemented using the languages Java and VRML, and can be executed through the Internet in PCs or workstations with a Java-enable Web browser. Applications that can benefit from this system are interface agents, virtual reality systems and animation software, among others.

Keywords: Computer Graphics, Real Time, Facial Animation, Facial Expressions, Morphing, Virtual Actors.

1. Introduction

Computer generated human characters is one of the most important areas in Computer Graphics lately. Agents or avatars are present in multimedia titles, graphic user interfaces, tele-presence and shared virtual worlds systems. While these agents can represent a breakthrough in human-computer interaction, it is not easy to create and animate believable human characters. When we are dealing with computer-generated agents, we turn on our human responses. We expect them to act like humans, moving the lips when they are talking, expressing emotions, having a personality. We expect human behavior from a computer program.

A powerful method for adding a human personality in a computer generated character is facial animation. Psychological researchs show that humans express most of the communication and the emotions by the face [K. Mehrabian, J. Ferris (1967)]. However, creating believable human faces in a computer is not an easy task. We have a remarkable ability to recognize one face between thousands of similar faces, and we can detect very subtle changes in facial expressions [F. Parke, K. Waters (1996)]. It's correct to say that all humans are specialists in human faces.

The construction and animation of human facial models in computers is not a new area. Since Parke's first studies [F. Parke (1972)], several techniques have been used to add realism to 3D computer human faces. However, animating a face convincingly in real-time is still a problem today.

The most used techniques to animate human faces are morphing between fixed and variable polygon topology [F. Parke (1972)][F. Parke (1974)][N. M. Thalmann et al. (1989)] and muscles simulations [K. Waters (1987)][C. Wang et al. (1994)]. While muscle simulations can give good results, it is computationally expensive and difficult to implement. Morphing between faces are easy to implement and computationally cheap, but usually requires the animator to model all the facial expressions and mouth positions before hand.

We propose a facial animation system that allows easy creation and animation of facial expressions. It is simple to implement and not too expensive computationally, being adequate for complete human simulations (with body movements, behavior, and so on). Facial expressions are created interactively, based on a simple set of pre-modeled expressions that when combined can lead to a wide range of facial emotions and mouth positions. The animation system uses a layered approach, making it easy to combine different actions, for instance to say a phrase while the character is angry. Actions in different layers are composed using techniques similar to image composition.

2. System Overview

To develop a facial animation system, we have to consider at least two aspects: modeling and animation.

The model is the representation of the head. It can be done in different ways: modeling from photographs, 3D scanning or modeling using special software. Models can use different representations, like polygon mesh and

B-splines. The animation can also be done in several different ways, like morphing geometry between different pre-modeled faces, simulating the muscles of the face, or using sensors or trackers.

Because of the real-time nature of the InterFace system, we use polygon-based models and geometric morphing between faces for animation.

The InterFace system has two modules: the expression modeler and the animation module.

We define an *expression* as any change in the face geometry. Simpler expressions can be combined visually and interactively to create more complex expressions.

The animation is done by composing groups of actions. An action is a script of animation the virtual actor can perform, like crying, talking, blinking or moving the eyes. Actions of the same kind are joined together in groups. Each group of actions has a transparence value to be used when the actions are composed, in a way similar to image processing algorithms.

The InterFace system is based on a previous work done in the Improv System [K. Perlin, A. Goldberg (1996)] and the Aria project [R. Ruschionni et al. (1997)]. It is implemented using VRML [VRML (1997)] for the description of the scene and basic expressions, and Java [Java (1995)] for the animation and modeling modules. These languages were chosen because they are system independent and have a good performance. The InterFace system can be accessed on-line, in the URL **http://www.lsi.usp.br/~wey/interface/**. The system requires a Java compatible browser with a VRML 2.0 plug-in. The platform used for development was a Pentium PC with Windows 95 and Netscape Communicator.

3. Modeling facial expressions

The process of animating a face usually requires modeling all facial expressions and mouth positions by hand. When the scene is long or complex, this approach can easily become impractical - the number of expressions to be modeled is too large.

To solve this problem, Frederick Parke [F. Parke (1974)] used parametric models of the face. By controlling a small set of parameters (like jaw rotation, eyelid opening and mouth positions) it is possible to create different facial expressions easily. Each parameter controls a region of the face, allowing transformations like rotation, scaling, vertices position offsets and interpolation.

The InterFace system uses the same approach, but in a different way. Complex expressions are modeled combining a small set of pre-modeled expressions, called the *Basic Library of Expressions* (BLE) - it is very similar to what Frederick Parke called "parameters". This specific set of BLE was based on the research done by Ken Perlin [K. Perlin (1997)]. We adapted the BLE defined by Perlin to 3D, since Perlin's work was done in a 2D character. Of course, the system is not limited to the BLE. We can use other expressions if necessary. The BLE is just a good set of expressions for creating emotions and mouth positions.

The BLE is composed by the following expressions: eyes move up, eyes move left, rotation of the left eyebrow, rotation of the right eyebrow, blink left, blink right, lower eyelid move up left, lower eyelid move up right, sneer left, sneer right, smile left, smile right, mouth position like "ahh", mouth position like "ohh", head rotation in Z, Y and X axis. A screenshot of the InterFace expression modeler is shown in figure 1.

Each expression in the BLE is stored by the system as differences from the original face of the character (called the canonical expression). These differences can be vertices offset or rotation, translation and scaling of objects. For instance, given the canonical face and the face with the mouth like "ahh", the system will calculate which vertices moved (in this case, only the vertices from the mouth) and store these differences as an array of vertices offsets and vertices indexes. The vertices that do not move are not stored.

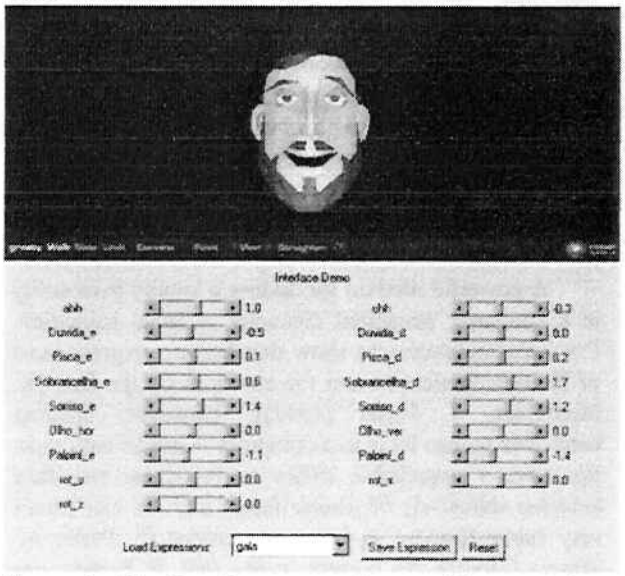

Figure 1: The InterFace expression modeling module. Each slider correspond to an expression from the basic library

The system modeler module presents 17 sliders, each one representing the weight of each expression of the basic library. While Parke's implementation used a value from 0 to 1 in each parameter, our system uses a

different approach: each expression of the BLE has a weight on the model. Each weight can range from -∞ to +∞, although the useful values almost never exceed -5 to +5. Negative weights create the oposite effect of positive weights: if I use a value of -1 in the expression EYES_RIGHT, the actor will look to the left.

The final expression is composed by simply adding the differences stored for each expression in the basic library, multiplied by the respective weights. The order that the weighted expressions are added makes no difference in the final expression. Figure 2 shows an example of the result of adding three expressions.

Although the BLE is relatively small, we were able to model several different and interesting expressions. Some of them are shown in the figure 3.

We also modeled a small set of mouth positions that can be used efficiently for synchronizing the lip movements and the voice, as described in [P. Blair (1989)] and [B. Robertson (1997)]. It is a set of eight mouth positions that mimics most of the phonemes of the English language, creating the illusion that the character is actually speaking. This set has been used for traditional cartoon animation for many years.

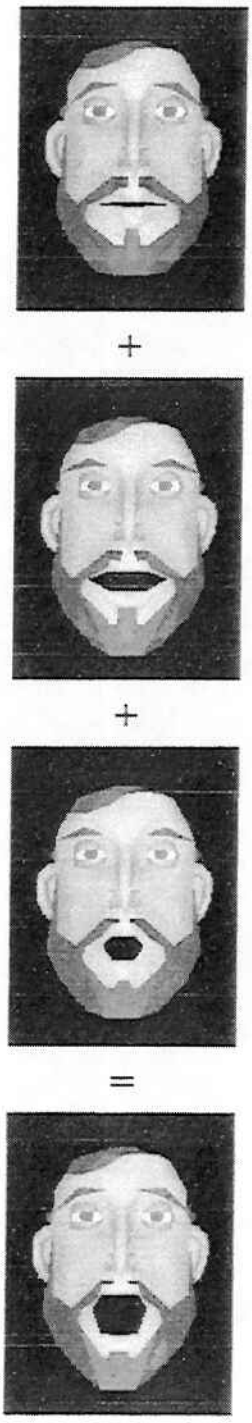

Figure 2: the result of adding three expressions

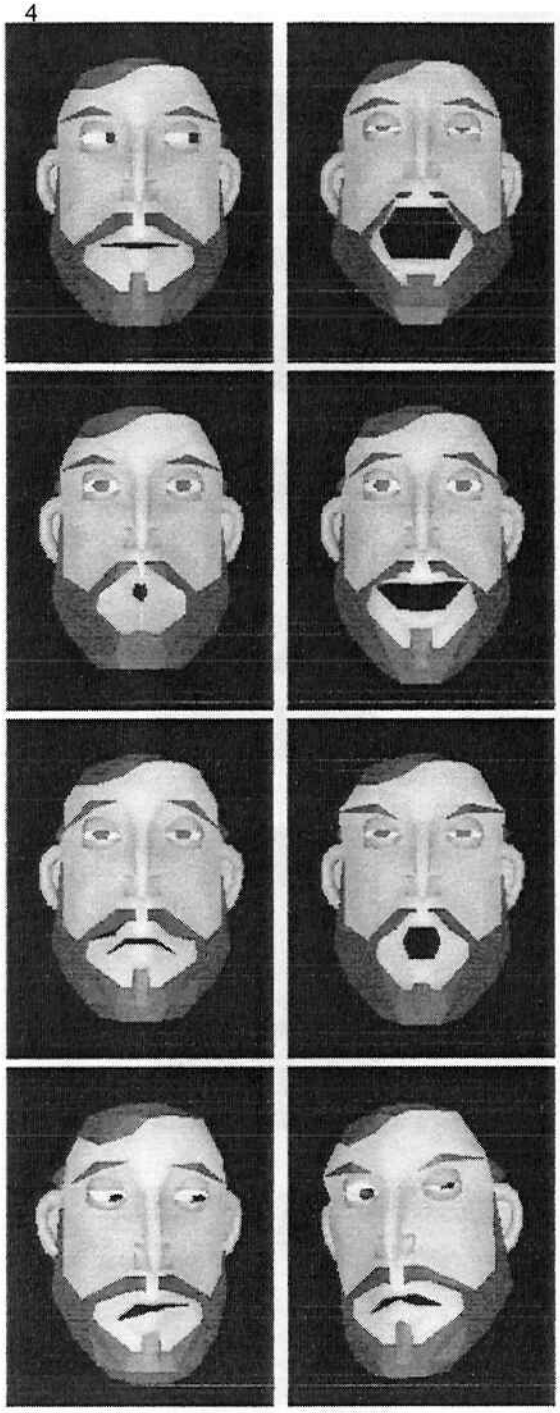

Figure3: Some expressions modeled using the BLE

All expressions created in this module are stored as arrays of weights of BLE. The last expression shown in the image above is described by the following array of BLE weights: [0.2 -0.3 3.8 4.1 -0.3 0.4 -0.4 -0.7 0.1 0.6 0.6 0.2 -0.4 0.6 -0.2 0.1 0.1].

4. Animating the face

Once we have some expressions modeled in the module described above, it is time to animate our character. As stated before, the InterFace system is composed of two modules: the expression modeler and the animation engine.

The animation engine is based on the concept of layers of actions, developed by Ken Perlin in his recent work on virtual actors [K. Perlin (1994)][K. Perlin (1995)][K. Perlin (1996)]. We adapted the concept of layers of actions to solve the specific problem of facial animation.

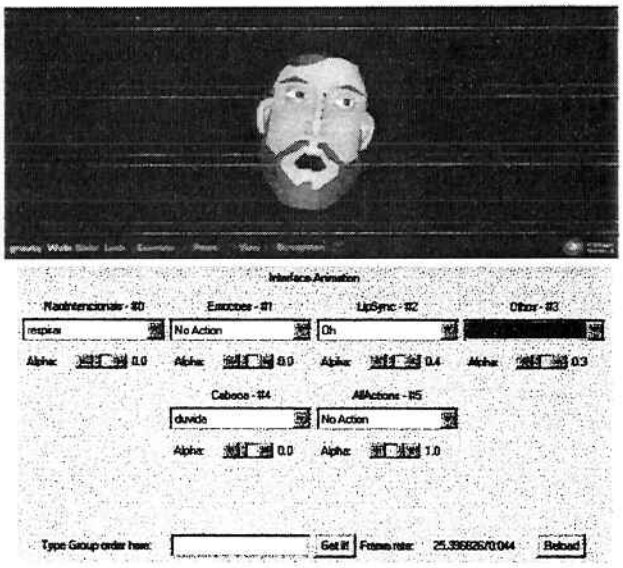

Figure 4: The Interface Animation module

4.1 Actions

The basic element of the animation system is the Action. An action is a script of animation the virtual actor can perform. Examples of actions are talk, look at something, breath, sneeze, say a phrase, smile and so on.

Actions are based on the expressions created in the expression modeler. What an action do is modify the intensity of expressions with the time, using mathematical functions (called *curves* in our system to be more intuitive for animators).

The curves available in the InterFace system are: constant, linear interpolation, B-splines, sine, cosine, non-predictable impulse and Perlin Noise [K. Perlin (1985)]. Since Java is an object-oriented language, it is very easy to add more mathematical functions as needed – just create another class inherited from the *curve* abstract class that implements the function.

The expression curve is involved by an envelope as shown below:

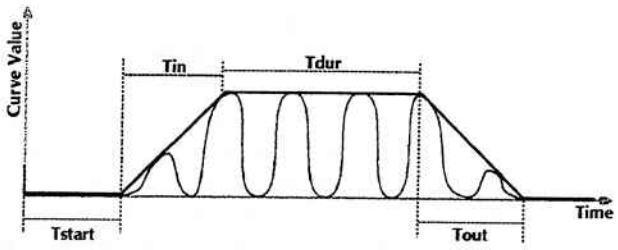

Figure 5: How the envelope affects the curve

This envelope defines a time to start (where the weight of the expression is zero, no matter the value of the function), a time to "fade in" (where the value of the function is multiplied by a linear interpolation from 0 to 1), a "duration" time (where the value of the function is not changed) and a "fade out" time (where the value of the function is multiplied by a linear interpolation from 1 to 0). The duration time can be infinite. The curve values are not limited to [0,1], although most of time this range is used.

Several expressions can be used in the same action. When two or more expressions are combined in the same action, the system just add the weights, therefore the order that the expressions are specified is not important.

The action itself is involved in an envelope. For each action we specify a time to fade in, a duration time (which can be infinite) and a time to fade out. These times are especially important in the transitions from one action to another.

The system uses a simple language to create actions. The syntax is:

```
ActionName Tin Tdur Tout
  Expr1  Tst  Tin  Tdur  Tout  curve
params
  Expr2  Tst  Tin  Tdur  Tout  curve
params
  ...
```

Tin is the time to fade in, *Tdur* is the duration of the action or the curve of the expression and *Tout* is the time to fade out. All times are given in seconds. The parameters of the curve are dependent to the type of curve used.

To illustrate how actions are done, we show below the code that defines the action *sneeze*:

```
Action Sneeze 0.1 9.6 0.3
  rotx 0 0 10 0 spline 0 0 3.5 1 4 -2
5 -2 7 0
  blink 3 0 2.5 2 constant 1
  Ahhh 0 3 0.5 0.5 constant 1
  whistle 3.5 0.5 1 2 constant 1
  roty 7 0.2 1 0.2 noise 0.3 1
  rotz 7 0.2 1 0.2 noise 0.3 0.6
```

The action has a total time of 10 seconds (0.1 to fade in, 9.6 sustain and 0.3 to fade out). We defined in the expression modeler that the expression *rotx* rotates the head in about 30 degrees in the X axis. We used a B-spline curve to animate this expression: in time 0 seconds, the value of the weight for this expression is 0. From 0 to 3.5 seconds, the weight is smoothly interpolated from 0 to 1 (which is the movement for the "ahhh" phase in a sneeze). From 3.5 to 4 seconds, the weight is smoothly interpolated from 1 to –2, which is the movement for the "choo" in the sneeze. And so on. The *blink* expression closes the eyes during the sneeze. The *Ahhh* and *whistle* perform the mouth movements for "ahhh-choo" and the *roty* and *rotz* expressions gives an impression of itching the nose after the sneeze.

One of the key elements to add realism to the characters is the use of Perlin noise [K. Perlin (1985)]. In order to make actions like blinking and breathing realistic, the timing must be non-predictable, but it must be controlled as well, so the actor will not take too long to blink or blink too fast – these variations are handled very well with the noise function. As a result, the use of Perlin noise helps avoiding the impression of "mechanical" movements.

Actions can be interrupted at any time. When an action is interrupted, it automatically starts to "fade out", unless, of course, the action is already in this state. The fade out time in actions is used to avoid rough transitions when an action is interrupted.

4.2 Layers of Actions

In order to improve performance and make the process of animation easier, actions of the same kind, like mouth movements, head movements or actions expressiong emotions are grouped together. Each group represents a *layer* when the actions are composed (the words *layer* and *group* represents the same concept in this paper) . Actions that belong to a group are usually not performed at the same time, except during the transition from one action to another – where one action is "fading in", while the other is "fading out". Of course, actions from different groups can be performed simultaneously.

While the system allows any action to be grouped together, we achieved the best results using the following groups: non-intentional actions (like breathing and blinking), emotions (like crying, smiling and

sleeping), Lip movements (the eight lip positions described above that mimics the phonemes), eyes movements and head movements.

Each layer has a transparency or alpha value, ranging from 0 to 1. Groups with alpha set to 1 are completely transparent and with alpha set to 0 are completely opaque. The alpha value is used in the composition of actions, described below.

4.3 Composition of Actions

The virtual action can perform several actions at the same time. Each action that is been executed may contribute to the final scene, depending of the alpha value of the group it belongs. Actions executed in an almost opaque layer will have more influence in the final scene than actions in an almost transparent layer. The model for composing the actions is similar to techniques used in composition of layers of images with transparency [M. Levoy (1988)].

In a given time, the system verifies in each group the actions that are being performed. If no action is performed in a group, it skips to the next. If an action is performed, probably this action is multiplying an array of modeled expressions by curves, resulting in an array of weights in the BLE. The system accumulates the weights of the BLE from group to group using the following algorithm:

```
For each group {
   [W_scene] = [W_scene] * α_group + [W_group] * (1 - α_group)
}
```

Where $[W_{scene}]$ is the array of weights of the BLE of the scene, $[W_{group}]$ is the array of weights of the BLE calculated for the actions that are been performed by this group and α_{group} is the transparency value of the group. Of course, groups with alpha value of 0 do not influence the final scene. The actions use transparency also: for instance, actions that only make movements on the lips are transparent to the rest of the face.

The order which the groups are calculated is important! Changing the order can give a totally different result. Groups that are calculated later usually affect the scene more than groups calculated earlier. The group order can be changed on the system in real time.

This technique simplifies considerably the work of an animator: the virtual actor can perform actions like crying, talking, blinking and moving the eyes at the same time. The animator acts more like a director to the scene, coordinating what the virtual actor will do, instead of actually moving the objects and vertices.

5. Conclusions and Future Work

The InterFace system was designed to be used by animators or to be integrated with other systems. In order to stick to this goal, we decided to make the interface and the inputs to the system as simple and visual as possible. Although the actions are not created interactively at the moment, actions created for one actor can be easily used in another, so libraries of actions can be made, simplifying the process of animating other actions considerably. Using Java and VRML for development made it possible to share our program with users all over globe through the Internet, allowing an ease way to distribute the software and receive feedback from testers.

The InterFace system will be improved in different ways. We are working on a script language that can start and stop actions and change the group order and transparency. Scripts can be used to drive a scene directly or can be combined in libraries to be used as "building blocks" of larger scenes. Since InterFace is a real-time system, we can read and execute scripts in real-time. This way, any program that uses CG humans as its user interface (like programs that simulate human behavior, computer games, virtual agents or avatars) can use InterFace to render the facial expressions of the virtual actor. We are also working on a visual editor of actions, sticking again with our objective to create a system easy to use for animators.

In a near future we will add the ability of changing the color of the face, thus simulating vascular activity [P. Kalra, N. M. Thalmann (1994)]. This would increase the realism of the virtual actor – when the actor is feeling fear, his face would become white, when he is angry or shy, his face become red and so on.

Finally, we are working on recreating the environment of the Aria project, presented in Siggraph 1996 [R. Ruschionni et al (1997)] on this system, allowing it to run in PCs through the Internet instead of expansive graphics workstations.

6. Acknowledgments

This work would never be possible without the help and suggestions of Ken Perlin, professor of The New York University. He is also responsible for the name InterFace. Arnaldo Massato Oka created the 3D model. Everyone involved in Aria also helped in the development of this project, especially Marcelo Zuffo and Ruggero Ruschionni. Daniel Wey would like also to thank Raquel Pires Gonçalves for the supported and for lending her animation books. This project was partially supported by the administration of the University of São Paulo.

7. References:

References are listed in alphabetical order:

[B. Robertson (1997)] B. Robertson, *Mastering Lip Sync*, Computer Graphics World, August 1997, 26-36.

[C. Wang et al. (1994)] C. L. Y. Wang, David R. Forsey: *Langwidere: A New Facial Animation System*, IEEE Computer Graphics and Applications, September 1994, 59-68.

[F. Parke (1974)] F. I. Parke, *A parametric model for human faces*, PhD Thesis, University of Utah, December 1974.

[F. Parke (1972)] F. I. Parke, *Computer generated animation of faces*, Master Thesis, University of Utah, June 1972.

[F. Parke, K. Waters (1996)] F. I. Parke, K. Waters, *Computer Facial Animation*, Published by A. K. Peters, 1996.

[Java (1995)] Java - http://www.javasoft.com/.

[K. Waters (1987)] K. Waters, *A muscle model for animating three-dimensional facial expressions*, SIGGRAPH Proceedings 1987, 21(4):17-24.

[K. Perlin (1985)] K. Perlin, *An image synthesizer*, Siggraph '85 proceedings, 1985, 19(3):287-296.

[K. Perlin, A. Goldberg (1996)] K. Perlin, Athomas Goldberg, *Improv: A System for Scripting Interactive Actors in Virtual Worlds*, Siggraph 1996 Computer Graphics proceedings CDROM.

[K. Perlin (1994)] K. Perlin, *Danse Interactif*, Siggraph '94 Electronic Theater Video, 1994.

[K. Perlin (1997)] K. Perlin, *Layered Compositing of Facial Expressions*, Siggraph '97 technical sketches, 1997 (online at http://mrl.nyu.edu/improv/sig97-sketch)

[K. Perlin, 1995] K. Perlin, *Real Time Responsible Animation with Personality*, IEEE Transactions on Visualization and Computer Graphics, 1995, 1(1) (online at http://mrl.nyu.edu/improv/perlin.ieee.tar.gz).

[M. Levoy (1988)] M. Levoy, *Volume Rendering: Display of Surface from Volume Data*, IEEE Computer Graphics and Applications May 1988, 26-36.

[K. Mehrabian, J. Ferris (1967)] K. Mehrabian, J. Ferris, *Inference of Attitudes from non-verbal communication in two channels*, Journal of Consulting Psycology, 1967, 3(31):248-252.

[N. M. Thalmann et al. (1989)] N. Magnenat-Thalmann, H. minh, M. de Angelis and D. Thalmann, *Design, Transformation and Animation of Human Faces*, The Visual Computer, 1989, 32-39.

[P. Kalra, N. M. Thalmann (1994)] P. Kalra, N. Magnenat-Thalmann, *Modeling of Vascular Expressions in Facial Animation*, IEEE Computer Graphics and Applications, September 1994, 50-57.

[P. Blair (1989)] P. Blair, *How To Draw Film Cartoons*, Walter Foster Publishing, 1989, 17.

[R. Ruschionni et al (1997)] R. A. Ruschionni, J. D. R. Wey, M. K. Zuffo, E. Toledo, *Some Experiences Implementing Virtual Worlds: The Aria Project*, IFIP 1997 Proceedings CDROM, Florianópolis, SC, Brazil – http://www.lsi.usp.br/~aria/.

[VRML (1997)] VRML97 - The Virtual Reality Modeling Language Specification - http://www.vrml.org/.

BOLETINS TÉCNICOS - TEXTOS PUBLICADOS

BT/PSI/0001 – Observabilidade Topológica de Osawa em Redes não Lineares – ARMANDO HANDAYA, FLÁVIO A. M. CIPPARRONE

BT/PSI/0002 – Desenvolvimento de uma Microbalança de Quartzo para Detectar Gases – ROBERTO CHURA CHAMBI, FRANCISCO JAVIER RAMIREZ FERNANDEZ

BT/PSI/0003 – Sistema para Desenvolvimento de Sensores Inteligentes – ANTONIO CARLOS GASPARETTI, FRANCISCO JAVIER RAMIREZ FERNANDEZ

BT/PSI/0004 – A 1.6GHz Dual Modulus Prescaler Using the Extended True Single-Phase Clock CMOS Circuit Technique (E-TSPC) – JOÃO NAVARRO SOARES JÚNIOR, WILHELMUS ADRIANUS M. VAN NOIJE

BT/PSI/0005 – Modelamento em Linguagem VHDL de uma Unidade de Policiamento para Redes Locais ATM – ÉDSON TAKESHI NAKAMURA, MARIUS STRUM

BT/PSI/0006 – Otimização das Operações Coletivas para um Aglomerado de 8 Computadores usando uma Rede Ethernet 10 Mbps baseada em Hub – MARTHA TORRES, SERGIO TAKEO KOFUJI

BT/PSI/0007 – Short Temporal Coherence Optical Source With External Fiber Optics Cavity – CARMEM LÚCIA BARBOSA, JOSÉ KEBLER DA CUNHA PINTO

BT/PSI/0008 – Hidrogenated Carbon Films Used as Mask in Wafer Processing With Integrated Circuits: Post-Processing – JUAN M. JARAMILLO O., RONALDO D. MANSANO, EDGAR CHARRY R.

BT/PSI/0009 – Redes Neurais em VLS – ANTONIO RAMIREZ HIDALGO, FRANCISCO JAVIER RAMIREZ FERNANDEZ

BT/PSI/0010 – Caracterização de Filmes Obtidos a Partir da Deposição por Plasma de Hexametildissilazana – SANDRINO NOGUEIRA, MARIA LÚCIA PEREIRA DA SILVA