Real-Time Wrinkles For Expressive Virtual Characters

by

Rowan Hughes, B.Sc.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

September 2011

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Rowan Hughes

August 30, 2011

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Rowan Hughes

August 30, 2011

Acknowledgments

To Antoinette, for your ceaseless understanding and support. To my parents for always believing in me, even when I didn't. To all of my fellow masters candidates, all of whom helped drag me kicking and screaming through a gruelling academic year. I would also like to thank my supervisor, Rachel McDonnell, for all of her help and ideas throughout the course of this dissertation.

Rowan Hughes

University of Dublin, Trinity College
September 2011

Real-Time Wrinkles For Expressive Virtual Characters

Rowan Hughes
University of Dublin, Trinity College, 2011

Supervisor: Rachel McDonnell

This thesis presents a method to both add realism and to enhance user perception of emotional states of virtual characters through the addition of fine details, such as wrinkles and furrows. We also attempt to add realism to the characters by leveraging state of the art methodologies in skin rendering.

In our implementation we build on the previous work by Dutreve, Meyer and Bouakaz[1]. The method uses a small set of reference poses which are compared with the current pose at run time in order to produce a set of similarity metrics. Wrinkle maps are then blended and applied according to these metrics. Pose evaluation is carried out using the mesh skeleton's bone, or "virtual muscle", transformations. Skinning blend weights are then used in order to associate rendered fragments and reference poses. Blending of the wrinkle maps builds on the work by Oat. [2] through the use of dynamically created mask data with a partial-derivative normal mapping method.

The technique described is both efficient in terms of computation as well as storage costs and can easily be inserted into a conventional real-time skinning based animation pipeline. We also describe a method to extend the technique into other domains, with particular reference to the small scale wrinkles formed on clothing around articulated joints of virtual characters.

Realistic skin rendering techniques have taken a dramatic leap forward over the past number of years and our technique leverages some of the most recent advancements in the field, specifically the work done by Jimenez et al.[3, 4], Penner and Borshukov[5] and dEon et al.[6], in order to create a more realistic virtual character.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Compelling facial animation is an extremely important and challenging aspect of computer graphics. Both games and animated feature films rely on convincing characters to help tell a story and an important part of character animation is the characters ability to use facial expression. Without even realizing it, we often depend on the subtleties of facial expression to give us important contextual cues about what someone is saying to us. For example, a wrinkled brow can indicate surprise while a furrowed brow may indicate confusion or inquisitiveness.

Much important work has been accomplished in this field allowing artists much more control over the more subtle aspects of facial animation. Real-time wrinkle techniques have been successfully used in a number of modern AAA video game titles, notably the Crysis and Drake's Fortune series. In the original pioneering work Oat [2] describes a technique for artist controllable wrinkles. This technique involves compositing multiple wrinkle maps using a system of masks and artist animated weights to create a final wrinkled normal map that is used to render a human face. Jimenez et a.l [7] extended upon this technique producing staggering results both through the introduction of a number of key optimisations to the original system and also through the addition of some state of the art skin rendering techniques. Their work clearly shows that modern real-time graphics techniques for the rendering of virtual characters are beginning to overcome the uncanny valley [8]. While not yet at the point of total photo-realism it is not difficult to envision that gap being bridged in the near future.

The goal of this thesis is to implement these techniques and explore the possibility

of driving our system through the use of data readily available from the character being rendered, that is the current position of its underlying skeletal structure.

1.1 Motivation

The current state of the art, as previously mentioned, provides extremely impressive results [7]. Their technique drives the system through the examination of triangle deformation [9] post and pre-skinning. This technique can only be used with a morph target based animation pipeline. We investigate the possibility of extending this system, with all of the associated skin rendering techniques, and apply it to a traditionally skinned mesh. In order to achieve this we must derive a new way of obtaining weights with which to blend in the models associated wrinkle maps. In this work we build upon the technique as described by Dutreve, Meyer and Bouakaz [1] in order to drive the wrinkle weights system.

Our technique, like previous techniques, is heavily dependent on the quality of the assets being employed. One notable problem with using skinning weights as drivers for blending in wrinkle masks is that it is quite common for blend weights associated with a bone not to be attached to regions on the mesh that should wrinkle if the bone moves. We examine the possibility of associating masks in the place of skinning weights in order to overcome these problems, this also has the benefit of allowing total artistic control over the display of wrinkles on the mesh.

In summary while it is not always necessary, or indeed desired, to convince the observer that they are viewing a real character it is always desirable to convey to the observer the current emotional state of the virtual character. One of the main issues that can cause emotional disconnections between observer and character is lack of facial expression on the part of the virtual character. Through implementation and extension of the original technique we will look to create a more convincing connection between observer and virtual character. This will be achieved through the addition of real-time wrinkles to virtual characters animated through a traditional skinning based pipeline.

1.2 Goals

The primary goal of this dissertation is to implement an application that can add emotional information, through the addition of facial wrinkles and furrows, to a virtual character, animated using skinning. We expect that this will add to a users perception of the emotional states of said virtual characters.

The secondary goal is to investigate whether the technique is applicable in domains other than facial wrinkles.

The tertiary goal is to experiment with optimisations and extensions to the technique looking both at the wrinkle techniques and also experimenting with the state of the art in skin rendering in order to produce a more convincing result and a technique suitable to be used in a real-time environment.

Finally, we wish to ensure that the implementation performs well and the overhead of running it is not prohibitive. If this can be achieved, algorithm optimisations are investigated to extract the best performance, as measured by frame rate.

1.3 Dissertation Outline

This dissertation is laid out as follows:

Chapter 2 : Background and Related Work describes the problem domain specifically related to real-time wrinkles and an overview of the main techniques that can be used to model this phenomenon. This chapter also contains some basics regarding some of the other techniques used in order to create a more visually realistic virtual character such as skin rendering techniques, screen space ambient occlusion, etc.

Chapter 3 : Design presents a high level view of the design and requirements of the system and the implementation plan.

Chapter 4 : Implementation describes the basic implementation, the application framework to host the technique and provides user manipulation functions and test scenarios.

Chapter 5 : Evaluation describes the output of the algorithm with reference to three specific areas, perceptual evaluation, performance and comparisons with previous work.

Chapter 6 : Conclusions summarises the work and main results in the thesis and presents ideas for future work to improve upon the current implementation.

Chapter 2

Background and Related Work

During the latter half of the 19th century, a French neurologist name Guillaume Duchenne performed a series of experiments that involved applying electrical impulses to his subject's facial muscles with the aim of identifying the muscles of the face responsible for various human expressions [10]. One of the more surprising results of these experiments was that the most subtle of differences in facial expression can express very disparate emotional states. For instance, a genuine smile utilizes not only the muscles of the mouth, but also that of the eyes and so the difference between a sarcastic smile and a genuine smile differs physically in a very subtle, but very important way.

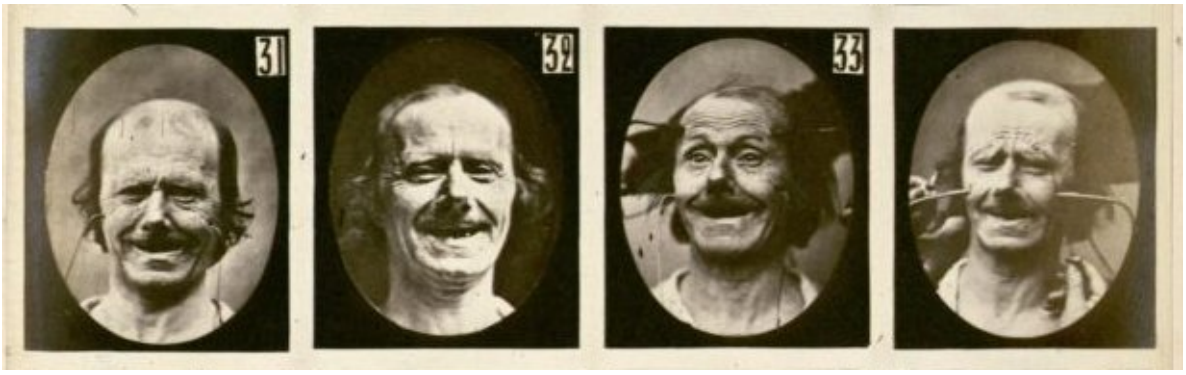

Figure 2.1: Photographs from the 1862 book Mecanisme de la Physionomie Humaine by Guillaume Duchenne. Through electric stimulation, Duchenne determined which muscles were responsible for different facial expressions.

While a good deal of research over the past number of years has focused on adding realistic real-time wrinkles to virtual characters, it is only very recently that we are

seeing these techniques being introduced into modern applications such as video games, etc. These techniques are almost exclusively based on work originally described by Oat [11]. In this chapter, we will look at this technique in more detail, the other major work and breakthroughs that have been seen in this field, and briefly some of the other techniques implemented by this dissertation to create more expressive and realistic virtual characters.

2.1 Motion

2.1.1 Wrinkle Maps

Bump maps and normal maps [12] are well-known and widely used techniques to add small surface details to otherwise far less detailed surfaces through the storing of surface normals, in tangent space, in a texture. These techniques are widely used in real-time applications to add surface detail to human faces. The use of static, artist created maps have one major drawback, however, in that they do not accurately represent the dynamic surface of a human face, they only "fit" the pose for which they were drawn.

Figure 2.2: Title character from Drake's Fortune series, by Naughty Dog, rendered with and without wrinkle maps.

In [13, 1, 14, 11, 7] we are introduced to the idea of wrinkle maps. Wrinkle maps in these cases are standard bump maps, that is an artist generated texture containing the encoded wrinkle patterns for a given model.

In order to create the appearance of wrinkles on an animated mesh in a real-time environment, these wrinkle maps are blended into the image based upon some weighting scheme, discussed in detail later. The weighting scheme uses some scalar range, i.e. from [-1,1], [0,1], etc. In [11] two wrinkle maps are used; one representing a stretched pose and one representing a squashed pose, with a blend value of 0 indicating that the wrinkle map(s) do not at all contribute to the final rendered image and -1 or 1 indicating that the squash/stretch wrinkle maps contribute totally to the final rendered image respectively. The technique described in [7] expands upon the scheme in [11] and allows for the use of any number of wrinkle maps. In this case, wrinkle weights are no longer blended between negative and positive values indicating which of two wrinkle maps to use, instead a range of [0,1] is used for each desired wrinkle map.

2.1.2 Wrinkle Masks

Wrinkles on a character's face must be independently controllable and therefore the wrinkle maps must be broken up into multiple regions. In order to achieve this goal, a mask (or indeed multiple masks) is used. In [11, 7, 13] among others, the authors make use of wrinkle masks which are stored as simple 2D textures. For each desired independently controlled region, a texture channel is used in order to store a scalar value representing the area of control. In [7], we see that the authors define eight independently controlled wrinkle regions and so two textures are used to achieve this. Any number of masks can be used with this methodology.

In the above cases, we are dealing with artist defined masks in order to define wrinkle zones. These masks are created by artists based on the desired regions where they wish for wrinkles to appear. However, if we look at the work in [14], we can see that the masks are dynamically created through the use of skinning weights which determine where wrinkle zones occur on the mesh. This methodology will be described in more detail in the implementation section.

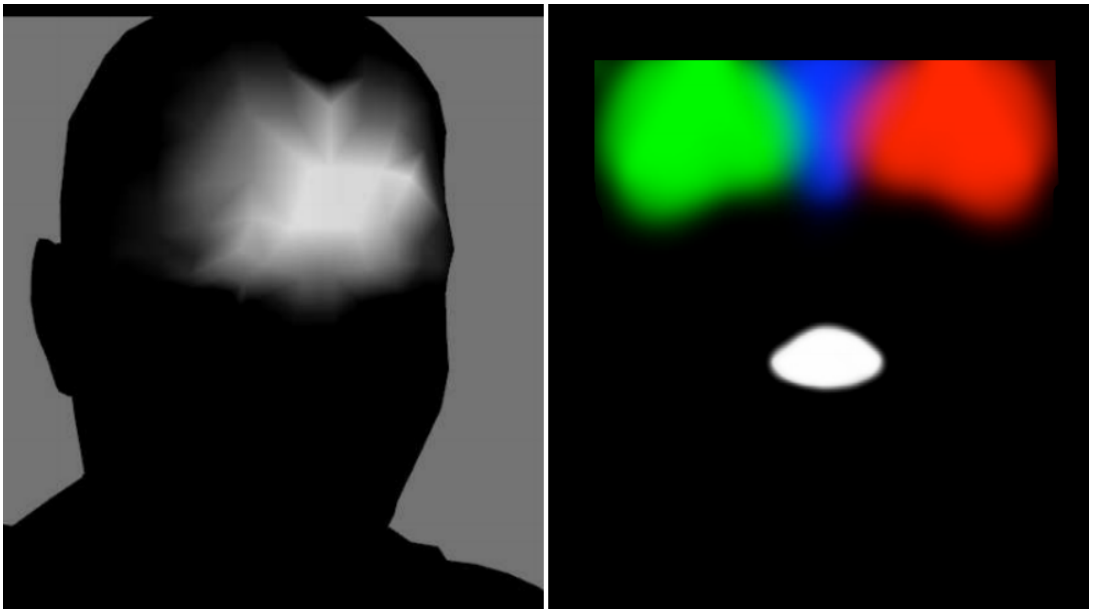

Figure 2.3: Left: Dynamically defined mask zones based on skinning weights. Right: Artist defined wrinkle mask texture.

2.1.3 Dynamic Wrinkle Weights

In order to effectively use our technique in a dynamic real-time environment we must have some method in which to determine which areas of the wrinkle maps to display and calculate the weight of the contribution, if any, for any particular pixel. Different methodologies have been suggested in this regard and one of the main drivers behind this, outside aesthetic or performance factors, is the fact that different animation techniques require different methodologies.

In [11], two very different techniques are described in order to achieve this feat. The first is performance driven, where an actor is filmed performing a scene, and weights for wrinkles were generated through the use of facial recognition software. This works quite well for scripted scenes but is not suitable for most other scenarios, for example, in conjunction with real-time motion capture. The second method described (derived from Microsoft's Sparse Morph Target Demo [9]) involves computing, using a geometry shader, the wrinkle weights based upon the deformation of the triangles on the mesh before and after skinning. The area of each triangle on the mesh is calculated before and after skinning and the resulting differences are used to approximate squashing or

stretching. This technique is effective when using blend shape \morph target type animation.

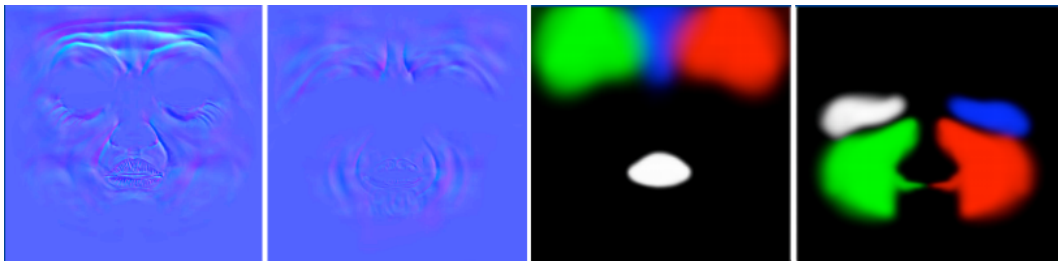

Figure 2.4: Left - "Squash" and "Stretch" wrinkle maps. Right - Two wrinkle mask textures representing eight, four per texture, independently controllable wrinkling regions.

The techniques described in [1] and [14] describe a very different methodology in determining wrinkle weights. They require that the character mesh be connected to one or more bones with a convex combination of weights. A number (two in these cases) of reference poses for the character are stored by this system and are compared at every frame against the current pose. These reference poses are highly pronounced poses that would result in a very strong wrinkle contribution, that is the maximum squash and stretch pose. Each desired bone is compared at every frame with the reference pose(s) on the CPU. Once influences have been calculated for each bone in the model these must be related to vertices on the mesh in order to determine influence on a per-vertex level. Importantly, by doing this on a per-bone basis it allows for independently controlled regions achieving similar results as the texture based techniques. The resulting set of influences are then sent to the GPU in order to determine the final wrinkle weight used for wrinkle map blending.

It is worthy of note that in all the techniques described above it is necessary to make a decision on how to blend in any additional normal maps. Simply blending maps would result in an overall loss of detail. The wrinkle map/s should contribute in addition to the base normal map. For this reason, it is not possible to simply average the maps and achieve a desirable result. In all of the above described techniques the following method was used to overcome this issue, that is the degree to which a normal is "bumpy" is defined through the z component with the direction of the bump being encoded in the x and y components. The directional components are simply blended giving the desired direction but the z components are multiplied which leads to an

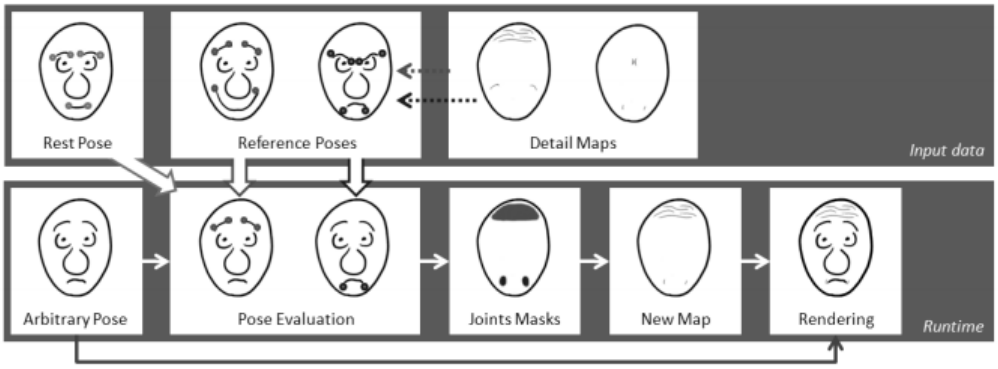

Figure 2.5: Visual representation of technique described in [1]. At runtime, each pose of the animation is compared with the reference poses, bone by bone, and skinning influences are used as masks to apply the bones poses evaluation. Wrinkle maps are blended on the GPU allowing rendering of the current frame with dynamic wrinkles.

increase in detail derived from the two maps rather than a flattening out. This can be written mathematically as:

$$\vec{WN} = normalize(W.x + N.x, W.y + N.y, W.zN.z) \tag{2.1}$$

where $\vec{WN}$ is the final normal, W is the normal provided by the blending of wrinkle maps and N is the normal of the base normal map for the mesh.

2.1.4 Beyond Normal Maps

Although in the papers outlined above all of the techniques used normal maps in order to simulate wrinkles this leaves room for considerable experimentation with alternative types of mapping beyond simple normal mapping. There are many techniques that could be considered in this regard such as Parallax Occlusion Mapping [15] and Relief Texture Mapping [16]. Some experimentation with these particular methods was carried out for the purpose of this dissertation, providing decidedly mixed results with the overall added cost in terms of computation not proving cost effective in terms of increased realism.

One of the issues with using normal maps is that fact that the underlying geometry is not changed and thus the model's silhouette remains unchanged despite the presence of wrinkles. While this may not be an issue with small scale wrinkles on the forehead

it could be with larger wrinkles, such as jowls, especially if robust collision detection is required.

2.1.5 Alternate Wrinkle Methodologies

The use of wrinkle map textures only forms one of the methodologies used to create animated wrinkles, although it is notably by far the most common. Other solutions exist, primarily the techniques involve some form of vertex displacement [17, 18, 19].

In [17], wrinkling is achieved through the use of length-preserving geometric constraints along with artist placed wrinkle features at locations on an animated mesh where wrinkling is desired. This method relies heavily on the underlying tessellation of the mesh in question, as the geometry is being displaced in order to model the wrinkles. Therefore a sufficient degree of tessellation is required in order to achieve the desired result, that is the finest degree of wrinkling required.

In [18], we are introduced to a method to add wrinkles to the face of a virtual character. The actual wrinkling is simulated in much the same way as [17]. The work is worthy of note, however, as it outlined the individual wrinkle areas that were then adopted as "mask" areas in future work.

In [20], the authors introduce a new and novel approach to the generation of dynamic, high resolution wrinkles, on a simulated cloth mesh during run time. The method works by attaching a higher resolution mesh to the base mesh and allowing the vertices of the attached, or wrinkle, mesh to deviate from their attached positions within a constrained range. The shape of the attached mesh is determined using a static solver, originally described in [21], and the look of the wrinkles can be user defined and tweaked. This forms an interesting new approach to adding wrinkles to surfaces in real-time and may be interesting to see if the scheme can be used in order to create expressive wrinkles for human faces.

2.2 Appearance

Although the main thrust of this thesis was involved in the addition of wrinkles to a skinned mesh, we were very interested in creating a highly realistic virtual character. In order to achieve this, some state of the art methodologies in the field of skin rendering

were leveraged. In the following section, we will briefly outline the main work upon which our skin shading implementation is based.

Human skin is a complex surface to render in a perceptually believable manner. The main source of this difficulty stems from the fact that skin exhibits a property, as do most non-metallic materials to varying degrees, known as subsurface scattering. Subsurface scattering is a mechanism of light transport in which light penetrates the surface of a translucent object, is scattered by interacting with the material, and exits the surface at a different point. Skin has an added level of complexity in that it is a heterogeneous material. Heterogeneous materials scatter light differently depending on the part of the material that they are passing through. Therefore, in the case of skin a number of light scattering stages must be modelled in order to produce a visually accurate result.

2.2.1 The BRDF/BSSRDF

The Bidirectional Reflection Distribution Function (BRDF),models the interaction between light and surfaces. The BRDF describes the ratio of reflectance along the output vector with respect to the irradiance of the incoming light vector. Common examples of BRDFs that are used in computer graphics today include the Blinn-Phong specular model [22], the Gouraud diffuse model [23], and the Oren-Nayar diffuse model [24].

The BSSRDF, or Bidirectional Surface Scattering Reflection Distribution Function, was introduced by Nicodemus [25] to describe the relationship between the incidence flux and outgoing radiance at a point on a surface primarily taking into account subsurface scattering within the material. The BSSRDF is a generalised form of the BRDF previously described in that the point of light incidence and radiance can be different. Mathematically it is described as:

$$dL_o(x_o, \vec{\omega}_o) = S(x_i, \vec{\omega}_i, x_o, \vec{\omega}_o) d\Phi(x_i, \vec{\omega}_i) \tag{2.2}$$

where L_o is the outgoing radiance in the direction $\vec{\omega}_o$ at the point x_o and $\Phi(x_i, \vec{\omega}_i)$ is the incidence flux in the direction $\vec{\omega}_i$ at the point x_i. Resolving the outgoing radiance involves solving this using a double integral over both surface area A and incoming

Figure 2.6: Render of a human head model with a diffuse BRDF (Lambertian).

directions 2π resulting in:

$$L_o(x_o, \vec{\omega}_o) = \int_A \int_{2\pi} S(x_i, \vec{\omega}_i, x_o, \vec{\omega}_o) L_i(x_i, \vec{\omega}_i)(\vec{n}.\vec{\omega}_i) d\omega_i A(x_i) \tag{2.3}$$

It should also be noted that the BRDF is a special case of the BSSRDF, where the light enters and exits a surface at the same point. Further details are described by Jensen et al. [26].

2.2.2 Texture Space/Screen Space Diffusion

Texture space diffusion is a technique first developed by Borshukov and Lewis while creating the CG footage for the movie "The Matrix Reloaded" [27]. It primarily works on the observation that one of the more stand out features of subsurface scattering is an overall softening or blurring of the object to be rendered. The technique unwraps the mesh and calculates irradiance in texture space, this texture is then diffused in texture space using a simple parameterised approximation. The irradiance texture and

the diffused texture/s are then combined to form the final result which produces a final rendering that displays some of the translucency that we associate with skin. This technique only modelled a dipole approximation of light diffusion, however, and so left room for considerable improvement.

Figure 2.7: Top: d'Eon and Luebke's [6] algorithm blurs high-frequency details in texture space. The image sequence shows the initial rendered irradiance map and two blurred versions. Bottom: Jimenez et al. [3] screen-space implementation where the blurring is applied to selected areas directly on the rendered image.

d'Eon and Luebke [6] significantly improved on this technique by developing a multipole diffusion technique, to simulate scattering in multi layered materials. As in [27] this was achieved through the use of sum of Gaussians approximation, using separable 1D convolutions. The results here produced images comparable in quality to some of the original offline techniques as described by Donner and Jensen [28] in a real-time environment. However, this technique does have a number of serious drawbacks not least that a large part of the algorithm must be performed for each character in the

scene.

In more recent research, Jimenez et al. [4] translated the light diffusion from texture to screen space. By applying the technique in screen space rather than texture space a number of key optimisations are attained. Firstly, a number of typical GPU optimisations are reintroduced such as back-face culling and viewport clipping. Secondly, a large amount of the algorithm only has to be computed once rather than on a per model basis. These optimisations vastly improved the performance of d'Eon and Luebke's technique, while achieving comparable rendering results. Again, they applied the convolution kernel using the same six 1D Gaussian convolutions as in d'Eon and Luebke's research, combining them with a weighted sum in a second pass. A matte texture is also stored on rendering of models in order to ensure that convolution in screen space only occurs over the desired regions of the image.

Further research was carried out by Jimenez et al. [4] that further optimised upon this algorithm and extended the technique to incorporate real-time translucency. Incorporating translucency was outside the scope of this dissertation.

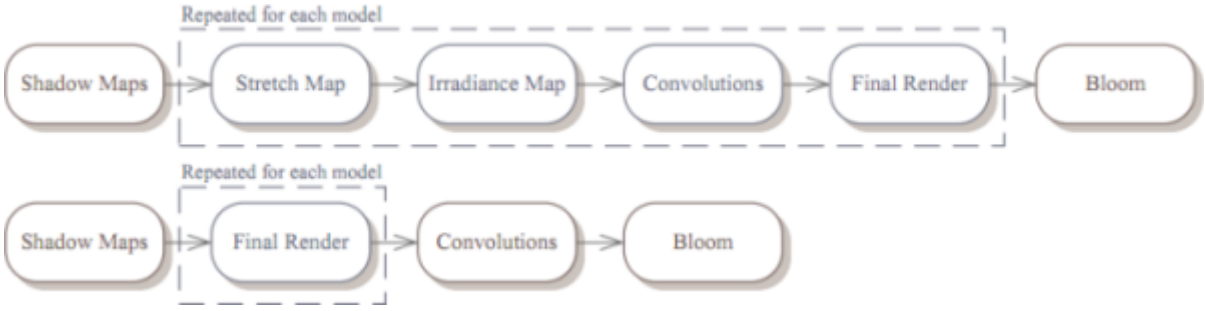

Figure 2.8: Visual comparison of the texture-space and screen-space rendering pipelines in order to simulate sub-surface scattering in human skin. We can instantly see the obvious performance benefits associated with the screen-space rather than texture-space implementation. Diagram from [3]

2.3 Screen Space Ambient Occlusion (SSAO)

Ambient occlusion is a shading method used in 3D computer graphics which helps add realism to local reflection models by taking into account attenuation of light due to occlusion. Ambient occlusion attempts to approximate the way light radiates in real

life, especially off what are normally considered non-reflective surfaces. Mittring et al. [29] introduced the idea of screen-space ambient occlusion, a GPU based approach which approximates the effect in a computationally efficient manner capable of running in real-time. SSAO was implemented in this project in order to determine the effect it would have on the overall perceptual realism of the rendered character.

2.4 Real-Time Wrinkles in Video Games

Real-time wrinkles in video games are a very modern development. All of the current methods used by the video game industry in order to produce this effect are based upon the texture based approach as pioneered by Oat [11].

Figure 2.9: Real-Time wrinkles as generated by the Crysis engine. Their method only works when animations are driven through the use of morph targets.

Morph target, or blend shape, animation is a very popular modern animation technique. A number of expressions for a character model are constructed by artists and these shapes are then interpolated to create the desired animations. At this point, we believe that all of the current implementations of real-time wrinkles in current

video games rely on this type of animation in order to be rendered rather than more conventional skinning techniques.

2.5 Summary

This chapter has given an overview of the current state of the art in real-time wrinkles and described the major concepts and ideas associated with these techniques. A brief overview of the skin rendering techniques that were implemented as part of this project was also outlined.

Texture based approaches to rendering real-time wrinkles is a very powerful and yet very simple concept that has already become popular, and remains an important area of research in modern video games as the industry strives for ever greater degrees of realism for virtual characters.

Chapter 3

Design

This chapter aims to provide an understanding of how the project was designed and the tools which were necessary in order to build the system. We also discuss the overriding identified project requirements, the tools and procedures that were used in order to generate suitable assets and finally the proposed perceptual evaluation tests.

3.1 Plan

The plan for this dissertation was broken down into three distinct areas, described as follows:

1. Investigation: A comprehensive literature review was necessary in order to fully understand the state of the art in the area. In depth investigation into those techniques that were to be built upon was conducted in order to understand fully the benefits and drawbacks of these solutions as well as to understand where improvements could be made. Finally, investigations were carried out to determine where disparate techniques could be brought together in order to achieve the desired final result.

2. Implementation: During the build phase, it was necessary to outline all of the various stages of development of the rendering technique itself as well as all of the various additional components that would be needed both for user control over the rendering system and debugging. These were broken up as follows:

(a) Camera and camera control system.

(b) Menu system allowing dynamic modification of shader parameters.

(c) A number of solutions were required to be built in order to precompute some of the textures required for the algorithm to work (e.g. Beckmann specular map, precomputed skin map, etc.).

(d) Debug viewer allowing intermediate steps in the rendering pipeline to be visualised.

(e) Benchmarking framework to capture performance data.

3. Validation: Create tests in order to validate both the perceptual advantages of the technique and overall rendering performance of the implementation.

3.2 Requirements

1. The overriding requirement of this dissertation is to produce a technique capable of rendering real-time wrinkles on a traditionally skinned mesh both in a computationally efficient and a perceptually valid manner.

2. The second requirement is to explore the possibility of utilizing the technique in domains other than facial wrinkles, more specifically the feasibility of the technique in rendering the small scale wrinkles visible around the clothing near the articulated joints of virtual characters is to be investigated.

3. The third requirement is to implement a modified screen-space diffusion algorithm as described by Jimenez et al.[3] in order to render human skin in a perceptually realistic manner. Optimisations and extensions to the existing technique are also to be investigated.

4. The final major requirement is to ensure that the algorithm produced runs satisfactorily compared with the existing implementations and if possible to try to improve upon the existing work in this regard.

3.3 Tools & Assets

The algorithm was built using the following combination of programming tools:

- C#
- XNA 4.0
- HLSL
- .X model file
- Visual Studio 10 (2010) development environment

Two three dimensional human character models were provided in a 3ds max model format for use in this dissertation. The models were provided by Trinity College Dublin's GV2 research group. In order to successfully use these assets for the purposes of this project it was first necessary to extract the head portion of the mesh along with its associated texture assets. It was then necessary to construct a skeleton for the head and skin the model appropriately. At this point a number of animation sequences were constructed both in order to allow visualisation of the wrinkles in runtime and to allow for perception testing. In order achieve these aims the 3ds max studio tool-set was used.

In order to create the wrinkle maps the deformed head model, that is the model at it's most extreme animated positions, were exported to Autodesk's Mudbox modelling toolset. In Mudbox the fine wrinkle details were created and the final wrinkle maps were exported as textures from this program.

It must be noted that this technique relies very heavily on the quality of the underlying assets in order to achieve the desired result. A complete set of artist generated assets would, realistically, be needed for deployment of this technique in an industrial environment.

3.4 Methodology

A loose agile methodology was used in order to develop the project. Agile development entails breaking the project up into stages or sprints with a specific time frame. These

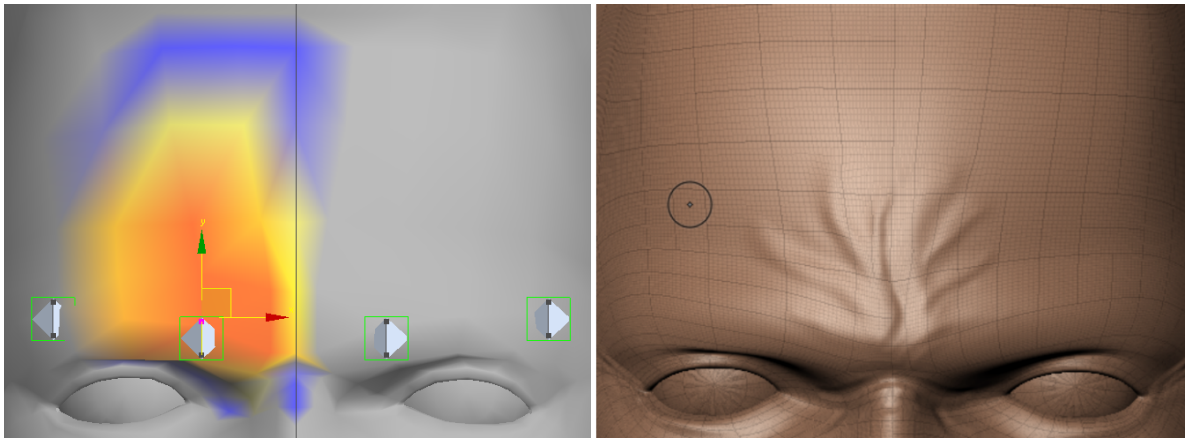

Figure 3.1: Left: Skinning weights associated with our mesh as authored in 3ds max. (Weights associated with any one vertex must always add to 1) Right: Wrinkles were authored using the Mudbox toolset

stages are further broken into individual smaller tasks again with a specific time frame. This methodology ensures that at the end of each stage a fully working system is in place upon which to build the next iteration.

3.5 Perceptual Validation

A user study was conducted in order to determine the effect that the addition of fine details, i.e. wrinkles, that appear and deform with a mesh during animation has to a viewers perception of the emotional state of the virtual character.

3.5.1 Participants

Six volunteering participants (five male, four female) took part in the experiment, all of the participant being unaware as to the experiments purpose . All of the subjects had experience with computer graphics and all reported normal or corrected to normal vision. The age of the participants ranged from 20 to 32.

3.5.2 Stimuli

The test consisted of 48 test stimuli (video clips). Two clips were produced for four different emotional states (Disgust, Happiness, Anger, Sadness), both with and without wrinkles and also with different types of skin shader applied. The final video contained repeated clips and the clip order was randomised. Participants were asked the following:

- Identify the emotional state of the character.
- Rate the intensity of the perceived emotion on a scale of 1 - 7.

Chapter 4

Implementation

In this chapter, we give an overview of the main algorithms and techniques that were implemented as part of this dissertation.

4.1 Overview

We aimed to construct an efficient technique for the display of perceptually realistic real-time wrinkles on a traditionally skinned mesh. In order to realise this aim some additional work from other from fields, particularly in the area of skin rendering, was leveraged. In addition to the construction of the application a number of new optimisations were implemented and tested in order to improve the final result. Fig 4.1 shows a high level diagram of our rendering framework.

4.2 Implementation Framework

In order to test, debug and validate the algorithm as well as provide a user interface that would allow anybody to intuitively work with the system an application framework was constructed. The various components of this framework are described below.

4.2.1 Menu

A user menu was implemented in order to allow a user to both manipulate individual shader parameters and switch various features on and off in real-time during application

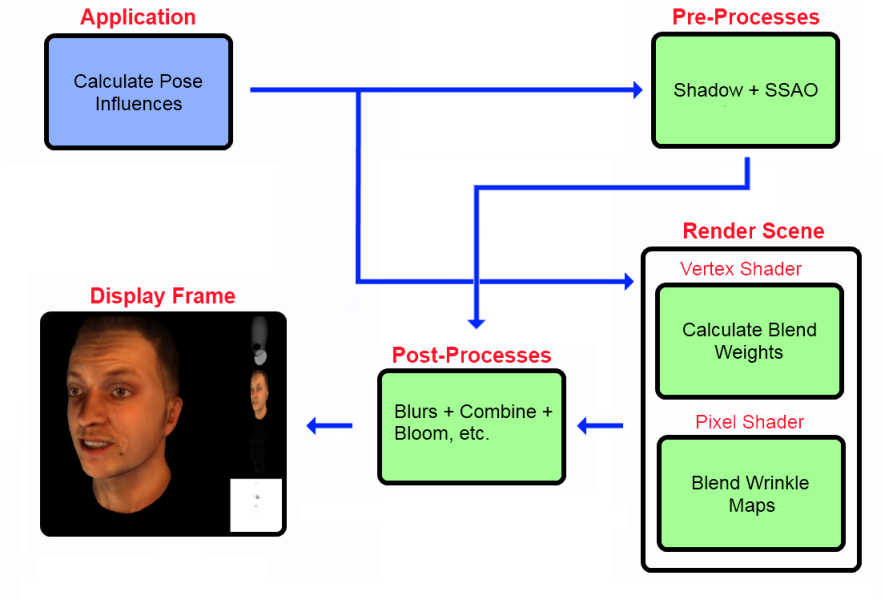

Figure 4.1: High level diagram of our rendering pipeline.

execution. This greatly speeds up the debugging process as various parameter values can be tested without having to reload the application at each stage.

4.2.2 Camera

A first person camera allows for the free positioning of the camera in the scene. The camera is primarily designed to be mouse driven with orbiting, panning and zooming being controlled via the various mouse buttons. The camera can also be manipulated however following the typical first person gaming conventions; the keys w, s, a, d for movement and mouse for free looking. The scheme enables the user to view the model from any point in the scene.

4.2.3 Debug

In order to allow the viewing of intermediate stages of the algorithm, a number of panels are displayed on the right hand side of the screen. These panels are capable of displaying all of the textures stored in the various render targets created during a rendering pass. This feature proves a good tool for debugging purposes and to allow

viewers to understand the stages in the compositing of the final image. The images that are displayed in these panels can be changed via the use interface.

4.3 Motion

4.3.1 Skeletal Pose Evaluation

As mentioned previously, in order for texture based wrinkle approaches to work, a set of wrinkle weights or coefficients are needed to drive the wrinkle animations. In this case, these weights are determined by comparing skeleton bone positions at run time with a set of reference poses.

Figure 4.2: Reference pose models as rendered from 3ds max.

The algorithm that was implemented was closely based on the technique described by Dutreve et al.[1]. In this implementation, two reference poses were used, representing the maximum stretch and squash positions of a virtual characters face, for comparison purposes. At run time, a set of pose influences are calculated for every desired bone on the CPU. A very important advantage of running comparisons and calculating the influences sets at the bone level rather than at the pose level is that it allows for the automatic determination of different areas of interest, assuming the model is skinned correctly. Multiple reference poses can therefore automatically be composited together allowing for a greater range of emotional states from a very limited number of poses.

Influence for any arbitrary pose P_k for the bone B_i at any frame f is defined by the following equation

$$I_{P_k}(B_{if}) = \begin{cases} 0 & \text{if } \vec{B}_{iP_o} = \vec{B}_{iP_k} \\ \alpha_{ifk} & \text{if } 0 \leq \alpha_{ifk} \leq 1 \\ 0 & \text{if } \alpha_{ifk} < 0 \\ 1 & \text{if } \alpha_{ifk} > 1 \end{cases}$$

with

$$\alpha_{ifk} = min\left(1, max\left(0, \frac{(\vec{B}_{if} - \vec{B}_{iP_o}) \cdot (\vec{B}_{iP_k} - \vec{B}_{iP_o})}{||(\vec{B}_{iP_k} - \vec{B}_{iP_o})||}\right)\right) \tag{4.1}$$

where $\cdot$ represents the dot product and $||..||$ represents the Euclidean distance. This algorithm works through the determination of the size of the segment between the orthogonal projection of $\vec{B}_i f$ onto the segment $[\vec{B}_{iP_k}, \vec{B}_{iP_o}]$.

This algorithm is run for every desired bone against each of our reference poses. At this point we are left with a complete set of wrinkle weights which can be passed as an vector array to the GPU, in order to drive the system.

4.3.2 Wrinkle Masks

Once the wrinkle coefficients have been derived we must use some methodology in order to associate them with specific areas on the mesh upon which we want the wrinkles to appear. There are two alternate approaches in this area which one can take, for the purposes of this dissertation both were implemented and compared with one another in order to determine relative performance.

Dynamic Masks

The quality of the skinning of the mesh is critical to the success of this technique. Wrinkles and expressive details are fundamentally related to facial deformations in that wrinkles form due to facial movement, we can assume that a similar relationship exists for mesh deformations. Mesh deformation in skinning animation is tied to the underlying skeletal structure of the mesh and how these bones are then attached to individual vertices. Assuming the model has been skinned correctly we can associate

skinning weights and pose influences using the following equation which calculates influence of the pose P_k for the vertex v_{ij} with skinning weight w_{ij} at frame f as

$$Influence(v_{jf}, P_k) = \sum_{i=1}^{n} (\lambda_{ji} \times \theta(I_{P_k}(B_{if}))) \tag{4.2}$$

where $Influence(v_{jf}, P_k)$ is clamped to range $[0, 1]$, $\theta(x)$ represents a modulation function and $\lambda_{ij} = 1$ if $w_{ij} > 0$ else $\lambda_{ij} = 0$. This equation was proposed in a more recent work, again by Dutreve et al.[14], that improves upon the original equation in that wrinkles become more pronounced during the animation in a way that is coherent with how real wrinkles appear due to bone deformation on a human face.

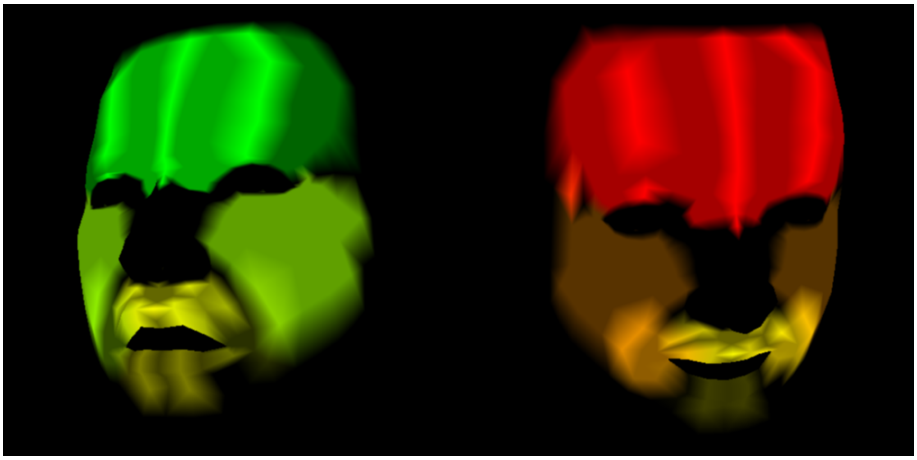

Figure 4.3: Visualization of the wrinkle masks generated by our system. Left: Mostly Stretch influence, bright green indicates full stretch pose inf. Right: Mostly Squash influence.

Modulation Function

The modulation function mentioned above allows us to alter the way in which wrinkles are blended into the mesh. A simple linear blending, according to bone displacement, does a reasonably good job of simulating the appearance of wrinkles but through the use of a simple non-linear modulation function we can generate a more realistic blending of wrinkles. Dutreve et al.[14] suggested the following modulation function in order to intensify the blending of wrinkles as bone positions neared their maximum

displacements.

$$\theta(x) = \frac{cos(\pi(1+x)) + 1}{2} \tag{4.3}$$

while this does provide a subtlety more accurate solution we found that this function does not fully capture the true nature of how wrinkles appear due to bone displacement. Through observation, it was noted that the intensity of the appearance of wrinkles has two separate peaks rather than the one peak simulated by Eqn. 4.3. In order to create an appropriate quadratic function to represent the curve the Mathematica 8 was used. A set of data points were selected to approximate the desired curve. Using the inbuilt Mathematica function FindFit on the dataset, a quadratic formula (equation 4.4) was produced. This was transcribed in to HLSL and added to the application. We found that the following equation, which models the two separate peaks, observable in wrinkle appearance, provided a better overall result.

$$2.62235x - 4.87773x^2 + 3.26168x^3 \tag{4.4}$$

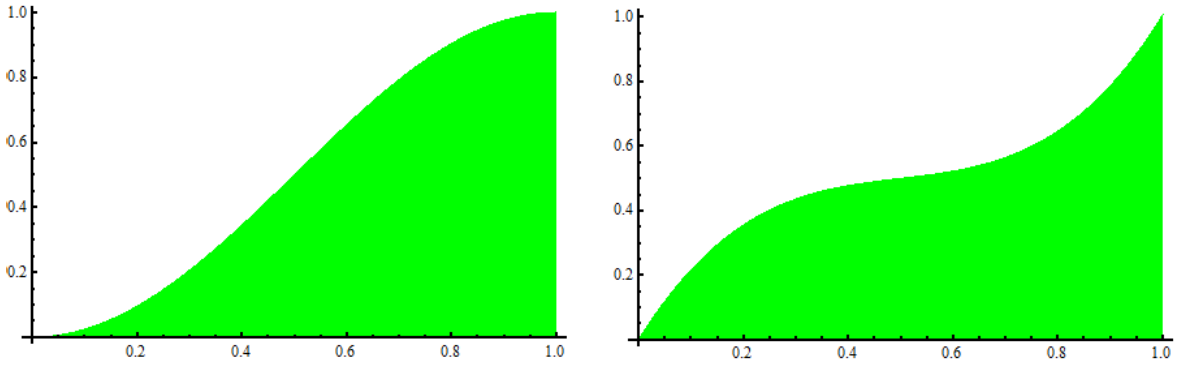

Figure 4.4: Left: Graph of Dutreve et al. modulation function $\theta(x)$. Right: Graph plotting our revised modulation function.

Texture Masks

Another method for associating wrinkles with areas on the mesh is to store areas of influence in a texture. Each area of influence takes up one color channel per texture, therefore one texture can define up to four areas of influence. One advantage to using this technique is that due to it's simplicity, these masks can easily be authored by

an artist in a very timely fashion. If the skinning weights on a mesh do not match correctly with the desired areas of influence it is a far more time consuming task to fix, it is also not desirable to alter artist skinning.

In order to cater for the eventuality that skinning weights do not cover the desired areas, as well as to analyse relative performance, testing was carried out as to whether we could still used the bone derived weighting coefficients with masking textures and obtain the desired result. Upon passing the pose influence data to the GPU an extra channel is set aside to store the bone index, thus these indices can be used in order to associate weighting coefficients and mask areas. For the face, two separate texture masks are used to outline eight areas of wrinkle influence.

As mentioned earlier, using this technique means that the artist has more control over the areas of influence. In addition to this, some redundancy can be removed from the system. Bones that may be closely related both in position and in their relative skeletal movements may be grouped together as one using mask areas, this means that the number of bones that have to be analysed in order to achieve the desired result is reduced.

One major drawback, however, is the need for a certain amount of hard coding on both the application and GPU side. Bones must be explicitly associated with mask areas, which is not ideal.

4.3.3 Wrinkle Map Blending

Now that the information required for blending in our wrinkle maps has been obtained, we must have some way of associating these weights with the wrinkle maps and blending these maps with the mesh's underlying normal map.

The base normal map of any character mesh often encodes many important fine details such as pores, scars, blemishes, etc. In the blending of our normal maps, we do not want any of this data to be lost. For this reason, a simple blending of maps is not suitable as this would result in an overall loss of surface detail. In order to prevent this from happening, we blend the maps in the following fashion:

$$\vec{n} = normalize(\vec{w_0}.xy + ... + \vec{w_n}.xy + \vec{b}.xy, \vec{w_0}.z \times ... \times \vec{w_n}.z \times \vec{b}.z) \qquad (4.5)$$

where $\vec{n}$ is the final normal, $[\vec{w_o}, ..., \vec{w_n}]$ represents the associated wrinkle maps and

finally $\vec{b}$ is the models base normal map. This works as the direction of a bump in a normal map is entirely encoded in the x and y components, a straight blending for these components gives us the correct new bump direction. The magnitude of the bump is stored in the z component and the lower the value of z, the greater the bump magnitude. A simple blending here would lead to an overall "flattening out" of the bump, through the use of a multiplication here we decrease the z component and end up with a result that retains the surface detail in the fashion desired.

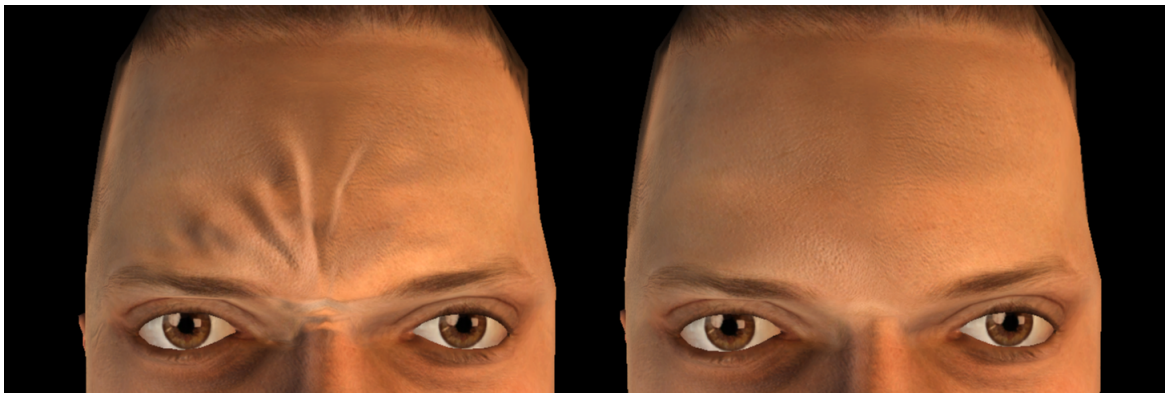

Figure 4.5: Left: Effect with and without real-time wrinkles, displayed using a simple Lambertian reflectance model.

Partial Derivative Normal Maps

While the above methodology worked quite well, in the work outlined by Oat et al.[7] an alternate scheme was put forward that provided better quality wrinkles and better overall performance. Partial Derivative Normal Maps (PDN), first proposed by Acton[30], provide a way to improve performance by having fewer instructions required in order to reconstruct the normal vector. PDN maps were computed from tangent space normal maps using a simple pixel shader. Conversion is performed through the use of the following functions:

$$n'_x = \frac{n_x}{n_z} \qquad\qquad n'_y = \frac{n_y}{n_z}$$

At runtime, reconstruction of a single partial-derivative normal n' to a tangent-space normal $\hat{n}$ is completed as follows:

$$n = (n'_x, n'_y, 1),$$
$$\hat{n} = \frac{n}{||n||}$$

At this point, blending of the various maps (base, stretch and squash in our case) consists of summing their x and y components followed by a normalization step.

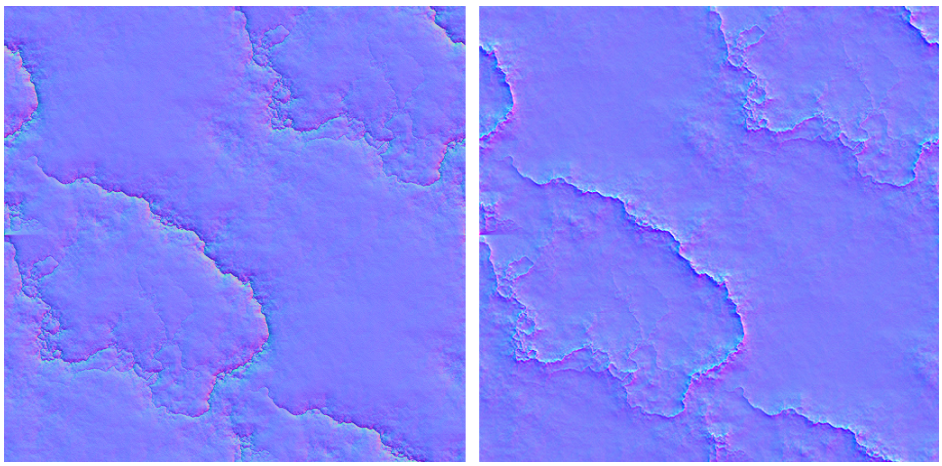

Figure 4.6: Right: Tangent-space normal map. Left: Partial derivative normal map

4.3.4 Beyond Facial Wrinkles

We were interested to determine if we could extend our technique to other domains than simply facial wrinkles. One area that stood out as an obvious candidate was virtual character clothing. Oat [2] describes a technique for generating such wrinkles. His technique derives wrinkle weights through the examination of triangle deformation post and pre skinning, calculated using a DirectX 10 geometry shader. This technique was not suitable for our implementation, as our chosen application framework only contains support for shader models 1.0 through 3.0. As geometry shaders were introduced with shader model 4.0 they could not be utilized as part of this dissertation. A scheme that derived wrinkle weights through the use of bone positions similar to our facial wrinkling technique was desired.

On examining the problem, an immediate observation was the fact that articulated joints on human virtual characters come in two distinct varieties; the hinge joint and

the ball and socket joint. Each joint type requires a different solution in order to derive effective blend weights.

Hinge Joint

The hinge joint is a type of synovial joint in the human body, it is also the simplest of it's class in terms of range of movement. The major hinge joints in the human body are the knee and elbow joints.

The hinge joint proves an easy case to solve for as it has a very restricted range of movement, being essentially 2D in nature. Hinge joints rotate on a single plane and within a β to 180° range where β° represents the joints minimum angle of extension. Maximum wrinkling should occur as the angle between the two bones approaches β.

The following simple equation determines the influence weight, α_{ij}, for the hinge joint involving the bones B_i and B_j and maps the result to the $(0, 1)$ range:

$$\alpha_{ij} = min\left(\frac{1 - (\vec{B_i} \cdot \vec{B_j})}{1 - cos(\pi - \beta)}, 1\right) \tag{4.6}$$

where $\vec{B_i}$ and $\vec{B_j}$ are the normalized directions of the bones at the joint.

Wrinkles at hinge joints appear on both the inner and the outer side of the joint. As the joint extends, the wrinkles on the outer side of the joint become more visible and those on inner side become less so. In order to achieve this effect we simply use $1 - \alpha_{ij}$ as a wrinkle coefficient for the inner side of the joint.

Now that we have derived wrinkle coefficients, we must associate them with areas on the mesh. The texture mask technique described in Section 4.3.4 is used in order to accomplish this. Skinning weights are not useful in this case as these longer bones (forearm, calf, thigh, etc.) are associated with far larger areas on the mesh than those upon which we require wrinkles to appear.

Ball & Socket Joint

The other type of joint we solve for is the ball and socket joint, i.e. the pelvis, shoulder, etc. This type of joint prove a more challenging task to derive wrinkle weights for, as the range of movement is far more extensive than that of simple 2D joints, such as the hinge joint.

Our solution breaks the rotation into two separate rotations. If one takes the case of the pelvis, we need to know how far the spine has bent and in which direction. Determining how far the spine has bent is a simple task, we use the formula as seen above, (Eqn. 4.6), to determine joint rotation. To derive a metric for the direction of rotation we take the position of the spine bone and project it into a plane perpendicular to the pelvis. A rotation weight, ϕ, is then derived from the dot product of the spine direction and the forward vector of the pelvis. The degree to which the spine is bent is used as the wrinkle coefficient, however, it still must be associated with an area on the mesh. In order to determine this area we use ϕ in conjunction with the spine bone's position.

The texture masking technique is again used. In this case, the texture contains four channels representing the front, back, left and right areas of the waist, with a good degree of overlap in this case to facilitate the animation. We derive our final wrinkle blend value, α, using the following values:

$$\alpha = \begin{cases} m_f \times w & \text{if } \phi > 0.75 \\ m_b \times w & \text{if } \phi < -0.75 \\ m_l \times w & \text{if } \phi > -0.75 \;\&\&\; \phi < 0.75 \;\&\&\; \gamma < 0 \\ m_r \times w & \text{if } \phi > -0.75 \;\&\&\; \phi < 0.75 \;\&\&\; \gamma > 0 \end{cases}$$

where m_f, m_b, m_l and m_r are the masking channel values, w is the weighting coefficient value and γ represents the x co-ordinate value of the spine bones position on the projected plane with respect to the pelvis (the forward vector of the pelvis representing the z-axis).

It is worthy of note that the same technique, used here for ball and socket joints, can be used to add wrinkles to pivot, saddle and condyloid joints without any changes, other than the adjustment of threshold parameters.

The technique as described provides a robust general solution to add realistic wrinkles to hinge joints. The solution provided for ball and socket joints is not as generalizable and while still an effective technique, requires a degree of manual adjustment in order to achieve the desired result.

Figure 4.7: Screen capture of wrinkles around hinge joints (knees) and a ball and socket joint (waist) rendered on a virtual character.

4.4 Appearance

Our technique at this point allowed for some very convincing wrinkle effects. A stated goal, however, was to create a technique capable of rendering perceptually realistic wrinkles on virtual characters. To achieve this end, a skin rendering technique as well as ambient occlusion was implemented in order to achieve a more realistic final result. The implementation of these elements will be outlined briefly in the following section.

4.4.1 Skin Rendering

The skin rendering technique implemented as part of this dissertation follows method as described by Jimenez et al. [3] quite closely, with only minor changes. The technique can be broken down into the following stages:

1. Render Shadow Maps

2. Render Scene & Specular Map
3. Run Convolution Passes
4. Combine Results
5. Bloom

Shadow Maps

If using point lights, the scene's depth map is rendered from the perspective of each light in the scene. If using directional lights only, then the depth is rendered from the camera's perspective only. It is critical that depth maps be stored in a linear fashion so that they can be used to perform kernel correction later in the pipeline, this can be easily achieved by diving the depth's output z value by the far clipping plane distance.

The depth maps are also used to perform the ambient occlusion pass which will be described in further detail in the next section.

Render Scene & Specular Map

Multiple render targets are used to capture the scenes lighting. The first pass calculates all incoming light, excluding specular, and renders the scene. This forms the base image that will be convolved in the next phase as well as the final combination pass.

The Phong specular model [31] is by far the most widely used model in modern graphics applications, however, we improve upon this through the use the Kelemen/Szirmay-Kalos specular model [32]. The technique approximates the physically based Torrance/Sparrow model [33] through the use of a precomputed texture, representing the Beckmann distribution function. It then uses the Schlick/Fresnel approximation [34] in order to calculate the specular BRDF in a real-time, inexpensive manner.

Convolution

This stage approximates the multi-pole diffusion profile of skin through the use of a sum-of-Gaussians approach. A diffusion profile, in this context, provides an approximation of how light scatters upon entering a highly scattering material, in our case human skin. Light, upon entering skin, is scattered in a highly isotropic fashion [28].

The diffusion profile allows us to model how light emerges from a material as a function of the distance from the point at which the light enters the material. Scattering is also highly color dependant, with red light scattering much further than green and blue light.

d'Eon et al. [6] found that six Gaussians were needed to match were needed to accurately match the three-layer model for skin described by Donner [28]. For this screen-space implementation we convolve the image using the same six Gaussians as d'Eon et al., see Fig. 4.8.

As we are working in screen space the size of the convolution kernel must be modulated with regard to depth and gradient. The deeper the pixel is in the image or the greater the the gradient in the depth map, the narrower the kernel should be. The following simple formulas, introduced by Jimenez et al. [3], caters for both of the these situations:

$$s_x = \frac{\alpha}{d(x,y) + \beta \cdot abs(\nabla_x d(x,y))}, \qquad s_y = \frac{\alpha}{d(x,y) + \beta \cdot abs(\nabla_x d(x,y))} \tag{4.7}$$

where $d(x,y)$ is the depth of the pixel in the depth map, α indicates the global subsurface scattering level in the image, and β modulates how this subsurface scattering varies with depth gradient. The operators ∇_x and ∇_y compute the depth gradient, and are implemented on the GPU using the functions ddx and ddy, respectively.

	Variance (mm^2)	Blur Weights Red	Green	Blue
	0.0064	0.233	0.455	0.649
	0.0484	0.100	0.336	0.344
	0.187	0.118	0.198	0
	0.567	0.113	0.007	0.007
	1.99	0.358	0.004	0
	7.41	0.078	0	0

Figure 4.8: Sum-of-Gaussians Parameters for Three-Layer Skin Model.

Combine

At this point the convolved images are now combined to simulate the sub-surface scattering effect. Each of the convolved scenes had an associated variance, that is the size of the Gaussian kernel used in the convolution. These images are then multiplied by the RGB profiles outlined in Fig. 4.8 depending on their associated variance and are linearly summed. At this stage the result is combined with the previously calculated specular component.

Bloom

The final stage in the skin rendering process is the bloom filter. The bloom filter first applies a threshold to the scene image to extract bright colours. The extracted image is then convolved using two separable 1D Gaussian convolution kernels. Finally, the images are composited to extract the final result.

Figure 4.9: Character as rendered using our skin rendering implementation.

4.5 Summary

This chapter presents a practical, real-time technique to add wrinkles to a traditionally animated virtual character as well as the fundamentals of the appearance enhancing techniques used.

The benefits of bone based wrinkle coefficient derivation can be combined with masking techniques in order to drive the system, which may prove a more attractive solution to potential users due to the increased level of artist control.

Bones provide a powerful tool in the generation of meaningful wrinkle coefficients due to their inherent relationship to the underlying mesh. It is demonstrated that techniques, similar to that used for facial wrinkles, are capable of being utilized in other domains with minimal changes.

Chapter 5

Evaluation

In this section we will present the results and findings of our evaluation procedure. The algorithm is evaluated from both a performance and perceptual standpoint.

5.1 Performance Evaluation

Carrying out a technical evaluation is an important task for any new piece of software. It allows developers to identify bottlenecks and other performance issues that may exist within a system. It can also prove a useful tool in identifying areas where there is room for fundamental improvement and optimisation. All performance evaluation tests were carried out on a machine with an Intel Xeon 2.67 GHz processor, an Nvidia GeForce GTX 560 Ti GPU and 4.00 GB DDR2 RAM.

5.1.1 Facial Wrinkles

Table 5.1 details the performance benchmarks of the facial wrinkles application, in frames per second (FPS). The data was obtained through the rendering of a scene using three directional light sources, a depth map resolution of 512 x 512, 3080 triangles per head and the final image being rendered at a resolution of 1280x800. The test consisted of rotating an animated head around the Y axis at a constant rate. The tests were run for 60 seconds and the frame rate sampled.

As we can see from the results in table 5.1, the addition of the wrinkles themselves is extremely fast, reducing overall performance by only 2.8% in our setup. Our use

Run	Test Parameters	FPS
1	Basic Diffuse Lighting & No Wrinkles	2942
2	Basic Diffuse Lighting & Wrinkles (Skinning Weights)	2860
3	Basic Diffuse Lighting & Wrinkles (Texture Masks)	2843
4	Basic Diffuse Lighting & Wrinkles & 1 Depth Pass	2694
5	Basic Diffuse Lighting & Wrinkles & 2 Depth Passes	2476
6	BSSRDF Lighting & Wrinkles	490
7	BSSRDF Lighting & Wrinkles $ SSAO	308

Table 5.1: Performance, in frames per second, of various configurations of the facial wrinkles algorithm.

of a number of early out clauses in the vertex shader together with the use of partial derivative normal maps did not even register on our test rig. They did however show a very small increase in performance in a much lower specification test machine. With this said, however, we are only rendering one model with a limited number of bones in a simple scene, these performance optimisations could prove very beneficial when rendering more complex scenes.

We can also see that the use of dynamic masks slightly outperforms the texture mask implementation. Though for our application this proved to be the case, one big advantage the texture masking approach has over using skinning weights is that it can be used to group bones together. If optimisations are implemented in this area, then as the number of bones scales upward this observed performance relationship would likely be reversed.

In attempting to benchmark the calculation of pose influences, carried out on the CPU, the performance impact proved negligible to the point that it could not be measured on our test rig using a single character. This proved a surprising result, and demonstrates the fact that bone-based wrinkle coefficient derivation is a viable alternative to triangle analysis. Thus, the method shows it is capability of being used as part of a technique to display wrinkles in a modern real-time application.

Looking again at the performance data we can see that the main cost of the algorithm is in computing the BSSRDF. This is to be expected as a large number of passes must be completed (six to twelve 1-D blur passes, one combination pass) in order to compute the final lighting model. Performance can improved at this phase

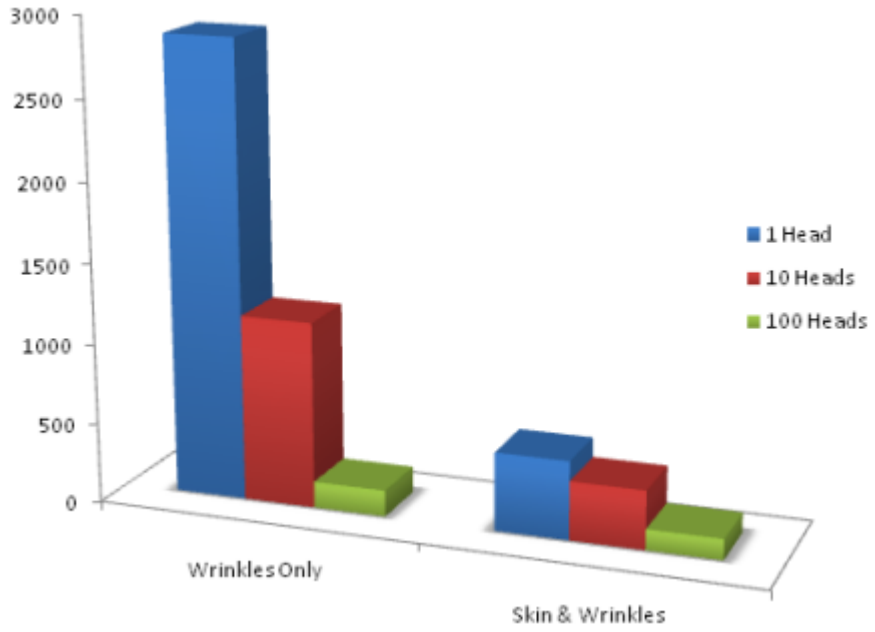

Figure 5.1: Graph modelling performance with increasing scene complexity.

through limiting the resolution of the render targets, however this does lead to a slight degradation of final image quality.

Finally, we examine how performance scales with increasing scene complexity (see Fig. 5.1). In order to perform this test a scene was rendered with increasing numbers of virtual characters displayed on-screen (1 head, 10 heads, 100 heads). For each case, the scene was rendered once with wrinkles only and once with the screen-space BSSRDF simulation and wrinkles. The tests were run for 60 seconds and the frame rate sampled. Taking the case where we are not using the skin shading model, we observe a near linear degradation (in fact it performs slightly better than this) in performance with increasing scene complexity. This observed O_n type behaviour that our algorithm exhibits is a desirable result and indicates that the algorithm does not have any major flaws in its design.

With the addition of the skin shading lighting model to the equation, we can clearly see the benefits of using Jimenez et al.'s screen-space skin rendering technique [3] over it's earlier texture space counterparts [27, 6]. As only one portion of the algorithm is required to be performed for every model in the scene, the performance impact of increasing scene complexity is minimised. In our test, the frame rate drops by 66% for

the rendering of 100 animated heads over just a single head.

5.1.2 Clothing Wrinkles

Table 5.1 details the performance benchmarks of the joint wrinkles application. The data was obtained through the rendering of a scene using three directional light sources, 8290 triangles per character and the final image being rendered at a resolution of 1280x800. The test conditions were identical to those outlined for the facial wrinkles performance test.

Run	Test Parameters	FPS
1	Render Scene	3777
2	Render Scene with Joint Wrinkles (2 Hinge, 1 Ball & Socket)	3771
3	Render Scene with Joint Wrinkles (100 Hinge, 50 Ball & Socket)	3702

Table 5.2: Performance, in frames per second, of the joint evaluation algorithm.

We can see that the performance impact of joint wrinkle weight evaluation for a single character with two hinge joints and a single ball and socket joint is 0.16%. In order to see how the algorithm would scale we ran the test again for a single character with one hundred hinge joints and fifty ball & socket joints, under this scenario performance impact figure rises to 1.99%

From these figures, it is reasonable to infer that the algorithm is more than suitable to be incorporated in to a real-time application with little or no modification from a performance standpoint.

5.2 Perceptual Evaluation

For the purposes of our perceptual evaluation, a simple test was designed in order to determine the following:

- If emotional content, simulated in virtual characters, can be better portrayed through the addition of expressive real-time wrinkles.
- Can this be further improved through the use of a realistic skin shading model.

A video was created containing forty eight, ten second, individual clips. The video was shown to a representative group of users who were asked to answer a number of questions based on the clips shown in the video.

In each clip a virtual character emotes happiness, anger, sadness or disgust. Taking into account that we need to determine both the effects of wrinkles and a realistic skin shading model, there exists a total of sixteen permutations. Each of these permutations has an associated clip and is displayed at least twice in the video. Users were asked to watch the video and perform the following tasks:

- Identify the virtual characters current emotional state.
- Rate the intensity of that emotion on a scale of 1 to 7. (1 := Almost no emotional intensity perceived; 7 := Very strong emotional intensity perceived)

As we can see from the results (Fig 5.2, Fig. 5.3) the addition of wrinkles to the character does appear to have an impact on user perception. The addition of wrinkles consistently, across every emotion, lead to an increase in reported intensity. The total average disparity in expressiveness averaged 1.3 points higher with wrinkles than without, without factoring in categorical recognition errors.

This disparity varies quite widely across the emotions. The sadness animation only displays quite subtle wrinkles, interestingly this is the emotion that had the least average increase in reported expressiveness through the addition of wrinkles. We must consider that the emotions being conveyed by the character are exaggerated expressions and so a high reported degree of expressiveness would be expected. With that said the 18.6% reported increase in expressiveness through the introduction of wrinkles would seem to be significant.

One other feature that became apparent, was that the addition of expressive wrinkles does not appear to have any impact on the categorical recognition of a virtual character's emotional state. Viewers made errors in categorical recognition with a similar frequency, regardless of the presence of wrinkles or not. Categorical recognition error averaged at 13% with anger and disgust the most frequently misjudged. This finding would appear to ally with the findings of Courgeon et al. [35], though similarly to that study we can not conclude with any certainty that wrinkles have no effect on categorical recognition due to the limitations of our test.

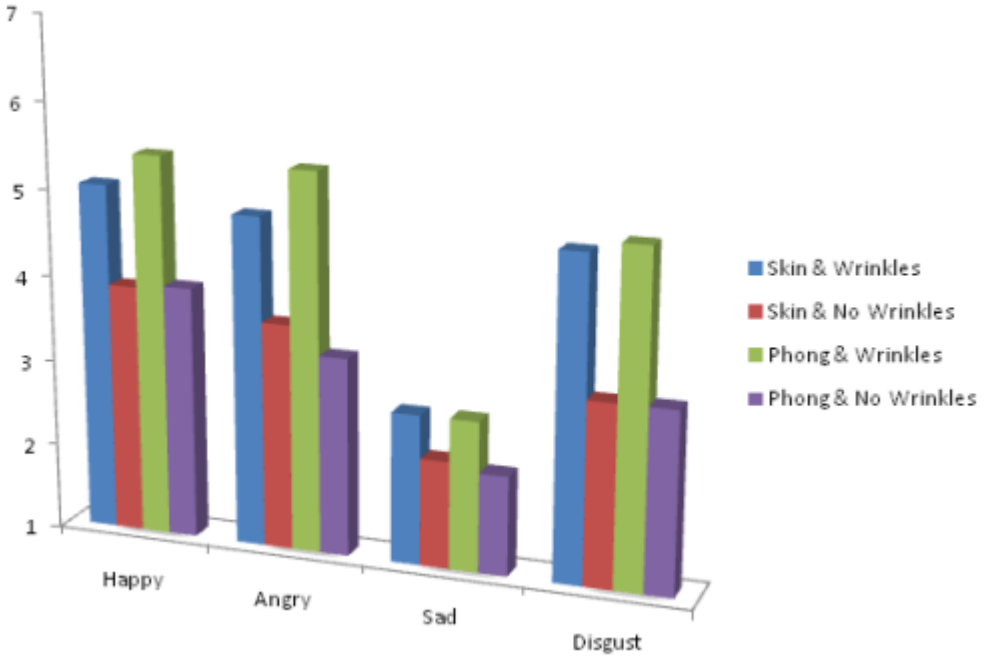

Figure 5.2: Graph modelling average user response to observed emotion under various shader permutations. All answers considered.

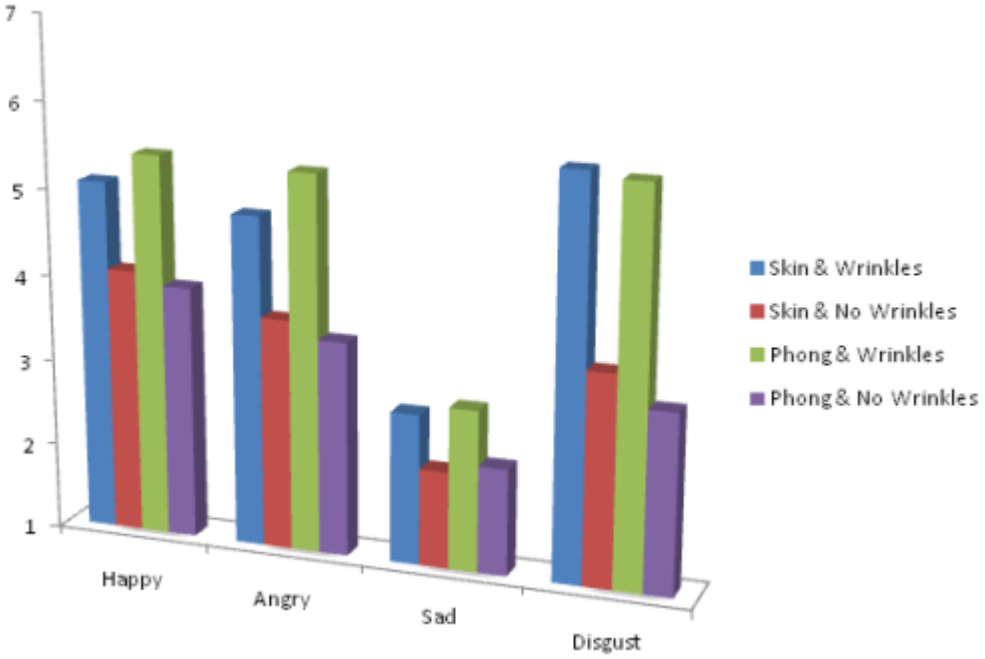

Figure 5.3: Graph modelling average user response to observed emotion under various shader permutations. Answers considered only on correct categorical recognition.

The addition of a skin shader does not seem to offer any additional benefit in terms of user perception. This is not completely surprising, however, as it has been observed that visual realism is not the most important aspect in conveying expression in virtual characters [36, 37].

Improvements

A number of key changes and extensions could be made to enhance the results of future perception tests in this area:

- Only a male character was used, for a more complete test a female character could also be displayed.
- Characters of various ages should be used, as age plays a key role in the appearance of wrinkles.
- We show only one emotion at a time. This does not truly capture how people emote in real scenarios. Through the use of motion capture technology one could capture true data in order to drive the character animations. The use of this kind of data could not only provide more realistic animations, allowing viewers to better connect with the observed character, but could also allow the capture of scripted scenes where the character emotes during the course of, for example, a real conversation.
- A larger sample size of test participants.

We would suggest that the enhanced data that these changes could provide would lead to better insight into the true effect that the addition of wrinkles has from a perceptual standpoint.

Chapter 6

Conclusions

We have implemented a technique that renders real-time wrinkles on a traditionally animated mesh utilizing additional data readily available from the model being rendered (i.e., the current positional data of the underlying skeletal structure).

As shown in Chapter 5, bone-based wrinkle weight derivation is very fast. Both facial pose evaluation and joint evaluation have a very minor performance impacts. Initially it was feared that, as an additional loop was required to be performed in the vertex shader for every bone to be examined for wrinkles, the performance of the algorithm would become prohibitive as it scale. This did not turn out to be the case, however, and the performance benefit of a number of early out clauses could only barely be measured. Overall, this proves that the technique is quite capable of being inserted into a modern, real-time application.

As an alternative to the use of skinning weights to create areas of influence for wrinkles, texture masking can be combined with bone-based weight derivation. This allows artists more complete control over the effect and can be used to reduce redundancy in the system.

We have demonstrated that the technique is capable of being used in disparate domains, through the use of a similar technique to render wrinkles around the articulated joints of virtual characters. This shows that at it's core, our technique provides an adjustable, extensible methodology to apply real-time wrinkles to an animated mesh in a variety of domains.

6.1 Summary

The main goals of this thesis, as set out in Chapter 1, have been achieved. A method has been outlined that can add emotional information, through the addition of facial wrinkles and furrows, to virtual characters animated using matrix palette skinning. We show that, with little modification, our technique can be used in other domains than simply facial wrinkles. We further show that the benefits of bone-based wrinkle coefficient derivation and the texture masking technique described by Oat [2] can be brought together to provide a potentially attractive solution.

6.2 Future Work

We set out potential further improvements and extensions that could be made to the technique.

6.2.1 Tessellation

With the introduction of Microsoft Direct3D 11 [38, 39] tessellation is now universally supported across all hardware platforms designed for this generation and has become a huge area of innovation in modern real-time graphics.

At it's simplest, tessellation involves dicing a polygon into a number of smaller polygons. On it's own, this may not seem like a particularly powerful concept but when combined with displacement mapping the possibilities open up rather quickly. A displacement map is a texture that stores height information for a model. When applied to a surface, the mesh's vertices are displaced up or down based on the height information in the map. Displacement mapping has been around for a long time but has never really caught on, for the simple reason that in order for it to be effective, the number of polygons that a mesh would have to contain would prove prohibitive. With the advent of hardware tessellation this problem has been largely overcome.

One significant problem when using normal mapping is that the model's underlying silhouette is not changed by the technique. Normal mapping creates only the illusion of "bumpy" surfaces, and is only truly partially successful in this. Through the use of displacement mapping and hardware tessellation in the place of normal mapping, the

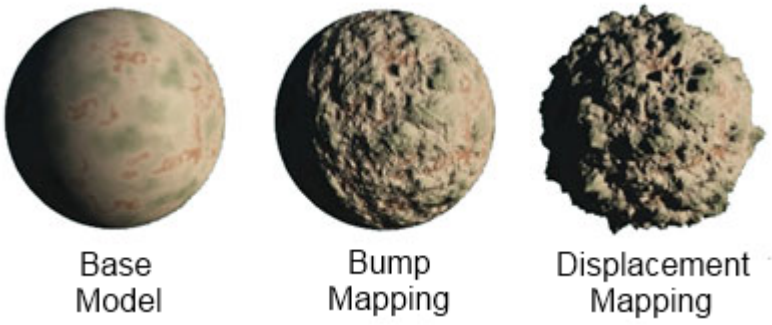

Figure 6.1: Images rendered of sphere using texture mapping, normal mapping and displacement mapping

actual desired geometry can be displayed in real-time.

Our technique could certainly be extended through the use of tessellation to achieve a more realistic final result. It may even be possible to re-examine some of the length preserving techniques mentioned earlier, section 2.1.5, as a genuine solution to achieving realistic real-time wrinkles.

6.2.2 Generalisation

As discussed in Section 4.3.4, our solution for ball and socket joints requires a certain amount of user tuning for it to be successfully applied to a mesh. We believe that there is a potential solution to the problem.

Our implementation utilizes static masks, stored as textures, to define areas of influence on the mesh upon which wrinkles occur. With some careful setup, it may be possible to move these mask areas in real-time, based upon joint orientation. Offsetting of this sort is a simple operation to perform from a technical standpoint and may provide a smoother, more convincing result.

The drawback here is that texture assets would have to be set up in a very specific way. Generally speaking, areas that may be close to each other in 3D space may be entirely disconnected in texture space. For any kind of mask offsetting solution to work, texture masks would likely be required to be authored so that joint masking areas remain close in both texture space and 3D space.

6.2.3 Incongruent Areas of Influence

As frequently mentioned, the quality of assets is critical to achieving the desired result. Currently, if using two or more wrinkle maps that cover multiple areas of influence, it is essential that wrinkle maps are set up so that they complement each other. One issue that can arise if this is not the case is that, as bones move in opposite directions, incongruent wrinkle areas can appear on the mesh. Looking at the image in Fig. 6.2, we can observe this happening. This is a particularly extreme case, but demonstrates the possible issues that could arise.

Figure 6.2: Wrinkle map incongruence due to facial bones moving in opposite directions.

It should be possible to solve this problem even in the case of uncomplimentary wrinkle maps. One possible solution to this problem would be to compare adjacent bones and ensure that if they are moving in opposite directions that only one wrinkle map is sampled (i.e., the map associated with the bone who's relative movement is greater).

This solution has some major issues however. Firstly, there will still be an obvious *seam* where the weight of the wrinkles drops from one area to another. Secondly, the wrinkles that are formed, while providing a far better result than those in Fig. 6.2, may not provide perceptually believable wrinkles. Finally, another degree of complexity is added, which is never desirable.

Overall, it is far better to ensure that the initial, artist authored, wrinkle maps complement each other so as to avoid this issue completely.

Appendix - Perception Test Form

Original Date:

Perception Test – Evaluation Form

For each clip please circle the intensity level of the emotion perceived.

1	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7	**...**	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7
2	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7	**...**	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7
3	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7	**...**	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7
...	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7	**...**	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7
...	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7	**...**	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7
...	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7	**46**	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7
...	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7	**47**	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7
...	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7	**48**	• Happy 1 – 2 – 3 – 4 – 5 – 6 – 7 • Angry 1 – 2 – 3 – 4 – 5 – 6 – 7 • Sad 1 – 2 – 3 – 4 – 5 – 6 – 7 • Disgust 1 – 2 – 3 – 4 – 5 – 6 – 7

Appendix - Perception Test Results

3-Way Repeated Measures Analysis Of Variance (ANOVA)

Statistical results
Main effect of emotion Current effect: F(3, 21)=24.722, p=.00000 Sad considered significantly less intense than all others (p<0.0002 in all cases) **Main effect of the Addition of Wrinkles** Current effect: F(1, 7)=21.349, p=.00242 emotions with wrinkles considered more intense in general (p<0.003) No effect of the rendering style used **Interaction between wrinkles and emotion** Current effect: F(3, 21)=13.929, p=.00003 All emotions were considered more intense with wrinkles than without, except for Sadness where there was no difference

Bibliography

[1] Ludovic Dutreve, Alexandre Meyer, and Sada Bouakaz. Real-time dynamic wrinkles of face for animated skinned mesh. In *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part II*, ISVC '09, pages 25–34, Berlin, Heidelberg, 2009. Springer-Verlag.

[2] Christopher Oat. Real-time wrinkles. In *Course 28: Advanced Real-Time Rendering in 3D Graphics and Games*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[3] Jorge Jimenez, Veronica Sundstedt, and Diego Gutierrez. Screen-space perceptual rendering of human skin. *ACM Transactions on Applied Perception*, 6:23:1–23:15, October 2009.

[4] Jorge Jimenez, David Whelan, Veronica Sundstedt, and Diego Gutierrez. Real-time realistic skin translucency. *IEEE Computer Graphics and Applications*, 30(4):32–41, 2010.

[5] Eric Penner and George Borshukov. *GPU Pro 2*, chapter Pre-Integrated Skin Shading. AK Peters Ltd., 2011.

[6] Eugene d'Eon, David Luebke, and Eric Enderton. A system for efficient rendering of human skin. In *ACM SIGGRAPH 2007 sketches*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[7] Jorge Jimenez, Jose I. Echevarria, Christopher Oat, and Diego Gutierrez. *GPU Pro 2*, chapter Practical and Realistic Facial Wrinkles Animation. AK Peters Ltd., 2011.

[8] Jun'ichiro Seyama and Ruth S. Nagayama. The uncanny valley: Effect of realism on the impression of artificial human faces. *Presence: Teleoperators and Virtual Environments*, 16:337–351, August 2007.

[9] Microsoft DirectX 10 SDK Team. Sparse morph targets sample, 2007.

[10] Guillaume B. Duchenne. *De l'lectrisation localise et de son application a la pathologie et a la thrapeutique*. 1855.

[11] Christopher Oat. Animated wrinkle maps. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 33–37, New York, NY, USA, 2007. ACM.

[12] James F. Blinn. Simulation of wrinkled surfaces. *SIGGRAPH Computer Graphics*, 12:286–292, August 1978.

[13] Clausius Duque, G. Reis, Jos Mario, De Martino, and Harlen Costa Batagelo. Real-time simulation of wrinkles.

[14] Ludovic Dutreve, Alexandre Meyer, and Sada Bouakaz. Easy acquisition and real-time animation of facial wrinkles. *Compututer Animation and Virtual Worlds*, 22:169–176, April 2011.

[15] Natalya Tatarchuk. Dynamic parallax occlusion mapping with approximate soft shadows. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, I3D '06, pages 63–69, New York, NY, USA, 2006. ACM.

[16] Fábio Policarpo, Manuel M. Oliveira, and João L. D. Comba. Real-time relief mapping on arbitrary polygonal surfaces. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 935–935, New York, NY, USA, 2005. ACM.

[17] Caroline Larboulette and Marie-Paule Cani. Real-time dynamic wrinkles. In *Proceedings of the Computer Graphics International*, pages 522–525, Washington, DC, USA, 2004. IEEE Computer Society.

[18] Ming Li, BaoCai Yin, DeHui Kong, and XiaoNan Luo. Modeling expressive wrinkles of face for animation. In *Proceedings of the Fourth International Conference on Image and Graphics*, ICIG '07, pages 874–879, Washington, DC, USA, 2007. IEEE Computer Society.

[19] Yu Wang A, Charlie C. L. Wang B, and Matthew M. F. Yuen A. Fast energy-based surface wrinkle modeling.

[20] Matthias Müller and Nuttapong Chentanez. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 85–92, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[21] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18:109–118, April 2007.

[22] James F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Computer Graphics*, 11:192–198, July 1977.

[23] Henri Gouraud. *Continuous shading of curved surfaces*, pages 87–93. ACM, New York, NY, USA, 1998.

[24] Michael Oren and Shree K. Nayar. Generalization of the lambertian model and implications for machine vision. *International Journal of Computer Vision*, 14:227–251, April 1995.

[25] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Radiometry. chapter Geometrical considerations and nomenclature for reflectance, pages 94–145. Jones and Bartlett Publishers, Inc., , USA, 1992.

[26] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 511–518, New York, NY, USA, 2001. ACM.

[27] George Borshukov and J. P. Lewis. Realistic human face rendering for "the matrix reloaded". In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[28] Craig Donner and Henrik Wann Jensen. Light diffusion in multi-layered translucent materials. *ACM Transactions on Graphics.*, 24:1032–1039, July 2005.

[29] Martin Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 97–121, New York, NY, USA, 2007. ACM.

[30] Mike Acton. *Ratchet and Clank Future: Tools of Destruction Technical Debriefing.* 2008.

[31] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18:311–317, June 1975.

[32] Csaba Kelemen, Laszlo Szirmay-Kalos, and Lszl Szirmay-kalos. A microfacet based coupled specular-matte brdf model with importance sampling, 2001.

[33] Kenneth E. Torrance and Ephraim M. Sparrow. Radiometry. chapter Theory for off-specular reflection from roughened surfaces, pages 32–41. Jones and Bartlett Publishers, Inc., , USA, 1992.

[34] Christophe Schlick. An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum*, 13:233–246, 1994.

[35] Matthieu Courgeon, Stéphanie Buisine, and Jean-Claude Martin. Impact of expressive wrinkles on perception of a virtual character's facial expressions of emotions. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, pages 201–214, Berlin, Heidelberg, 2009. Springer-Verlag.

[36] Maia Garau, Mel Slater, Vinoba Vinayagamoorthy, Andrea Brogni, Anthony Steed, and M. Angela Sasse. The impact of avatar realism and eye gaze control on perceived quality of communication in a shared immersive virtual environment. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, pages 529–536, New York, NY, USA, 2003. ACM.

[37] Vinoba Vinayagamoorthy, Anthony Steed, and Mel Slater. *Building Characters: Lessons Drawn from Virtual Environments*, pages 119–126. Cognitive Science Society, 2005.

[38] Allison. Klein. Introduction to the direct3d 11 graphics pipeline. In *Gamefest 2008*, 2008.

[39] Kevin Gee. Direct3d 11 tessellation. In *Gamefest 2008*, 2008.