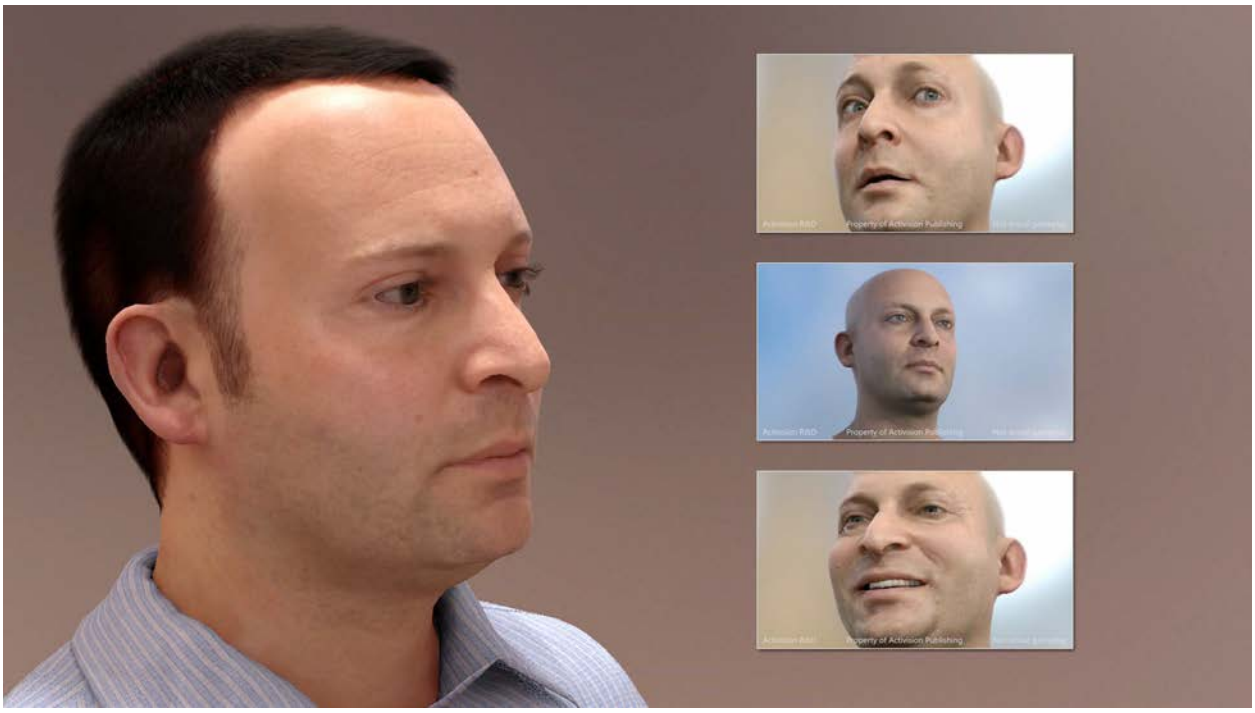


Digital Ira and Beyond:
Creating Photoreal Real-Time Digital Characters

Summary Statement

This course explains a complete process for creating next-generation realtime digital human characters, using the Digital Ira collaboration between USC ICT and Activision as an example, covering highres facial scanning, blendshape rigging, video-based performance capture, animation compression, realtime skin and eye shading, hair, latest results, and future directions.

Short Overview

This course will present the process of creating "Digital Ira" seen at the SIGGRAPH 2013 Real-Time live venue, covering the complete set of technologies from high resolution facial scanning, blendshape rigging, video-based performance capture, animation compression, realtime skin and eye shading, and hair rendering. The course will also present and explain late-breaking results and refinements and point the way along future directions which may increase the quality and efficiency of this kind of digital

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
SIGGRAPH 2014, August 10 – 14, 2014, Vancouver, British Columbia, Canada.
2014 Copyright held by the Owner/Author.
ACM 978-1-4503-2962-0/14/08

character pipeline. The actor from this project was scanned in 30 high-resolution expressions from which eight were chosen for real-time performance rendering. Performance clips were captured using multi-view video. Expression UVs were interactively corresponded to the neutral expression, retopologized to an artist mesh. An animation solver creates a performance graph representing dense GPU optical flow between video frames and the eight expressions; dense optical flow and 3D triangulation are computed, yielding per-frame spatially varying blendshape weights approximating the performance. The performance is converted to standard bone animation on a 4k mesh using a bone-weight and transform solver. Surface stress values are used to blend albedo, specular, normal, and displacement maps from the high-resolution scans per-vertex at run time. DX11 rendering includes SSS, translucency, eye refraction and caustics, physically based two-lobe specular reflection with microstructure, DOF, antialiasing, and grain. The course will explain each of processes, mentioning why each design choice was made and pointing to alternative components which may have been employed in place of any of the steps. We will also cover emerging technologies in performance capture and facial rendering. Attendees will receive a solid understanding of the techniques used to create photoreal digital characters in video games and other applications, and the confidence to incorporate some of the techniques into their own pipelines.

Project URL

http://gl.ict.usc.edu/Research/DigitalIra/

Intended Audience

Digital Character Artists, Game Developers, Texture Painters, and Researchers working on Performance Capture, Facial Modeling, and Real-Time Shading research

Prerequisites

Some experience with video game pipelines, facial animation, and shading models. The course is designed so that attendees with a wide range of experience levels will take away useful information and lessons from the course.

Course Schedule

1. Introduction/Overview - von der Pahlen

2. Facial Scanning and Microgeometry Capture - Debevec

3. Facial scan correspondence with Vuvuzela (live demo) - Alexander

4. Performance capture and animation solving - Fyffe

5. Compressing animation to a bone rig - Danvoye

6. Skin shading - Jimenez

7. Driving Expression Blending - Danvoye

8. Rendering Eyes - Jimenez

9. Rendering Hair - Jimenez

10. Latest Results and Future Work - von der Pahlen

12. Q&A - All

Instructor Bios:

JAVIER VON DER PAHLEN is Director or R&D at Activision Central Studios, leading a photreal character program since 2009. Javier started working on computer graphics in the Architecture program at Cornell University in the late 80s. Before joining Activision he co-created Softimage Face Robot in 2005, the first face commercially available facial animation software.

JORGE JIMENEZ is a real-time graphics researcher at Activision Blizzard. He received his PhD degree in Real-Time Graphics from Universidad de Zaragoza (Spain) in 2012. His interests include real-time photorealistic rendering, special effects, and squeezing rendering algorithms to be practical in game environments. He has contributions in conferences, books, and journals, including SIGGRAPH and GDC, the GPU Pro series, the Game Developer magazine, and the journal Transaction on Graphics. He co-organized the course "Filtering Approaches for Real-Time Anti-Aliasing at SIGGRAPH 2011. Some of his key achievements include Jimenez's MLAA, SMAA, and the separable subsurface scattering technique.

ETIENNE DANVOYE joined Activision Central Studio's R&D team in 2009. He has been involved in improving every step of the pipeline for realistic characters, from the high resolution scanning hardware to the tools to process the animation and texture data into a runtime-ready form. Before that, he spent seven years at Artificial Mind&Movement (now Behavior Interactive) as Lead Engine Programmer, with focus on animation, particles and physics. Areas of expertise include animation engines, and efficient game engine pipelines.

PAUL DEBEVEC is a Research Professor in the University of Southern California's Viterbi School of Engineering. He has worked on facial capture and rendering research beginning with his SIGGRAPH 2000 paper "Acquiring the Reflectance Field of the Human Face" which gave rise to the Light Stage systems recognized with an Academy Scientific and Engineering Award in 2010.

GRAHAM FYFFE is a computer scientist in the Graphics Lab of the USC Institute for Creative Technologies. He previously worked at Sway Studio in Los Angeles, CA, during which time he received a Visual Effects Society award in 2007 for Outstanding Visual Effects in a Music Video. He received his masters in computer science at the University of New Brunswick, Canada, which gave him a background in computer graphics and artificial intelligence. His research interests include computer graphics, computer vision, and physics simulation, especially as applied towards visual effects. His recent work focuses on facial geometry scanning and performance capture.

OLEG ALEXANDER is a technical artist specializing in facial rigging and animation. He received his MFA in Computer Arts from Florida Atlantic University. From 2006 to 2009 he was lead technical artist at Image Metrics. During this time, Oleg created hundreds of facial rigs for film, game, and TV projects. He became an expert in the Facial Action Coding System, facial rigging, and facial animation. In 2008, he directed and rigged the Digital Emily project, a demo featuring a photorealistic CG facial performance. Currently, Oleg is a technical artist at USC Institute for Creative Technologies.

Digital Ira and Beyond: Creating Real-Time Photoreal Digital Actors

Oleg Alexander Graham Fyffe Jay Busch Xueming Yu Jorge Jimenez Etienne Danvoye Bernardo Antionazzi
Ryosuke Ichikari Andrew Jones Paul Debevec*
USC Institute for Creative Technologies
Mike Eheler Zybnek Kysela Javier von der Pahlen†
Activision, Inc.

Figure 1: *(Left) Three of eight high-res (0.1mm) light stage scans of the actor in static expressions. (Middle) Seven-camera HD performance recording. (Right) 180Hz video-driven blendshape model with screen-space subsurface scattering and advanced eye shading effects.*

Overview In 2008, the "Digital Emily" project [Alexander et al. 2009] showed how a set of high-resolution facial expressions scanned in a light stage could be rigged into a real-time photoreal digital character and driven with video-based facial animation techniques. However, Digital Emily was rendered offline, involved just the front of the face, and was never seen in a tight closeup. This SIGGRAPH 2014 Course will describe in detail the processes used by USC ICT and Activision to create the "Digital Ira" character shown at SIGGRAPH 2013's *Real-Time Live* venue, which achieved a real-time, largely photoreal digital human character which could be seen from any viewpoint, in any lighting, and could perform realistically from video performance capture even in a tight closeup. In addition, the character ran in a real-time game-ready production pipeline, ultimately achieving 180 frames per second for a full-screen character on a two-year old graphics card. For 2014, the course will show additional character examples, discuss lessons learned, and suggest directions for future work.

3D Scanning We began by scanning accomodating researcher Ari Shapiro in thirty high-resolution expressions using the USC ICT's Light Stage X system [Ghosh et al. 2011], producing 0.1mm resoution geometry and 4K diffuse and specular reflectance maps per expression. We chose eight expressions for the real-time performance rendering, maximizing the variety of fine-scale skin deformation observed in the scans. The expressions were merged onto an artistically built back-of-the head model. To record performances for the character, we shot seven views of 30fps video of the actor improvising lines using the same seven Canon 1Dx cameras used for the scans. We used a new tool called Vuvuzela to interactively and precisely correspond all expression texture (u,v) coordinates to the neutral expression, which was retopologized to a low-polygon clean artist mesh.

Performance Animation Our offline animation solver creates a performance graph from dense GPU optical flow between the video frames and the eight expressions. This graph gets pruned by analyzing the correlation between the video frames and the expression scans over twelve facial regions. The algorithm then computes dense optical flow and 3D triangulation yielding per-frame spatially varying blendshape weights approximating the performance.

The Game Rig To create the game-ready facial rig, we transferred the mesh animation to standard bone animation on a 4K polygon mesh using a bone weight and transform solver. The solver optimizes the smooth skinning weights and the bone animated transforms to maximize the correspondence between the game mesh and the reference animated mesh.

Real-Time Rendering The rendering technique uses surface stress values to blend diffuse texture, specular, normal, and displacement maps from the different high-resolution expression scans per-vertex at run time. As a result, realistic wrinkles appear around the actor's eyes when he squints and on his foreheard when he raises his eyebrows; the color of the skin also changes with expression due to shifting blood content. The DirectX11 rendering takes into account light transport phenomena happening in the skin and eyes, from large scale events like the reflection of light of the own face into the eyes, to the shadowing and occlusion happening in the skin pores. In particular, it includes separable subsurface scattering [Jimenez et al. 2012] in screen-space, translucency, eye refraction and caustics, advanced shadow mapping and ambient occlusion, a physically-based two-lobe specular reflection with microstructure, depth of field, post effects, temporal antialiasing (SMAA T2x), and film grain.

Acknowledgements (Omitted for review)

References

ALEXANDER, O., ROGERS, M., LAMBETH, W., CHIANG, M., AND DEBEVEC, P. 2009. Thedigital emily project: photoreal facial modeling and animation. In *ACM SIGGRAPH 2009 Courses*, ACM, New York, NY, USA, SIGGRAPH '09, 12:1–12:15.

GHOSH, A., FYFFE, G., TUNWATTANAPONG, B., BUSCH, J., YU, X., AND DEBEVEC, P. 2011. Multiview face capture using polarized spherical gradient illumination. *ACM Trans. Graph. 30*, 6 (Dec.), 129:1–129:10.

JIMENEZ, J., JARABO, A., GUTIERREZ, D., DANVOYE, E., AND VON DER PAHLEN, J. 2012. Separable subsurface scattering and photorealistic eyes rendering. In *ACM SIGGRAPH 2012 Courses*, ACM, New York, NY, USA, SIGGRAPH 2012.

*debevec@ict.usc.edu †Javier.Pahlen@activision.com

SIGGRAPH 2013

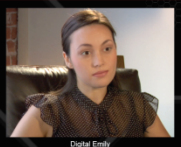
USC Institute for Creative Technologies

Digital Ira: Creating a Real-Time Photoreal Digital Actor

USC Institute for Creative Technologies

Overview

In 2008, the "Digital Emily" project [Alexander et al. 2009] showed how a set of high-resolution facial expressions scanned in a light stage could be rigged into a real-time photoreal digital character and driven with video-based facial animation techniques. However, Digital Emily was rendered offline, involved just the front of the face, and was never seen in a tight closeup. In this collaboration between Activision and USC ICT shown at SIGGRAPH 2013's Real-Time Live venue, we endeavoured to create a real-time, photoreal digital human character which could be seen from any viewpoint, in any lighting, and could perform realistically from video performance capture even in a tight closeup. In addition, we wanted this to run in a real-time game-ready production pipeline, ultimately achieving 180 frames per second for a full-screen character on a two-year old graphics card.



Digital Emily
[Alexander et al. 2009]

3D Scanning

Three of eight high-res (0.1mm) light stage scans of the actor in static expressions.

We began by scanning accomodating researcher Ari Shapiro in thirty high-resolution expressions using the USC ICT's Light Stage X system [Ghosh et al. 2011], producing 0.1mm resoution geometry and 4K diffuse and specular reflectance maps per expression. We chose eight expressions for the real-time performance rendering, maximizing the variety of fine-scale skin deformation observed in the scans. The expressions were merged onto an artistically built back-of-the head model. To record performances for the character, we shot seven views of 30fps video of the actor improvising lines using the same seven Canon 1Dx cameras used for the scans. We used a new tool called Vuvuzela to interactively and precisely correspond all expression texture (u,v) coordinates to the neutral expression, which was retopologized to a low-polygon clean artist mesh.

Performance Animation

Our offline animation solver creates a performance graph from dense GPU optical flow between the video frames and the eight expressions. This graph gets pruned by analyzing the correlation between the video frames and the expression scans over twelve facial regions. The algorithm then computes dense optical flow and 3D triangulation yielding per-frame spatially varying blendshape weights approximating the performance.

Seven-camera HD performance recording.

Game Rig

To create the game-ready facial rig, we transferred the mesh animation to standard bone animation on a 4K polygon mesh using a bone weight and transform solver. The solver optimizes the smooth skinning weights and the bone animated transforms to maximize the correspondence between the game mesh and the reference animated mesh, giving the most natural movement available.

Real-Time Rendering

Realtime DirectX11 rendering

The rendering technique uses surface stress values to blend diffuse texture, specular, normal, and displacement maps from the different high-resolution expression scans per-vertex at run time. As a result, realistic wrinkles appear around the actor's eyes when he squints and on his foreheard when he raises his eyebrows; the color of the skin also changes with expression due to shifting blood content.

Dynamic skin wrinkling

The DirectX11 rendering takes into account light transport phenomena happening in the skin and eyes, from large scale events like the reflection of light of the own face into the eyes, to the shadowing and occlusion happening in the skin pores. In particular, it includes separable subsurface scattering [Jimenez et al. 2012] in screen-space, translucency, eye refraction and caustics, advanced shadow mapping and ambient occlusion, a physically-based two-lobe specular reflection with microstructure, depth of field, post effects, temporal antialiasing (SMAA T2x), and film grain.

Caustics within eyes responding to enviornment

Acknowledgements

We thank Borom Tunwattanapong, Koki Nagano, Domi Piturro, Alejo von der Pahlen, Joe Alter, Curtis Beeson, Mark Daly, Mark Swain, Jen-Hsun Huang, Ari Shapiro, Valerie Dauphin, and Kathleen Haase for their important assistance and contributions to this work. This work was supported by USA RDECOM, USC, and Activision, Inc; no endorsement is implied.

Oleg Alexander Graham Fyffe Jay Busch Xueming Yu Ryosuke Ichikari
Andrew Jones Paul Debevec
USC Institute for Creative Technologies

Jorge Jimenez Etienne Danvoye Bernardo Antionazzi Mike Eheler
Zbynek Kysela Xian-Chun Wu Javier von der Pahlen
Activision Inc.

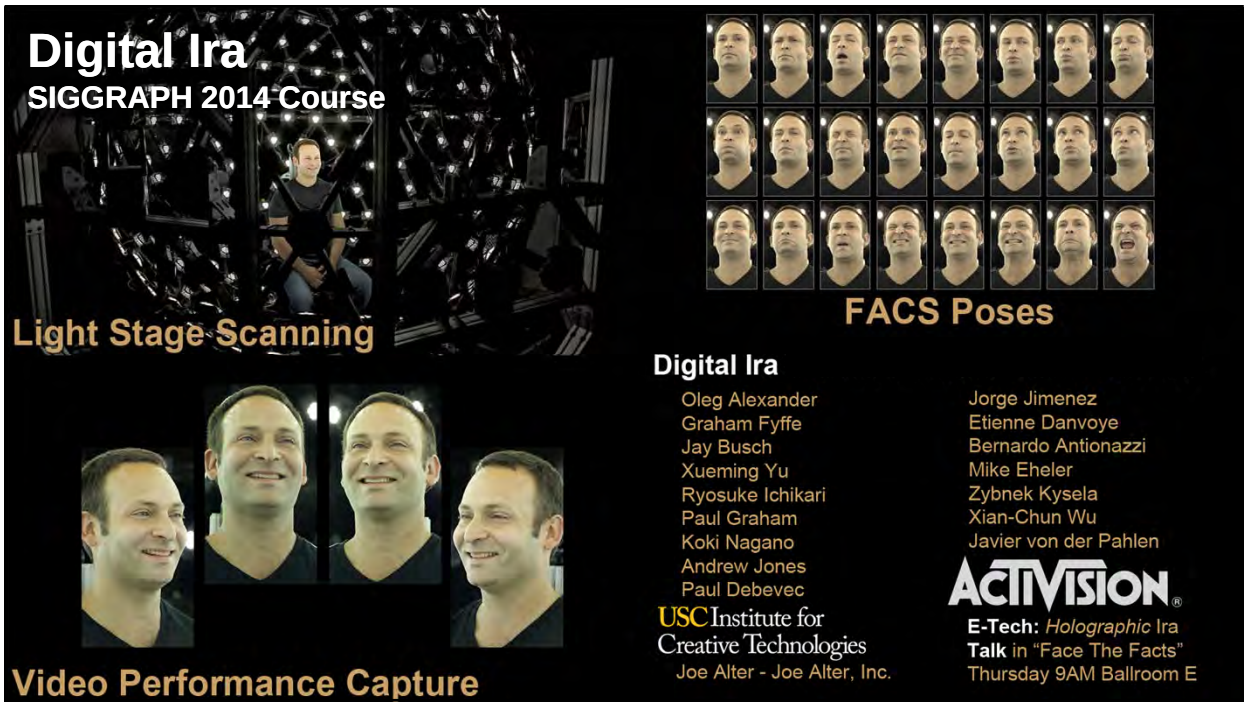
http://www.ict.usc.edu/

References:

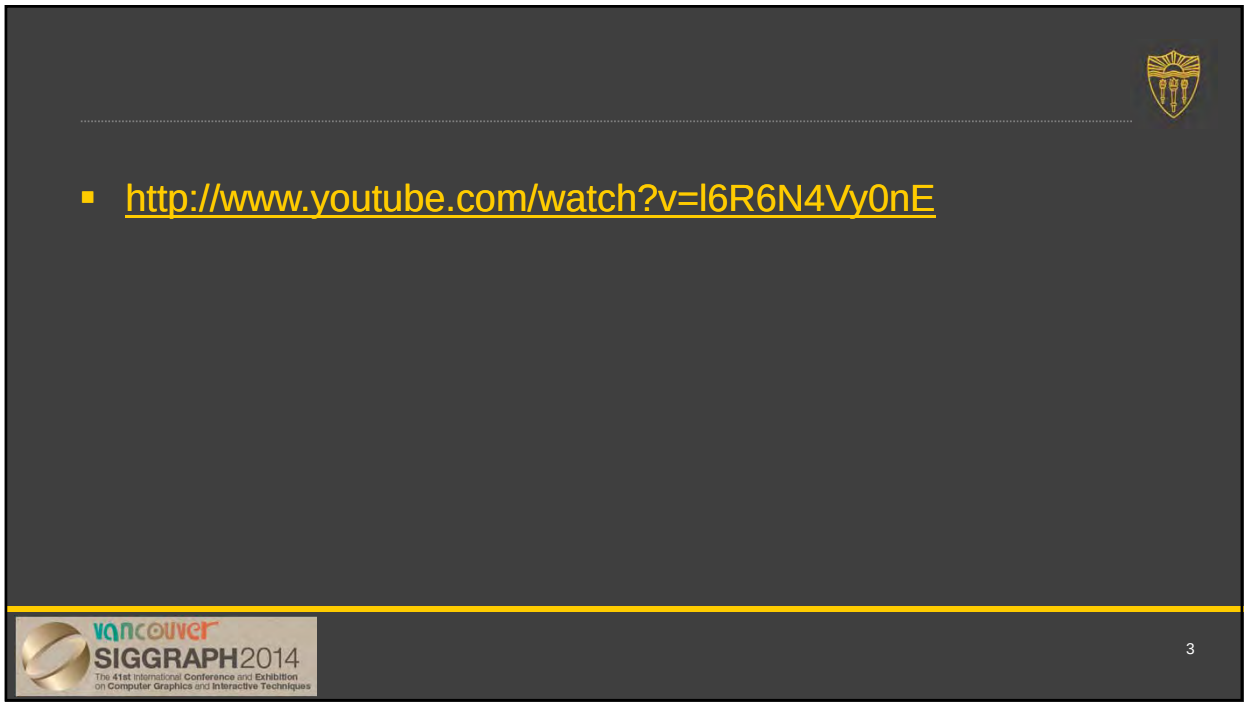
ALEXANDER, O., ROGERS, M., LAMBETH, W., CHIANG, M., AND DEBEVEC, P. 2009. The digital emily project: photoreal facial modeling and animation. In ACM SIGGRAPH 2009 Courses, ACM, New York, NY, USA, SIGGRAPH '09, 12:1–12:15.

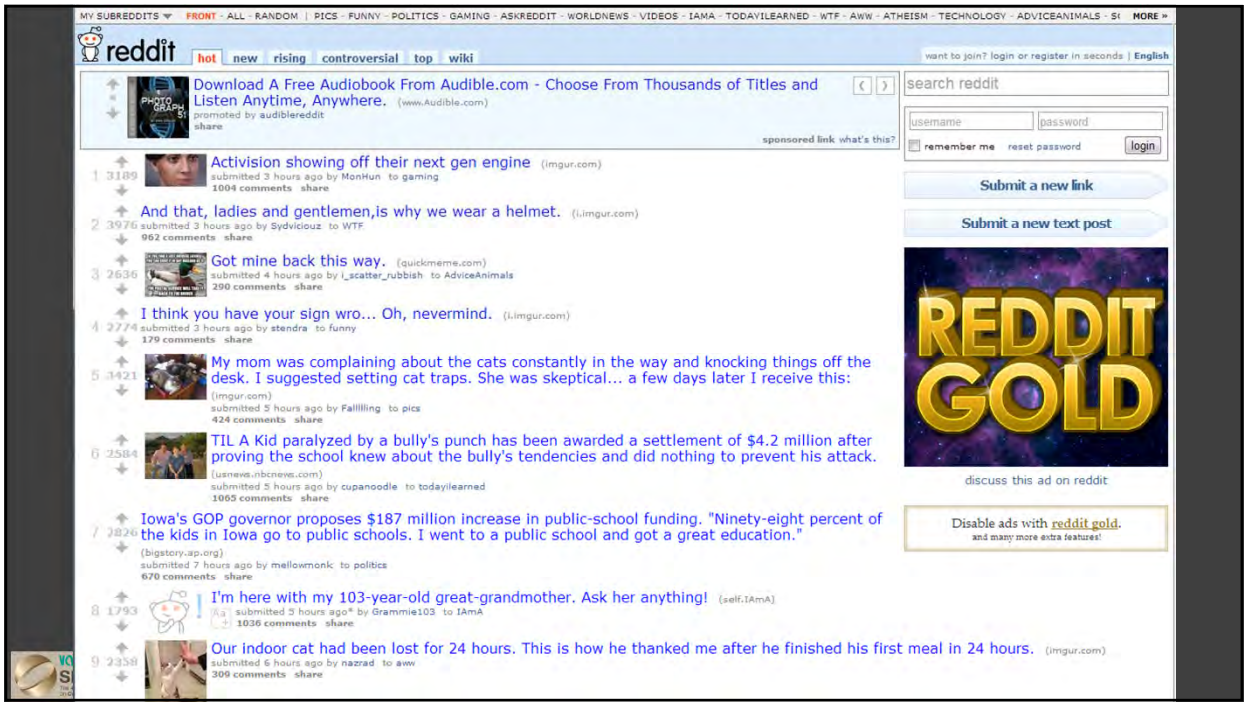
GHOSH, A., FYFFE, G., TUNWATTANAPONG, B., BUSCH, J., YU, X., AND DEBEVEC, P. 2011. Multiview face capture using polarized spherical gradient illumination. ACM Trans. Graph. 30, 6 (Dec.), 129:1–129:10.

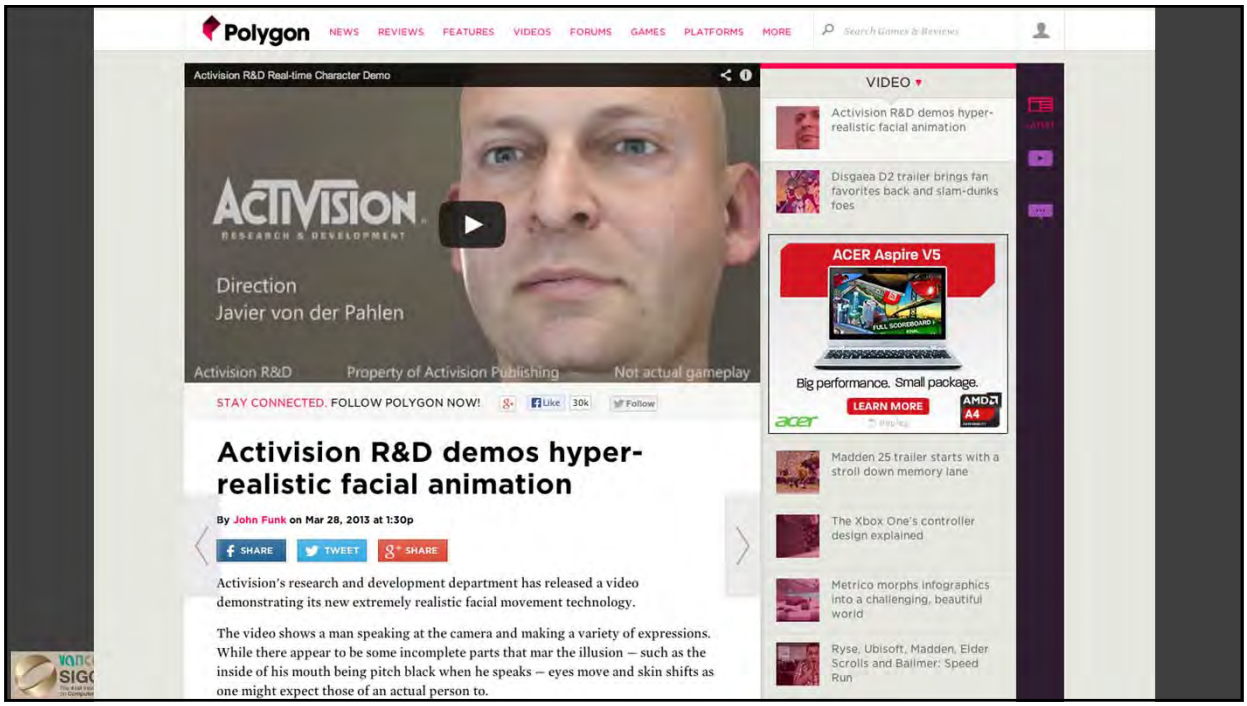
JIMENEZ, J., JARABO, A., GUTIERREZ, D., DANVOYE, E., AND VON DER PAHLEN, J. 2012. Separable subsurface scattering and photorealistic eyes rendering. In ACM SIGGRAPH 2012 Courses, ACM, New York, NY, USA, SIGGRAPH '12.

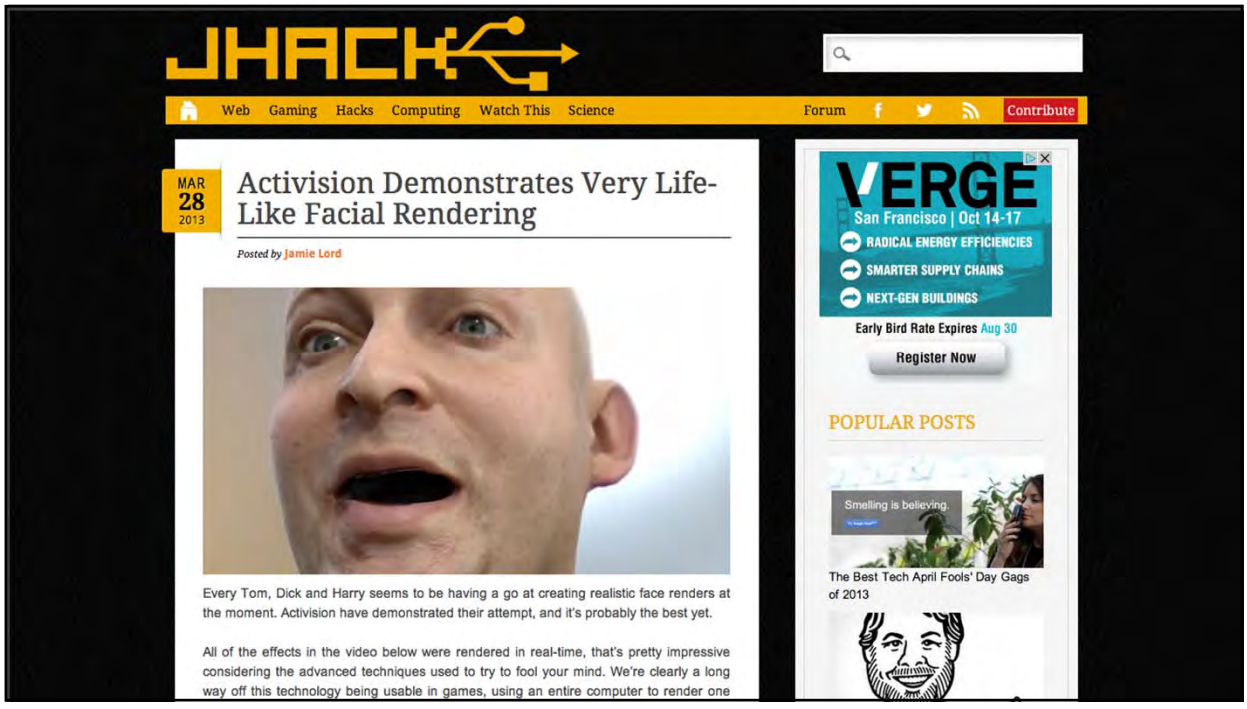

Digital Ira
SIGGRAPH 2014 Course
Light Stage Scanning
FACS Poses
Video Performance Capture
Digital Ira
Oleg Alexander
Graham Fyffe
Jay Busch
Xueming Yu
Ryosuke Ichikari
Paul Graham
Koki Nagano
Andrew Jones
Paul Debevec
USC Institute for Creative Technologies
Joe Alter - Joe Alter, Inc.
Jorge Jimenez
Etienne Danvoye
Bernardo Antionazzi
Mike Eheler
Zybnek Kysela
Xian-Chun Wu
Javier von der Pahlen
ACTIVISION
E-Tech: Holographic Ira
Talk in "Face The Facts"
Thursday 9AM Ballroom E

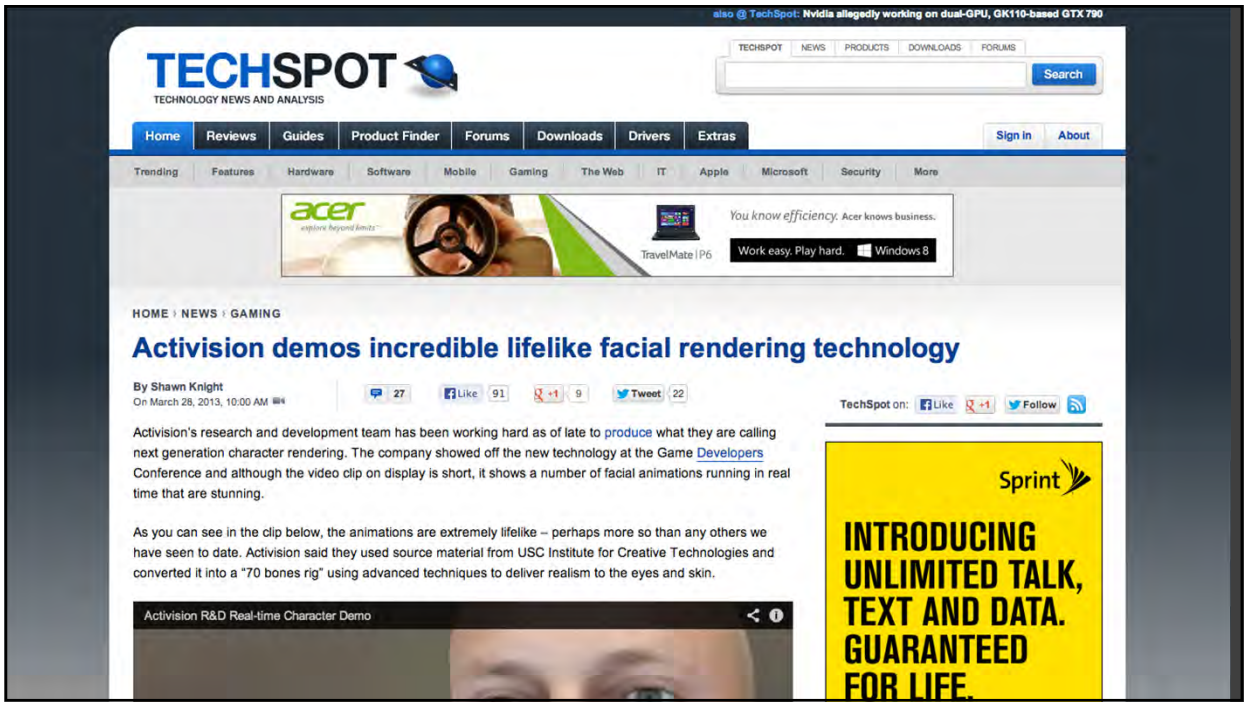

GDC 14
GAME DEVELOPERS CONFERENCE
SAN FRANCISCO, CA
MARCH 17-21, 2014
EXPO DATES: MARCH 19-21
2014
ABOUT
ATTEND
CONFERENCE
EXHIBIT
EVENTS
Session Scheduler
Next Generation Character Rendering
Speakers
PASS TYPE
TRACK
FORMAT
DAY
VANCOUVER
SIGGRAPH2014

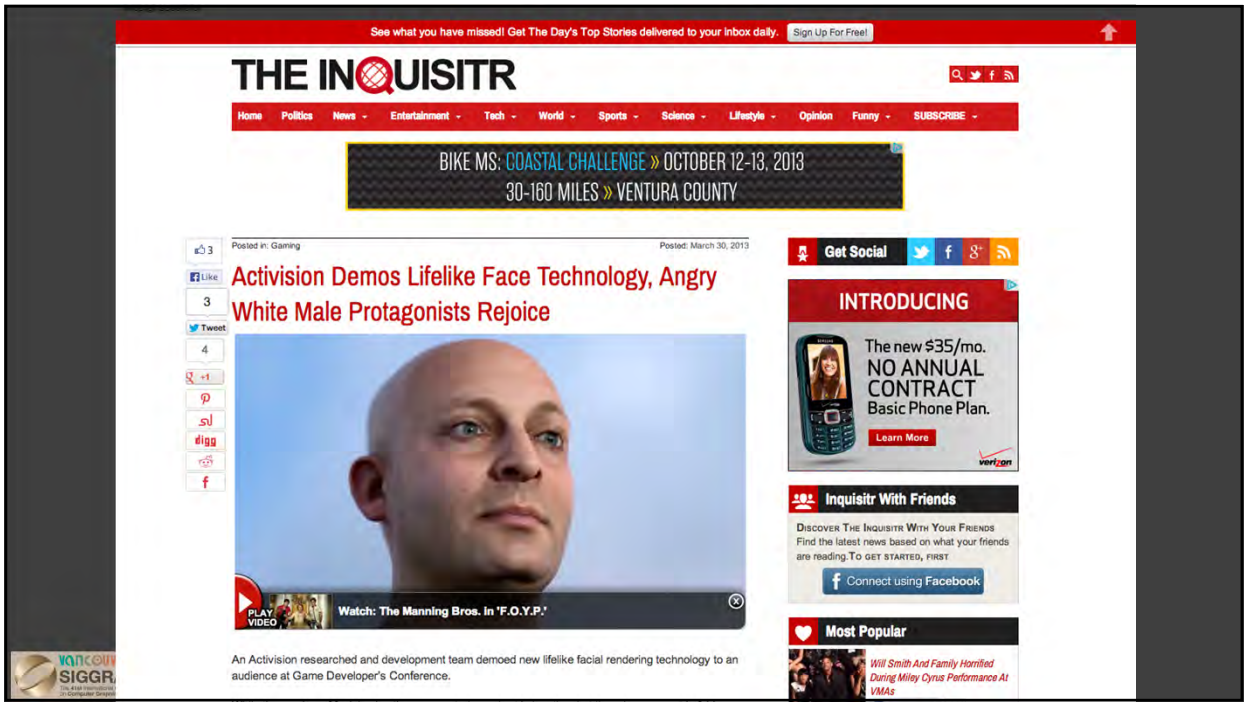

http://www.youtube.com/watch?v=l6R6N4Vy0nE
vancouver
SIGGRAPH2014
The 41st International Conference and Exhibition on Computer Graphics and Interactive Techniques
3


MY SUBREDDITS
FRONT - ALL - RANDOM | PICS - FUNNY - POLITICS - GAMING - ASKREDDIT - WORLDNEWS - VIDEOS - IAMA - TODAYILEARNED - WTF - AWW - ATHEISM - TECHNOLOGY - ADVICEANIMALS
MORE
reddit
hot new rising controversial top wiki
want to join? login or register in seconds | English
Download A Free Audiobook From Audible.com - Choose From Thousands of Titles and Listen Anytime, Anywhere. (www.Audible.com)
promoted by audiblereddit
share
sponsored link what's this?
search reddit
username password
remember me reset password
login
Submit a new link
Submit a new text post
Activision showing off their next gen engine (imgur.com)
submitted 3 hours ago by MonHun to gaming
1004 comments share
And that, ladies and gentlemen,is why we wear a helmet. (i.imgur.com)
submitted 3 hours ago by Sydvicious to WTF
962 comments share
Got mine back this way. (quickmeme.com)
submitted 4 hours ago by i_scatter_rubbish to AdviceAnimals
290 comments share
I think you have your sign wro... Oh, nevermind. (i.imgur.com)
submitted 3 hours ago by stendra to funny
179 comments share
My mom was complaining about the cats constantly in the way and knocking things off the desk. I suggested setting cat traps. She was skeptical... a few days later I receive this:
(imgur.com)
submitted 5 hours ago by Fallllling to pics
424 comments share
TIL A Kid paralyzed by a bully's punch has been awarded a settlement of $4.2 million after proving the school knew about the bully's tendencies and did nothing to prevent his attack.
(usnews.nbcnews.com)
submitted 5 hours ago by cupanoodle to todayilearned
1065 comments share
Iowa's GOP governor proposes $187 million increase in public-school funding. "Ninety-eight percent of the kids in Iowa go to public schools. I went to a public school and got a great education."
(bigstory.ap.org)
submitted 7 hours ago by mellowmonk to politics
670 comments share
I'm here with my 103-year-old great-grandmother. Ask her anything! (self.IAmA)
submitted 5 hours ago* by Grammie103 to IAmA
1036 comments share
Our indoor cat had been lost for 24 hours. This is how he thanked me after he finished his first meal in 24 hours. (imgur.com)
submitted 6 hours ago by nazrad to aww
309 comments share
REDDIT GOLD
discuss this ad on reddit
Disable ads with reddit gold.
and many more extra features!


Polygon
NEWS
REVIEWS
FEATURES
VIDEOS
FORUMS
GAMES
PLATFORMS
MORE
Activision R&D Real-time Character Demo
ACTIVISION
RESEARCH & DEVELOPMENT
Direction
Javier von der Pahlen
Activision R&D
Property of Activision Publishing
Not actual gameplay
VIDEO
Activision R&D demos hyper-realistic facial animation
Disgaea D2 trailer brings fan favorites back and slam-dunks foes
ACER Aspire V5
Big performance. Small package.
LEARN MORE
STAY CONNECTED. FOLLOW POLYGON NOW!
Activision R&D demos hyper-realistic facial animation
By John Funk on Mar 28, 2013 at 1:30p
SHARE
TWEET
SHARE
Activision's research and development department has released a video demonstrating its new extremely realistic facial movement technology.
The video shows a man speaking at the camera and making a variety of expressions. While there appear to be some incomplete parts that mar the illusion — such as the inside of his mouth being pitch black when he speaks — eyes move and skin shifts as one might expect those of an actual person to.
Madden 25 trailer starts with a stroll down memory lane
The Xbox One's controller design explained
Metrico morphs infographics into a challenging, beautiful world
Ryse, Ubisoft, Madden, Elder Scrolls and Ballmer: Speed Run


JHACK
Web
Gaming
Hacks
Computing
Watch This
Science
Forum
Contribute
MAR
28
2013
Activision Demonstrates Very Life-Like Facial Rendering
Posted by Jamie Lord
Every Tom, Dick and Harry seems to be having a go at creating realistic face renders at the moment. Activision have demonstrated their attempt, and it's probably the best yet.
All of the effects in the video below were rendered in real-time, that's pretty impressive considering the advanced techniques used to try to fool your mind. We're clearly a long way off this technology being usable in games, using an entire computer to render one
VERGE
San Francisco | Oct 14-17
RADICAL ENERGY EFFICIENCIES
SMARTER SUPPLY CHAINS
NEXT-GEN BUILDINGS
Early Bird Rate Expires Aug 30
Register Now
POPULAR POSTS
Smelling is believing.
The Best Tech April Fools' Day Gags of 2013


also @ TechSpot: Nvidia allegedly working on dual-GPU, GK110-based GTX 790
TECHSPOT
TECHNOLOGY NEWS AND ANALYSIS
Search
Home
Reviews
Guides
Product Finder
Forums
Downloads
Drivers
Extras
Sign in
About
Trending
Features
Hardware
Software
Mobile
Gaming
The Web
IT
Apple
Microsoft
Security
More
You know efficiency. Acer knows business.
Work easy. Play hard.
Windows 8
TravelMate P6
HOME › NEWS › GAMING
Activision demos incredible lifelike facial rendering technology
By Shawn Knight
On March 28, 2013, 10:00 AM
Activision's research and development team has been working hard as of late to produce what they are calling next generation character rendering. The company showed off the new technology at the Game Developers Conference and although the video clip on display is short, it shows a number of facial animations running in real time that are stunning.
As you can see in the clip below, the animations are extremely lifelike – perhaps more so than any others we have seen to date. Activision said they used source material from USC Institute for Creative Technologies and converted it into a "70 bones rig" using advanced techniques to deliver realism to the eyes and skin.
Activision R&D Real-time Character Demo
Sprint
INTRODUCING UNLIMITED TALK, TEXT AND DATA. GUARANTEED FOR LIFE.


See what you have missed! Get The Day's Top Stories delivered to your inbox daily.
Sign Up For Free!
THE INQUISITR
Home
Politics
News
Entertainment
Tech
World
Sports
Science
Lifestyle
Opinion
Funny
SUBSCRIBE
BIKE MS: COASTAL CHALLENGE » OCTOBER 12-13, 2013
30-160 MILES » VENTURA COUNTY
Posted in: Gaming
Posted: March 30, 2013
Activision Demos Lifelike Face Technology, Angry White Male Protagonists Rejoice
Watch: The Manning Bros. in 'F.O.Y.P.'
An Activision researched and development team demoed new lifelike facial rendering technology to an audience at Game Developer's Conference.
Get Social
INTRODUCING
The new $35/mo.
NO ANNUAL CONTRACT
Basic Phone Plan.
Learn More
Inquisitr With Friends
Discover The Inquisitr With Your Friends
Find the latest news based on what your friends are reading. To get started, first
Connect using Facebook
Most Popular
Will Smith And Family Horrified During Miley Cyrus Performance At VMAs

The hi-tech scanner that turns actors into aliens, warts and all

Kaya Burgess

In the cinemas and video games of the near future, computer-generated characters will be so lifelike that individual skin cells and hair follicles will be visible, because of a new high-definition form of digital animation technology.

The distinction between real-life actors and computer graphics has already been blurred by films such as *Beowulf* and *Avatar* — which used computer-generated imagery (CGI) to create animated versions of leading actors — but until now they risked seeming rather plastic-looking.

But thanks to new super-high resolution facial scanning you will now be able to see every blemish and crease in Angelina Jolie's virtual cheek or Zoe Saldana's digital forehead.

Researchers at the University of Southern California and Imperial College London have developed techniques to scan centimetre-square patches of skin from the cheek, forehead, nose, chin and temple in such high resolution that a single skin cell covers three pixels on the screen.

The team has also polarised the light source used during the scanning to pick up not only the light reflecting off the skin's surface but also light that penetrates below the epidermis and scatters back, providing greater depth and tone to the final image.

"The scanning, which uses high-resolution stills cameras in a laboratory, also captures how the skin behaves under different types of light and during different facial expressions. The scanned patches can then be mapped on to a 3-D image of the actor, created with motion-capture technology.

As a result, computer-generated characters will no longer be so "plastic-looking", according to Paul Debevec, the associate director of graphics research at USC, whose earlier techniques were used on James Cameron's *Avatar*. "The bumpiness of the surface of the skin, at the micron scale, actually affects how light reflects off the surface," Professor Debevec explained.

"That's what makes it look healthy or oily or pasty or chalky. It makes someone look like a human being made out of organic material and not like a computer-generated zombie."

To make *Avatar*, artists had to go back to the CGI imagery of the blue-skinned Na'vi characters and add blemishes, such as moles or creases, by hand. This vastly increased the man-hours and expense of the film, which was nearly 60 per cent computer-generated and cost more than £150 million.

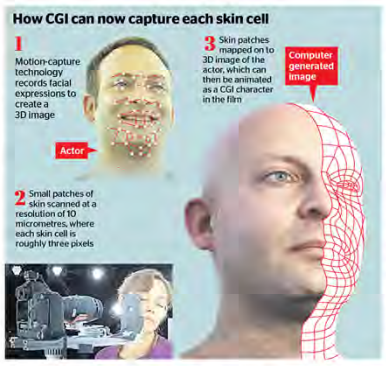
The process will now be much cheaper, Professor Debevec said, and video game developers at Activision have already created mathematical algorithms that can mimic many of the effects of the high-definition scanning, greatly reducing the time, expense and processing power needed.

This will allow hyper-realistic CGI characters to appear on video games consoles and could allow film directors to create CGI scenes in real time.

Professor Debevec said: "In the future it might be the less expensive movies that use CGI technology, while big budget movies will be the only ones who can still afford to go out on location and shoot in Paris or Bermuda and take up actors' time."

Abhijeet Ghosh, from the computing department at Imperial College London, helped to develop the "facial microgeometry scanning" process and was approached by the Avon cosmetics company to help it to analyse the effects of make-up on the skin.

He predicted that cosmetics customers may be able to use apps in future to see how their faces would look with different types of foundation. "When you start scanning skin at that scale, it could also have medical or dermatological applications," Dr Ghosh said.



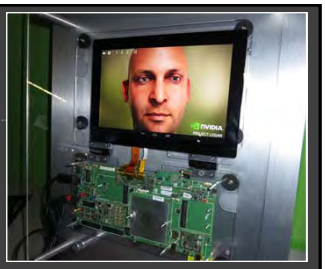
vancouver SIGGRAPH2014
The 41st International Conference and Exhibition on Computer Graphics and Interactive Techniques

- http://www.youtube.com/watch?v=SPeZNXmPzuI

vancouver SIGGRAPH2014
The 41st International Conference and Exhibition on Computer Graphics and Interactive Techniques

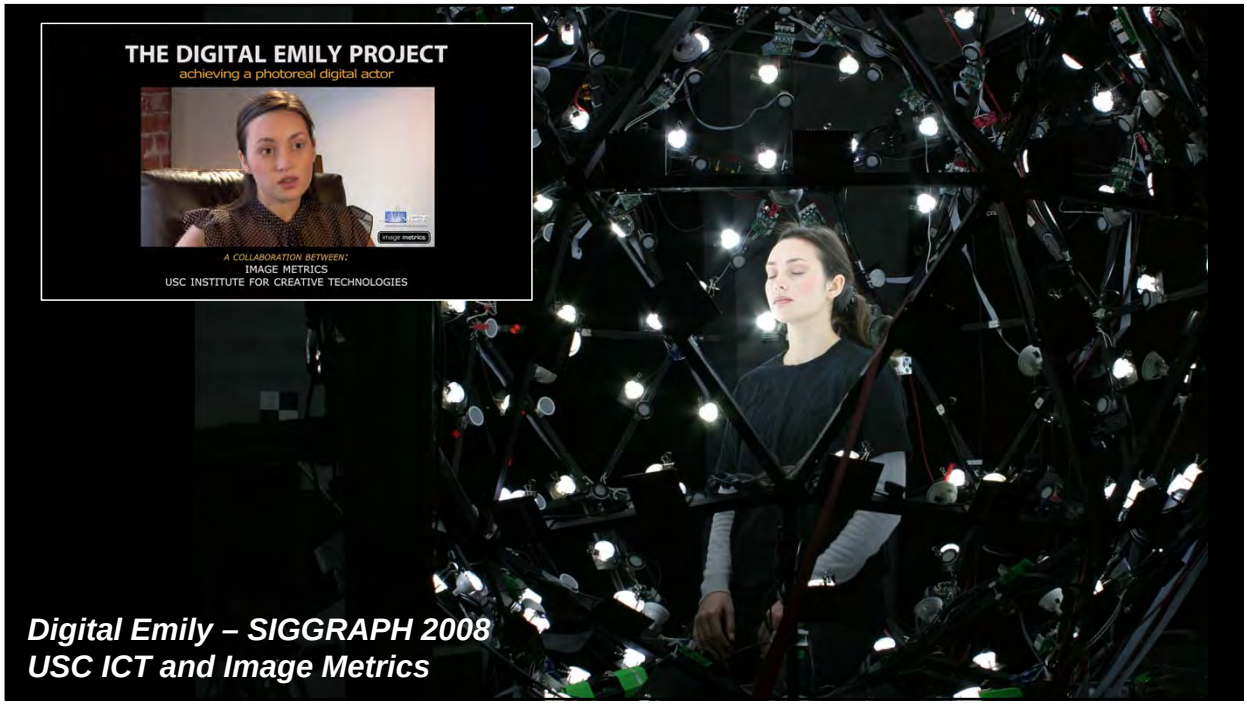
Digital Ira at SIGGRAPH 2013 Real-Time Live

- World's first (reasonably) photoreal real-time digital character, collaboratively developed with **Activision**, debuts at ***Real-Time Live*** to 2000+ attendees
- Leverages Graham et al.'s ***Measurement-Based Synthesis of Facial Microgeometry*** Eurographics 2013 for skin detail synthesis
- First version with hair!
- **NVIDIA** shows improved Digital Ira in their SIGGRAPH booth on 4K monitor, including Digital Ira running on their "Project Logan" tablet prototype
- ► Come to **"Digital Ira" Project Overview** at ICT Monday 8/26 featuring Graham Fyffe, Oleg Alexander, and Javier von der Pahlen



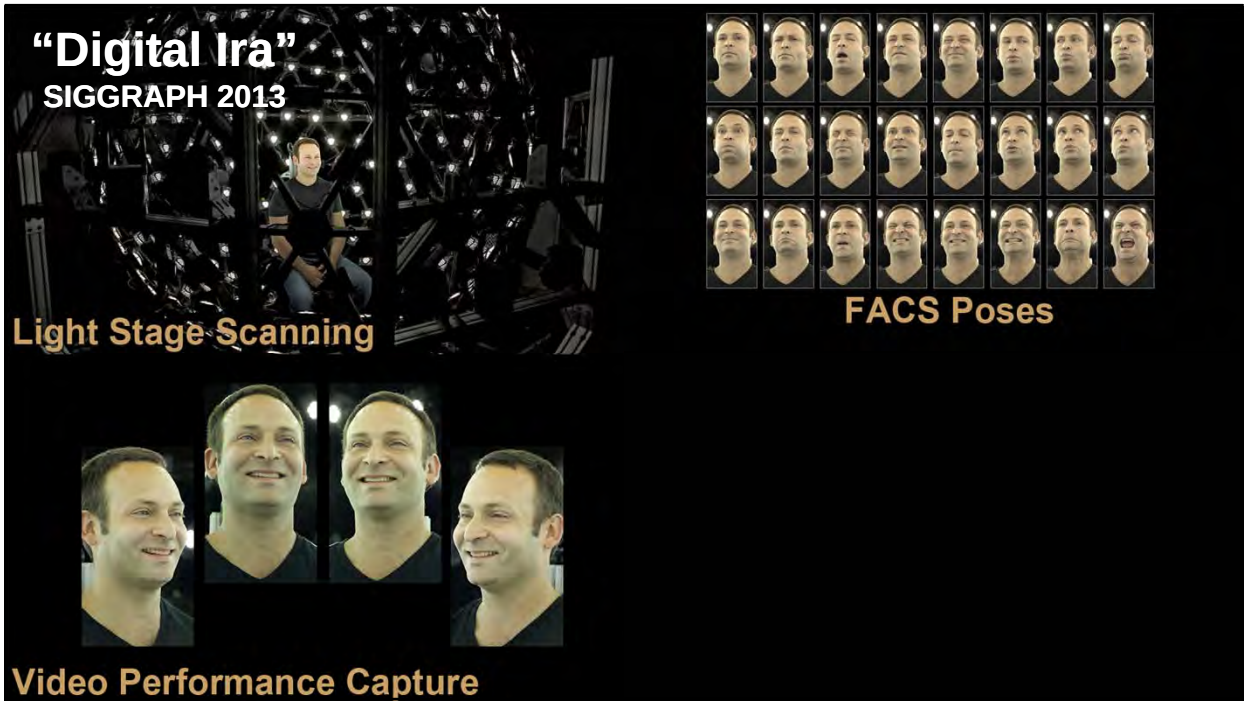
vancouver SIGGRAPH2014
The 41st International Conference and Exhibition on Computer Graphics and Interactive Techniques

11



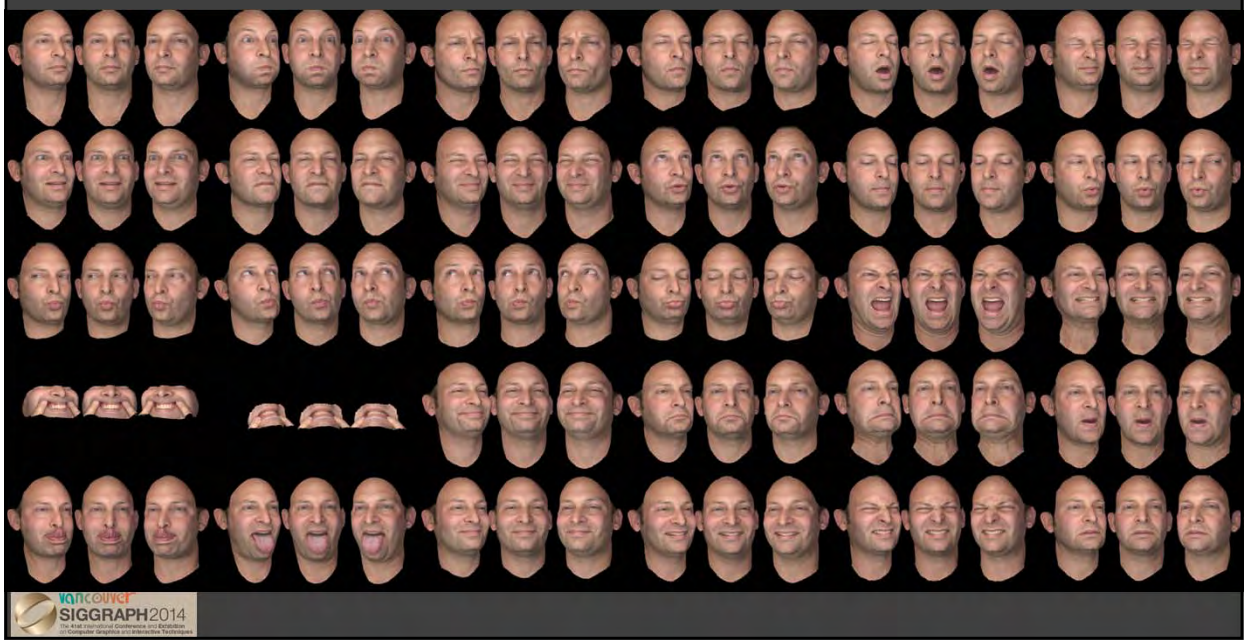
Digital Emily – SIGGRAPH 2008
USC ICT and Image Metrics


Light Stage X


"Digital Ira"
SIGGRAPH 2013
Light Stage Scanning
FACS Poses
Video Performance Capture

Latest Result: 30 High-Res Expressions Processed in One Week

Latest Result: 30 High-Res Expressions Processed in One Week



Measurement-Based Synthesis of Facial Microgeometry

Paul Graham, Borom Tunwattanapong, Jay Busch, Xueming Yu, Andrew Jones, Paul Debevec, Abhijeet Ghosh

USC Institute for Creative Technologies

Presented at Eurographics 2013 May 6-10, Girona (Spain)

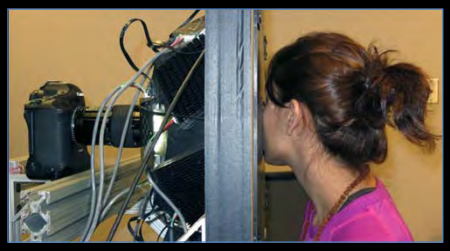
Rendering from Multi-view Scan

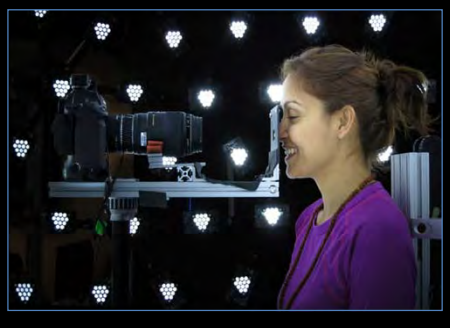
Photograph

Rendering with Enhance Microstructure

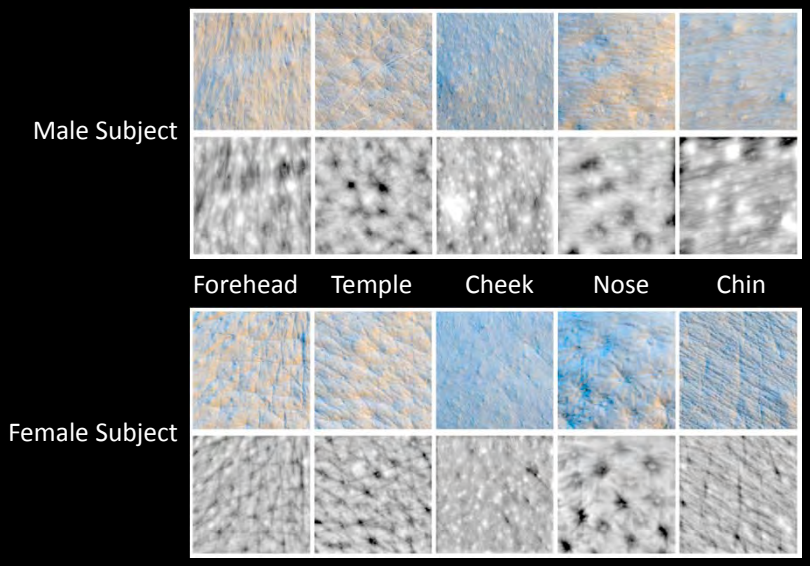
Recording skin microstructure

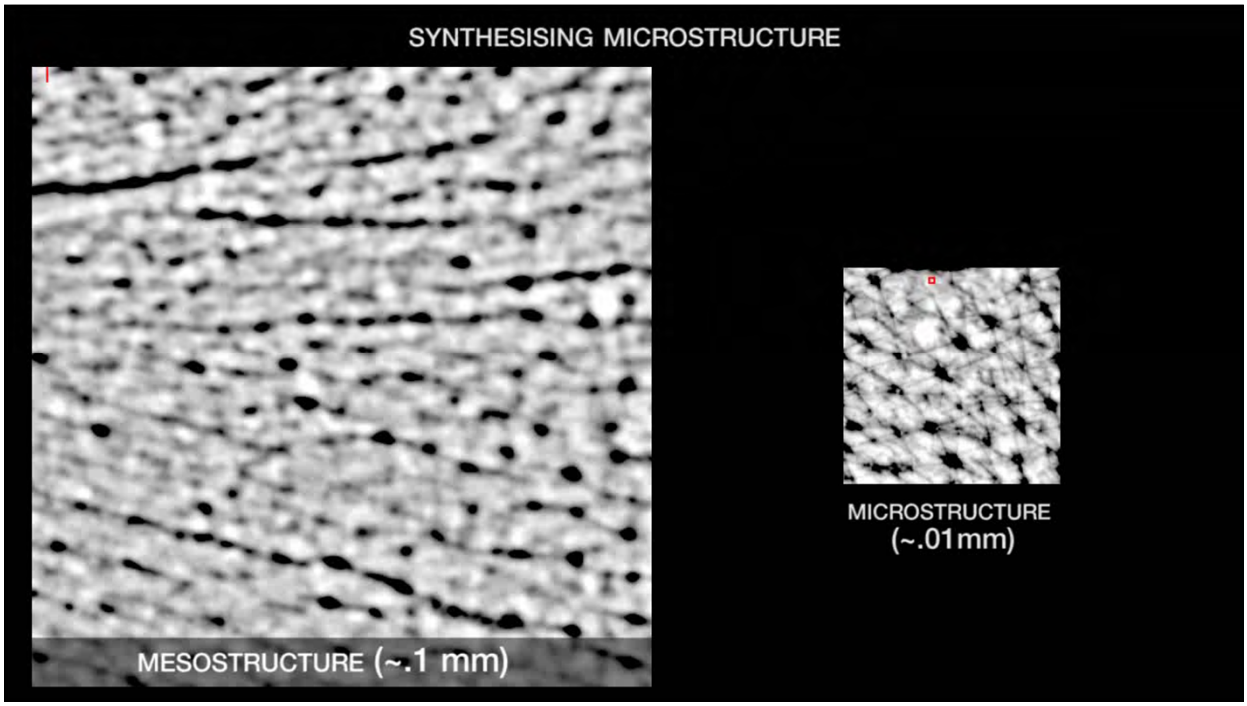
- 12-light hemispherical dome or
- Polarized LED sphere
  - higher lighting resolution for specular/oily skin
- Canon 1DMark III camera
  - Canon 100mm macro lens
- 24mm by 16mm aperture
  - 7 microns resolution





Specular Surface Normals and Displacement Maps



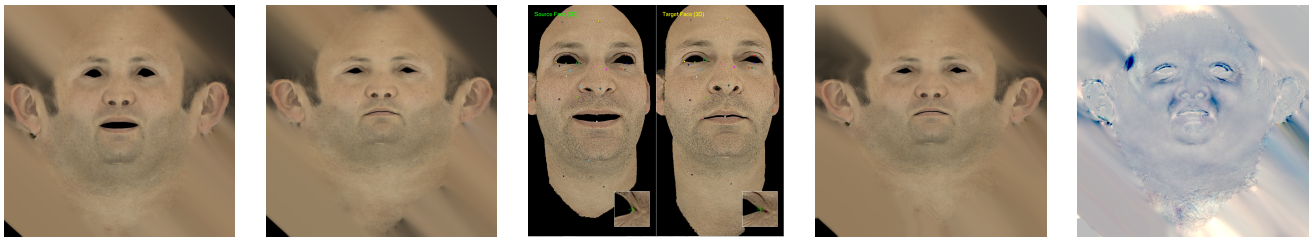


Vuvuzela Demo

Vuvuzela: A Facial Scan Correspondence Tool

Ryosuke Ichikari Oleg Alexander Paul Debevec

USC Institute for Creative Technologies oalexander@ict.usc.edu



(a) (b) (c) (d) (e)

Figure 1: *(a) Source. (b) Target. (c) Vuvuzela workflow. (d) Warped source. (e) Difference between warped source and target.*

1 Introduction

When scanning an actor's face in multiple static facial expressions, it is often desirable for the resulting scans to all have the same topology and for the textures to all be in the same UV space. Such "corresponded" scans would enable the straightforward creation of blendshape-based facial rigs. We present Vuvuzela, a semi-automated facial scan correspondence tool. Vuvuzela is currently being used in our facial rigging pipeline, and was one of the key tools in the Digital Ira project.

2 Our Approach

Our rig building process begins by scanning an actor's face in our Light Stage X device [Ghosh et al. 2011]. We capture a set of about 30 static facial expressions, roughly corresponding to Action Units from the Facial Action Coding System [Ekman and Friesen 1978]. We also capture a master "neutral" expression, which becomes the target scan in our correspondence pipeline.

Rather than storing our scans as geometry and textures, we choose instead to store our scans as images. Each one of our scans is stored as a set of 4K, 32 bit float EXR images, including diffuse, specular, specular normals, and a high resolution point cloud. The maps are in a cylindrically unwrapped UV space, representing our ear to ear data. However, the UV space differs slightly for each expression scan.

Vuvuzela exploits this image-based scan represantation by doing the scan correspondence in 2D rather than 3D. Vuvuzela takes as input two scans: one of the expressions as the source and the neutral expression as the target. Vuvuzela provides an OpenGL UI, allowing the user to interact with the scans in 3D. The scans are rendered with the diffuse textures only, and all of the correspondence processing uses only the diffuse textures.

The user clicks corresponding points in the source and target scans, such as corners of the eyes and lips, and other facial landmarks. We found that we don't need to put dots or markers on the face during scanning, because there is plenty of naturally occuring texture in the face, especially when over-sharpened. The placement of the correspondence points doesn't have to be exact—the points are used only as an initialization by our algorithm.

Once enough points have been placed, the user presses the Update button, which triggers our correspondence algorithm. The result is displayed to the user and the UI offers several modes to preview the quality of the correspondence, including a "blendshape" slider blending both geometry and/or texture. The user can then add, delete, or edit points, and repeat the process until a high quality correspondence is achieved.

Our algorithm has three steps and runs in 2D. First, we construct a Delaunay triangulation between the user supplied points and apply affine triangles to roughly pre-warp the source diffuse texture to the target. Second, we use GPU-accelerated optical flow to compute a dense warp field from the pre-warped source diffuse texture to the target. Finally, we apply the dense warp to each one of our source texture maps, including diffuse, specular, specular normals, and point cloud. The result is the source scan warped to the target UV space. The submillimeter correspondence is able to align individual pores across the majority of the face.

Some expressions are more challenging to correspond than others. Especially expressions with lots of occlusions, like mouth open to mouth closed. In such cases, optical flow will fail to get a good result. We assist optical flow in two ways. First, we paint black masks around occlusion regions in both source and target diffuse textures. Second, we mark some points as "pinned" and those points are rasterized into small black dots at runtime. Using both of these techniques in combination usually produces good results even in the toughest cases.

A useful byproduct of Vuvuzela is the ability to generate blendshapes directly from the corresponded scans. First, we remesh the neutral scan, creating an artist mesh with artist UVs. Then we load the artist mesh into Vuvuzela and export the blendshapes for all the scans by looking up vertex positions in the warped point clouds. All the texture maps are also warped into the artist UV space, which is simply an additional affine triangles 2D warp. The result is a set of blendshapes and texture maps ready to hand off to the facial rigger.

References

EKMAN, P., AND FRIESEN, W. 1978. *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Consulting Psychologists Press, Palo Alto.

GHOSH, A., FYFFE, G., TUNWATTANAPONG, B., BUSCH, J., YU, X., AND DEBEVEC, P. 2011. Multiview face capture using polarized spherical gradient illumination. *ACM Trans. Graph. 30*, 6 (Dec.), 129:1–129:10.

SIGGRAPH 2013

USC Institute for Creative Technologies

Vuvuzela: A Facial Scan Correspondence Tool

Ryosuke Ichikari Oleg Alexander Paul Debevec
USC Institute for Creative Technologies

USC Institute for Creative Technologies

Introduction

Digital Ira

When scanning an actor's face in multiple static facial expressions, it is often desirable for the resulting scans to all have the same topology and for the textures to all be in the same UV space. Such "corresponded" scans would enable the straightforward creation of blendshape-based facial rigs. We present Vuvuzela, a semiautomated facial scan correspondence tool. Vuvuzela is currently being used in our facial rigging pipeline, and was one of the key tools in the Digital Ira project.

(a) source map (b) target map (c) resulting warped source (d) difference between warped source and target

***Figure 1**: the original neutral expression (a) and the second expression needing to be placed into the same texture space (b) are imported into Vuvuzela. After processing, the original second expression is warped into the same texture space (c) as the neutral expression (a). The difference from the computed warp can be seen to the right (d).*

Our Approach

Data acquisition and preparation Our rig building process begins by scanning an actor's face in our Light Stage X device [Ghosh et al. 2011]. We capture a set of about 30 static facial expressions, roughly corresponding to Action Units from the Facial Action Coding System [Ekman and Friesen 1978]. We also capture a master "neutral" expression, which becomes the target scan in our correspondence pipeline.

Rather than storing our scans as geometry and textures, we choose instead to store our scans as images. Each one of our scans is stored as a set of 4K, 32 bit float EXR images, including diffuse, specular, specular normals, and a high resolution point cloud. The maps are in a cylindrically unwrapped UV space, representing our ear to ear data. However, the UV space differs slightly for each expression scan.

User interface Vuvuzela exploits this image-based scan represantation by doing the scan correspondence in 2D rather than 3D. Vuvuzela takes as input two scans: one of the expressions as the source and the neutral expression as the target. Vuvuzela provides an OpenGL UI, allowing the user to interact with the scans in 3D. The scans are rendered with the diffuse textures only, and all of the correspondence processing uses only the diffuse textures.

The user clicks corresponding points in the source and target scans, such as corners of the eyes and lips, and other facial landmarks. We found that we don't need to put dots or markers on the face during scanning, because there is plenty of naturally occuring texture in the face, especially when over-sharpened. The placement of the correspondence points doesn't have to be exact—the points are used only as an initialization by our algorithm.

References:

EKMAN, P., AND FRIESEN, W. 1978. Facial Action Coding System: A Technique for the Measurement of Facial Movement. Consulting Psychologists Press, Palo Alto.

GHOSH, A., FYFFE, G., TUNWATTANAPONG, B., BUSCH, J., YU, X., AND DEBEVEC, P. 2011. Multiview face capture using polarized spherical gradient illumination. ACM Trans. Graph. 30, 6 (Dec.), 129:1–129:10.

Once enough points have been placed, the user presses the Update button, which triggers our correspondence algorithm. The result is displayed to the user and the UI offers several modes to preview the quality of the correspondence, including a "blendshape" slider blending both geometry and/or texture. The user can then add, delete, or edit points, and repeat the process until a high quality correspondence is achieved.

posed mesh with corresponded maps

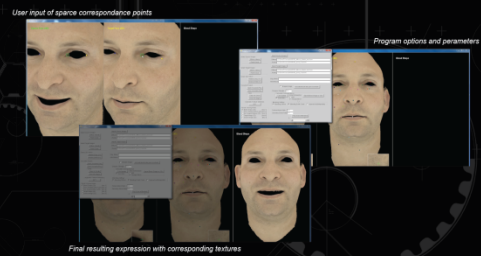
posed mesh with neutral expression maps

Correspondence algorithm Our algorithm has three steps and runs in 2D. First, we construct a Delaunay triangulation between the user supplied points and apply affine triangles to roughly pre-warp the source diffuse texture to the target. Second, we use GPU-accelerated optical flow to compute a dense warp field from the pre-warped source diffuse texture to the target. Finally, we apply the dense warp to each one of our source texture maps, including diffuse, specular, specular normals, and point cloud. The result is the source scan warped to the target UV space. The submillimeter correspondence is able to align individual pores across the majority of the face.

User intervention Some expressions are more challenging to correspond than others. Especially expressions with lots of occlusions, like mouth open to mouth closed. In such cases, optical flow will fail to get a good result. We assist optical flow in two ways. First, we paint black masks around occlusion regions in both source and target diffuse textures. Second, we mark some points as "pinned" and those points are rasterized into small black dots at runtime. Using both of these techniques in combination usually produces good results even in the toughest cases.

Blend shapes A useful byproduct of Vuvuzela is the ability to generate blendshapes directly from the corresponded scans. First, we remesh the neutral scan, creating an artist mesh with artist UVs. Then we load the artist mesh into Vuvuzela and export the blendshapes for all the scans by looking up vertex positions in the warped point clouds. All the texture maps are also warped into the artist UV space, which is simply an additional affine triangles 2D warp. The result is a set of blendshapes and texture maps ready to hand off to the facial rigger.

Software Interface and Workflow



To appear in ACM TOG ?(?).

Driving High-Resolution Facial Scans with Video Performance Capture

Graham Fyffe Andrew Jones Oleg Alexander Ryosuke Ichikari Paul Debevec *
USC Institute for Creative Technologies

(a) (b) (c) (d)

Figure 1: *(a) High resolution geometric and reflectance information from multiple static expression scans is automatically combined with (d) dynamic video frames to recover (b) matching animated high resolution performance geometry that can be (c) relit under novel illumination from a novel viewpoint. In this example, the performance is recovered using only the single camera viewpoint in (d).*

Abstract

We present a process for rendering a realistic facial performance with control of viewpoint and illumination. The performance is based on one or more high-quality geometry and reflectance scans of an actor in static poses, driven by one or more video streams of a performance. We compute optical flow correspondences between neighboring video frames, and a sparse set of correspondences between static scans and video frames. The latter are made possible by leveraging the relightability of the static 3D scans to match the viewpoint(s) and appearance of the actor in videos taken in arbitrary environments. As optical flow tends to compute proper correspondence for some areas but not others, we also compute a smoothed, per-pixel confidence map for every computed flow, based on normalized cross-correlation. These flows and their confidences yield a set of weighted triangulation constraints among the static poses and the frames of a performance. Given a single artist-prepared face mesh for one static pose, we optimally combine the weighted triangulation constraints, along with a shape regularization term, into a consistent 3D geometry solution over the entire performance that is drift-free by construction. In contrast to previous work, even partial correspondences contribute to drift minimization, for example where a successful match is found in the eye region but not the mouth. Our shape regularization employs a differential shape term based on a spatially varying blend of the differential shapes of the static poses and neighboring dynamic poses, weighted by the associated flow confidences. These weights also permit dynamic reflectance maps to be produced for the performance by blending the static scan maps. Finally, as the geometry and maps are represented on a consistent artist-friendly mesh, we render the resulting high-quality animated face geometry and animated reflectance maps using standard rendering tools.

*e-mail:{fyffe,jones,oalexander,debevec}@ict.usc.edu

1 Introduction

Recent facial geometry scanning techniques can capture very high resolution geometry, including high-frequency details such as skin pores and wrinkles. When animating these highly detailed faces, highly accurate temporal correspondence is required. At present, the highest quality facial geometry is produced by static scanning techniques, where the subject holds a facial pose for several seconds. This permits the use of high-resolution cameras for accurate stereo reconstruction and active illumination to recover pore-level resolution surface details. Such techniques also capture high-quality surface reflectance maps, enabling realistic rendering of the captured faces. Alternatively, static facial poses may be captured using facial casts combined with detail acquired from surface imprints. Unfortunately, *dynamic* scanning techniques are unable to provide the same level of detail as static techniques, even when high-speed cameras and active illumination are employed.

The classic approach to capturing facial motion is to use markers or face paint to track points on the face. However, such techniques struggle to capture the motion of the eyes and mouth, and rely on a high-quality facial rig to provide high-frequency skin motion and wrinkling. The best results are achieved when the rig is based on high-resolution static scans of the same subject. A second approach is to capture a performance with one or more passive video cameras. Such setups are lightweight as they use environmental illumination and off-the-shelf video cameras. As the camera records the entire face, it should be possible to recover eye and mouth motion missed by sparse markers. Still, by itself, passive video cannot match the resolution of static scans. While it is possible to emboss some video texture on the face [Bradley et al. 2010][Beeler et al. 2011][Valgaerts et al. 2012], many facial details appear only in specular reflections and are not visible under arbitrary illumination.

We present a technique for creating realistic facial animation from a set of high-resolution scans of an actor's face, driven by passive video of the actor from one or more viewpoints. The videos can be shot under existing environmental illumination using off-the-shelf

HD video cameras. The static scans can come from a variety of sources including facial casts, passive stereo, or active illumination techniques. High-resolution detail and relightable reflectance properties in the static scans can be transferred to the performance using generated per-pixel weight maps. We operate our algorithm on a *performance flow graph* that represents dense correspondences between dynamic frames and multiple static scans, leveraging GPU-based optical flow to efficiently construct the graph. Besides a single artist remesh of a scan in neutral pose, our method requires no rigging, no training of appearance models, no facial feature detection, and no manual annotation of any kind. As a byproduct of our method we also obtain a non-rigid registration between the artist mesh and each static scan. Our principal contributions are:

- An efficient scheme for selecting a sparse subset of image pairs for optical flow computation for drift-free tracking.
- A fully coupled 3D tracking method with differential shape regularization using multiple locally weighted target shapes.
- A message-passing-based optimization scheme leveraging lazy evaluation of energy terms enabling fully-coupled optimization over an entire performance.

2 Related Work

As many systems have been built for capturing facial geometry and reflectance, we will restrict our discussion to those that establish some form of dense temporal correspondence over a performance.

Many existing algorithms compute temporal correspondence for a sequence of temporally inconsistent geometries generated by e.g. structured light scanners or stereo algorithms. These algorithms operate using only geometric constraints [Popa et al. 2010] or by deforming template geometry to match each geometric frame [Zhang et al. 2004]. The disadvantage of this approach is that the per-frame geometry often contains missing regions or erroneous geometry which must be filled or filtered out, and any details that are missed in the initial geometry solution are non-recoverable.

Other methods operate on video footage of facial performances. Methods employing frame-to-frame motion analysis are subject to the accumulation of error or "drift" in the tracked geometry, prompting many authors to seek remedies for this issue. We therefore limit our discussion to methods that make some effort to address drift. Li et al. [1993] compute animated facial blendshape weights and rigid motion parameters to match the texture of each video frame to a reference frame, within a local minimum determined by a motion prediction step. Drift is avoided whenever a solid match can be made back to the reference frame. [DeCarlo and Metaxas 1996] solves for facial rig control parameters to agree with sparse monocular optical flow constraints, applying forces to pull model edges towards image edges in order to combat drift. [Guenter et al. 1998] tracks motion capture dots in multiple views to deform a neutral facial scan, increasing the realism of the rendered performance by projecting video of the face (with the dots digitally removed) onto the deforming geometry. The "Universal Capture" system described in [Borshukov et al. 2003] dispenses with the dots and uses dense multi-view optical flow to propagate vertices from an initial neutral expression. User intervention is required to correct drift when it occurs. [Hawkins et al. 2004] uses performance tracking to automatically blend between multiple high-resolution facial scans per facial region, achieving realistic multi-scale facial deformation without the need for reprojecting per-frame video, but uses dots to avoid drift. Bradley et al. [2010] track motion using dense multi-view optical flow, with a final registration step between the neutral mesh and every subsequent frame to reduce drift. Beeler et al. [2011] explicitly identify anchor frames that are similar to a manually chosen reference pose using a simple image difference metric, and track the performance bidirectionally between anchor frames. Non-sequential surface tracking [Klaudiny and Hilton 2012] finds a minimum-cost spanning tree over the frames in a performance based on sparse feature positions, tracking facial geometry across edges in the tree with an additional temporal fusion step. Valgaerts et al. [2012] apply scene flow to track binocular passive video with a regularization term to reduce drift.

One drawback to all such optical flow tracking algorithms is that the face is tracked from one pose to another *as a whole*, and success of the tracking depends on accurate optical flow between images of the entire face. Clearly, the human face is capable of repeating different poses over different parts of the face asynchronously, which the holistic approaches fail to model. For example, if the subject is talking with eyebrows raised and later with eyebrows lowered, a holistic approach will fail to exploit similarities in mouth poses when eyebrow poses differ. In contrast, our approach constructs a graph considering similarities over multiple regions of the face across the performance frames and a set of static facial scans, removing the need for sparse feature tracking or anchor frame selection.

Blend-shape based animation rigs are also used to reconstruct dynamic poses based on multiple face scans. The company Image Metrics (now Faceware) has developed commercial software for driving a blend-shape rig with passive video based on active appearance models [Cootes et al. 1998]. Weise et al. [2011] automatically construct a personalized blend shape rig and drive it with Kinect depth data using a combination of as-rigid-as-possible constraints and optical flow. In both cases, the quality of the resulting tracked performance is directly related to the quality of the rig. Each tracked frame is a linear combination of the input blend-shapes, so any performance details that lie outside the domain spanned by the rig will not be reconstructed. Huang et al. [2011] automatically choose a minimal set of blend shapes to scan based on previously captured performance with motion capture markers. Recreating missing detail requires artistic effort to add corrective shapes and cleanup animation curves [Alexander et al. 2009]. There has been some research into other non-traditional rigs incorporating scan data. Ma et al. [2008] fit a polynomial displacement map to dynamic scan training data and generate detailed geometry from sparse motion capture markers. Bickel et al. [2008] locally interpolate a set of static poses using radial basis functions driven by motion capture markers. Our method combines the shape regularization advantages of blendshapes with the flexibility of optical flow based tracking. Our optimization algorithm leverages 3D information from static scans without constraining the result to lie only within the linear combinations of the scans. At the same time, we obtain per-pixel blend weights that can be used to produce per-frame reflectance maps.

3 Data Capture and Preparation

We capture high-resolution static geometry using multi-view stereo and gradient-based photometric stereo [Ghosh et al. 2011]. The scan set includes around 30 poses largely inspired by the Facial Action Coding System (FACS) [Ekman and Friesen 1978], selected to span nearly the entire range of possible shapes for each part of the face. For efficiency, we capture some poses with the subject combining FACS action units from the upper and lower half of the face. For example, combining eyebrows raise and cheeks puff into a single scan. Examples of the input scan geometry can be seen in Fig. 2. A base mesh is defined by an artist for the neutral pose scan. The artist mesh has an efficient layout with edge loops following the wrinkles of the face. The non-neutral poses are represented as raw scan geometry, requiring no artistic topology or remeshing.

We capture dynamic performances using up to six Canon 1DX DSLR cameras under constant illumination. In the simplest case, we use the same cameras that were used for the static scans and switch to 1920×1080 30p movie mode. We compute a sub-frame-accurate synchronization offset between cameras using a correlation analysis of the audio tracks. This could be omitted if cameras with hardware synchronization are employed. Following each performance, we capture a video frame of a calibration target to calibrate camera intrinsics and extrinsics. We relight (and when necessary, repose) the static scan data to resemble the illumination conditions observed in the performance video. In the simplest case, the illumination field resembles one of the photographs taken during the static scan process and no relighting is required.

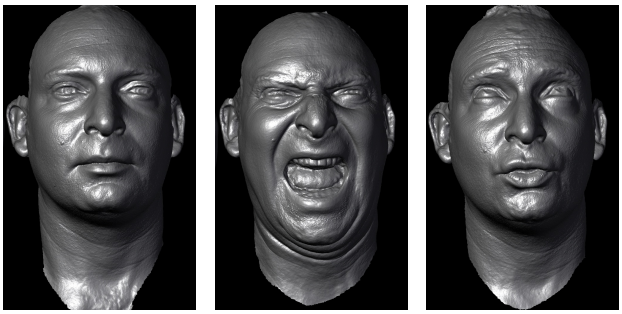


Figure 2: *Sample static scans (showing geometry only).*

4 The Performance Flow Graph

Optical-flow-based tracking algorithms such as [Bradley et al. 2010][Beeler et al. 2011][Klaudiny and Hilton 2012] relate frames of a performance to each other based on optical flow correspondences over a set of image pairs selected from the performance. These methods differ in part by the choice of the image pairs to be employed. We generalize this class of algorithms using a structure we call the *performance flow graph*, which is a complete graph with edges representing dense 2D correspondences between *all pairs* of images, with each edge having a weight, or confidence, of the associated estimated correspondence field. The graphs used in previous works, including anchor frames [Beeler et al. 2011] and non-sequential alignment with temporal fusion [Klaudiny and Hilton 2012], can be represented as a performance flow graph having unit weight for the edges employed by the respective methods, and zero weight for the unused edges. We further generalize the performance flow graph to include a dense *confidence field* associated with each correspondence field, allowing the confidence to vary spatially over the image. This enables our technique to exploit relationships between images where only a partial correspondence was able to be computed (for example, a pair of images where the mouth is similar but the eyes are very different). Thus our technique can be viewed as an extension of anchor frames or minimum spanning trees to minimize drift independently over different regions of the face.

A performance capture system that considers correspondences between all possible image pairs naturally minimizes drift. However, this would require an exorbitant number of graph edges, so we instead construct a graph with a reduced set of edges that approximates the complete graph, in the sense that the correspondences are *representative* of the full set with respect to confidence across the regions of the face. Our criterion for selecting the edges to include in the performance flow graph is that any two images having a high confidence correspondence between them in the complete graph of possible correspondences ought to have a *path* between them (a concatenation of one or more correspondences) in the constructed graph with nearly as high confidence (including the reduction in confidence from concatenation). We claim that correspondences between temporally neighboring dynamic frames are typically of high quality, and no concatenation of alternative correspondences can be as confident, therefore we always include a graph edge between each temporally neighboring pair of dynamic frames. Correspondences between frames with larger temporal gaps are well-approximated by concatenating neighbors, but decreasingly so over larger temporal gaps (due to drift). We further claim that whenever enough drift accumulates to warrant including a graph edge over the larger temporal gap, there exists a path with nearly as good confidence that passes through one of the predetermined *static scans* (possibly a different static scan for each region of the face). We justify this claim by noting the 30 static poses based on FACS ought to span the space of performances well enough that any region of any dynamic frame can be corresponded to some region in some static scan with good confidence. Therefore we do not include any edges between non-neighboring dynamic frames, and instead consider only edges between a static scan and a dynamic frame as candidates for inclusion (visualized in Fig. 3). Finally, as the drift accumulated from the concatenation described above warrants additional edges only sparsely over time, we devise a coarse-to-fine graph construction strategy using only a sparse subset of static-to-dynamic graph edges. We detail this strategy in Section 4.1.

Figure 3: *performance flow graph showing optical flow correspondences between static and dynamic images. Red lines represent optical flow between neighboring frames within a performance. Blue, green, and orange lines represent optical flow between dynamic and static images. Based on initial low-resolution optical flow, we construct a sparse graph requiring only a small subset of high resolution flows to be computed between static scans and dynamic frames.*

4.1 Constructing the Performance Flow Graph

The images used in our system consist of one or more dynamic sequences of frames captured from one or more viewpoints, and roughly similar views of a set of high-resolution static scans. The nodes in our graph represent static poses (associated with static scans) and dynamic poses (associated with dynamic frames from one or more sequences). We construct the performance flow graph by computing a large set of static-to-dynamic optical flow correspondences at a reduced resolution for only a single viewpoint, and then omit redundant correspondences using a novel voting algorithm to select a sparse set of correspondences that is representative of the original set. We then compute high-quality optical flow correspondences at full resolution for the sparse set, and include all

viewpoints. The initial set of correspondences consists of quarter-resolution optical flows from each static scan to every n^{th} dynamic frame. For most static scans we use every 5^{th} dynamic frame, while for the eyes-closed scan we use every dynamic frame in order to catch rapid eye blinks. We then compute normalized cross correlation fields between the warped dynamic frames and each original static scan to evaluate the confidence of the correspondences. These correspondences may be computed in parallel over multiple computers, as there is no sequential dependency between them. We find that at quarter resolution, flow-based cross correlation correctly assigns low confidence to incorrectly matched facial features, for example when flowing disparate open and closed mouth shapes. To reduce noise and create a semantically meaningful metric, we average the resulting confidence over twelve facial regions (see Fig. 4). These facial regions are defined once on the neutral pose, and are warped to all other static poses using rough static-to-static optical flow. Precise registration of regions is not required, as they are only used in selecting the structure of the performance graph. In the subsequent tracking phase, per-pixel confidence is used.

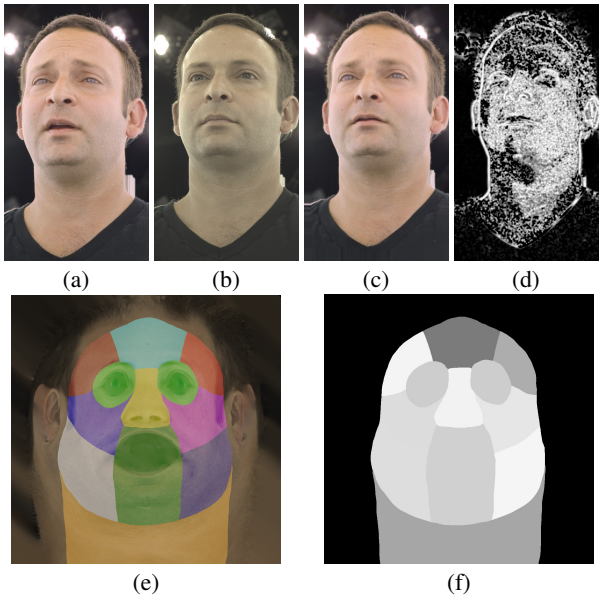


Figure 4: *We compute an initial low-resolution optical flow between a dynamic image (a) and static image (b). We then compute normalized cross correlation between the static image (b) and the warped dynamic image (c) to produce the per-pixel confidence shown in (d). We average these values for 12 regions (e) to obtain a per-region confidence value (f). This example shows correlation between the neutral scan and a dynamic frame with the eyebrows raised and the mouth slightly open. The forehead and mouth regions are assigned appropriately lower confidences.*

Ideally we want the performance flow graph to be sparse. Besides temporally adjacent poses, dynamic poses should only connect to similar static poses and edges should be evenly distributed over time to avoid accumulation of drift. We propose an iterative greedy voting algorithm based on the per-region confidence measure to identify good edges. The confidence of correspondence between the dynamic frames and any region of any static facial scan can be viewed as a curve over time (depicted in Fig. 5). In each iteration we identify the maximum confidence value over all regions, all scans, and all frames. We add an edge between the identified dynamic pose and static pose to the graph. We then adjust the recorded confidence of the identified region by subtracting a hat function scaled by the maximum confidence and centered around the maximum frame, indicating that the selected edge has been accounted for, and temporal

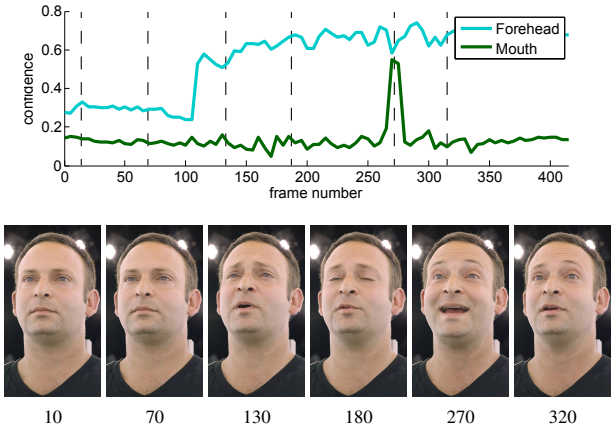


Figure 5: *A plot of the per-region confidence metric over time. Higher numbers indicate greater correlation between the dynamic frames and a particular static scan. The cyan curve represents the center forehead region of a brows-raised static scan which is active throughout the later sequence. The green curve represents the mouth region for an extreme mouth-open scan which is active only when the mouth opens to its fullest extent. The dashed lines indicate the timing of the sampled frames shown on the bottom row.*

neighbors partly so. All other regions are adjusted by subtracting similar hat functions, scaled by the (non-maximal) per-region confidence of the identified flow. This suppresses any other regions that are satisfied by the flow. The slope of the hat function represents a loss of confidence as this flow is combined with adjacent dynamic-to-dynamic flows. We then iterate and choose the new highest confidence value, until all confidence values fall below a threshold. The two parameters (the slope of the hat function and the final threshold value) provide intuitive control over the total number of graph edges. We found a reasonable hat function falloff to be a 4% reduction for every temporal flow and a threshold value that is 20% of the initial maximum confidence. After constructing the graph, a typical 10-20 second performance flow graph will contain 100-200 edges between dynamic and static poses. Again, as the change between sequential frames is small, we preserve all edges between neighboring dynamic poses.

After selecting the graph edges, final HD resolution optical flows are computed for all active cameras and for all retained graph edges. We directly load video frames using nVidia's h264 GPU decoder and feed them to the FlowLib implementation of GPU-optical flow [Werlberger 2012]. Running on a Nvidia GTX 680, computation of quarter resolution flows for graph construction take less than one second per flow. Full-resolution HD flows for dynamic-to-dynamic images take 8 seconds per flow, and full-resolution flows between static and dynamic images take around 23 seconds per flow due to a larger search window. More sophisticated correspondence estimation schemes could be employed within our framework, but our intention is that the framework be *agnostic* to this choice and *robust* to imperfections in the pairwise correspondences. After computing optical flows and confidences, we synchronize all the flow sequences to a primary camera by warping each flow frame forward or backward in time based on the sub-frame synchronization offsets between cameras.

We claim that an approximate performance flow graph constructed in this manner is more representative of the complete set of possible correspondences than previous methods that take an all-or-nothing approach to pair selection, while still employing a number of optical flow computations on the same order as previous methods (i.e. temporal neighbors plus additional sparse image pairs).

5 Fully Coupled Performance Tracking

The performance flow graph is representative of all the constraints we could glean from 2D correspondence analysis of the input images, and now we aim to put those constraints to work. We formulate an energy function in terms of the 3D vertex positions of the artist mesh as it deforms to fit all of the dynamic and static poses in the performance flow graph in a *common head coordinate system*, as well as the associated head-to-world rigid transforms. We collect the free variables into a vector $\theta = (\mathbf{x}_i^p, \mathbf{R}_p, \mathbf{t}_p | p \in \mathcal{D} \cup \mathcal{S}, i \in \mathcal{V})$, where $\mathbf{x}_i^p$ represents the 3D vertex position of vertex i at pose p in the common head coordinate system, $\mathbf{R}_p$ and $\mathbf{t}_p$ represent the rotation matrix and translation vector that rigidly transform pose p from the common head coordinate system to world coordinates, $\mathcal{D}$ is the set of dynamic poses, $\mathcal{S}$ is the set of static poses, and $\mathcal{V}$ is the set of mesh vertices. The energy function is then:

$$\mathbf{E}(\theta) = \sum_{(p,q)\in\mathcal{F}} (\mathbf{E}_{\text{corr}}^{pq} + \mathbf{E}_{\text{corr}}^{qp}) + \lambda \sum_{p\in\mathcal{D}\cup\mathcal{S}} |\mathcal{F}_p| \mathbf{E}_{\text{shape}}^{p} + \zeta \sum_{p\in\mathcal{S}} |\mathcal{F}_p| \mathbf{E}_{\text{wrap}}^{p} + \gamma |\mathcal{F}_g| \mathbf{E}_{\text{ground}}, \quad (1)$$

where $\mathcal{F}$ is the set of performance flow graph edges, $\mathcal{F}_p$ is the subset of edges connecting to pose p, and g is the ground (neutral) static pose. This function includes:

- dense correspondence constraints $\mathbf{E}_{\text{corr}}^{pq}$ associated with the edges of the performance flow graph,
- shape regularization terms $\mathbf{E}_{\text{shape}}^{p}$ relating the differential shape of dynamic and static poses to their graph neighbors,
- "shrink wrap" terms $\mathbf{E}_{\text{wrap}}^{p}$ to conform the static poses to the surface of the static scan geometries,
- a final grounding term $\mathbf{E}_{\text{ground}}$ to prefer the vertex positions in a neutral pose to be close to the artist mesh vertex positions.

We detail these terms in sections 5.2 - 5.5. Note we do not employ a stereo matching term, allowing our technique to be robust to small synchronization errors between cameras. As the number of poses and correspondences may vary from one dataset to another, the summations in (1) contain balancing factors (to the immediate right of each Σ) in order to have comparable total magnitude (proportional to $|\mathcal{F}|$). The terms are weighted by tunable term weights λ, ζ and γ, which in all examples we set equal to 1.

5.1 Minimization by Lazy DDMS-TRWS

In contrast to previous work, we consider the *three-dimensional coupling* between all terms in our formulation, over all dynamic and static poses simultaneously, thereby obtaining a robust estimate that gracefully fills in missing or unreliable information. This presents two major challenges. First, the partial matches and loops in the performance flow graph preclude the use of straightforward mesh propagation schemes used in previous works. Such propagation would produce only partial solutions for many poses. Second (as a result of the first) we lack a complete initial estimate for traditional optimization schemes such as Levenberg-Marquadt.

To address these challenges, we employ an iterative scheme that admits partial intermediate solutions, with pseudocode in Algorithm 1. As some of the terms in (1) are data-dependent, we adapt the outer loop of Data Driven Mean-Shift Belief Propagation (DDMSBP) [Park et al. 2010], which models the objective function in each iteration as an increasingly-tight Gaussian (or quadratic) approximation of the true function. Within each DDMS loop, we use Gaussian Tree-Reweighted Sequential message passing (TRW-S) [Kolmogorov 2006], adapted to allow the terms in the model to be constructed lazily as the solution progresses over the variables. Hence we call our scheme *Lazy DDMS-TRWS*. We define the ordering of the variables to be pose-major (i.e. visiting all the vertices of one pose, then all the vertices of the next pose, etc.), with static poses followed by dynamic poses in temporal order. We decompose the Gaussian belief as a product of 3D Gaussians over vertices and poses, which admits a pairwise decomposition of (1) as a sum of quadratics. We denote the current belief of a vertex i for pose p as $\bar{\mathbf{x}}_i^p$ with covariance $\mathbf{\Sigma}_i^p$ (stored as inverse covariance for convenience), omitting the i subscript to refer to all vertices collectively. We detail the modeling of the energy terms in sections 5.2 - 5.5, defining $\bar{\mathbf{y}}_i^p = \mathbf{R}_p \bar{\mathbf{x}}_i^p + \mathbf{t}_p$ as shorthand for world space vertex position estimates. We iterate the DDMS loop 6 times, and iterate TRW-S until 95% of the vertices converge to within 0.01mm.

Algorithm 1 Lazy DDMS-TRWS for (1)

$\forall_{p,i} : (\mathbf{\Sigma}_i^p)^{-1} \leftarrow \mathbf{0}$.
for DDMS outer iterations **do**
  // Reset the model:
  $\forall_{p,q} : \mathbf{E}_{\text{corr}}^{pq}, \mathbf{E}_{\text{shape}}^{p}, \mathbf{E}_{\text{wrap}}^{p} \leftarrow$ undefined (effectively 0).
  for TRW-S inner iterations **do**
    // Major TRW-S loop over poses:
    for each $p \in \mathcal{D} \cup \mathcal{S}$ in order of increasing $o(p)$ **do**
      // Update model where possible:
      for each $q | (p,q) \in \mathcal{F}$ **do**
        if $(\mathbf{\Sigma}^p)^{-1} \neq \mathbf{0}$ and $\mathbf{E}_{\text{corr}}^{pq}$ undefined **then**
          $\mathbf{E}_{\text{corr}}^{pq} \leftarrow$ model fit using (2) in section 5.2.
        if $(\mathbf{\Sigma}^q)^{-1} \neq \mathbf{0}$ and $\mathbf{E}_{\text{corr}}^{qp}$ undefined **then**
          $\mathbf{E}_{\text{corr}}^{qp} \leftarrow$ model fit using (2) in section 5.2.
      if $(\mathbf{\Sigma}^p)^{-1} \neq \mathbf{0}$ and $\mathbf{E}_{\text{wrap}}^{p}$ undefined **then**
        $\mathbf{E}_{\text{wrap}}^{p} \leftarrow$ model fit using (8) in section 5.4.
      if $\exists_{(p,q)\in\mathcal{F}} | (\mathbf{\Sigma}^q)^{-1} \neq \mathbf{0}$ and $\mathbf{E}_{\text{shape}}^{p}$ undefined **then**
        $\mathbf{E}_{\text{shape}}^{p} \leftarrow$ model fit using (5) in section 5.3.
      // Minor TRW-S loop over vertices:
      Pass messages based on (1) to update $\bar{\mathbf{x}}^p, (\mathbf{\Sigma}^p)^{-1}$.
      Update $\mathbf{R}_p, \mathbf{t}_p$ as in section 5.6.
    // Reverse TRW-S ordering:
    $o(s) \leftarrow \|\mathcal{D} \cup \mathcal{S}\| + 1 - o(s)$.

5.2 Modeling the Correspondence Term

The correspondence term in (1) penalizes disagreement between optical flow vectors and projected vertex locations. Suppose we have a 2D optical flow correspondence field between poses p and q in (roughly) the same view c. We may establish a 3D relationship between $\mathbf{x}_i^p$ and $\mathbf{x}_i^q$ implied by the 2D correspondence field, which we model as a quadratic penalty function:

$$\mathbf{E}_{\text{corr}}^{pq} = \frac{1}{|\mathcal{C}|} \sum_{\substack{c\in\mathcal{C} \\ i\in\mathcal{V}}} \sum (\mathbf{x}_i^q - \mathbf{x}_i^p - \mathbf{f}_{\underset{i}{pq}}^{c})^{\mathsf{T}} \mathbf{F}_{\underset{i}{pq}}^{c} (\mathbf{x}_i^q - \mathbf{x}_i^p - \mathbf{f}_{\underset{i}{pq}}^{c}), \quad (2)$$

where $\mathcal{C}$ is the set of camera viewpoints, and $\mathbf{f}_{\underset{i}{pq}}^{c}, \mathbf{F}_{\underset{i}{pq}}^{c}$ are respectively the mean and precision matrix of the penalty, which we estimate from the current estimated positions as follows. We first project $\bar{\mathbf{y}}_i^p$ into the image plane of view c of pose p. We then warp the 2D image position from view c of pose p to view c of pose q using the correspondence field. The warped 2D position defines a world-space view ray that the same vertex i ought to lie on in pose q. We transform this ray back into common head coordinates (via

$-\mathbf{t}_q$, $\mathbf{R}_q^\mathsf{T}$) and penalize the squared distance from $\mathbf{x}_i^q$ to this ray. Letting $\mathbf{r}_{\substack{pq\\i}}^c$ represent the direction of this ray, this yields:

$$\mathbf{f}_{\substack{pq\\i}}^c = (\mathbf{I} - \mathbf{r}_{\substack{pq\\i}}^c \mathbf{r}_{\substack{pq\\i}}^{c\,\mathsf{T}})(\mathbf{R}_q^\mathsf{T}(\mathbf{c}_q^c - \mathbf{t}_q) - \bar{\mathbf{x}}_i^p), \tag{3}$$

where $\mathbf{c}_q^c$ is the nodal point of view c of pose q, and $\mathbf{r}_{\substack{pq\\i}}^c = \mathbf{R}_q^\mathsf{T}\mathbf{d}_{\substack{pq\\i}}^c$ with $\mathbf{d}_{\substack{pq\\i}}^c$ the world-space direction of the ray in view c of pose q through the 2D image plane point $f_{pq}^c[\mathrm{P}_p^c(\bar{\mathbf{y}}_i^p)]$ (where square brackets represent bilinearly interpolated sampling of a field or image), f_{pq}^c the optical flow field transforming an image-space point from view c of pose p to the corresponding point in view c of pose q, and $\mathrm{P}_p^c(\mathbf{x})$ the projection of a point $\mathbf{x}$ into the image plane of view c of pose p (which may differ somewhat from pose to pose). If we were to use the squared-distance-to-ray penalty directly, $\mathbf{F}_{\substack{pq\\i}}^c$ would be $\mathbf{I} - \mathbf{r}_{\substack{pq\\i}}^c \mathbf{r}_{\substack{pq\\i}}^{c\,\mathsf{T}}$, which is singular. To prevent the problem from being ill-conditioned and also to enable the use of *monocular* performance data, we add a small regularization term to produce a non-singular penalty, and weight the penalty by the confidence of the optical flow estimate. We also assume the optical flow field is locally smooth, so a *large* covariance $\mathbf{\Sigma}_i^p$ inversely influences the precision of the model, whereas a *small* covariance $\mathbf{\Sigma}_i^p$ does not, and weight the model accordingly. Intuitively, this weighting causes information to propagate from the ground term outward via the correspondences in early iterations, and blends correspondences from all sources in later iterations. All together, this yields:

$$\mathbf{F}_{\substack{pq\\i}}^c = \min(1, \det(\mathbf{\Sigma}_i^p)^{-\frac{1}{3}}) v_{\substack{p\\i}}^c \tau_{pq}^c[\mathrm{P}_p^c(\bar{\mathbf{y}}_i^p)](\mathbf{I} - \mathbf{r}_{\substack{pq\\i}}^c \mathbf{r}_{\substack{pq\\i}}^{c\,\mathsf{T}} + \epsilon\mathbf{I}), \tag{4}$$

where $v_{\substack{p\\i}}^c$ is a soft visibility factor (obtained by blurring a binary vertex visibility map and modulated by the cosine of the angle between surface normal and view direction), τ_{pq}^c is the confidence field associated with the correspondence field f_{pq}^c, and ϵ is a small regularization constant. We use $\det(\mathbf{\Sigma})^{-1/3}$ as a scalar form of precision for 3D Gaussians.

5.3 Modeling the Differential Shape Term

The shape term in (1) constrains the differential shape of each pose to a spatially varying convex combination of the differential shapes of the neighboring poses in the performance flow graph:

$$\mathbf{E}_{\mathrm{shape}}^p = \sum_{(i,j)\in\mathcal{E}} \left\| \mathbf{x}_j^p - \mathbf{x}_i^p - \mathbf{l}_{ij}^p \right\|^2, \tag{5}$$

$$\mathbf{l}_{ij}^p = \frac{\epsilon(\mathbf{g}_j - \mathbf{g}_i) + \sum_{q|(p,q)\in\mathcal{F}} \mathrm{w}_{ij}^{pq}(\bar{\mathbf{x}}_j^q - \bar{\mathbf{x}}_i^q)}{\epsilon + \sum_{q|(p,q)\in\mathcal{F}} \mathrm{w}_{ij}^{pq}}, \tag{6}$$

$$\mathrm{w}_{ij}^{pq} = \frac{w_i^{pq} w_j^{pq}}{w_i^{pq} + w_j^{pq}}, \tag{7}$$

where $\mathcal{E}$ is the set of edges in the geometry mesh, $w_i^{pq} = \det(\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \mathbf{F}_{\substack{pq\\i}}^c + \mathbf{F}_{\substack{qp\\i}}^c)^{1/3}$ (which is intuitively the strength of the relationship between poses p and q due to the correspondence term), $\mathbf{g}$ denotes the artist mesh vertex positions, and ϵ is a small regularization constant. The weights w_i^{pq} additionally enable trivial synthesis of high-resolution reflectance maps for each dynamic frame of the performance by blending the static pose data.

5.4 Modeling the Shrink Wrap Term

The shrink wrap term in (1) penalizes the distance between static pose vertices and the raw scan geometry of the same pose. We model this as a regularized distance-to-plane penalty:

$$\mathbf{E}_{\mathrm{wrap}}^p = \sum_{i\in\mathcal{V}} (\mathbf{x}_i^p - \mathbf{d}_i^p)^\mathsf{T} \mathrm{g}_i^p (\mathbf{n}_i^p \mathbf{n}_i^{p\mathsf{T}} + \epsilon\mathbf{I})(\mathbf{x}_i^p - \mathbf{d}_i^p), \tag{8}$$

where $(\mathbf{n}_i^p, \mathbf{d}_i^p)$ are the normal and centroid of a plane fitted to the surface of the static scan for pose p close to the current estimate $\bar{\mathbf{x}}_i^p$ in common head coordinates, and g_i^p is the confidence of the planar fit. We obtain the planar fit inexpensively by projecting $\bar{\mathbf{y}}_i^p$ into each camera view, and sampling the raw scan surface via a set of precomputed rasterized views of the scan. (Alternatively, a 3D search could be employed to obtain the samples.) Each surface sample (excluding samples that are occluded or outside the rasterized scan) provides a plane equation based on the scan geometry and surface normal. We let $\mathbf{n}_i^p$ and $\mathbf{d}_i^p$ be the weighted average values of the plane equations over all surface samples:

$$\mathbf{n}_i^p = \sum_{c\in\mathcal{C}} \omega_{\substack{p\\i}}^c \mathbf{R}_p^\mathsf{T} \hat{\mathbf{n}}_p^c[\mathrm{P}_p^c(\bar{\mathbf{y}}_i^p)] \text{ (normalized)}, \tag{9}$$

$$\mathbf{d}_i^p = \Big(\sum_{c\in\mathcal{C}} \omega_{\substack{p\\i}}^c\Big)^{-1} \sum_{c\in\mathcal{C}} \omega_{\substack{p\\i}}^c \mathbf{R}_p^\mathsf{T}(\hat{\mathbf{d}}_p^c[\mathrm{P}_p^c(\bar{\mathbf{y}}_i^p)] - \mathbf{t}_p), \tag{10}$$

$$\mathrm{g}_i^p = \min(1, \det(\mathbf{\Sigma}_i^p)^{-\frac{1}{3}}) \sum_{c\in\mathcal{C}} \omega_{\substack{p\\i}}^c, \tag{11}$$

where $(\hat{\mathbf{n}}_p^c, \hat{\mathbf{d}}_p^c)$ are the world-space surface normal and position images of the rasterized scans, and $\omega_{\substack{p\\i}}^c = 0$ if the vertex is occluded in view c or lands outside of the rasterized scan, otherwise $\omega_{\substack{p\\i}}^c = v_{\substack{p\\i}}^c \exp(-\|\hat{\mathbf{d}}_p^c[\mathrm{P}_p^c(\bar{\mathbf{y}}_i^p)] - \bar{\mathbf{y}}_i^p\|^2)$.

5.5 Modeling the Ground Term

The ground term in (1) penalizes the distance between vertex positions in the ground (neutral) pose and the artist mesh geometry:

$$\mathbf{E}_{\mathrm{ground}} = \sum_{i\in\mathcal{V}} \left\| \mathbf{x}_i^g - \mathbf{R}_g^\mathsf{T} \mathbf{g}_i \right\|^2, \tag{12}$$

where $\mathbf{g}_i$ is the position of the vertex in the artist mesh. This term is simpler than the shrink-wrap term since the pose vertices are in one-to-one correspondence with the artist mesh vertices.

5.6 Updating the Rigid Transforms

We initialize our optimization scheme with all $(\mathbf{\Sigma}_i^p)^{-1} = 0$ (and hence all $\bar{\mathbf{x}}_i^p$ *moot*), fully relying on the lazy DDMS-TRWS scheme to propagate progressively tighter estimates of the vertex positions $\mathbf{x}_i^p$ throughout the solution. Unfortunately, in our formulation the rigid transforms $(\mathbf{R}_p, \mathbf{t}_p)$ enjoy no such treatment as they always occur together with $\mathbf{x}_i^p$ and would produce non-quadratic terms if they were included in the message passing domain. Therefore we must initialize the rigid transforms to some rough initial guess, and update them after each iteration. The neutral pose is an exception, where the transform is *specified by the user* (by rigidly posing the artist mesh to their whim) and hence not updated. In all our examples, the initial guess for all poses is simply the same as the user-specified rigid transform of the neutral pose. We update $(\mathbf{R}_p, \mathbf{t}_p)$ using a simple scheme that aligns the neutral artist mesh to the current result. Using singular value decomposition, we compute the closest rigid transform minimizing $\sum_{i\in\mathcal{V}} r_i \left\| \mathbf{R}_p \mathbf{g}_i + \mathbf{t}_p - \bar{\mathbf{R}}_p \bar{\mathbf{x}}_i^p - \bar{\mathbf{t}}_p \right\|^2$, where r_i is a rigidity weight value (high weight around the eye sockets and temples, low weight elsewhere), $\mathbf{g}_i$ denotes the artist mesh vertex positions, and $(\bar{\mathbf{R}}_p, \bar{\mathbf{t}}_p)$ is the previous transform estimate.

5.7 Accelerating the Solution Using Keyframes

Minimizing the energy in (1) over the entire sequence requires multiple iterations of the TRW-S message passing algorithm, and multiple iterations of the DDMS outer loop. We note that the performance flow graph assigns static-to-dynamic flows to only a sparse subset of performance frames, which we call *keyframes*. Correspondences among the spans of frames in between keyframes are reliably represented using concatenation of temporal flows. Therefore to reduce computation time we first miminize the energy at *only the keyframes and static poses*, using concatenated temporal flows in between keyframes. Each iteration of this reduced problem is far cheaper than the full problem, so we may obtain a satisfactory solution of the performance keyframes and static poses more quickly. Next, we keep the static poses and keyframe poses *fixed*, and solve the spans of in-between frames, omitting the shrink-wrap and grounding terms as they affect only the static poses. This subsequent minimization requires only a few iterations to reach a satisfactory result, and each span of in-between frames may be solved *independently* (running on multiple computers, for example).

6 Handling Arbitrary Illumination and Motion

Up to now, we have assumed that lighting and overall head motion in the static scans closely matches that in the dynamic frames. For performances in uncontrolled environments, the subject may move or rotate their head to face different cameras, and lighting may be arbitrary. We handle such complex cases by taking advantage of the 3D geometry and relightable reflectance maps in the static scan data. For every 5$^{\text{th}}$ performance frame, we compute a relighted rendering of each static scan with roughly similar rigid head motion and lighting environment as the dynamic performance. These renderings are used as the static expression imagery in our pipeline. The rigid head motion estimate does not need to be exact as the optical flow computation is robust to a moderate degree of misalignment. In our results, we (roughly) rigidly posed the head by hand, though automated techniques could be employed [Zhu and Ramanan 2012]. We also assume that a HDR light probe measurement [Debevec 1998] exists for the new lighting environment, however, lighting could be estimated from the subject's face [Valgaerts et al. 2012] or eyes [Nishino and Nayar 2004].

The complex backgrounds in real-world uncontrolled environments pose a problem, as optical flow vectors computed on background pixels close to the silhouette of the face may confuse the correspondence term if the current estimate of the facial geometry slightly overlaps the background. This results in parts of the face "sticking" to the background as the subject's face turns from side to side (Fig. 6). To combat this, we weight the correspondence confidence field by a simple soft segmentation of head vs. background. Since head motion is largely rigid, we fit a 2D affine transform to the optical flow vectors in the region of the current head estimate. Then, we weight optical flow vectors by how well they agree with the fitted transform. We also assign high weight to the region deep inside the current head estimate using a simple image-space erosion algorithm, to prevent large jaw motions from being discarded. The resulting soft segmentation effectively cuts the head out of the background whenever the head is moving, thus preventing the optical flow vectors of the background from polluting the edges of the face. When the head is not moving against the background the segmentation is poor, but in this case the optical flow vectors of the face and background agree and pollution is not damaging.

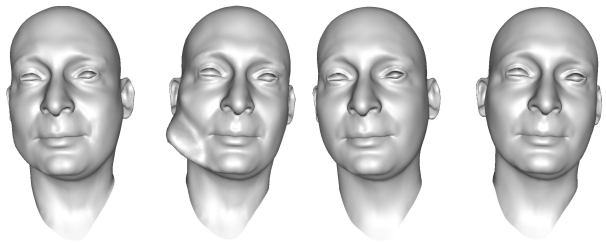


Figure 6: *(a, b) Two frames of a reconstructed performance in front of a cluttered background, where the subject turns his head over the course of ten frames. The silhouette of the jaw "sticks" to the background because the optical flow vectors close to the jaw are stationary. (c, d) A simple segmentation of the optical flow field to exclude the background resolves the issue.*

7 Results

We ran our technique on several performances from three different subjects. Each subject had 30 static facial geometry scans captured before the performance sessions, though the performance flow graph construction often employs only a fraction of the scans. An artist produced a single face mesh for each subject based on their neutral static facial scan.

7.1 Performances Following Static Scan Sessions

We captured performances of three subjects directly following their static scan sessions. The performances were recorded from six camera views in front of the subject with a baseline of approximately 15 degrees. Our method produced the performance animation results shown in Fig. 19 without any further user input.

7.2 Performances in Other Locations

We captured a performance of a subject using four consumer HD video cameras in an office environment. An animator rigidly posed a head model roughly aligned to every 5$^{\text{th}}$ frame of the performance, to produce the static images for our performance flow graph. Importantly, this rigid head motion does not need to be very accurate for our method to operate, and we intend that an automated technique could be employed. A selection of video frames from one of the views is shown in Fig. 7, along with renderings of the results of our method. Despite the noisy quality of the videos and the smaller size of the head in the frame, our method is able to capture stable facial motion including lip synching and brow wrinkles.

7.3 High-Resolution Detail Transfer

After tracking a performance, we transfer the high-resolution reflectance maps from the static scans onto the performance result. As all results are registered to the same UV parameterization by our method, the transfer is a simple weighted blend using the cross-correlation-based confidence weights w_i^{pq} of each vertex, interpolated bilinearly between vertices. We also compute values for w_i^{pq} for any dynamic-to-static edge pq that was not present in the performance flow graph, to produce weights for every frame of the performance. This yields detailed reflectance maps for every performance frame, suitable for realistic rendering and relighting. In addition to transferring reflectance, we also transfer geometric details in the form of a displacement map, allowing the performance tracking to operate on a medium-resolution mesh instead of the full scan resolution. Fig. 8 compares transferring geometric details

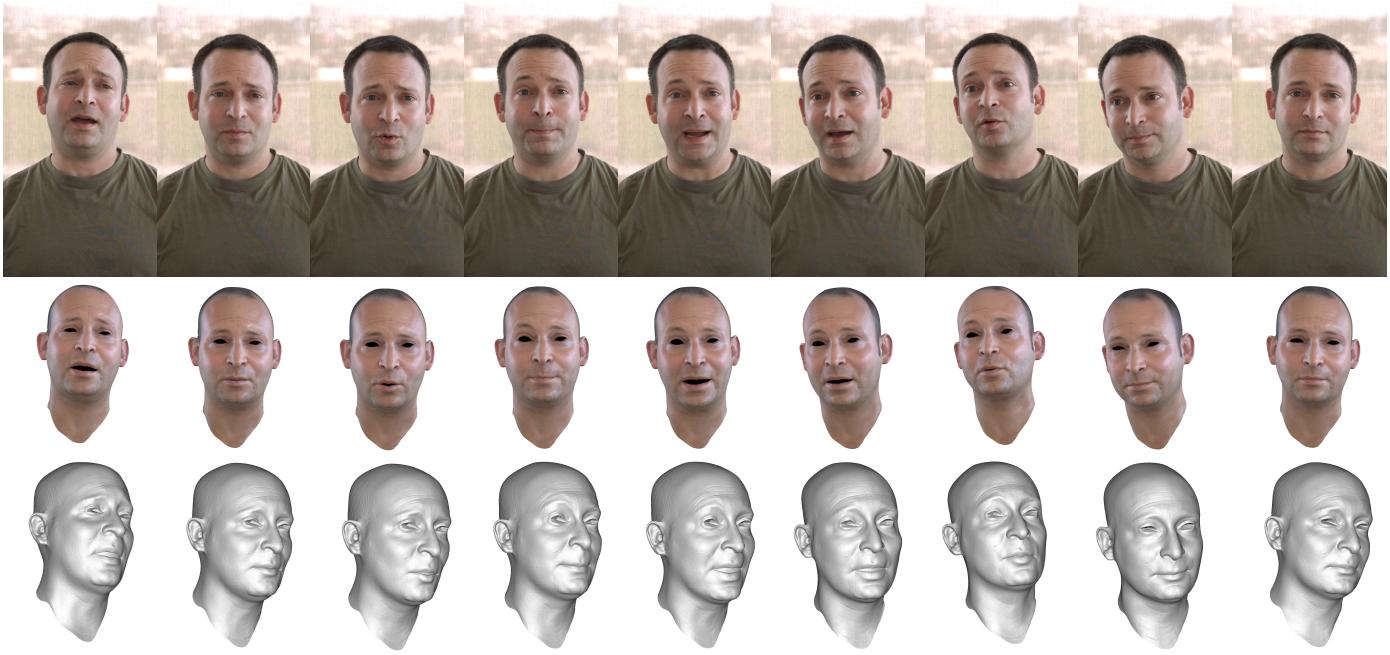
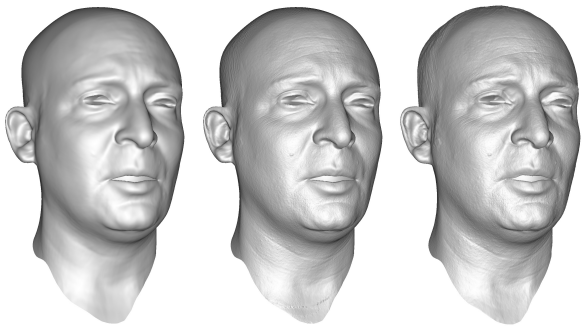


Figure 7: *A performance captured in an office environment with uncontrolled illumination, using four HD consumer video cameras and seven static expression scans. Top row: a selection of frames from one of the camera views. Middle row: geometry tracked using the proposed method, with reflectance maps automatically assembled from static scan data, shaded using a high-dynamic-range light probe. The reflectance of the top and back of the head were supplemented with artist-generated static maps. The eyes and inner mouth are rendered as black as our method does not track these features. Bottom row: gray-shaded geometry for the same frames, from a novel viewpoint. Our method produces stable animation even with somewhat noisy video footage and significant head motion. Dynamic skin details such as brow wrinkles are transferred from the static scans in a manner faithful to the video footage.*



(a) (b) (c)

Figure 8: *High-resolution details may be transferred to a medium-resolution tracked model to save computation time. (a) medium-resolution tracked geometry using six views. (b) medium-resolution geometry with details automatically transferred from the high-resolution static scans. (c) high-resolution tracked geometry. The transferred details in (b) capture most of the dynamic facial details seen in (c) at a reduced computational cost.*

from the static scans onto a medium-resolution reconstruction to directly tracking a high-resolution mesh. As the high-resolution solve is more expensive, we first perform the medium-resolution solve and use it to prime the DDMS-TRWS belief in the high-resolution solve, making convergence more rapid. In all other results, we show medium-resolution tracking with detail transfer, as the results are satisfactory and far cheaper to compute.

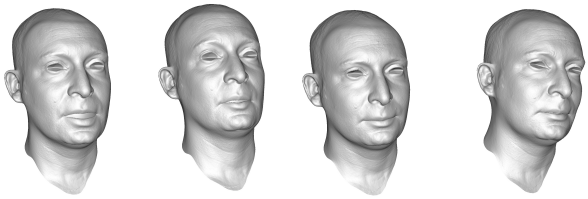


Figure 9: *Results using only a single camera view, showing the last four frames from Fig. 7. Even under uncontrolled illumination and significant head motion, tracking is possible from a single view, at somewhat reduced fidelity.*

7.4 Monocular vs. Binocular vs. Multi-View

Our method operates on any number of camera views, producing a result from even a single view. Fig. 9 shows results from a single view for the same uncontrolled-illumination sequence as Fig. 7. Fig. 10 shows the incremental improvement in facial detail for a controlled-illumination sequence using one, two, and six views. Our method is applicable to a wide variety of camera and lighting setups, with graceful degradation as less information is available.

7.5 Influence of Each Energy Term

The core operation of our method is to propagate a known facial pose (the artist mesh) to a set of unknown poses (the dynamic frames and other static scans) via the ground term and correspondence terms in our energy formulation. The differential shape term and shrink wrap term serve to regularize the shape of the solution. We next explore the influence of these terms on the solution.

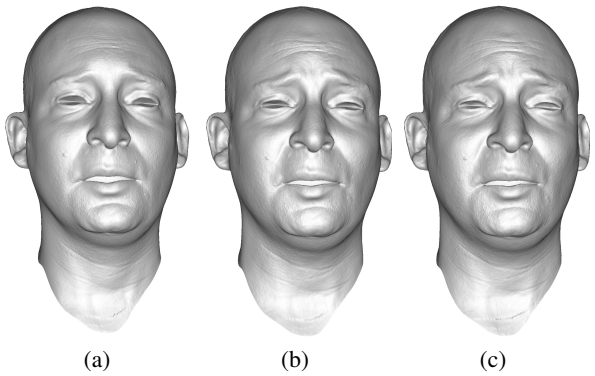


Figure 10: *Example dynamic performance frame reconstructed from (a) one view, (b) two views and (c) six views. Our method gracefully degrades as less information is available.*

Figure 11: *The artist mesh is non-rigidly registered to each of the other static expression scans as a byproduct of our method. The registered artist mesh is shown for a selection of scans from two different subjects. Note the variety of mouth shapes, all of which are well-registered by our method without any user input.*

Correspondence Term The correspondence term produces a consistent parameterization of the geometry suitable for texturing and other editing tasks. As our method computes a coupled solution of performance frames using static poses to bridge larger temporal gaps, the artist mesh is non-rigidly registered to each of the static scans as a byproduct of the optimization. (See Fig. 11 for examples.) Note especially that our method automatically produces a complete head for each expression, despite only having static facial scan geometry for the frontal face surface. As shown in Fig. 12, this consistency is maintained even when the solution is obtained from a different performance. Fig. 13 illustrates that the use of multiple static expression scans in the performance flow graph produces a more expressive performance, with more accentuated facial expression features, as there are more successful optical flow regions in the face throughout the performance.

Differential Shape Term In our formulation, the differential shape of a performance frame or pose is tied to a blend of its neighbors on the performance flow graph. This allows details from multiple static poses to propagate to related poses. Even when only one static pose is used (i.e. neutral), allowing temporal neighbors to influence the differential shape provides temporal smoothing without overly restricting the shape of each frame. Fig. 13 (c, d) illustrates the loss of detail when temporal neighbors are excluded from the differential shape term (compare to a, b).

Figure 12: *Top row: neutral mesh with checker visualization of texture coordinates, followed by three non-rigid registrations to other facial scans as a byproduct of tracking a speaking performance. Bottom row: the same, except the performance used was a series of facial expressions with no speaking. The non-rigid registration obtained from the performance-graph-based tracking is both consistent across expressions and across performances. Note, e.g. the consistent locations of the checkers around the contours of the lips.*

Shrink Wrap Term The shrink wrap term conforms the static poses to the raw geometry scans (Fig. 14). Without this term, subtle details in the static scans cannot be propagated to the performance result, and the recovered static poses have less fidelity to the scans.

7.6 Comparison to Previous Work

We ran our method on the data from [Beeler et al. 2011], using their recovered geometry from the first frame (frame 48) as the "artist" mesh in our method. For expression scans, we used the geometry from frames 285 (frown) and 333 (brow raise). As our method makes use of the expression scans only via image-space operations on camera footage or rasterized geometry, any point order information present in the scans is entirely ignored. Therefore in this test, it is as if the static scans were produced individually by the method of [Beeler et al. 2010]. We constructed a simple UV projection on the artist mesh for texture visualization purposes, and projected the video frames onto each frame's geometry to produce a per-frame UV texture map. To measure the quality of texture alignment over the entire sequence, we computed the temporal variance of each pixel in the texture map (shown in Fig.15 (a, b)), using contrast normalization to disregard low-frequency shading variation. The proposed method produces substantially lower temporal texture variance, indicating a more consistent alignment throughout the sequence, especially around the mouth. Examining the geometry in Fig.15 (c-f), the proposed method has generally comparable quality as the previous work, with the mouth-closed shape recovered more faithfully (which is consistent with the variance analy-

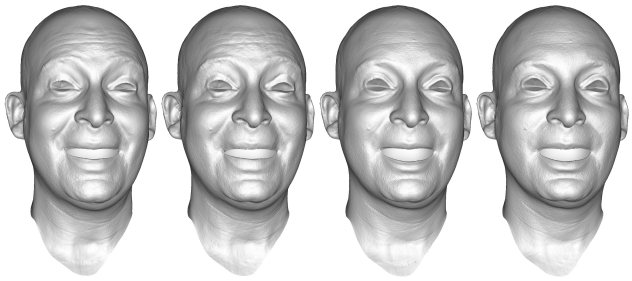


Figure 13: *Using multiple static expressions in the performance flow graph produces more detail than using just a neutral static expression. Multiple static expressions are included in the performance flow graph in (a, c), whereas only the neutral expression is included in (b, d). By including temporal neighbors and static scans in determining the differential shape, details from the various static scans can be propagated throughout the performance. Differential shape is determined by the static expression(s) and temporal neighbors in (a, b), whereas temporal neighbors are excluded from the differential shape term in (c, d). Note the progressive loss of detail in e.g. the brow region from (a) to (d).*

sis). We also compared to [Klaudiny and Hilton 2012] in a similar manner, using frame 0 as the artist mesh, and frames 25, 40, 70, 110, 155, 190, 225, 255 and 280 as static expressions. Again, no point order information is used. Fig. 16 again shows an overall lower temporal texture variance from the proposed method.

7.7 Performance Timings

We report performance timings in Fig. 17 for various sequences, running on a 16-core 2.4 GHz Xeon E5620 workstation (some operations are multithreaded across the cores). All tracked meshes have 65 thousand vertices, except Fig. 8(c) and Fig. 15 which have one million vertices. We report each stage of the process: "Graph" for the performance graph construction, "Flow" for the high-resolution optical flow calculations, "Key" for the performance tracking solve on key frames, and "Tween" for the performance tracking solve in between key frames. We mark stages that could be parallelized over multiple machines with an asterisk (*). High-resolution solves (Fig. 8(c) and Fig. 15) take longer than medium-resolution solves. Sequences with uncontrolled illumination (Fig. 7 and Fig. 9) take longer for the key frames to converge since the correspondence tying the solution to the static scans has lower confidence.

7.8 Discussion

Our method produces a consistent geometry animation on an artist-created neutral mesh. The animation is expressive and lifelike, and the subject is free to make natural head movements within a certain degree. Fig. 18 shows renderings from such a facial performance rendered using advanced skin and eye shading techniques as described in [Jimenez et al. 2012]. One notable shortcoming of our performance flow graph construction algorithm is the neglect of eye blinks. This results in a poor representation of the blinks in the final animation. Our method requires one artist-generated mesh per subject to obtain results that are immediately usable in production pipelines. Automatic generation of this mesh could be future work, or use existing techniques for non-rigid registration. Omitting this step would still produce a result, but would require additional cleanup around the edges as in e.g. [Beeler et al. 2011][Klaudiny and Hilton 2012].

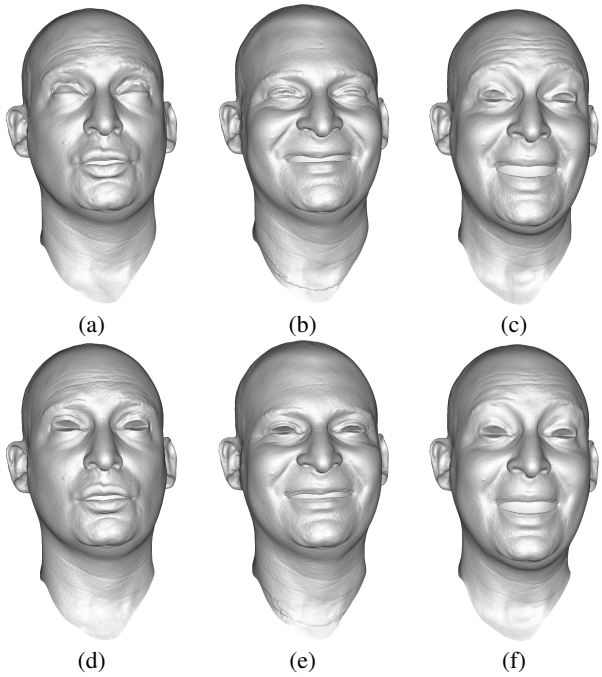


Figure 14: *The shrink wrap term conforms the artist mesh to the static scan geometry, and also improves the transfer of expressive details to the dynamic performance. The registered artist mesh is shown for two static poses in (a) and (b), and a dynamic pose that borrows brow detail from (a) and mouth detail from (b) is shown in (c). Without the shrink wrap term, the registration to the static poses suffers (d, e) and the detail transfer to the dynamic performance is less sucessful (f). Fine-scale details are still transferred via displacement maps, but medium-scale expressive details are lost.*

8 Future Work

One of the advantages of our technique is that it relates a dynamic performance back to facial shape scans using per-pixel weight maps. It would be desirable to further factor our results to create multiple localized blend shapes which are more semantically meaningful and artist friendly. Also, our algorithm does not explicitly track eye or mouth contours. Eye and mouth tracking could be further refined with additional constraints to capture eye blinks and more subtle mouth behavior such as "sticky lips" [Alexander et al. 2009]. Another useful direction would be to retarget performances from one subject to another. Given a set of static scans for both subjects, it should be possible to clone one subject's performance to the second subject as in [Seol et al. 2012]; providing more meaningful control over this transfer remains a subject for future research. Finally, as our framework is agnostic to the particular method employed for estimating 2D correspondences, we would like to try more recent optical flow algorithms such as the top performers on the Middlebury benchmark [Baker et al. 2011]. Usefully, the quality of our performance tracking can be improved any time that an improved optical flow library becomes available.

Acknowledgements

The authors thank the following people for their support and assistance: Ari Shapiro, Sin-Hwa Kang, Matt Trimmer, Koki Nagano, Xueming Yu, Jay Busch, Paul Graham, Kathleen Haase, Bill Swartout, Randall Hill and Randolph Hall. We thank the authors of [Beeler et al. 2010] and [Klaudiny et al. 2010] for graciously

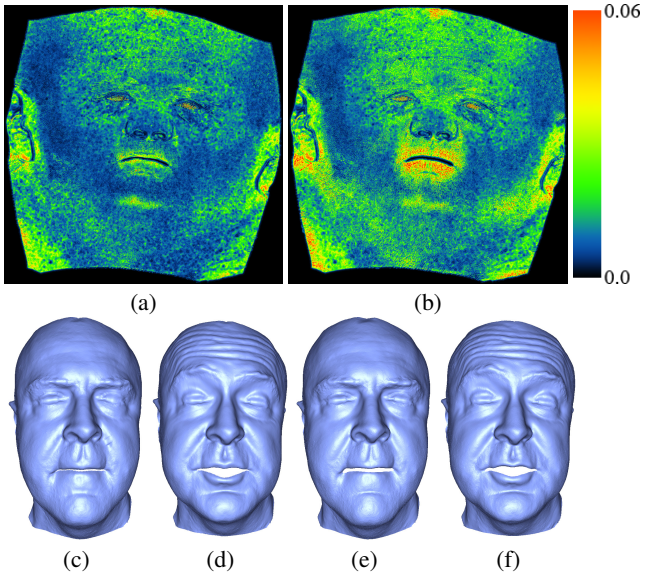


Figure 15: *Top row: Temporal variance of contrast-normalized texture (false color, where blue is lowest and red is highest), with (a) the proposed method and (b) the method of [Beeler et al. 2011]. The variance of the proposed method is substantially lower, indicating a more consistent texture alignment throughout the sequence. Bottom row: Geometry for frames 120 and 330 of the sequence, with (c, d) the proposed method and (e, f) the prior work.*

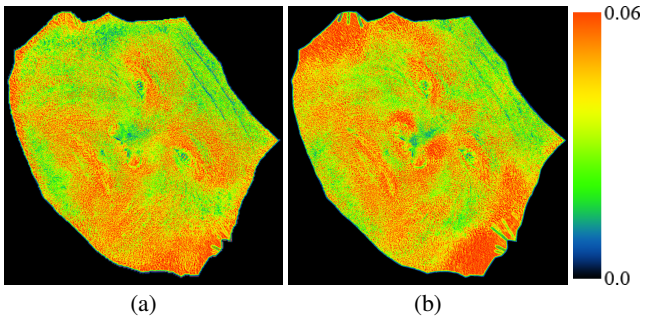


Figure 16: *Temporal variance of contrast-normalized texture (false color, where blue is lowest and red is highest), with (a) the proposed method and (b) the method of [Klaudiny et al. 2010]. As in Fig.15, the variance of the proposed method is generally lower.*

Sequence	Frames	Graph*	Flow*	Key	Tween*
Fig. 7	170	0.5 hr	8.0 hr	5.2 hr	1.2 hr
Fig. 8(b)	400	1.1 hr	24 hr	4.3 hr	4.3 hr
Fig. 8(c)	400	1.1 hr	24 hr	24 hr	26 hr
Fig. 9	170	0.5 hr	2.0 hr	3.6 hr	0.9 hr
Fig. 15	347	0.1 hr	15 hr	16 hr	17 hr
Fig. 16	300	0.2 hr	12 hr	3.0 hr	3.0 hr
Fig. 19 row 2	600	1.6 hr	36 hr	6.5 hr	7.0 hr
Fig. 19 row 4	305	0.8 hr	18 hr	3.3 hr	3.5 hr
Fig. 19 row 6	250	0.7 hr	15 hr	2.6 hr	2.8 hr

Figure 17: *Timings for complete processing of the sequences used in various figues, using a single workstation. A * indicates an operation that could trivially be run in parallel across many machines.*

providing the data for the comparisons in Figs. 15 and 16, respectively. We thank Jorge Jimenez, Etienne Danvoye, and Javier von der Pahlen at Activision R&D for the renderings in Fig. 18. This work was sponsored by the University of Southern California Office of the Provost and the U.S. Army Research, Development, and Engineering Command (RDECOM). The content of the information does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

References

ALEXANDER, O., ROGERS, M., LAMBETH, W., CHIANG, M., AND DEBEVEC, P. 2009. Creating a photoreal digital actor: The digital emily project. In *Visual Media Production, 2009. CVMP '09. Conference for*, 176–187.

BAKER, S., SCHARSTEIN, D., LEWIS, J. P., ROTH, S., BLACK, M. J., AND SZELISKI, R. 2011. A database and evaluation methodology for optical flow. *International Journal of Computer Vision 92*, 1 (Mar.), 1–31.

BEELER, T., BICKEL, B., BEARDSLEY, P., SUMNER, B., AND GROSS, M. 2010. High-quality single-shot capture of facial geometry. *ACM Trans. on Graphics (Proc. SIGGRAPH) 29*, 3, 40:1–40:9.

BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R. W., AND GROSS, M. 2011. High-quality passive facial performance capture using anchor frames. In *ACM SIGGRAPH 2011 papers*, ACM, New York, NY, USA, SIGGRAPH '11, 75:1–75:10.

BICKEL, B., LANG, M., BOTSCH, M., OTADUY, M. A., AND GROSS, M. 2008. Pose-space animation and transfer of facial details. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '08, 57–66.

BORSHUKOV, G., PIPONI, D., LARSEN, O., LEWIS, J. P., AND TEMPELAAR-LIETZ, C. 2003. Universal capture: image-based facial animation for "the matrix reloaded". In *SIGGRAPH*, ACM, A. P. Rockwood, Ed.

BRADLEY, D., HEIDRICH, W., POPA, T., AND SHEFFER, A. 2010. High resolution passive facial performance capture. In *ACM SIGGRAPH 2010 papers*, ACM, New York, NY, USA, SIGGRAPH '10, 41:1–41:10.

COOTES, T. F., EDWARDS, G. J., AND TAYLOR, C. J. 1998. Active appearance models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Springer, 484–498.

DEBEVEC, P. 1998. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '98, 189–198.

DECARLO, D., AND METAXAS, D. 1996. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, IEEE Computer Society, Washington, DC, USA, CVPR '96, 231–238.

EKMAN, P., AND FRIESEN, W. 1978. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press, Palo Alto.

GHOSH, A., FYFFE, G., TUNWATTANAPONG, B., BUSCH, J., YU, X., AND DEBEVEC, P. 2011. Multiview face capture using

Figure 18: *Real-time renderings of tracked performances using advanced skin and eye shading [Jimenez et al. 2012].*

polarized spherical gradient illumination. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, ACM, New York, NY, USA, SA '11, 129:1–129:10.

GUENTER, B., GRIMM, C., WOOD, D., MALVAR, H., AND PIGHIN, F. 1998. Making faces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '98, 55–66.

HAWKINS, T., WENGER, A., TCHOU, C., GARDNER, A., GÖRANSSON, F., AND DEBEVEC, P. 2004. Animatable facial reflectance fields. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, 309–320.

HUANG, H., CHAI, J., TONG, X., AND WU, H.-T. 2011. Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition. *ACM Trans. Graph. 30*, 4 (July), 74:1–74:10.

JIMENEZ, J., JARABO, A., GUTIERREZ, D., DANVOYE, E., AND VON DER PAHLEN, J. 2012. Separable subsurface scattering and photorealistic eyes rendering. In *ACM SIGGRAPH 2012 Courses*, ACM, New York, NY, USA, SIGGRAPH 2012.

KLAUDINY, M., AND HILTON, A. 2012. High-detail 3d capture and non-sequential alignment of facial performance. In *3DIMPVT*.

KLAUDINY, M., HILTON, A., AND EDGE, J. 2010. High-detail 3d capture of facial performance. In *3DPVT*.

KOLMOGOROV, V. 2006. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell. 28*, 10, 1568–1583.

LI, H., ROIVAINEN, P., AND FORCHEIMER, R. 1993. 3-d motion estimation in model-based facial image coding. *IEEE Trans. Pattern Anal. Mach. Intell. 15*, 6 (June), 545–555.

MA, W.-C., JONES, A., CHIANG, J.-Y., HAWKINS, T., FREDERIKSEN, S., PEERS, P., VUKOVIC, M., OUHYOUNG, M., AND DEBEVEC, P. 2008. Facial performance synthesis using deformation-driven polynomial displacement maps. *ACM Trans. Graph. 27*, 5 (Dec.), 121:1–121:10.

NISHINO, K., AND NAYAR, S. K. 2004. Eyes for relighting. *ACM Trans. Graph. 23*, 3, 704–711.

PARK, M., KASHYAP, S., COLLINS, R., AND LIU, Y. 2010. Data driven mean-shift belief propagation for non-gaussian mrfs. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 3547 –3554.

POPA, T., SOUTH-DICKINSON, I., BRADLEY, D., SHEFFER, A., AND HEIDRICH, W. 2010. Globally consistent space-time reconstruction. *Computer Graphics Forum (Proc. SGP)*.

SEOL, Y., LEWIS, J., SEO, J., CHOI, B., ANJYO, K., AND NOH, J. 2012. Spacetime expression cloning for blendshapes. *ACM Trans. Graph. 31*, 2 (Apr.), 14:1–14:12.

VALGAERTS, L., WU, C., BRUHN, A., SEIDEL, H.-P., AND THEOBALT, C. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graph. 31*, 6 (Nov.), 187:1–187:11.

WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. 2011. Realtime performance-based facial animation. In *ACM SIGGRAPH 2011 papers*, ACM, New York, NY, USA, SIGGRAPH '11, 77:1–77:10.

WERLBERGER, M. 2012. *Convex Approaches for High Performance Video Processing*. PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Spacetime faces: high resolution capture for modeling and animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 548–558.

ZHU, X., AND RAMANAN, D. 2012. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, 2879–2886.

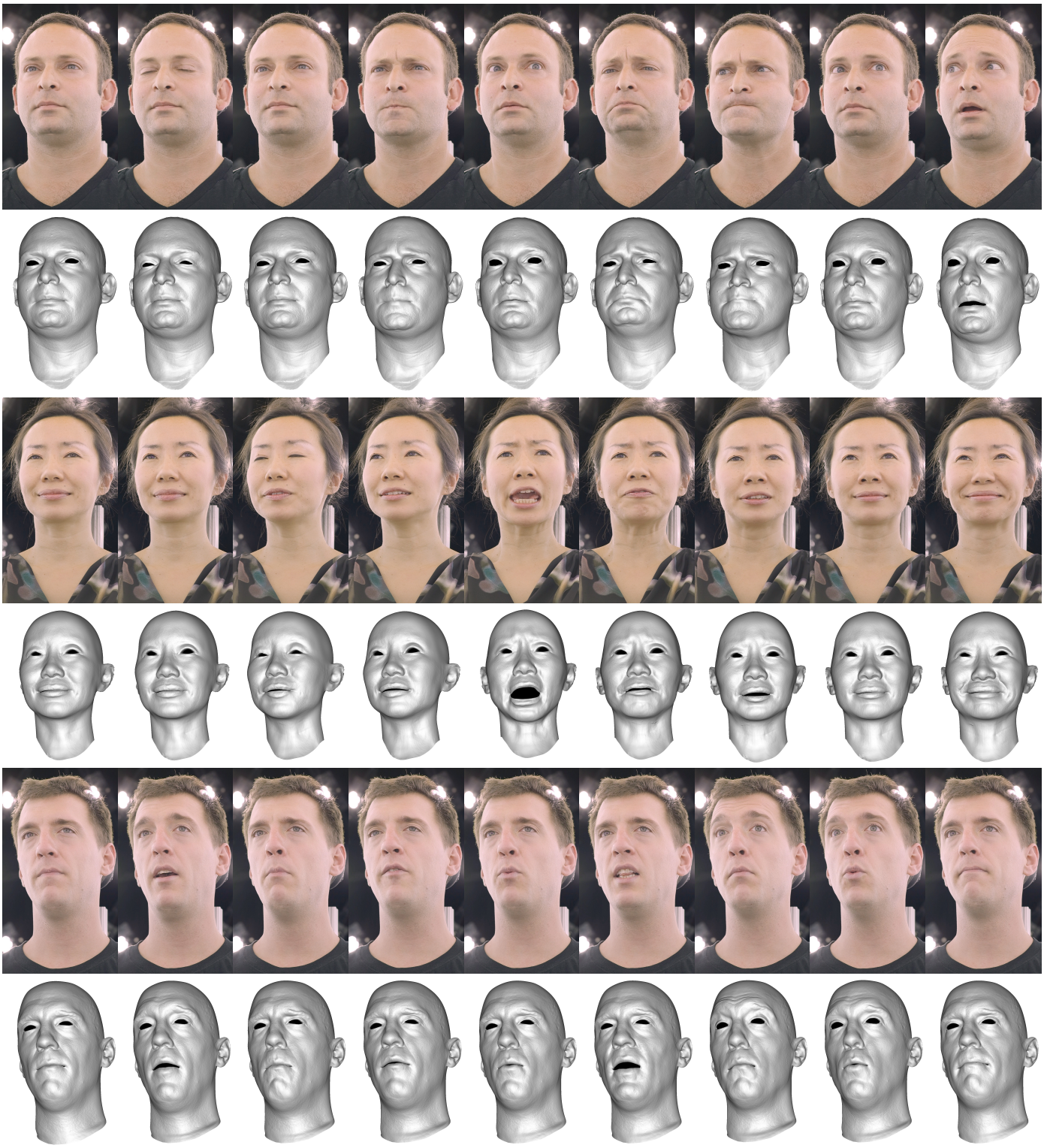


Figure 19: *Three tracked performances with different subjects, using six camera views and six to eight static expression scans per subject. Shown are alternating rows of selected frames from the performance video, and gray-shaded tracked geometry for the same frames. Our method produces a consistent geometry animation on an artist-created neutral mesh. The animation is expressive and lifelike, and the subject is free to make natural head movements within a certain degree.*

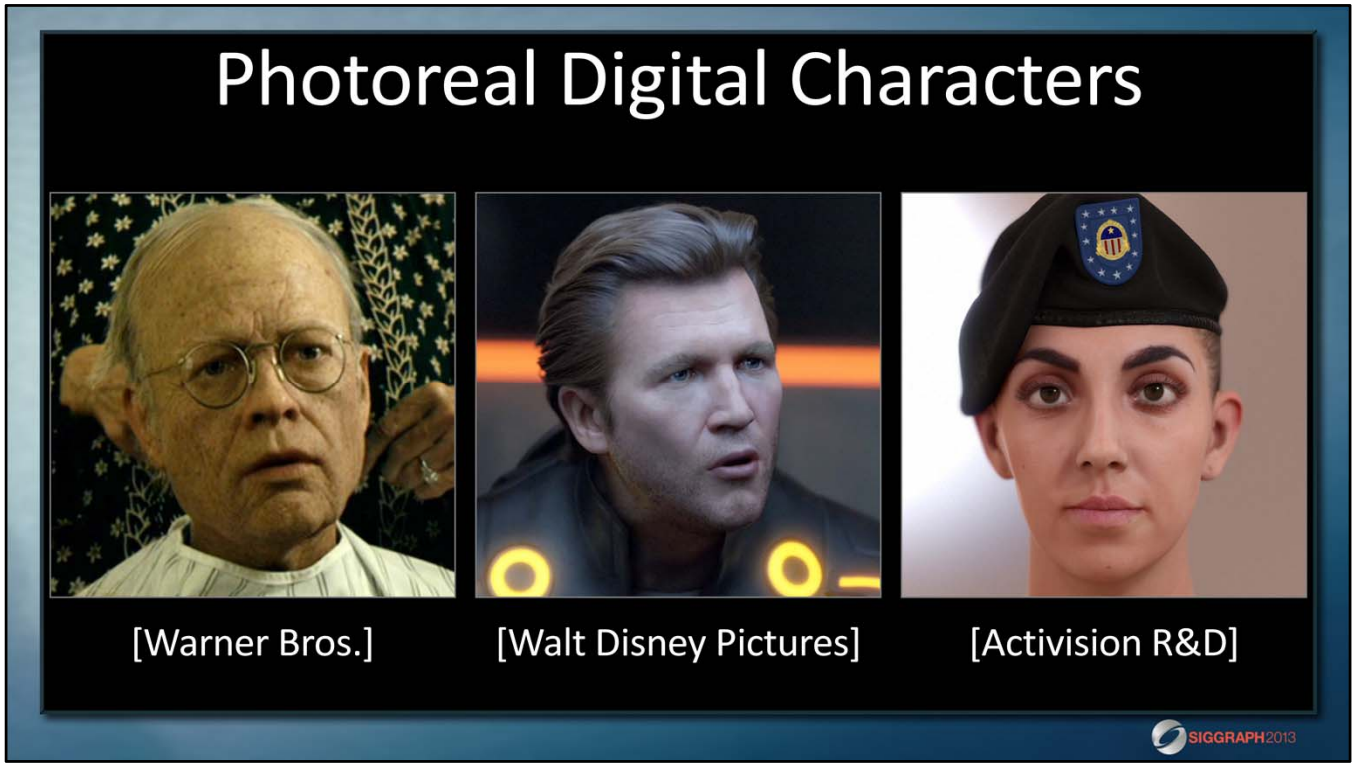
Driving High-Resolution Facial Blendshapes with Video Performance Capture

Graham Fyffe Andrew Jones
Oleg Alexander Ryosuke Ichikari Paul Debevec

USC Institute for Creative Technologies

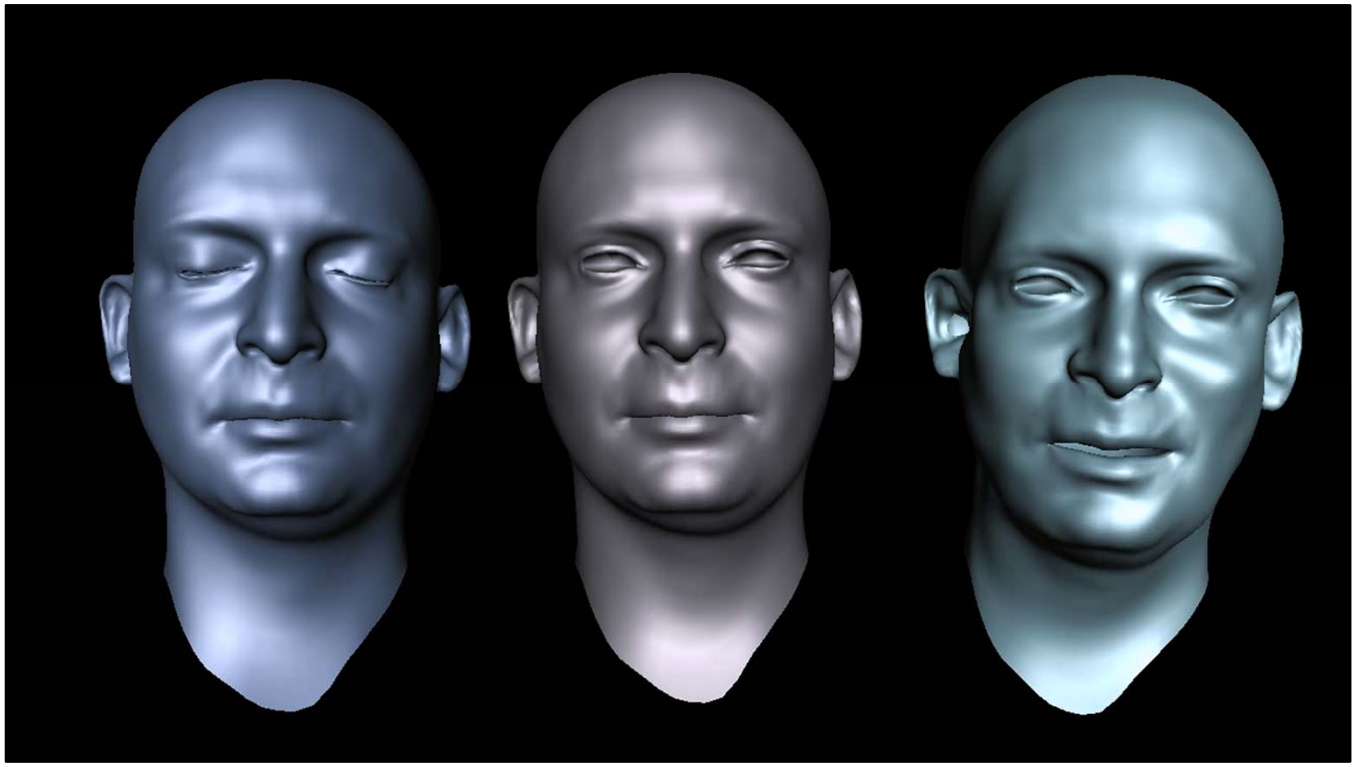
SIGGRAPH2013

Hi everyone. I'm going to present the latest work we've been doing at USC ICT in facial performance capture, using video streams to drive high resolution blendshape scans.



Digital characters are everywhere, and they are arguably approaching photorealism. But they remain difficult and costly to animate in a realistic fashion. When a character's physical appearance is almost like a real person, any problems in their movement will be that much more obvious.

At USC we create realistic characters based on Light Stage scans. The scans take only about two and a half seconds each, so we can capture expressive poses like the ones here. Now we obtain not only detailed geometry, but also detailed diffuse and specular reflectance properties, enabling realistic renderings of the face using a skin shader. Static scans are nice because they enable the highest quality and detail, which would be difficult to obtain in a dynamic setting. Then we had the idea to switch our cameras over to video mode, and use the videos to animate the faces.



We solve the animation on an artist mesh, using the high resolution scans as a sort of guide for the shape and also the tracking. This directly produces an animated mesh that fits into most existing pipelines.

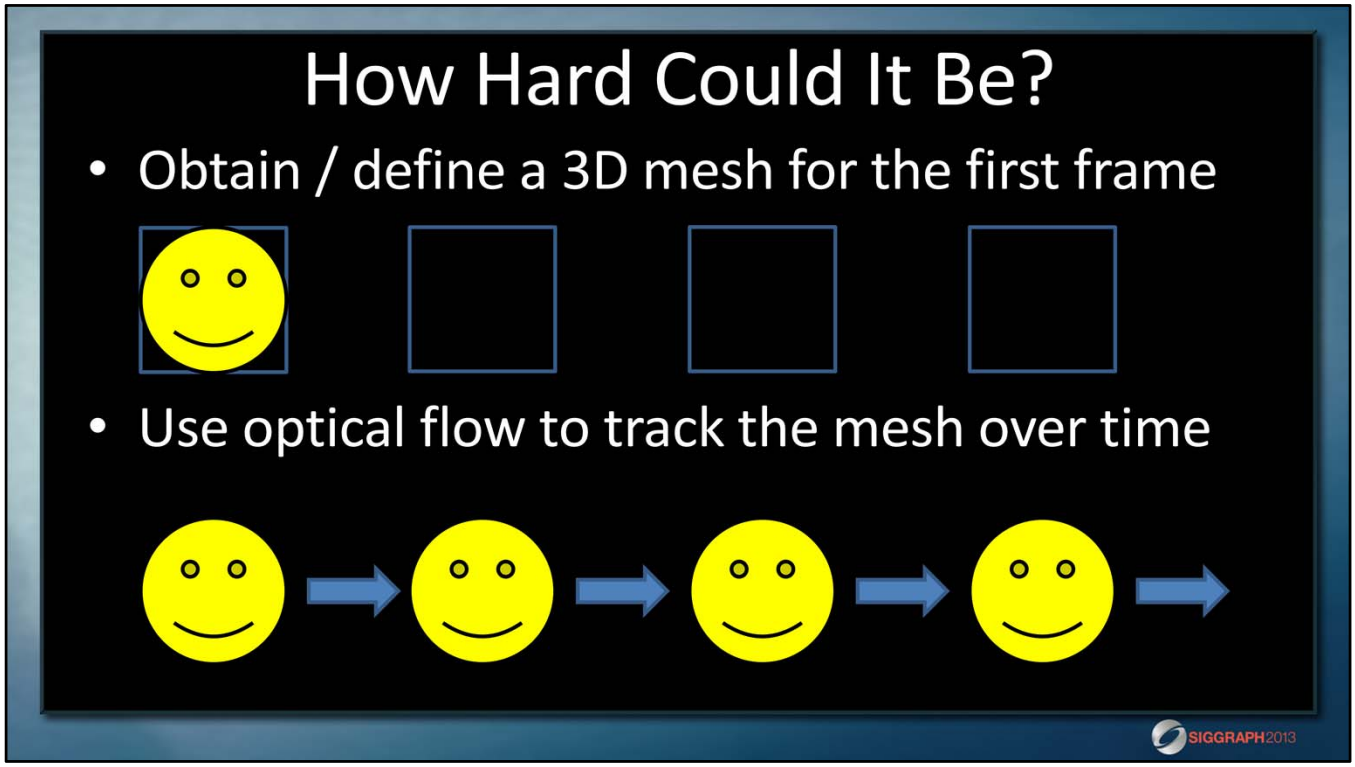
We also keep track of which parts of the face match which static scans, so we can transfer the appropriate wrinkle details and reflectance onto each frame of the performance.

Video Performance Capture

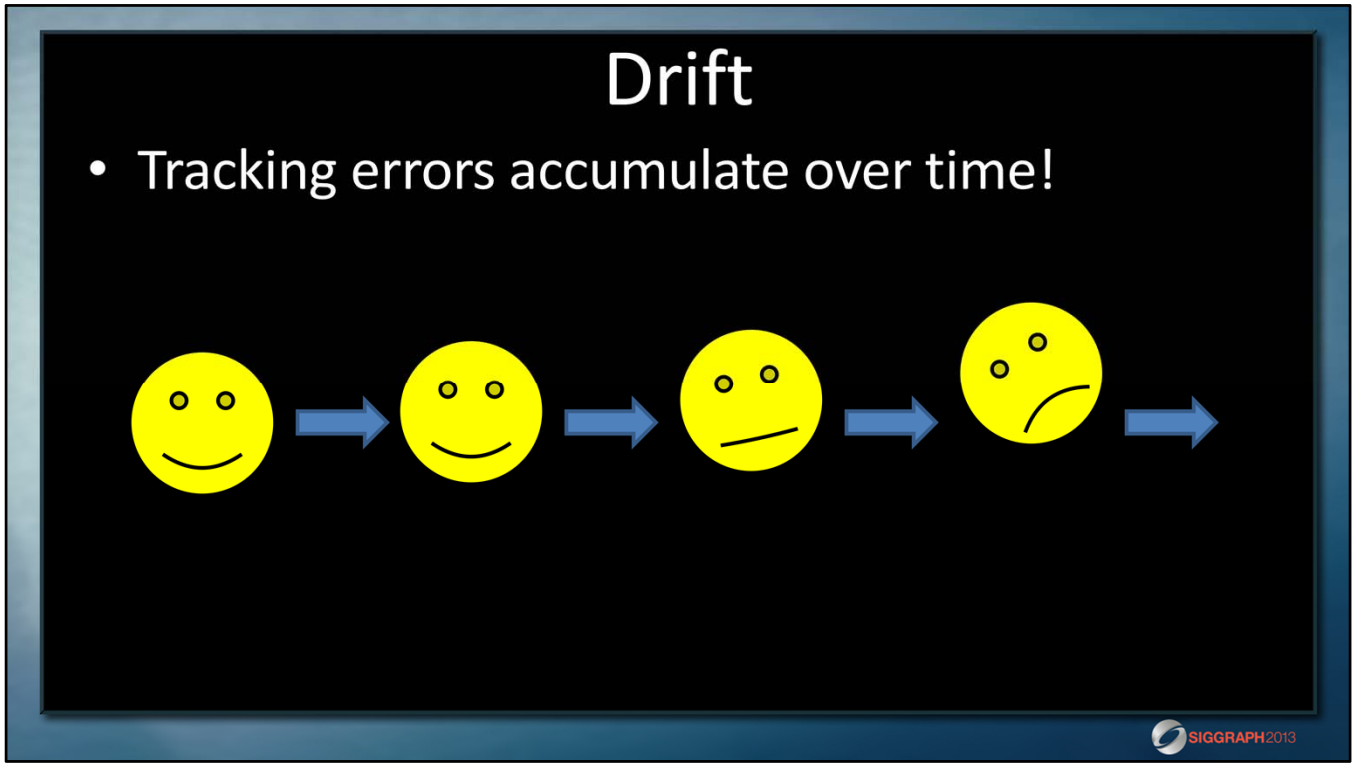
- Markerless
  - It's 2013
- HD Video
  - The skin's natural markers are visible (pores)

SIGGRAPH2013

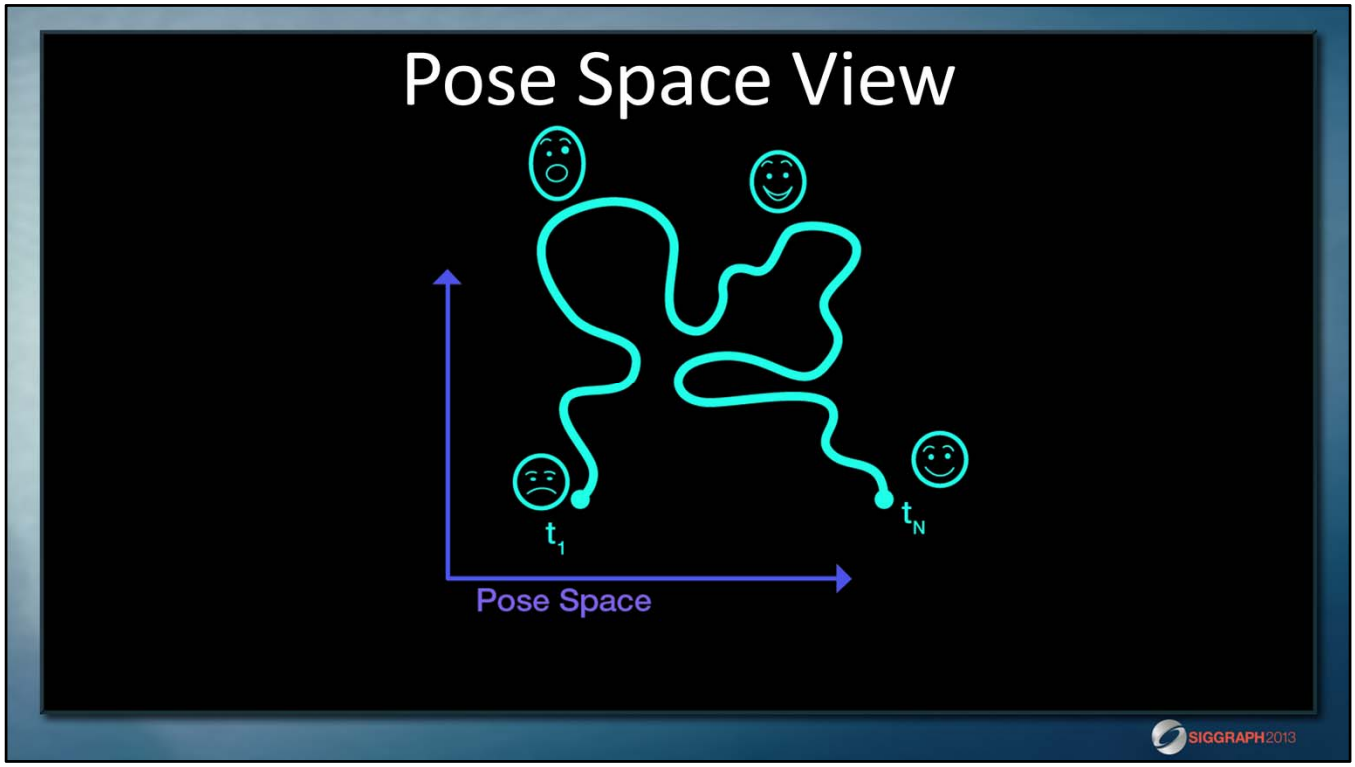
There are many performance capture techniques, but we focus only on markerless video based capture.
Consumer HD cameras have high enough resolution to show the skin's own natural markers, the skin pores.
We shoot our videos under fairly even, constant lighting.



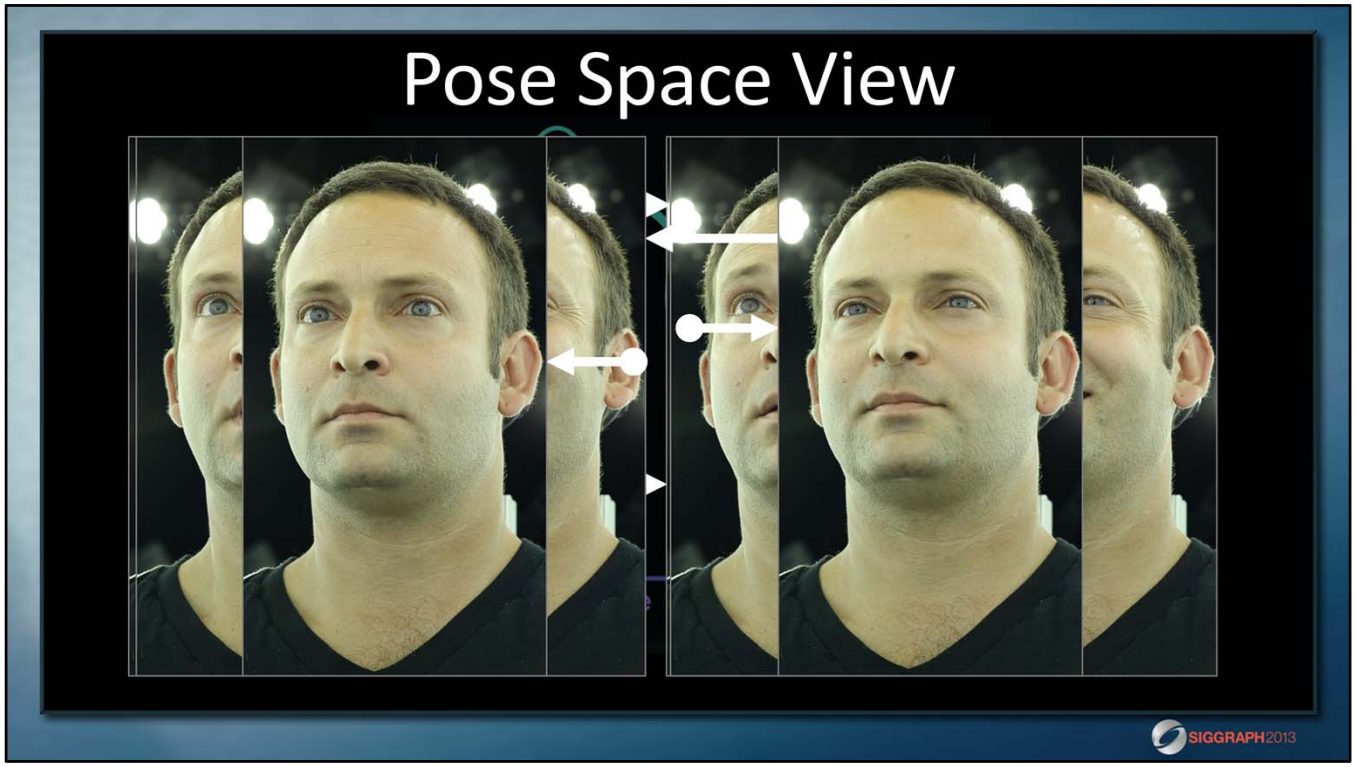
So how hard could this be? Can't you just set up a nice mesh for the first frame of the video, and use optical flow or something to propagate the mesh through time? Well, you can do that, but…



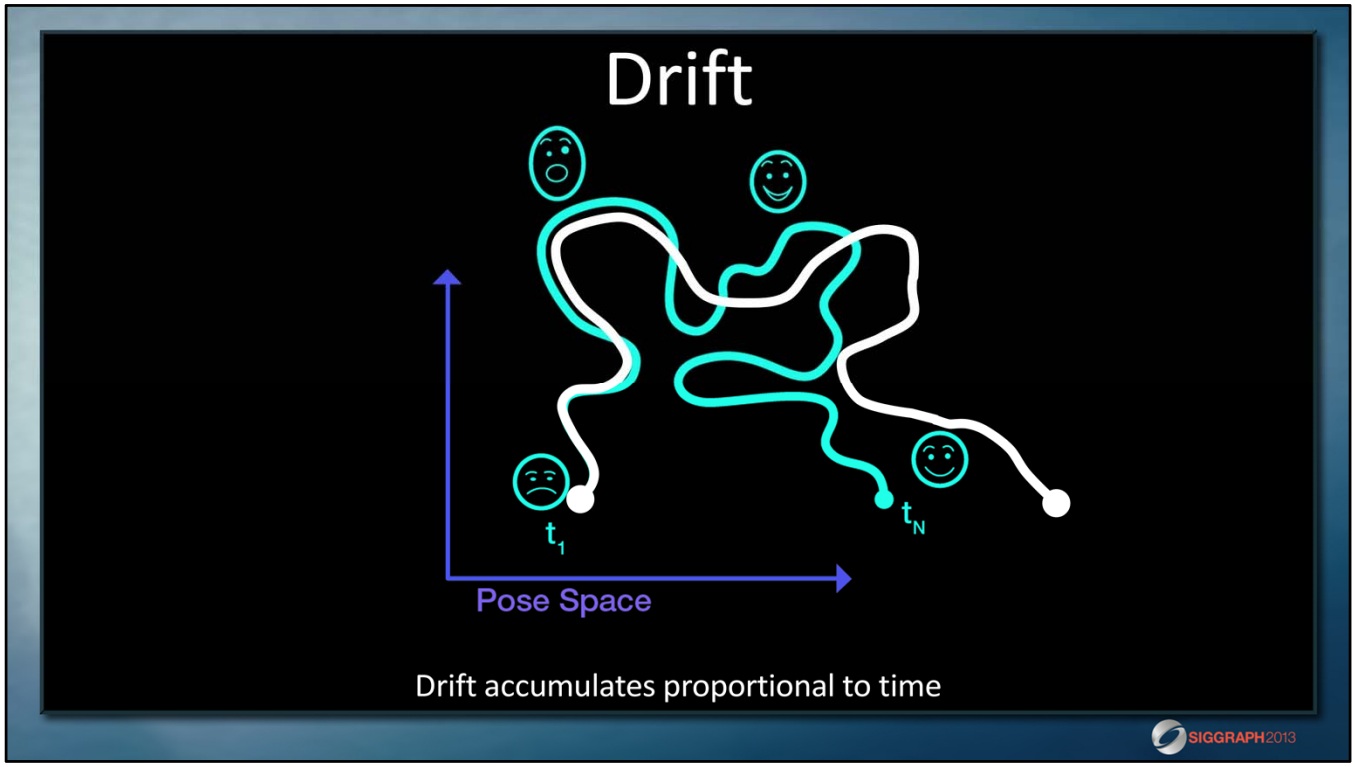
Even the slightest tracking errors will accumulate over time, resulting in what we call drift. The face mesh might drift away from the actual face in the video, which is one kind of drift. The face might also drift within the mesh parameterization, or texture space, which is equally bad since we want to keep the texture space totally consistent, so we can do things like blending textures from the different scans, or make consistent edits to the performance.



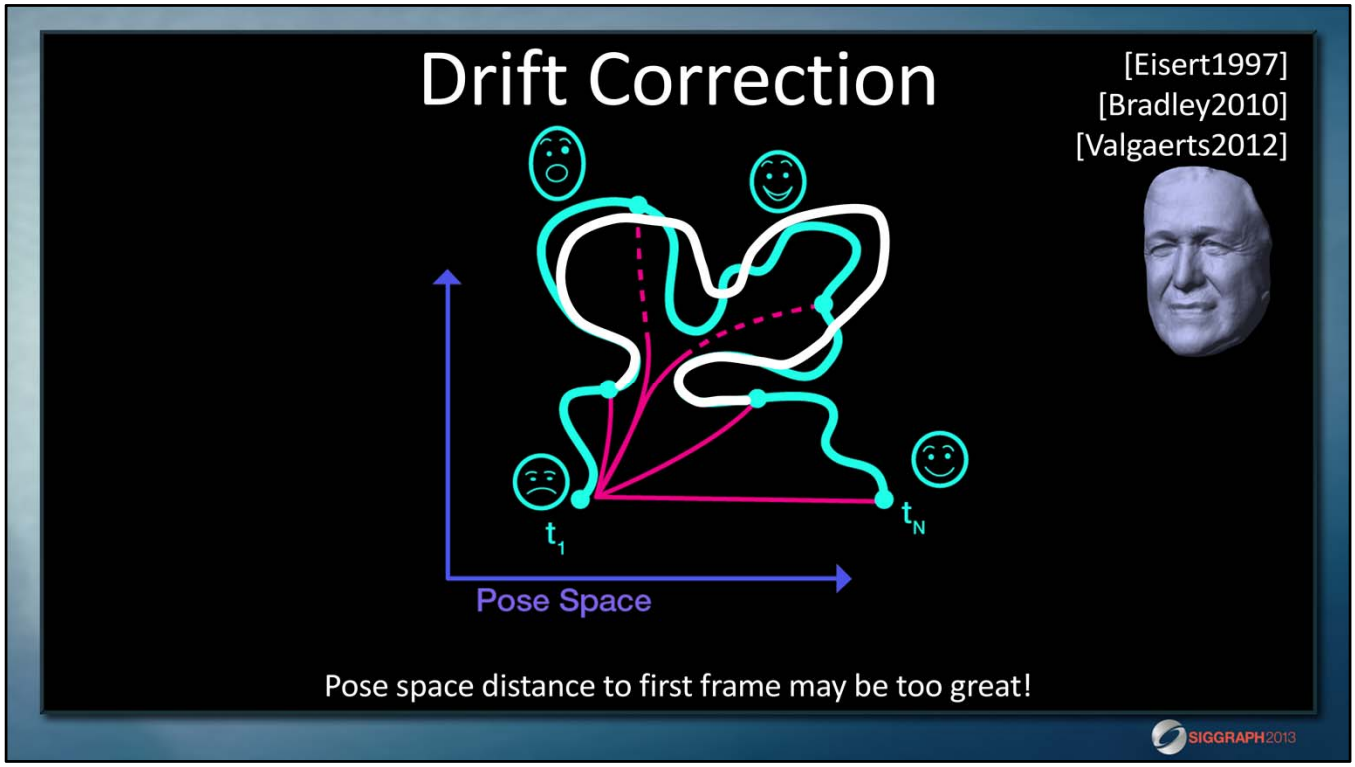
In order to understand a bit more about drift, let's look at the performance in another space, called pose space. Here, the performance timeline ends up as a curve through pose space.



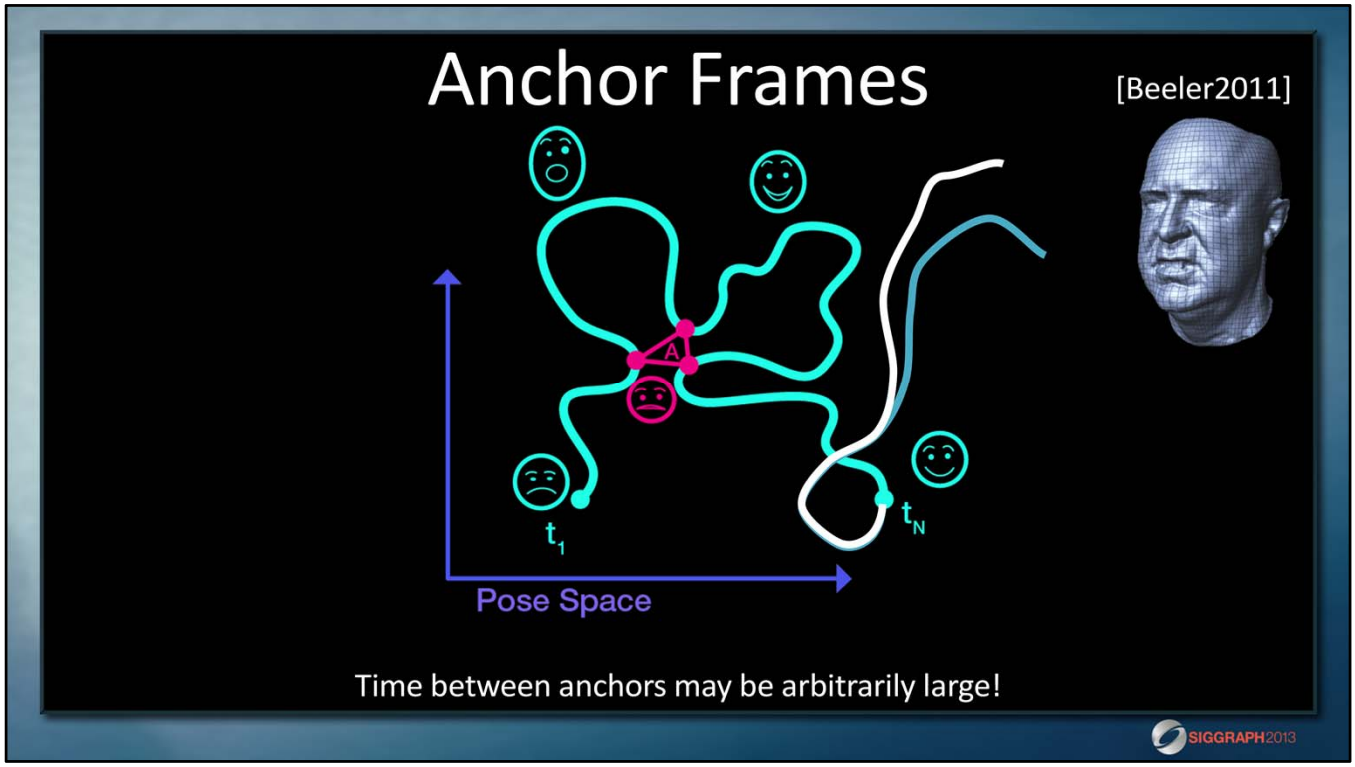
The idea of pose space is that face poses will be close together if they have similar appearance. Note that it's possible for poses to be close together in pose space even if they are far apart in time.



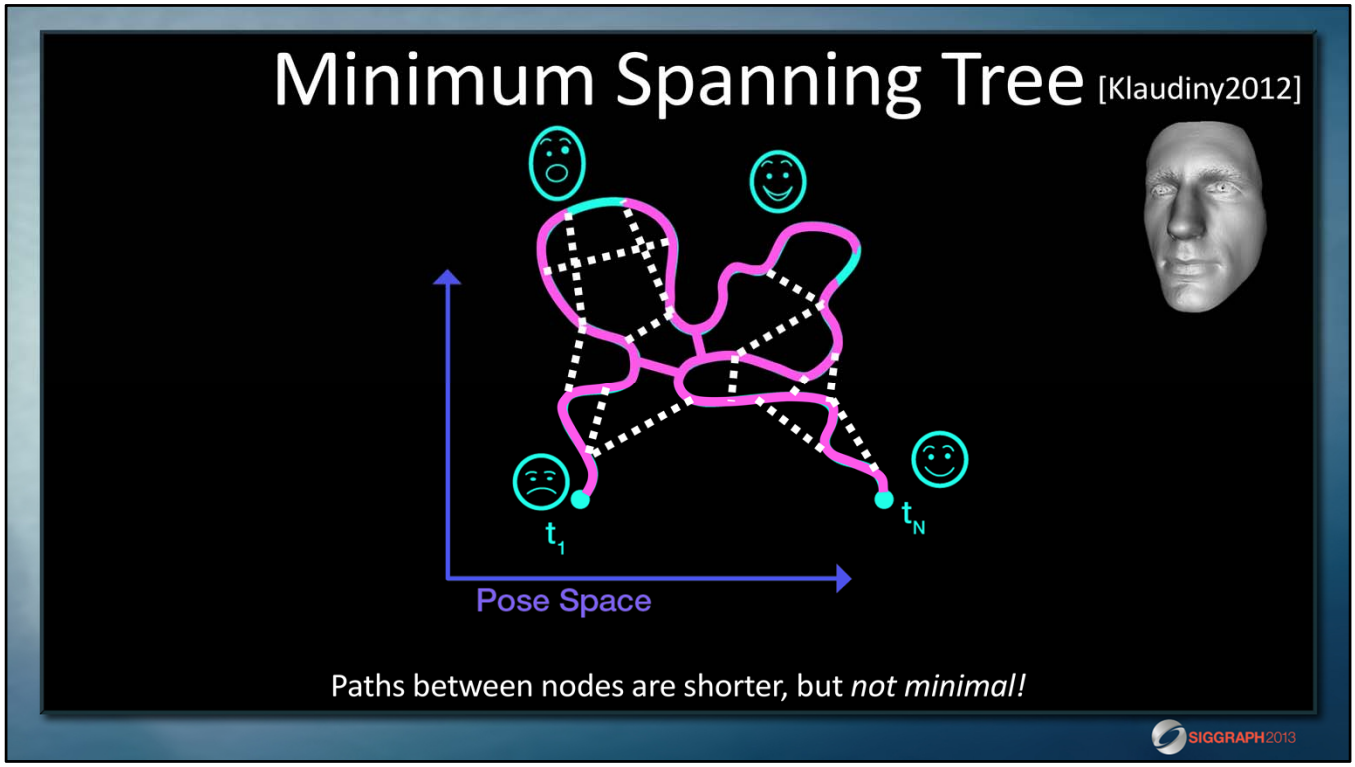
So again, if we track a performance starting from t1 all the way to tn, we get accumulated error, or drift. We'd like to avoid this.



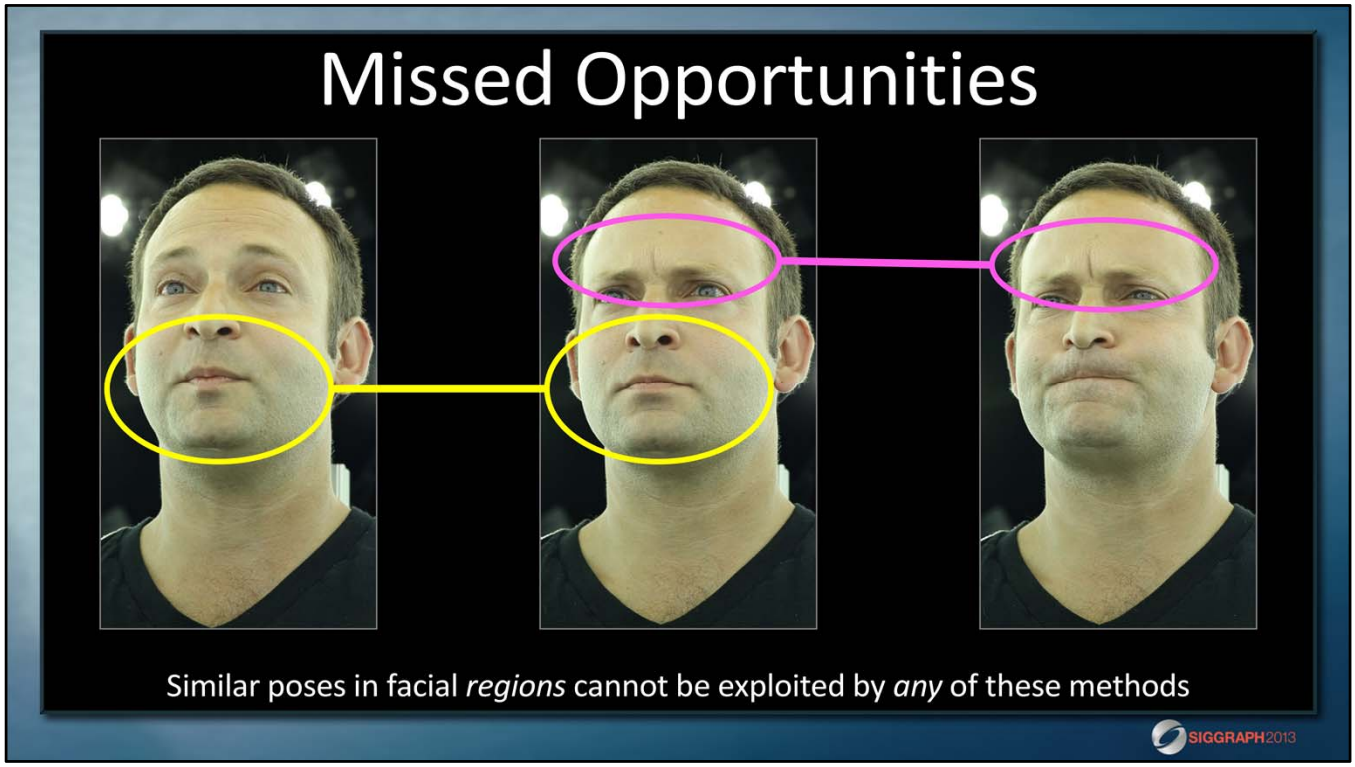
Previous work to avoid drift includes drift correction based on the first frame or a template. Every once in a while we compare the current frame to the template and make an adjustment. This works for frames that are close enough to the template to compute a good correspondence or optical flow. But if the appearance is too far away, like the dashed lines shown here, drift still occurs.



Another idea, from Beeler and colleagues is to carefully select a set of frames that resemble each other, called anchor frames. It's easy to correspond them to each other. Now, drift is limited to the loops of frames between the anchor frames, so drift is reduced. However, if the performance is very expressive, the facial pose might not return to a common appearance, and drift may still occur.



Another idea, from Klaudiny and colleagues, is to compute a minimum spanning tree over all the frames using an estimated distance matrix. This yields overall short paths between all the frames and a common root.
However, even the minimum spanning tree fails to capture the best available correspondences between any two frames.
To do that, you'd actually need to compute the **Essential Subgraph**, which contains all shortest paths,
And then come up with some sort of scheme to combine all these paths. That might be nice, but I'd like to point out something that we are so far ignoring entirely.

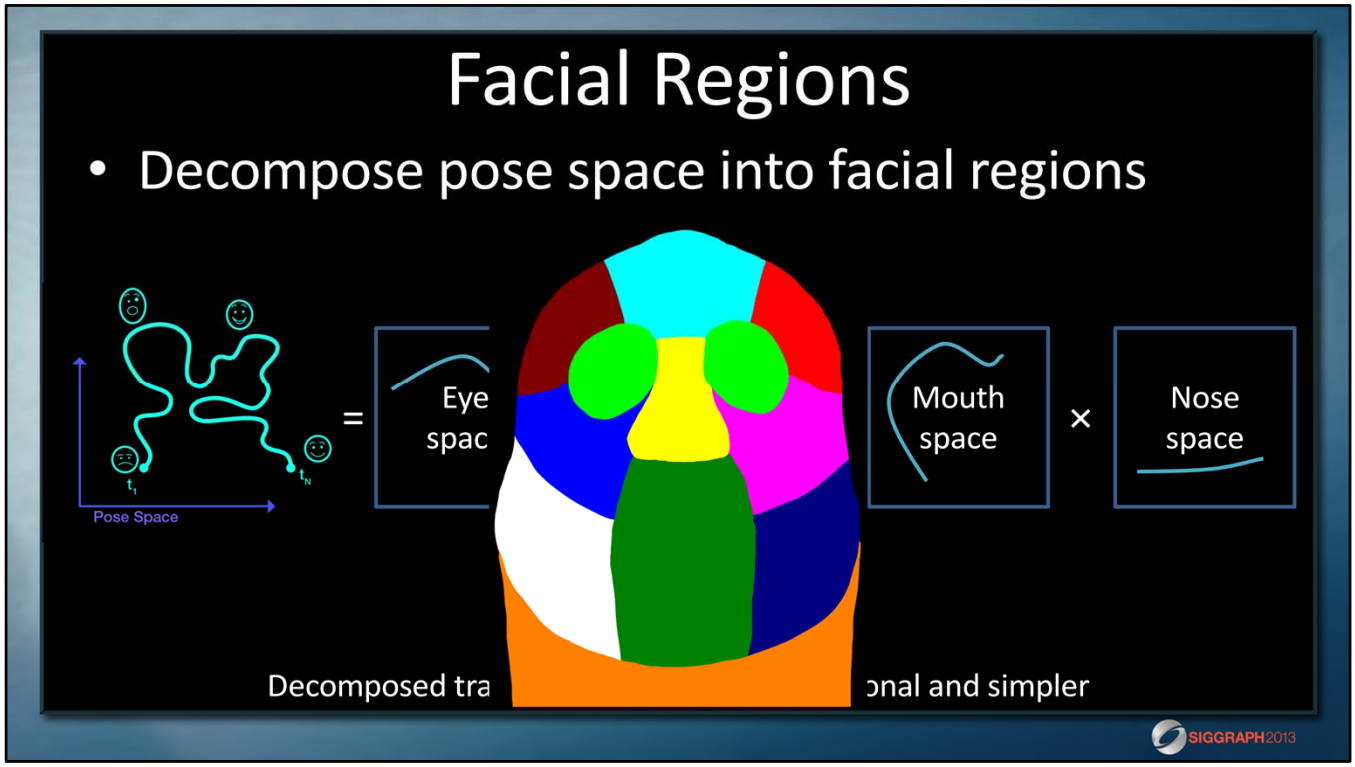


THIS SHOULD BE AT ABOUT 5:00 INTO THE PRESENTATION

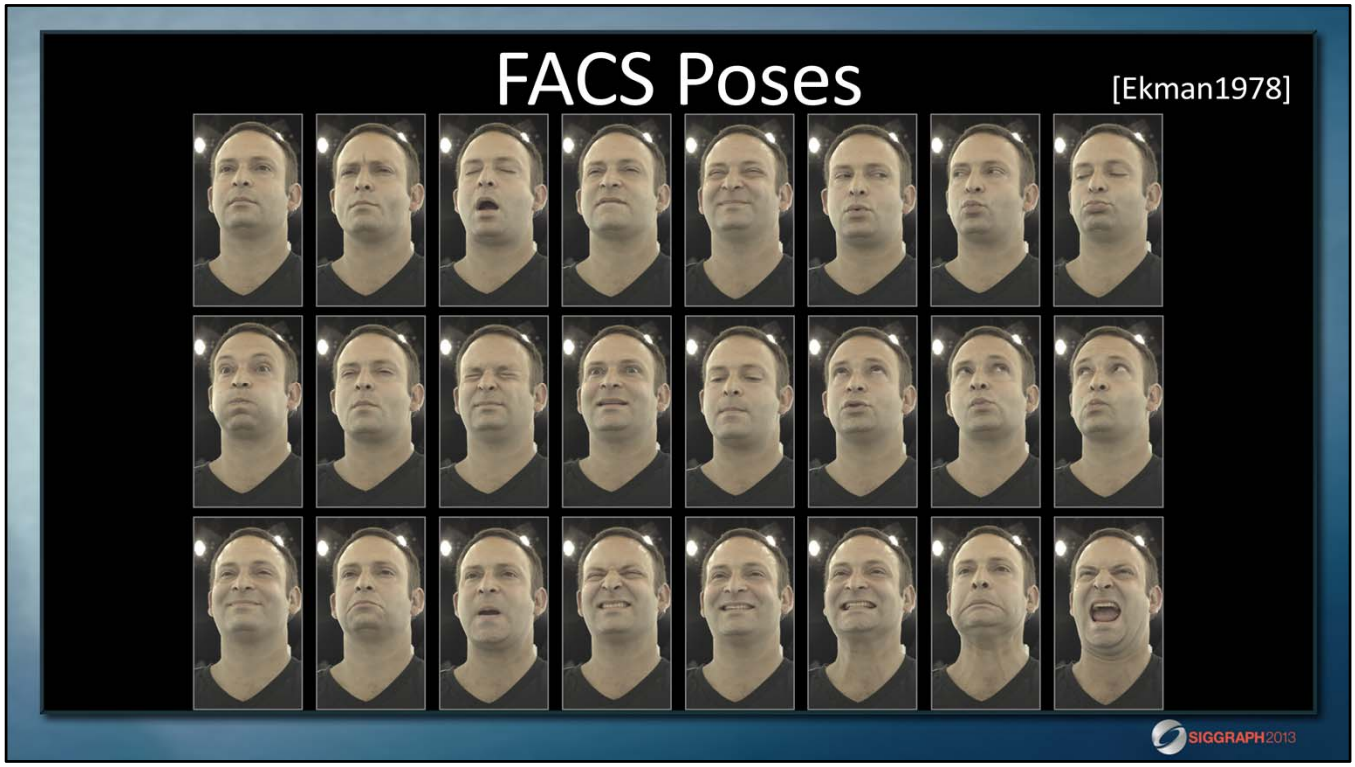
So far we are concerned about tracking from one pose to another in some order.

We *require* each tracking step to succeed for the *whole* face.

We cannot exploit obvious matches restricted to a smaller region of the face.



Of course in reality you have more than 4 regions, but this is for illustration.

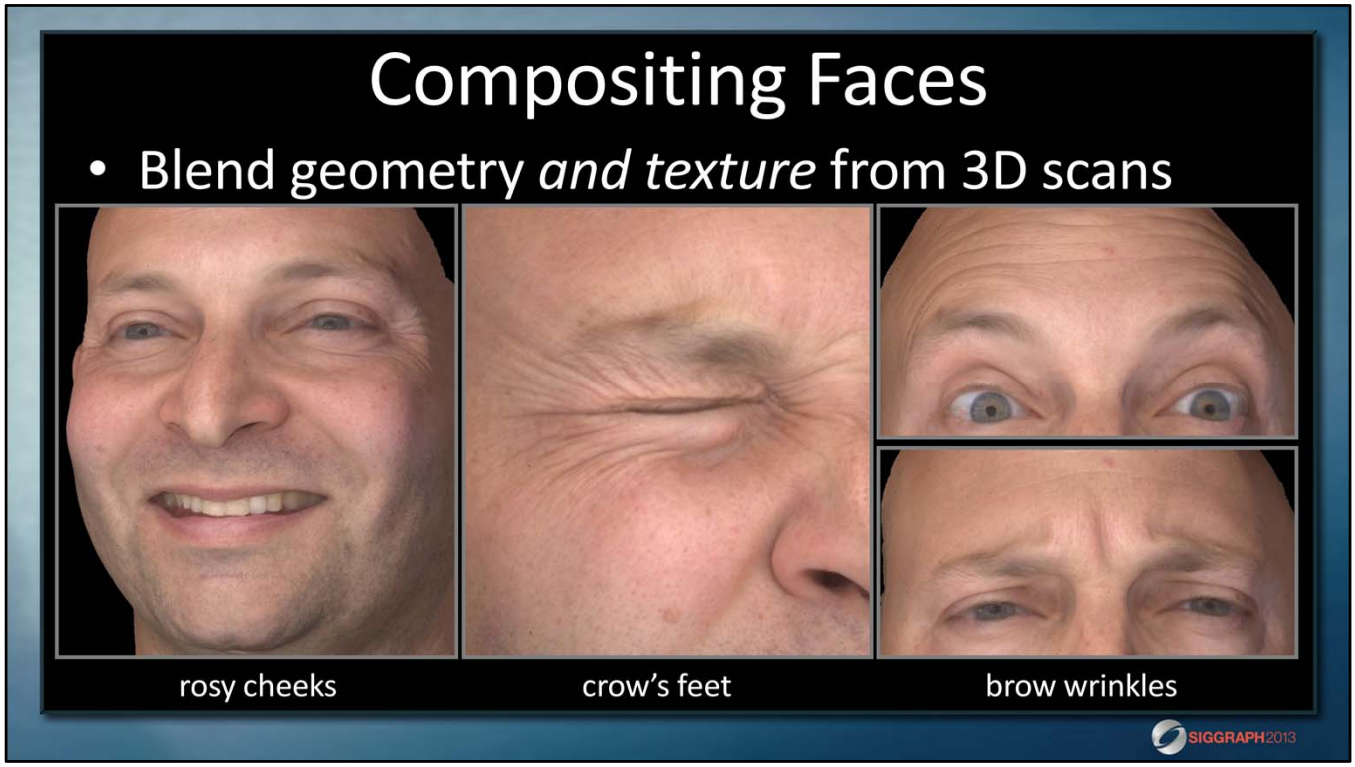


Spans the entire range of motion per facial region.

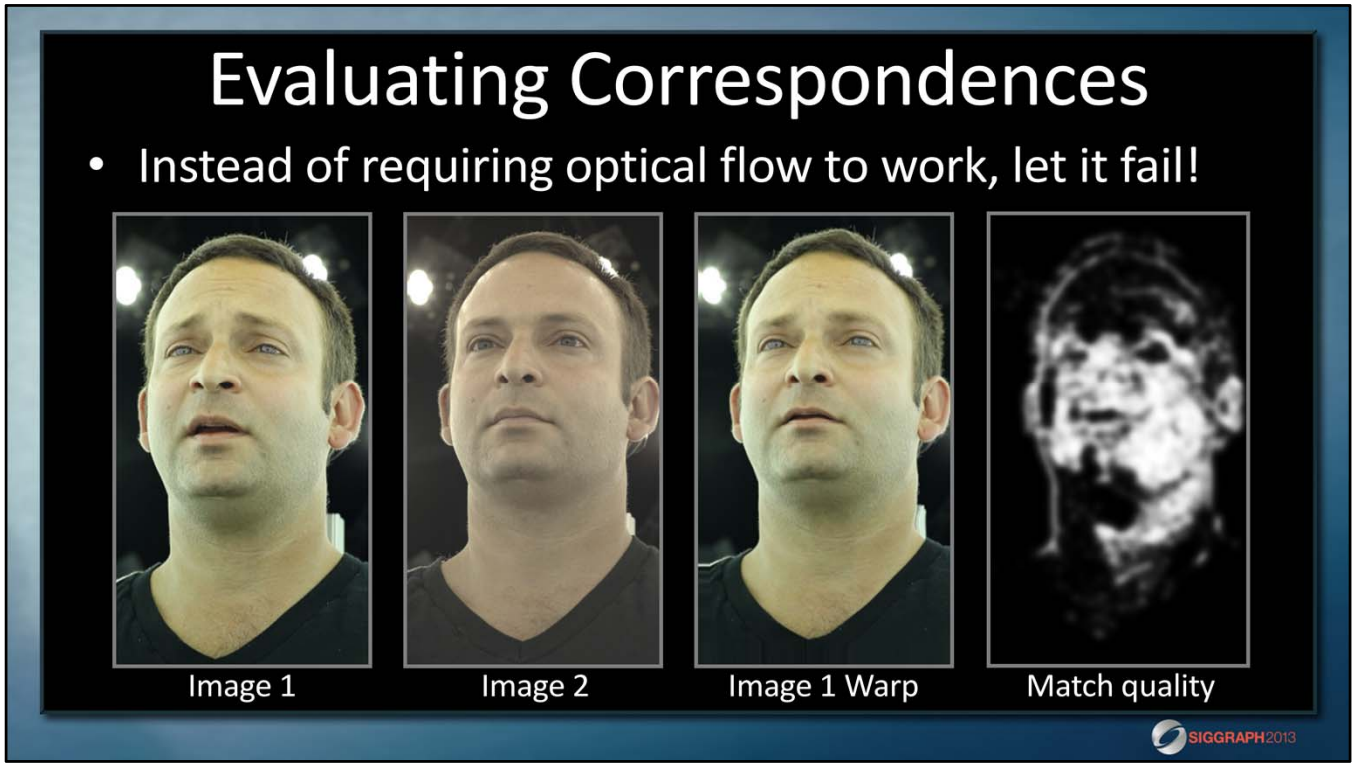
Paul Ekman and Wallace Friesen. Facial Action Coding System: A Technique for the Measurement of Facial Movement. Consulting Psychologists Press, Palo Alto, **1978**.

Compositing Faces

- Scan FACS poses in high resolution 3D

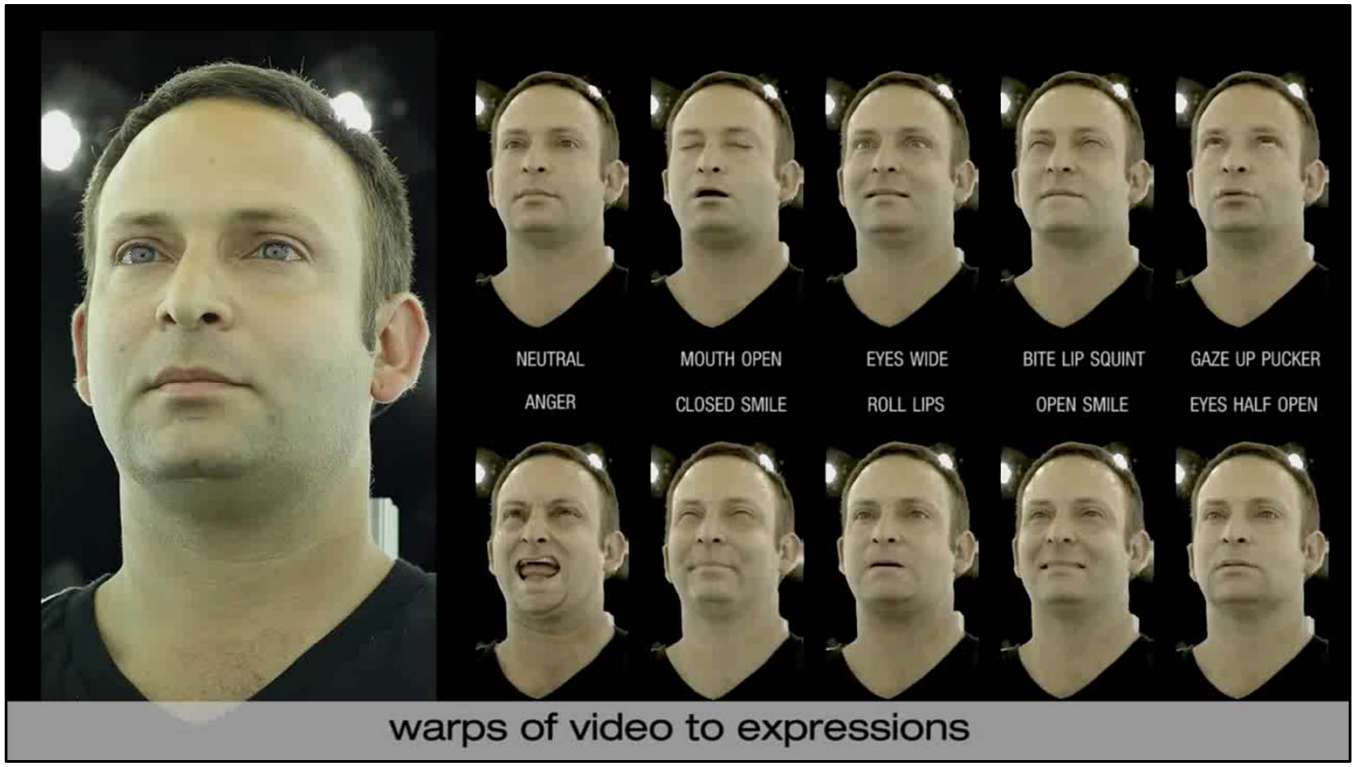


The nice thing about capturing blendshapes using scans is you get all the texture and color detail that changes when the face makes different expressions,
Such as bloodflow to the cheeks, and wrinkles around the eyes or the forehead.



We use low-res optical flow because we want sense correspondences, with a meaningful matching score.
White is better match, black means poor match.

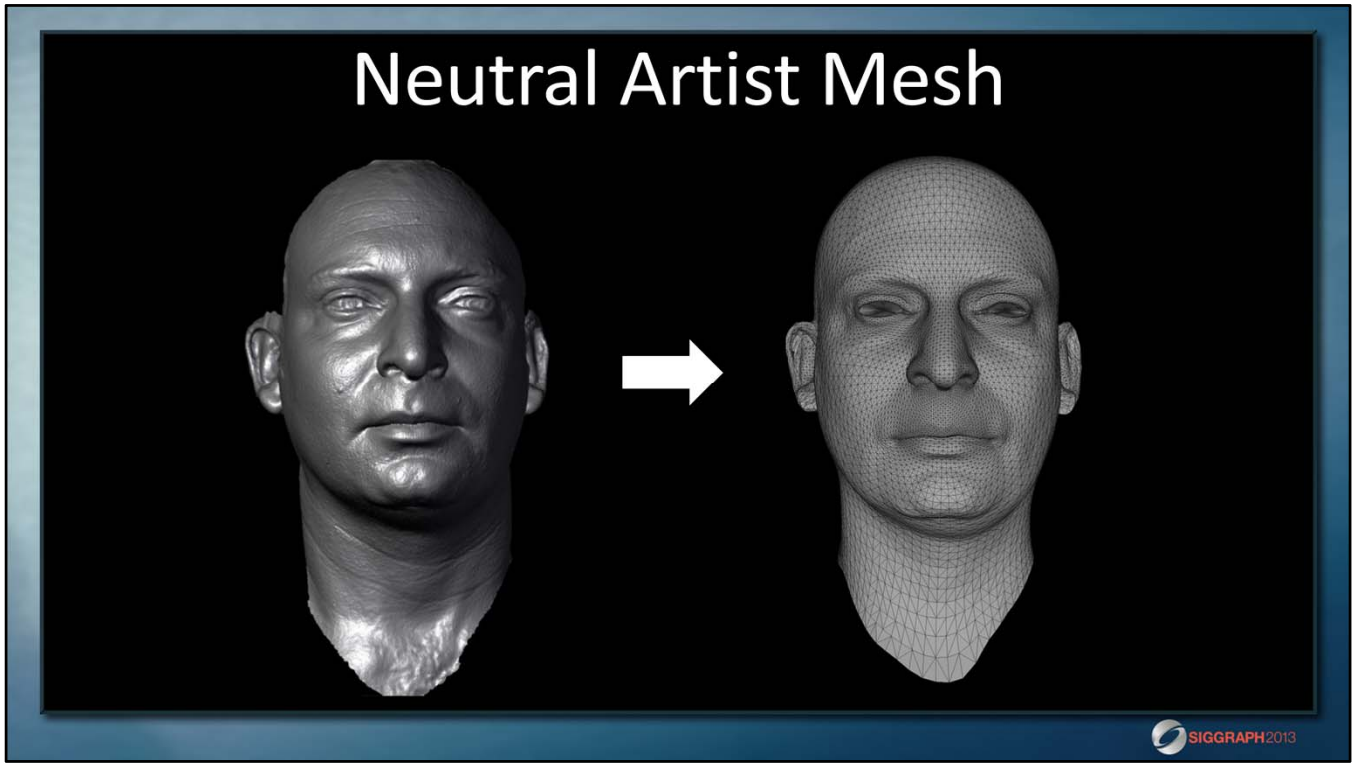
It's blurred because we want it that way?



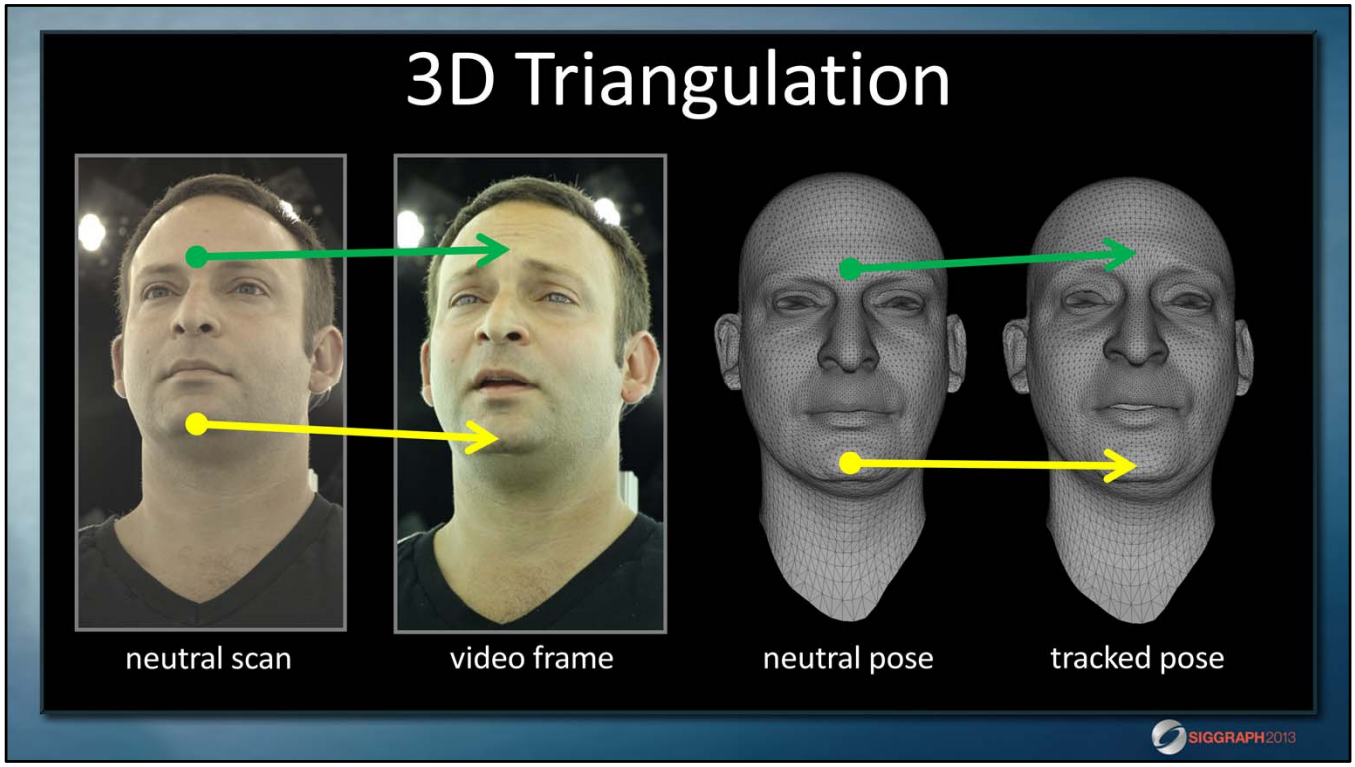
Bonus: We get per-region expression weights

Flow failures are temporally inconsistent, but that's okay because we're going to blend everything later.

Low-res flows are cheap using GPU.
Optical flow is becoming faster and more of a plug-and-play technology to putting many instances of optical flow inside a facial performance solver (at least an offline one) is reasonable.



Show the topology here



2D image correspondences imply 3D view ray correspondences

Laplacian regularization using the artist mesh and the other scans, to guarantee a full head mesh for all frames, including inner mouth and back of head

Note there is no temporal smoothing, as the tracking takes care of that.

Expression Blending Weights

SIGGRAPH2013

The real-time rendering, eye shading, and skin shading are done by Activision R&D.

Links to youtube channel and Jorge's blog

The real-time rendering, eye shading, and skin shading are done by Activision R&D.

Links to youtube channel and Jorge’s blog

The real-time rendering, eye shading, and skin shading are done by Activision R&D.

Links to youtube channel and Jorge’s blog

The real-time rendering, eye shading, and skin shading are done by Activision R&D.

Links to youtube channel and Jorge’s blog

Acknowledgements

- Ari Shapiro
- Activision R&D
- Bill Swartout
- Randall Hill
- Randolph Hall
- Koki Nagano
- Xueming Yu
- Paul Graham
- Jay Busch
- Borom Tunwattanapong
- U.S. Army Research, Development and Engineering Command (RDECOM)

SIGGRAPH2013

An Autostereoscopic Projector Array Optimized for 3D Facial Display

Koki Nagano Andrew Jones Jing Liu Jay Busch Xueming Yu Mark Bolas Paul Debevec
USC Institute for Creative Technologies

Figure 1: *Our projector array (left) generates autostereoscopic views of a virtual character (middle) and 4D facial performance capture (right)*

Introduction Video projectors are rapidly shrinking in size, power consumption, and cost. Such projectors provide unprecedented flexibility to stack, arrange, and aim pixels without the need for moving parts. We present a dense projector display that is optimized in size and resolution to display an autostereoscopic life-sized 3D human face with a wide 110 degree field of view. Applications include 3D teleconferencing and fully synthetic characters for education and interactive entertainment.

Related Work The commercial company Holografika has demonstrated various large-format screens up to $3m$ across with up to 80 large-format projectors [Balogh et al. 2007] but does not publish their projector specifications, diffusion materials, calibration process, or rendering algorithms, and has not specifically demonstrated the concentrated spatial and angular resolution for convincing autostereoscopic display of an interactive face. Our display is reproducible with off-the-shelf components, and a central goal of our E-Tech exhibit will be to show how such a display can be constructed easily and run from a single computer to encourage research in the field.

Projector Array Design Our display utilizes 72 Texas Instruments DLP Pico Projector Development Kit v2.0 devices to illuminate a $30cm \times 30cm$ anisotropic screen, and we removed the cases and built custom mounts to place the projectors just 14mm apart. We use the horizontal cylindrical ridges of a plastic 40 lineper inch lenticular screen painted black on its back side to achieve the anisotropic reflection with a high contrast ratio in ambient light. The light from each projector lens reflects back as a vertical strip of light, so to blend the lines together we use a 1° horizontal by 60° vertical light-shaping diffuser sheet from Luminit, which also increases the vertical diffusion. We wrote a GPU simulator program to show the effect of different projector array shapes, amounts of diffusion, and projector densities, which led us to place the projectors in a $124cm$ curve with a radius of $60cm$ to maximize the depth of field. We also prototyped the display using a single pico-projector on a motion control arm and long-exposure photography [Jurik et al. 2011] to simulate the 3D effect with real-world equipment. Our setup provides a high angular resolution of 1.66° between views, achieving not only binocular stereo but also compelling motion parallax.

GPU rendering Notably, we drive our projector array using a single computer with twenty-four 1920×480 video outputs from four AMD FirePro W600 Eyefinity graphics cards. We then split each video signal using a Matrox TripleHeadToGo box into three 640x480 outputs, yielding the 72 projector signals. We adapted the multiperspective vertex shader technique of [Jones et al. 2007] to work with fixed projector arrays and different display surface shapes. For a given mirror shape and diffusion profile, we approximate the reflected projector positions and use them to warp vertex positions to generate multiple-center of projection images. We project a series of AR toolkit markers from each projector to calibrate the display geometrically and photometrically. As shown in the accompanying video, we can produce a stable image with correct perspective for either flat or curved display surfaces. We can also determine the ideal horizontal diffusion width by simulating different anisotropic reflectance lobes.

Vertical Parallax Our display achieves autostereoscopic horizontal parallax without lag. For faces which can make eye contact, vertical parallax is also important. Unlike other systems, we render accurate vertical parallax by detecting the viewer head positions using a Microsoft Kinect. Given the height and distance of the multiple viewers, we warp the multiperspective rendering according to who will see each column of projected pixels. An example of tracked vertical parallax rendering can be seen in the accompanying video.

Future Work Our projector array construction and calibration approach admits alternate setups, including rear-projection, and full-body projection using higher-resolution projectors placed with greater density further away from a larger screen.

References

BALOGH, T., KOVÁCS, P., AND MEGYESI, Z. 2007. Holovizio 3d display system. In *Proceedings of the First International Conference on Immersive Telecommunications*, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 19.

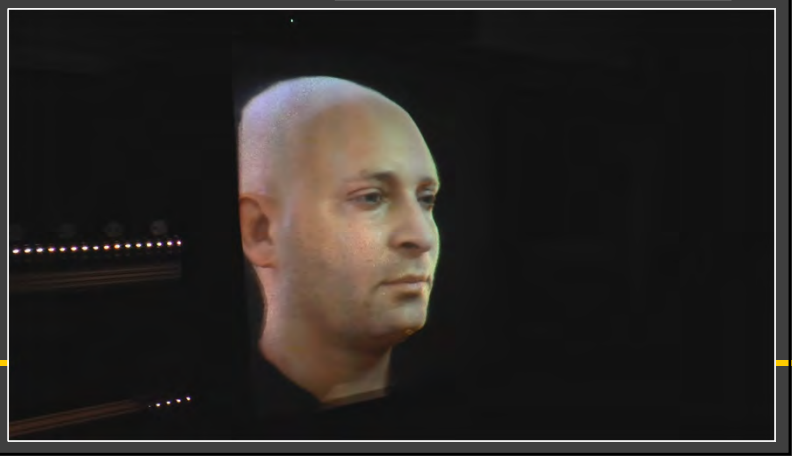
JONES, A., MCDOWALL, I., YAMADA, H., BOLAS, M., AND DEBEVEC, P. 2007. Rendering for an interactive 360 light field display. *ACM Transactions on Graphics (TOG) 26*, 3, 40.

JONES, A., LANG, M., FYFFE, G., YU, X., BUSCH, J., MCDOWALL, I., BOLAS, M., AND DEBEVEC, P. 2009. Achieving eye contact in a one-to-many 3d video teleconferencing system. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 64.

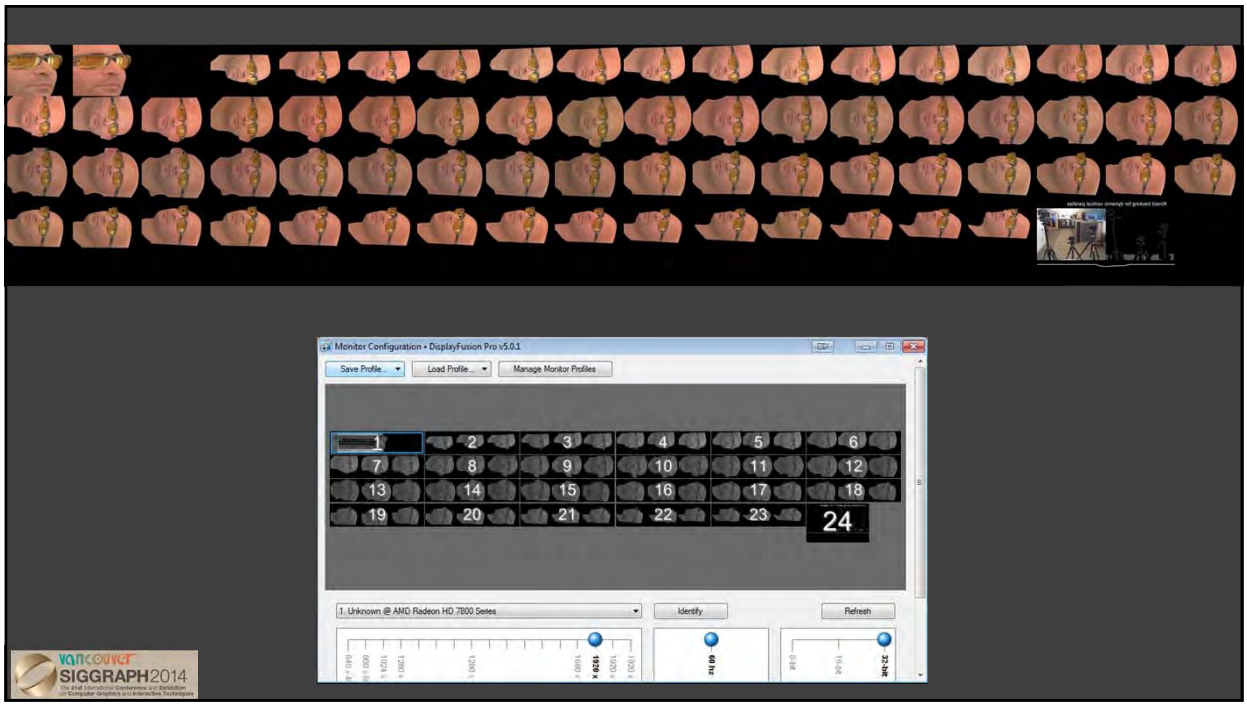
JURIK, J., JONES, A., BOLAS, M., AND DEBEVEC, P. 2011. Prototyping a light field display involving direct observation of a video projector array. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, IEEE, 15–20.

Digital Ira at SIGGRAPH 2013:
Pico-Projector 3D Display

- Collaboration with **Mark Bolas of MxR Lab**
- First autostereoscopic facial display shown at **SIGGRAPH Emerging Technologies**
- **72 projectors** in a mobile cart platform
- Driven by a **single computer** with 4 graphics cards
- Special version of **Digital Ira** created by Activision to run in real time in autostereo 3D on the pico-array
- Demoed also at **Director's Guild of America** "Digital Day"

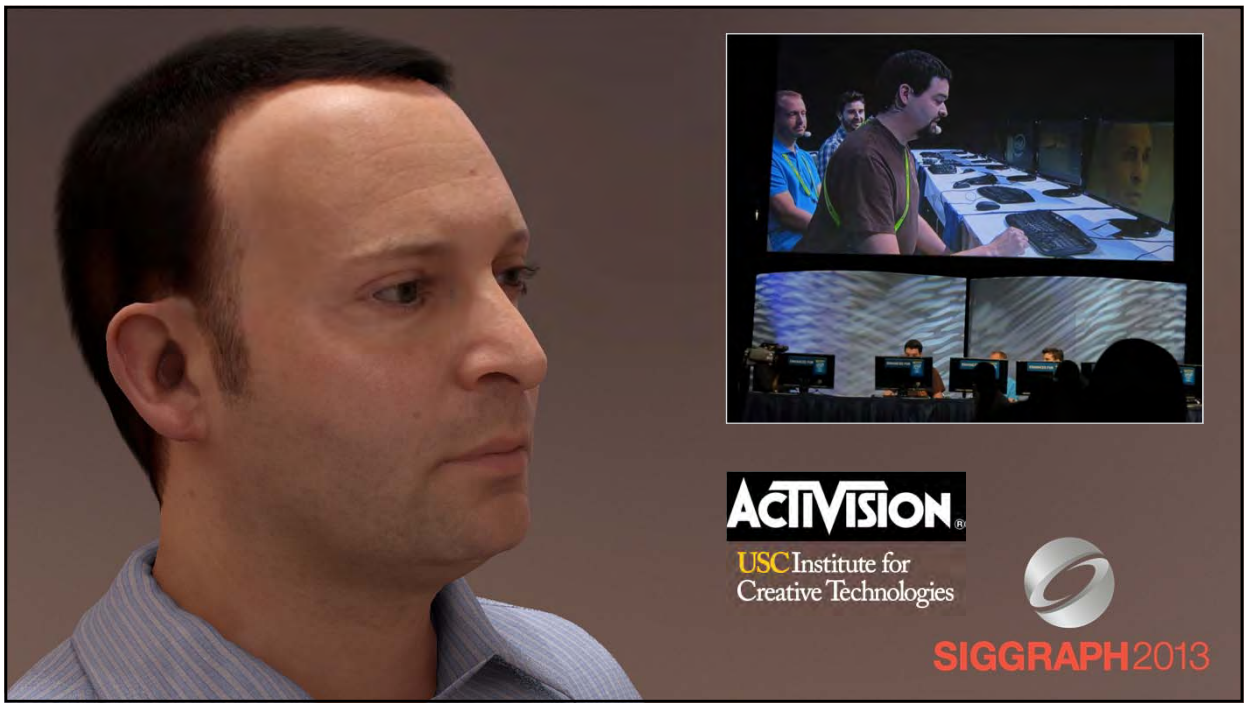


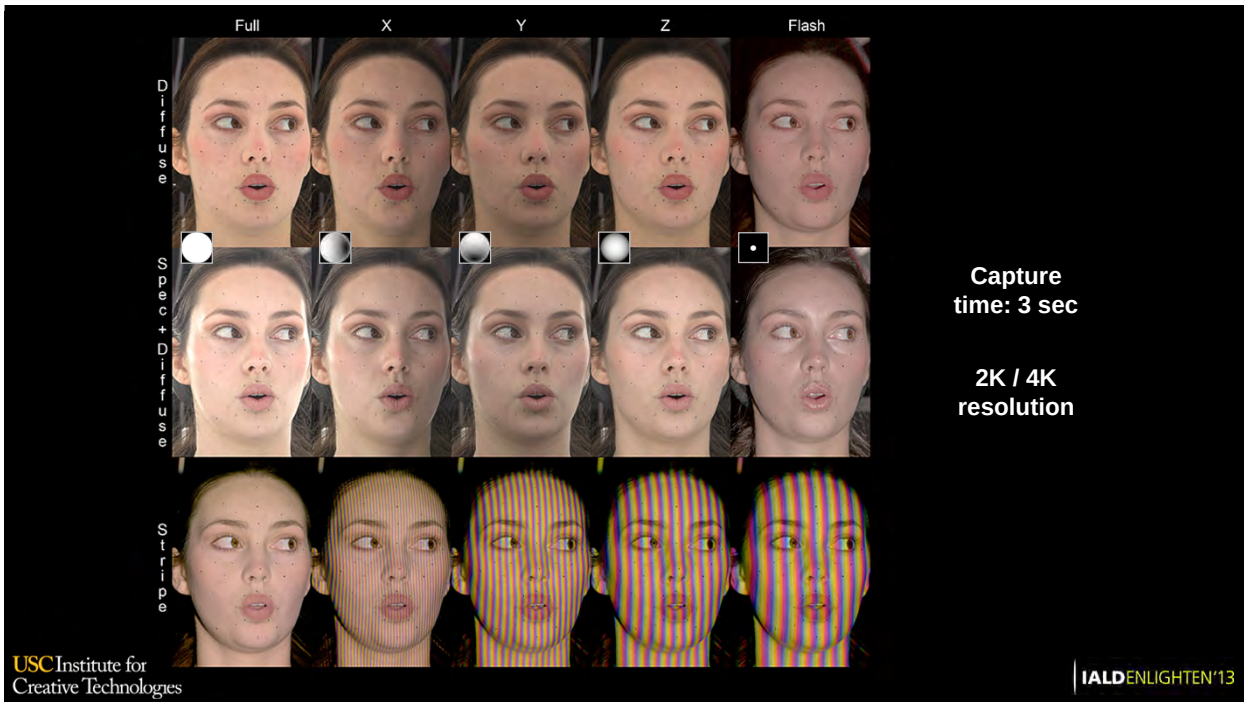
vancouver SIGGRAPH2014
The 41st International Conference and Exhibition on Computer Graphics and Interactive Techniques

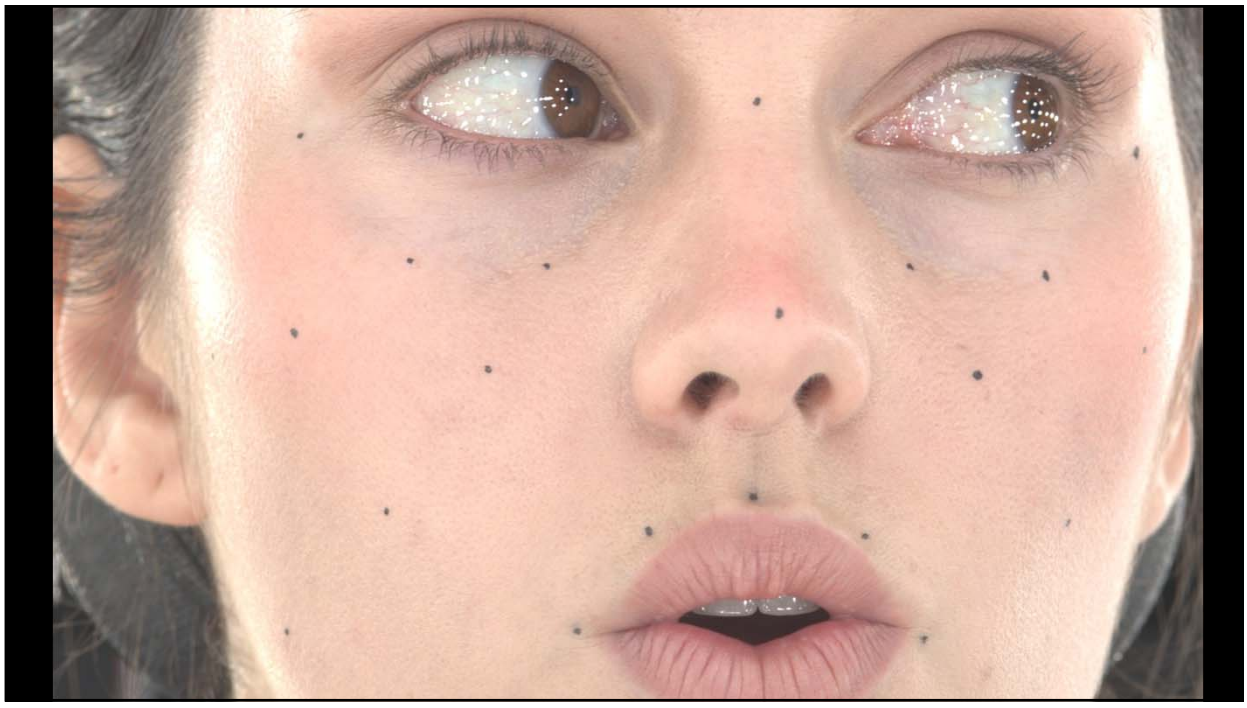

Monitor Configuration • DisplayFusion Pro v5.0.1
Save Profile...
Load Profile...
Manage Monitor Profiles
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
1. Unknown @ AMD Radeon HD 7800 Series
Identify
Refresh
1920 x
60 Hz
32-bit
VANCOUVER
SIGGRAPH2014

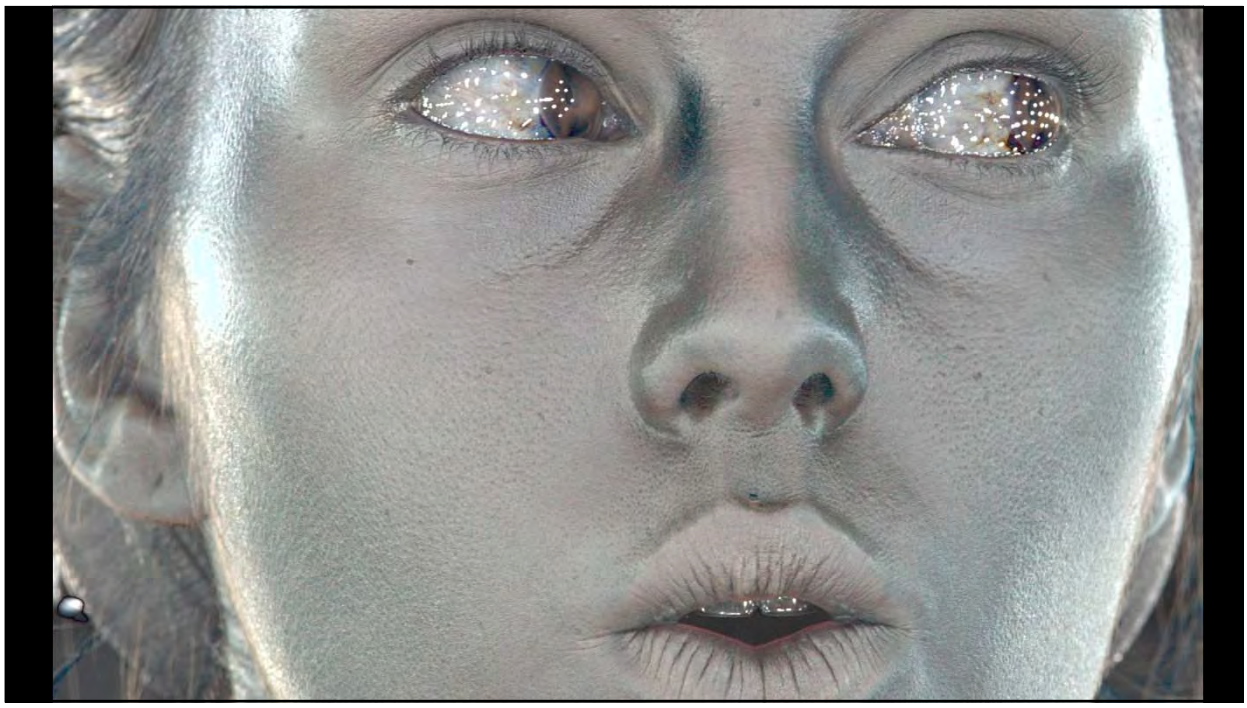
Digital Ira
Creating a Real-Time Photoreal Digital Actor

Activision, NVIDIA, and USC Insitute for Creative Technologies




Full
X
Y
Z
Flash
Diffuse
Spec+Diffuse
Stripe
Capture time: 3 sec
2K / 4K resolution
USC Institute for Creative Technologies
IALDENLIGHTEN'13





THE DIGITAL EMILY PROJECT

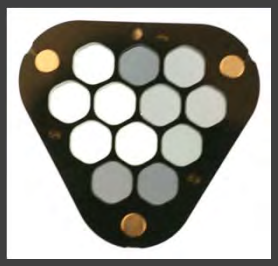
achieving a photoreal digital actor

A COLLABORATION BETWEEN:

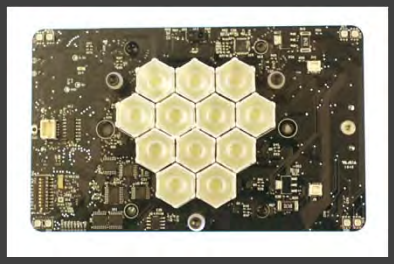
IMAGE METRICS

USC INSTITUTE FOR CREATIVE TECHNOLOGIES

Light Stage Acquisition



Polarization filter



Next-Generation Character Rendering

Jorge Jimenez
Technical Director
Activision Blizzard

Javier von der Pahlen
Director of R&D
Activision Blizzard

Hi,

This presentation is all about the details, and how a collection of very small things can be crucial for bringing characters to life.

Something I learnt these past years working with faces is that each character rendering feature counts,
even if it doesn’t seem to make sense when used individually.

Disclaimer

- Projectors don't do justice to the details of our renderings
- Similarly, slides are best viewed fullscreen, in order to appreciate all the details and subtleties
- This presentation reflects R&D that ATVI is exploring
- This content is for presentation purposes only and does not reflect actual design examples nor actual game assets

ACTIVISION | BLIZZARD

Jorge Jimenez

Graphics R&D

Javier von der Pahlen

Director of R&D

Etienne Danvoye

Technical Director

Bernardo Antoniazzi

Technical Art Director

Zbyněk Kysela

Modeler and Texture Artist

Mike Eheler

Programming & Support

ACTIVISION | BLIZZARD

Here you have the full team, with Javier Pahlen leading the research.

Render

ACTIVISION BLIZZARD

The presentation is composed of two parts: rendering and animation.

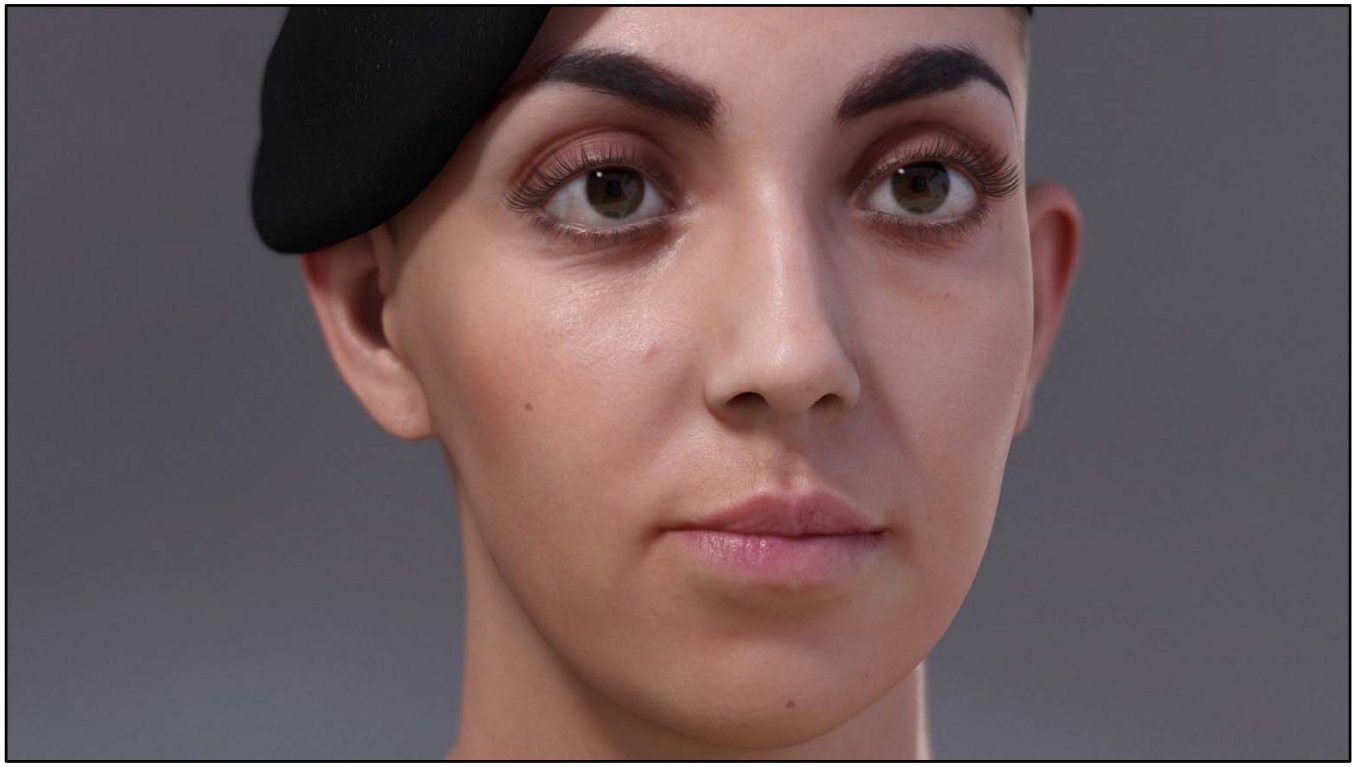
I'll present the rendering part first, and Javier will continue with the animation/compression part.

Our Previous Work

- **[Jimenez2009] *Screen-Space Perceptual Rendering of Human Skin***
  - **http://www.iryoku.com/sssss/**
- **[Jimenez2010] *Real-Time Realistic Skin Translucency***
  - **http://www.iryoku.com/translucency/**
- **[Jimenez2011] *Enhanced Subpixel Morphological Antialiasing***
  - **http://www.iryoku.com/smaa/**
- **[Jimenez2012] *Separable Subsurface Scattering and Photorealistic Eyes Rendering*:**
  - **http://advances.realtimerendering.com/s2012/index.html**

We tried to keep this as self-contained as possible, but given the time limit of talk, and the fact that this research is the result of many years of investigation, we will have to refer to our previous work for more details.

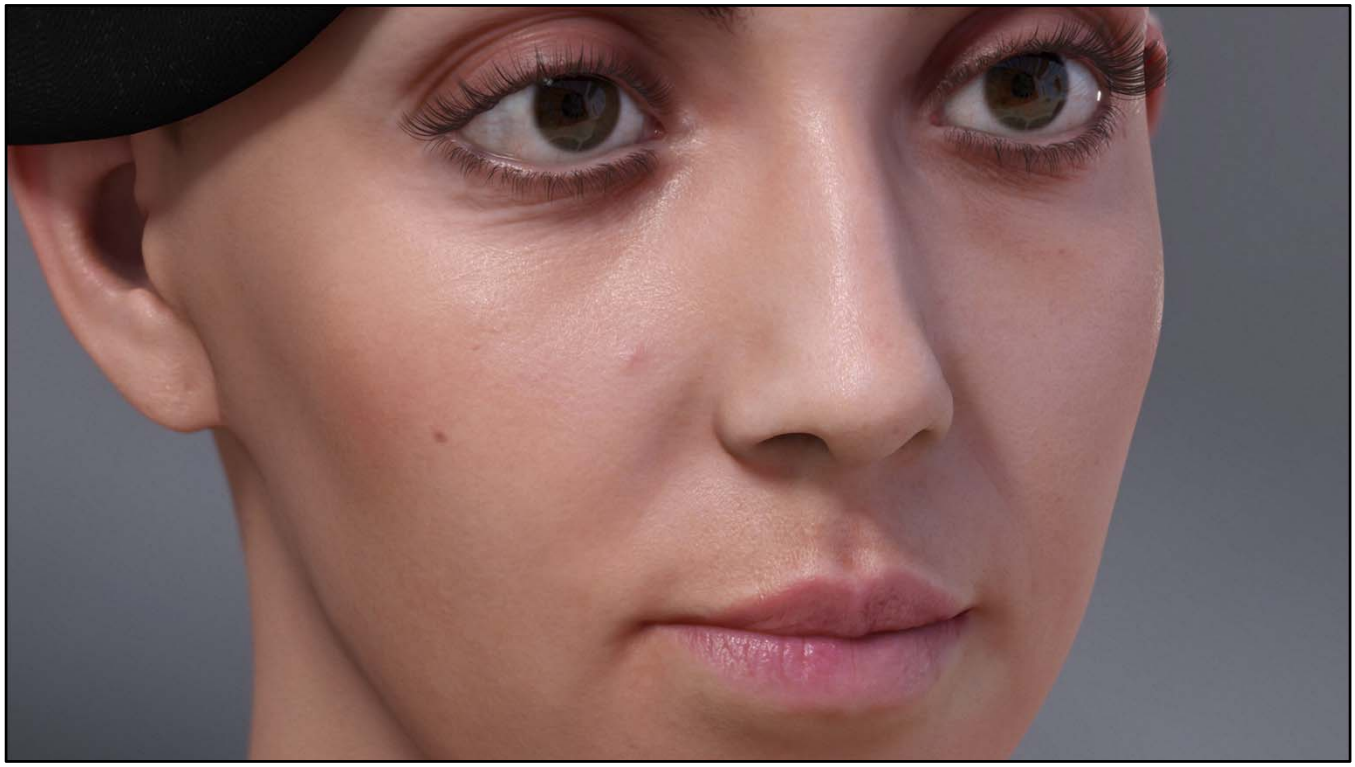
(For people reading this, if you feel lost at some point, just going through these references in order should help)



So, let me present Lauren, one of our test characters, running at 180 frames per second. (On a GeForce GTX 680 at 720p, using SMAA T2x.)

There are many things important for character rendering.

For example, the modeling, the shadows, the eyelashes…



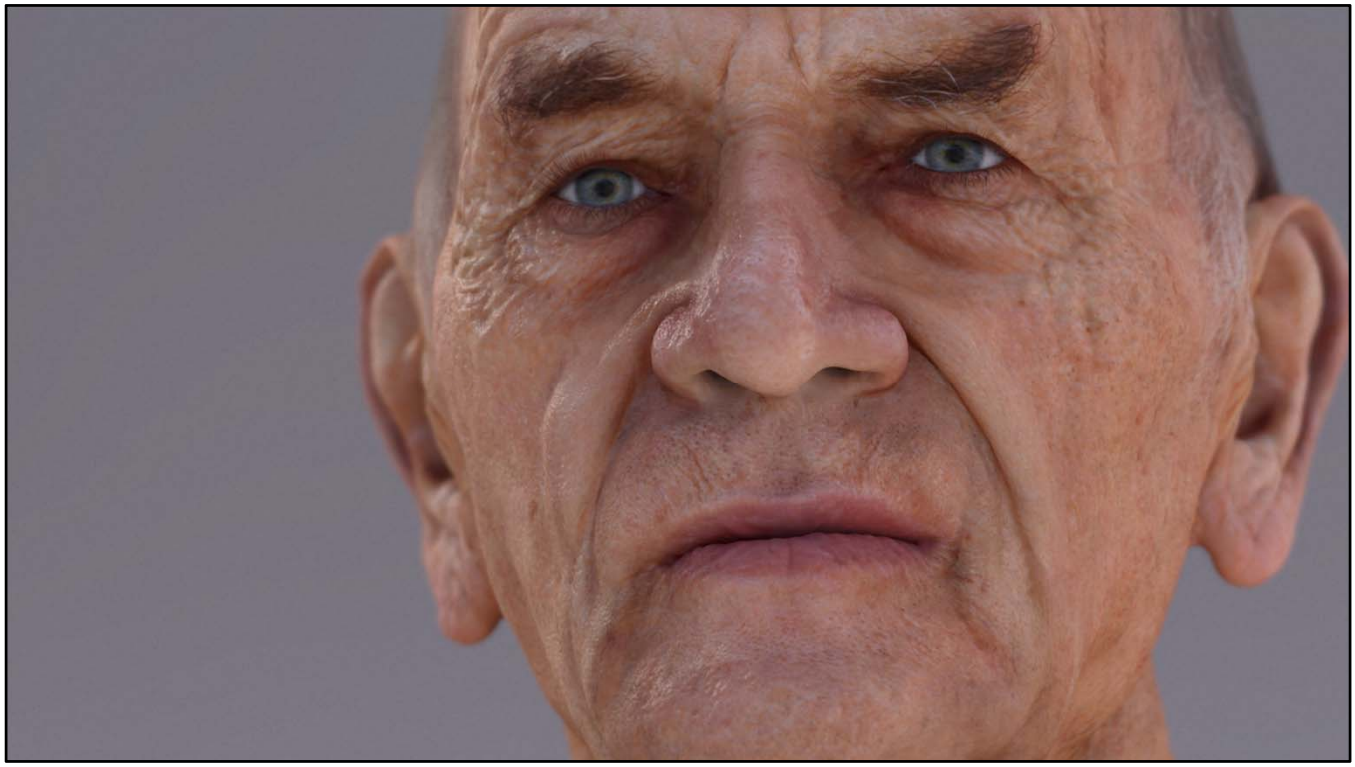
…the specular reflections, the occlusion, the skin, post fx and of course…

…the eyes.

Described as “windows to the soul”, they are, in my opinion, of extreme importance for showing emotions.

Notice that, to a certain degree, we manage to maintain realism even in tight macro shots, with very fine skin details.

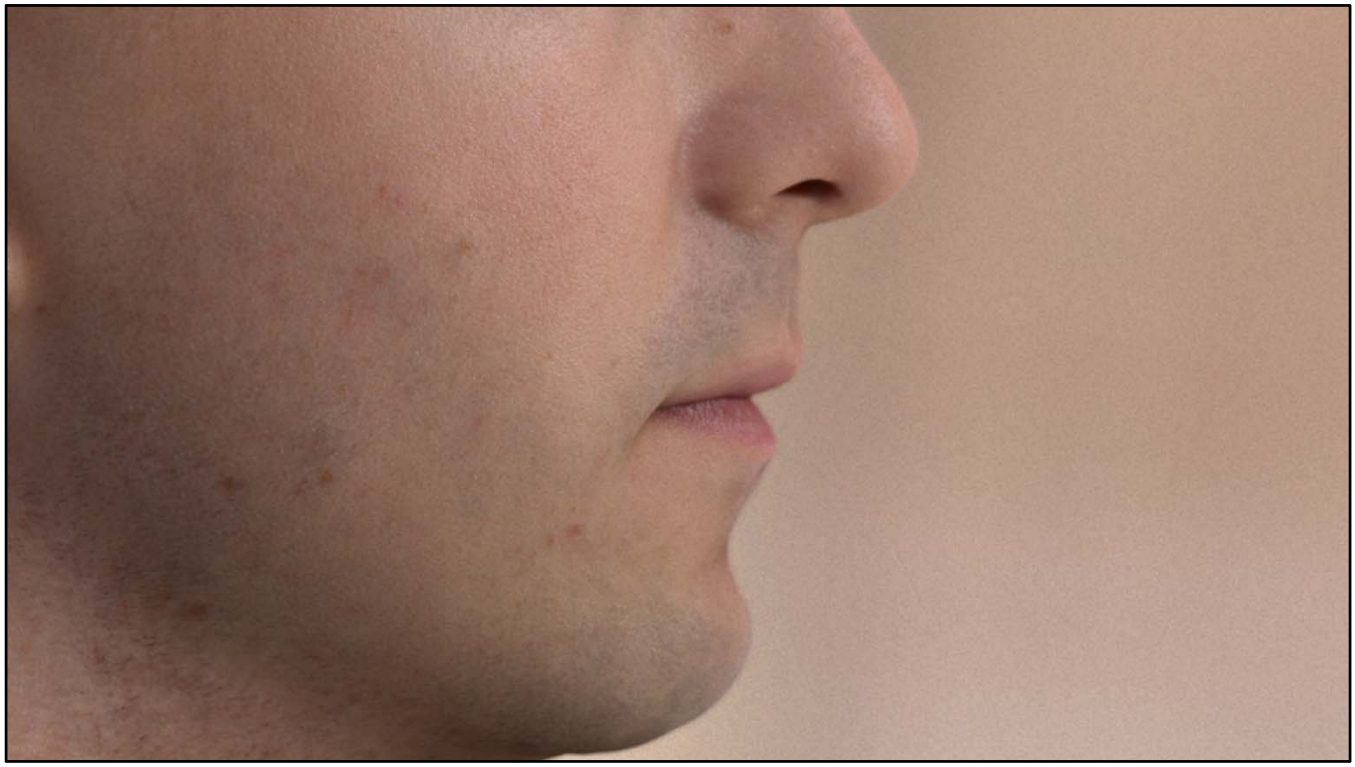
Here you can see the light transport in the eyes, including light refraction,
and the smooth colored shadows on the top of the eye, produced by subsurface scattering.



In this one, you can see the power the eyes can have for showing emotion,
the soft specular reflection of old skin, and the importance of depth of field in portrait shots.

This shows how soft lighting can further reduce the specularity of a face…

…and how important the eyes can be in a composition.



See the soft diffuse lighting, together with strong specular bumps,
and how film grain can add realism to a character shot.

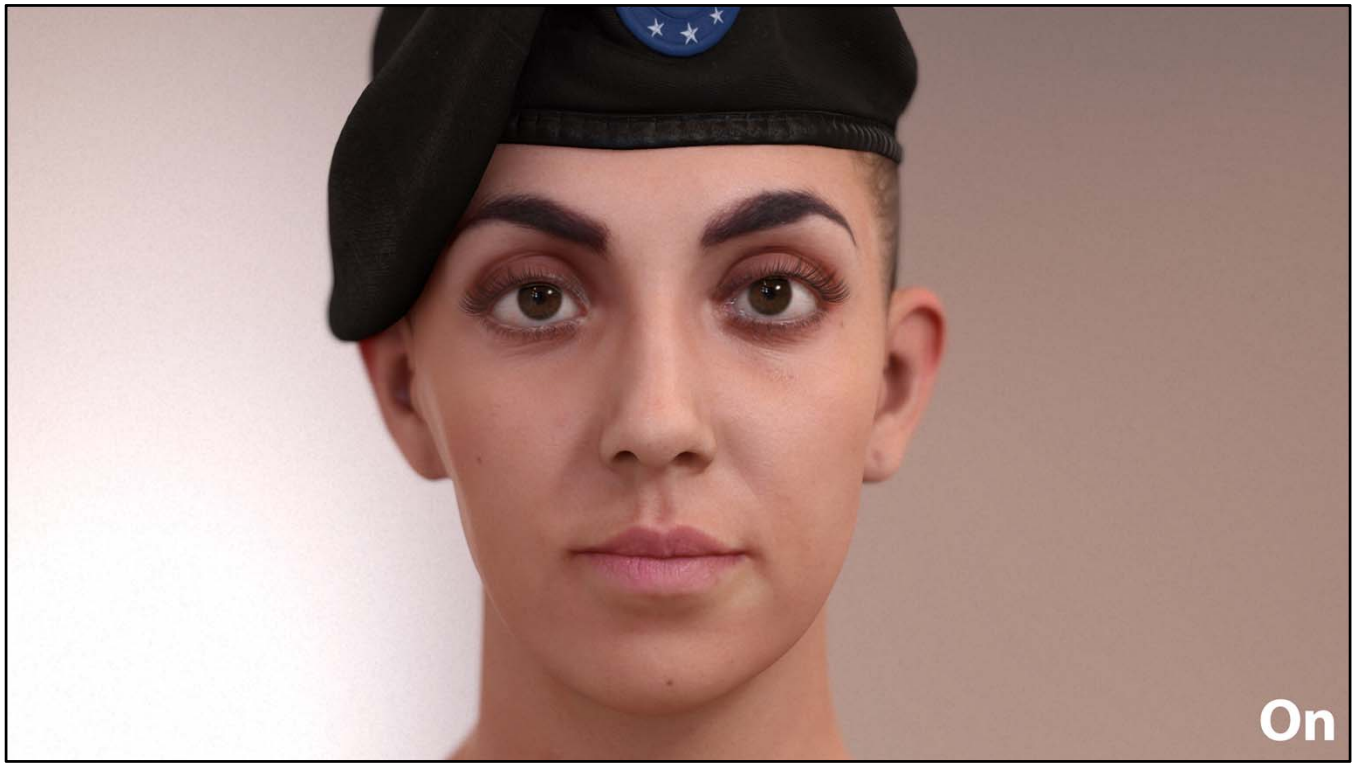
There is much to say about this one.

Notice the strong features of the skin, and how smooth it looks at the same time.

Achieving this, is for me, one of the biggest challenges we faced.

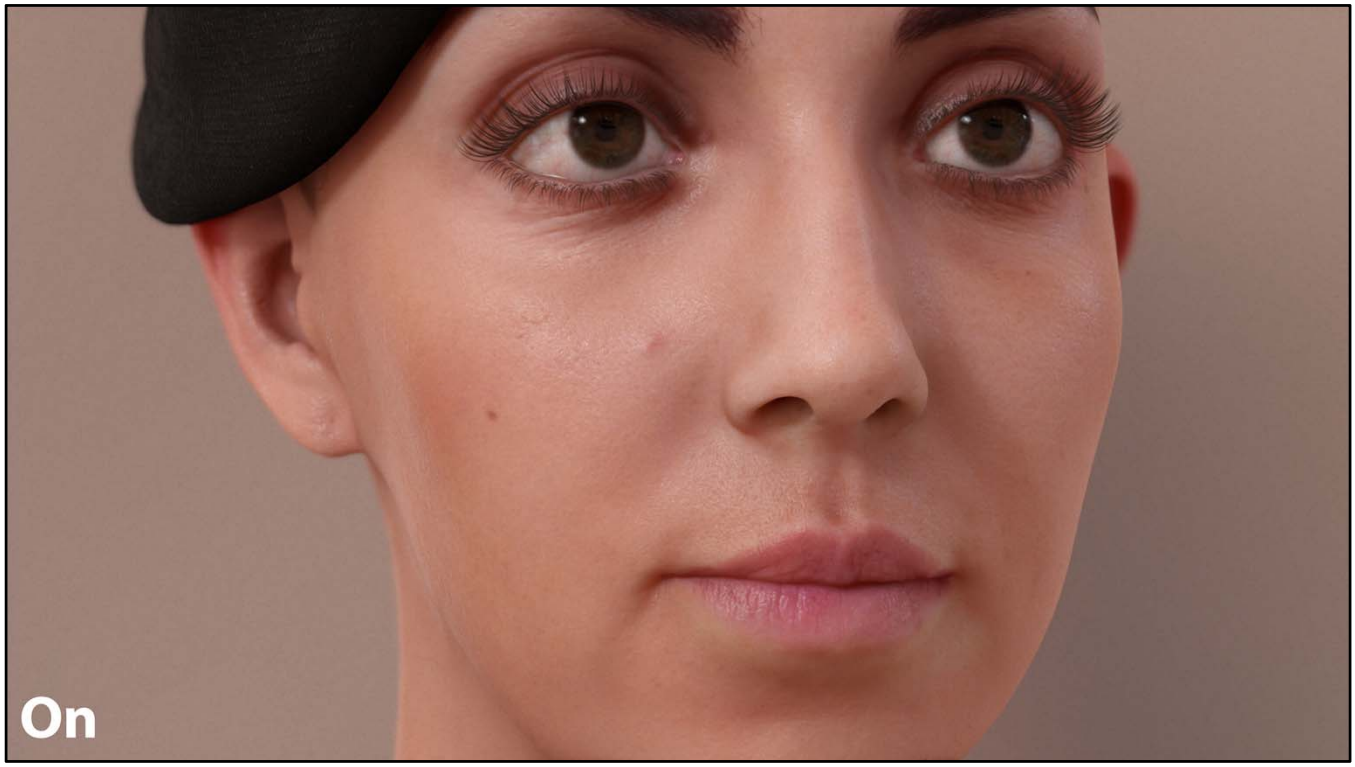
Also look at the reflections in the eyes, and how they reveal how wet they are,
which can help to convey a character's emotional state.

And last, but not least, notice how the light is travelling through the nose of the character, how light is refracting in the eyes, and the subtle, yet noticeable pore detail in the face.



I’m going to show you some images, turning on and off everything we do that's special for character rendering.


Off

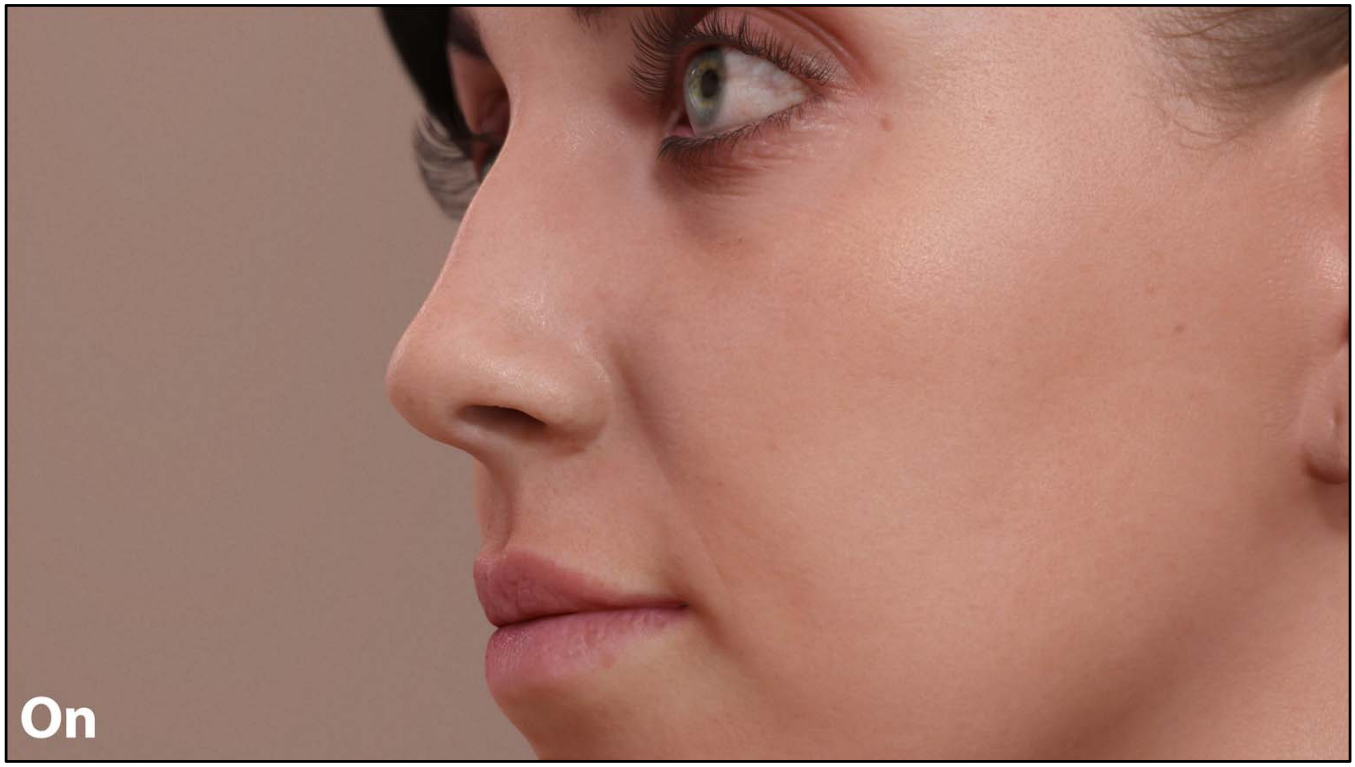


Notice how the character turns into…

…a doll when we turn off our special rendering features.

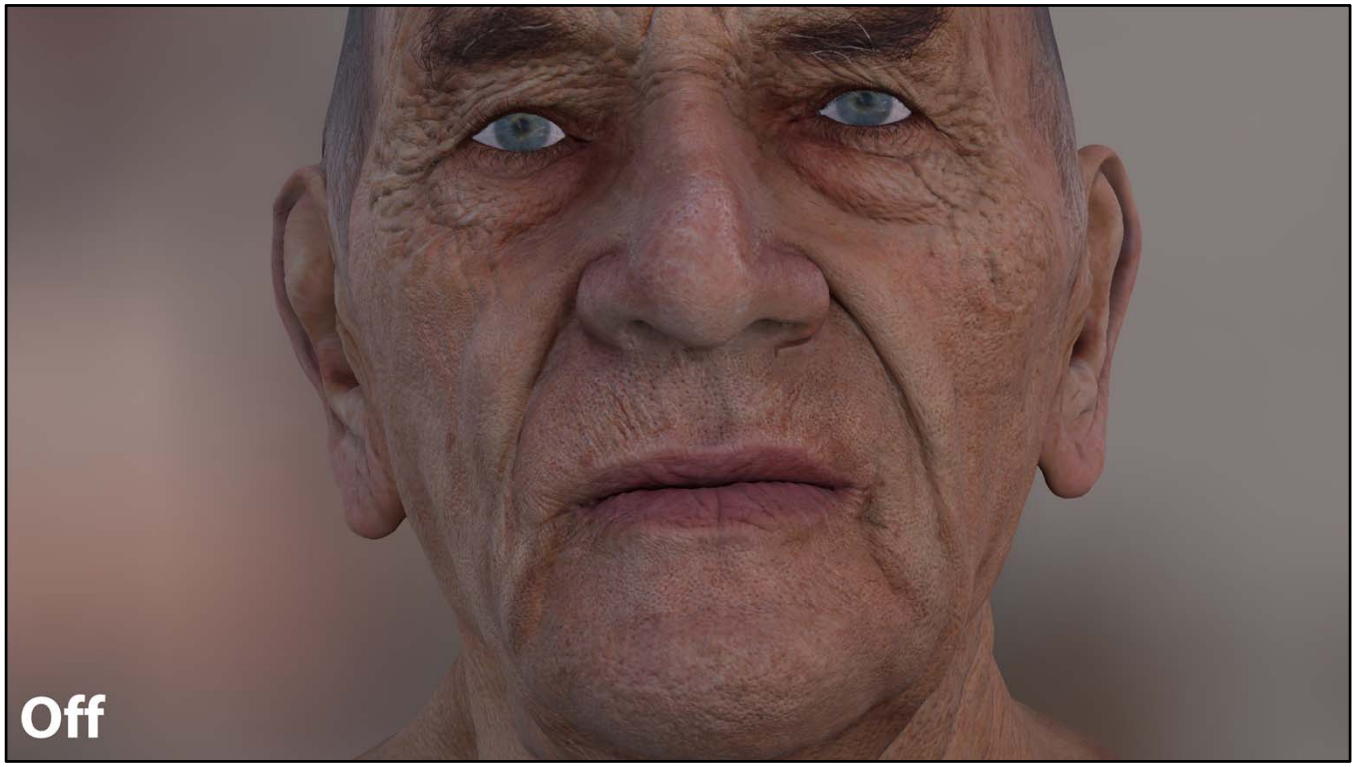
They are especially important…

…the closer you are to the character.

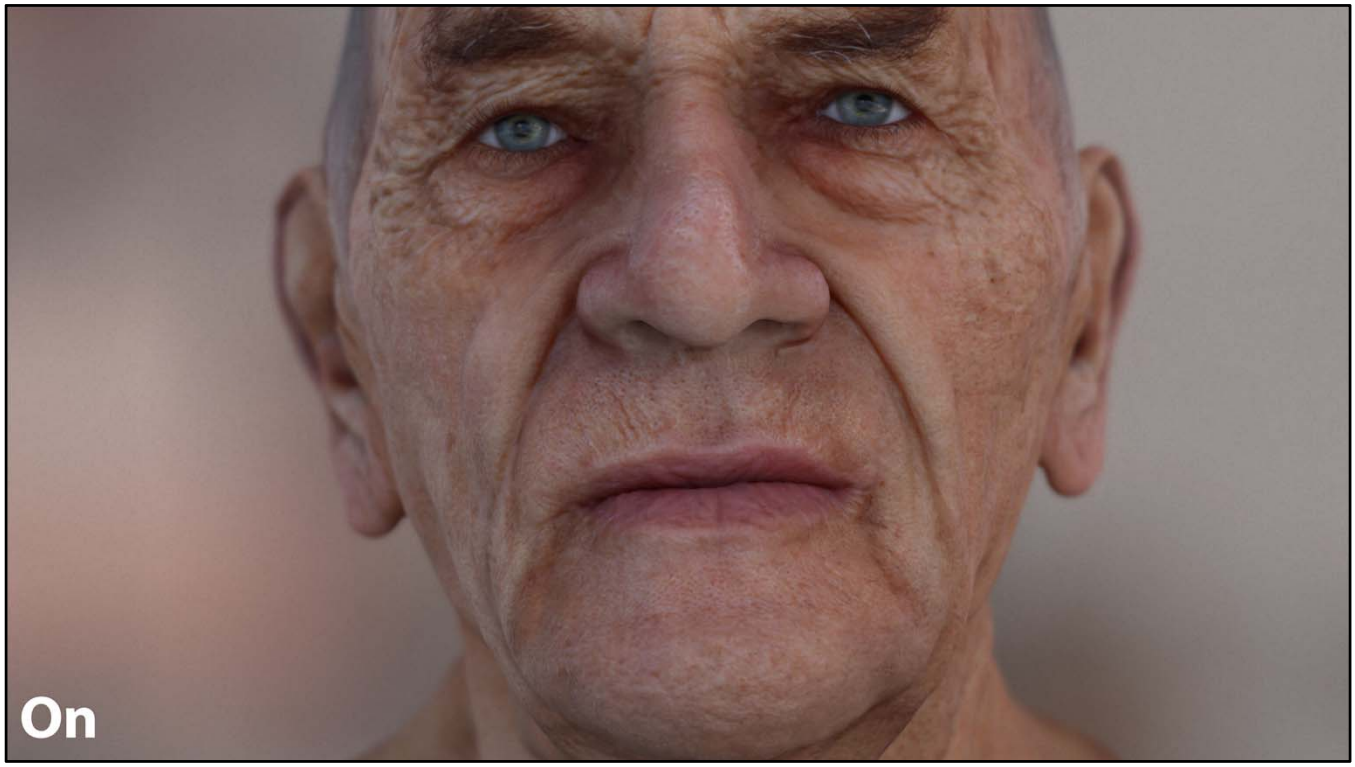


Look at the refraction in the eyes…

…and the specular details.



In my opinion, this technology has the potential to take a current generation character…



…to the next generation life.

What's going on?

- In the real world, nothing is perfect
- Too perfect renderings look unrealistic
- Fastest route to the uncanny valley

ACTIVISION | BLIZZARD

Art Surpassing Tech

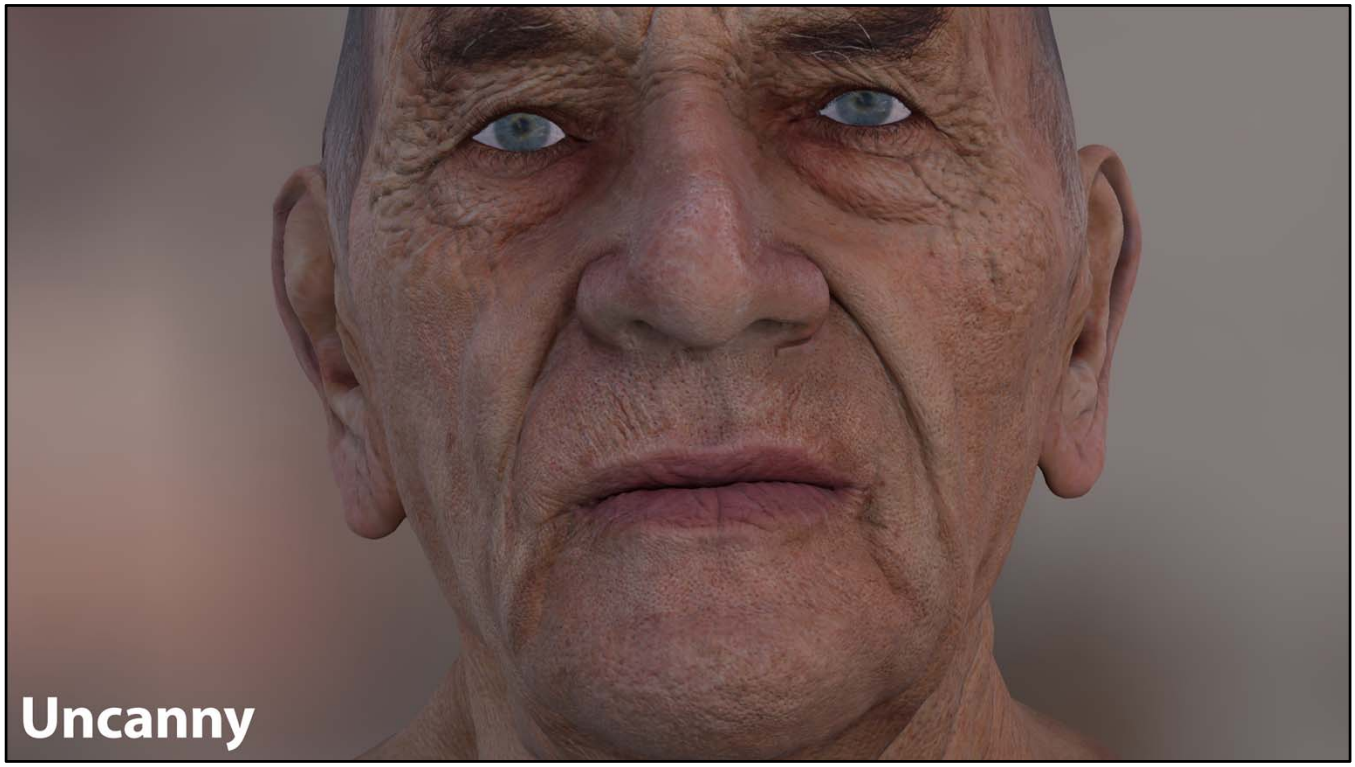
- **Ultra Detailed Meshes +**
- **Naive Technology =**
- **Uncanny Valley**

ACTIVISION | BLIZZARD

The level of detail of characters has steadily increased in the past few years.

There are now many games with incredibly beautiful and detailed faces.

Unfortunately, if we don’t raise the technology to the same level, what you will get is again...



..this.

High level features of the renderer

- Linear lighting workflow
- Physically based lighting
- Special post-processing effects

- Numbers from physical quantities
- 1-5 procedural lights and 1 pre-filtered probe light (256x256)

ACTIVISION | BLIZZARD

The main goal of our renderer was correctness.

To achieve that, we're using a linear lighting workflow, physically based lighting, and a special post-effects pipeline.

We paid a lot of attention to the numbers used as parameters to the renderer – most of them being actual physical quantities.

We avoided using magic numbers as much as possible.

We used between one to five procedural lights and one pre-filtered probe light, but we found that a single procedural light (together with the probe light) is usually enough.

Skin Rendering

ACTIVISION | BLIZZARD

The render part of the presentation has three parts:

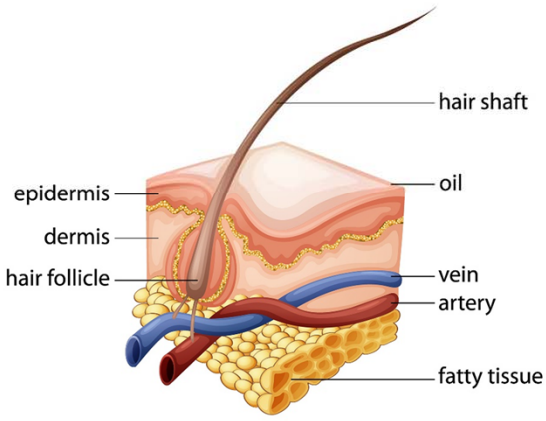
Skin, eyes and post effects.

So, let’s begin with the first part.

The skin.

The Skin

1. **Global Illumination**
2. **Specular Reflections**
3. **Subsurface Scattering (SSS)**



ACTIVISION | BLIZZARD

For us, skin rendering has three main components.

First: light bounces from some parts of the face onto others – i.e., global illumination.

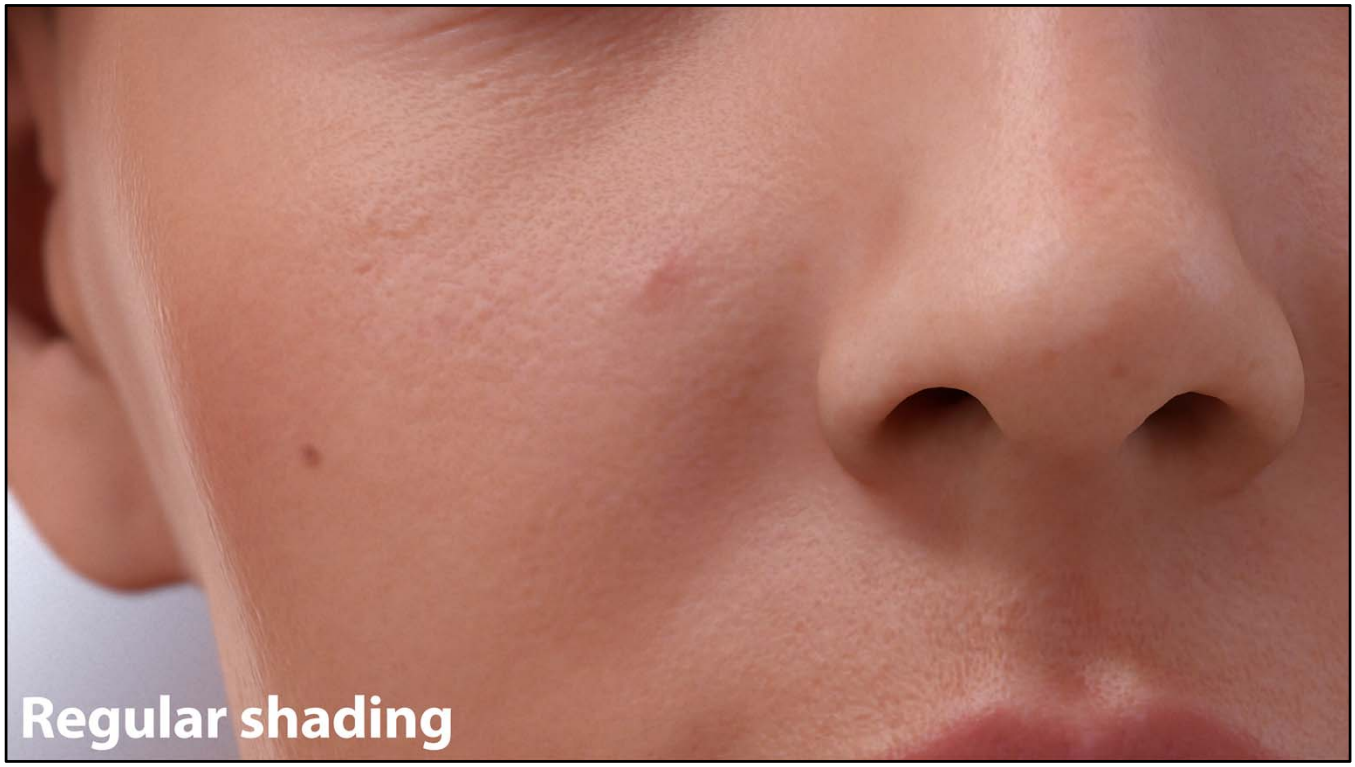
Second: part of this light reflects off the oily layer of the skin.

And finally: the remaining light enters the skin and scatters, which is a phenomenon called subsurface scattering.

Specular Reflections

ACTIVISION | BLIZZARD

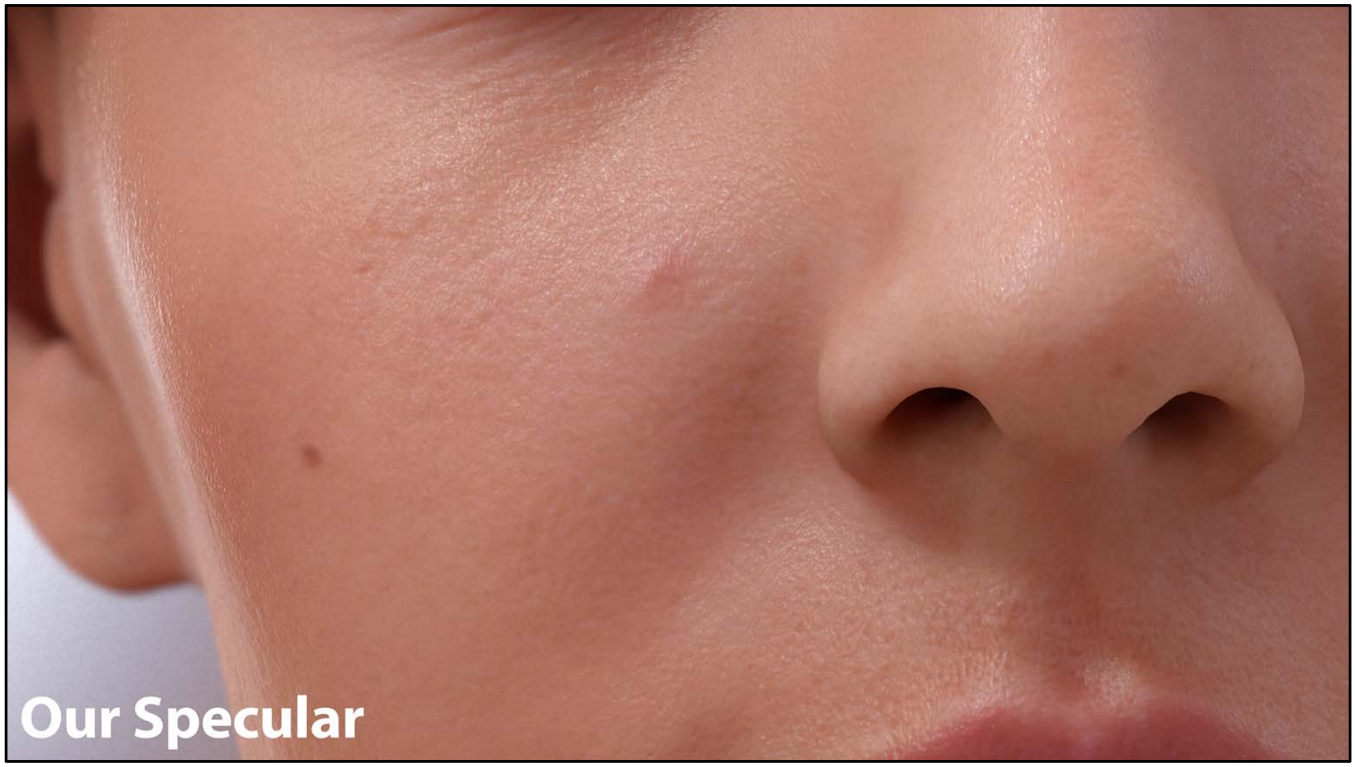
I'll start with the specular reflections.



So, what's the problem with the specular?

For us, it's that it looks like plastic.

So, how can we break this appearance, and turn it into something like…



This?

Physically based Specular

- **Black Ops 2 physically based specular BRDF for procedural lights**
  - **Improved version of Black Ops specular model, see [Lazarov2011] - *Physically-based lighting in Call of Duty: Black Ops***

$$\frac{F(\mathbf{l},\mathbf{h})G(\mathbf{l},\mathbf{v},\mathbf{h})D(\mathbf{h})}{4(\underline{\mathbf{n}\cdot\mathbf{l}})(\underline{\mathbf{n}\cdot\mathbf{v}})}$$

- **Environmental specular using a pre-filtered cube map**
  - **For more details: http://seblagarde.wordpress.com/2011/08/17/hello-world/**

ACTIVISION BLIZZARD

Thanks to Dimitar Lazarov and Sébastien Lagarde for code and support!

We begin with the production-proven Black Ops 2 specular model as the foundation.

It's physically based, which means it ensures accurate results with no manual tweaks.

(Many many thanks for the support of to Dimitar Lazarov, who answered every single question we had
and to Sébastien Legarde, for the countless email threads and interesting discussions)

Measurement-Based Synthesis of Facial Microgeometry

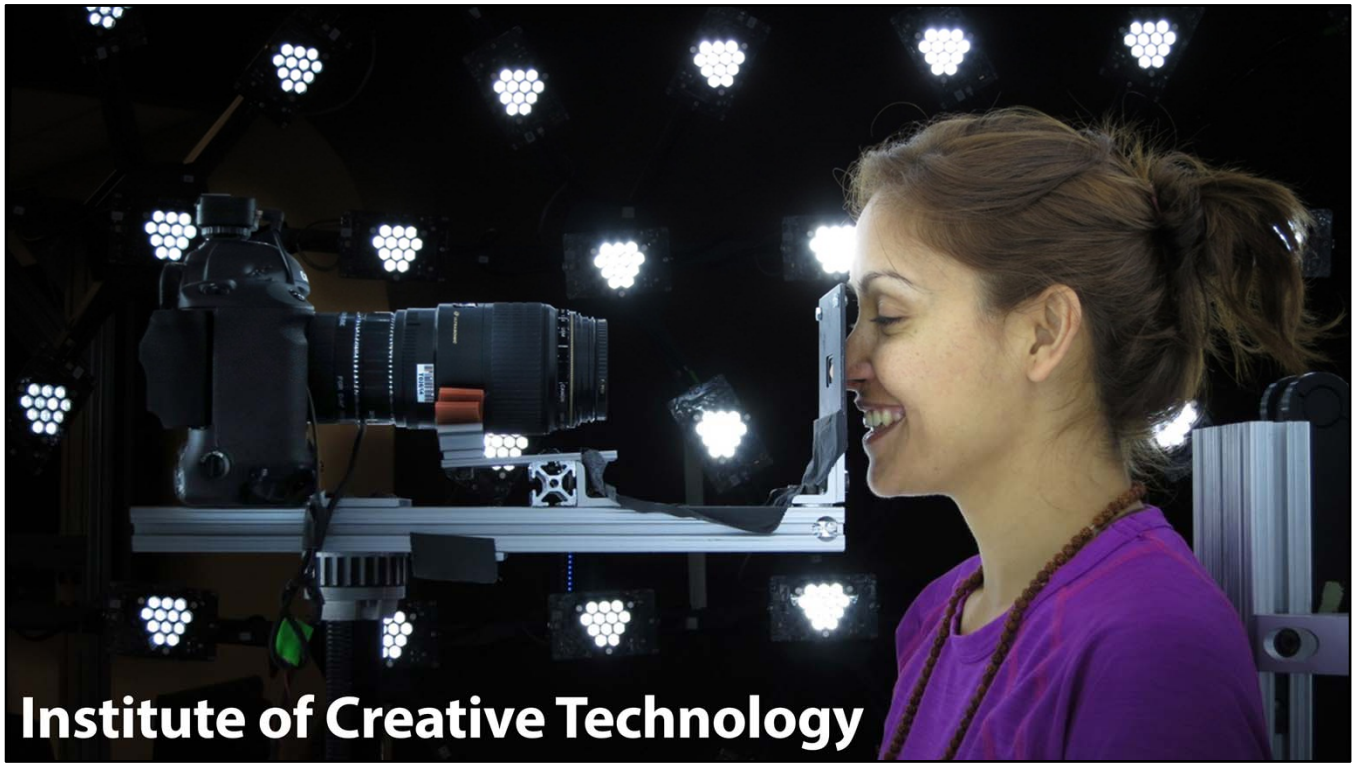
Paul Graham, Borom Tunwattanapong, Jay Busch, Xueming Yu, Andrew Jones, Paul Debevec, and Abhijeet Ghosh †

USC Institute for Creative Technologies, Los Angeles, CA, USA

(a) Rendering using 4K geometry | (b) Rendering using 4K geometry and microscale BRDF | (c) Rendering using synthesized microstructure | (d) Comparison photograph

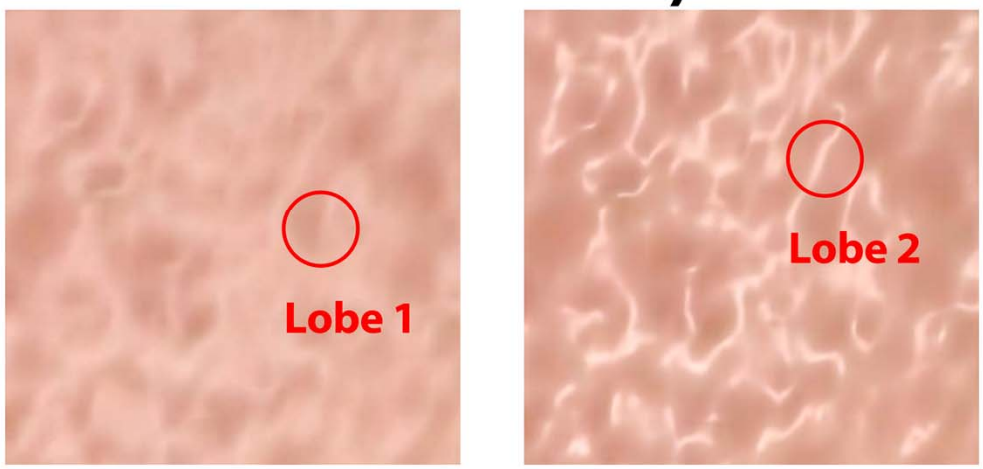
Figure 1: *(a) Rendering with scanned mesostructure (4K displacement map). (b) Rendering with scanned mesostructure (4K displacement map) and microstructure measured BRDF. (c) Rendering with synthesized microstructure (16K displacement map). (d) Photograph under flash illumination. See Fig. 3, Subject 2 for the skin patches used in microstructure synthesis.*

Using this model, we tried to mimic the results from a state-of-the-art technical report from the ICT (Institute of Creative Technologies),
where they describe a very accurate skin capture system.



In summary: they capture micro-scale details for tiny patches of the skin – in this case the nose – and smartly clone it over the face,
which allows us to accurately capture the skin reflection structure and properties.

Two Lobes Rendering (ICT Reference)



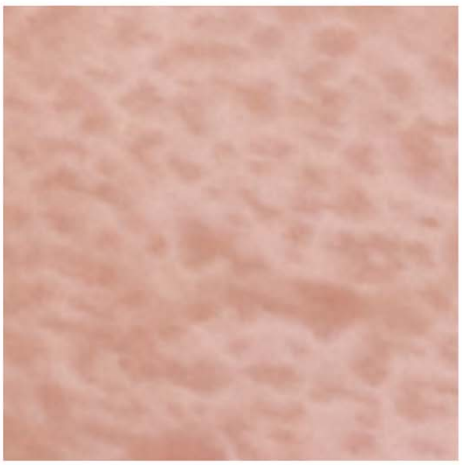
Regular | Microstructure BRDF Fit

From [Graham2013] *Measurement-Based Synthesis of Facial Microgeometry*

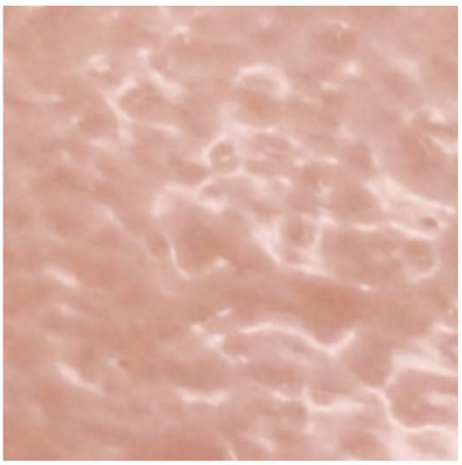
In my opinion, one of the key contributions of the paper is that it reveals the importance of having two specular lobes for rendering skin.

They obtain the lobe parameters using fitting techniques, but we managed to get similar results by visually adjusting them…

Two Lobes Rendering (Our Render)



One Lobe



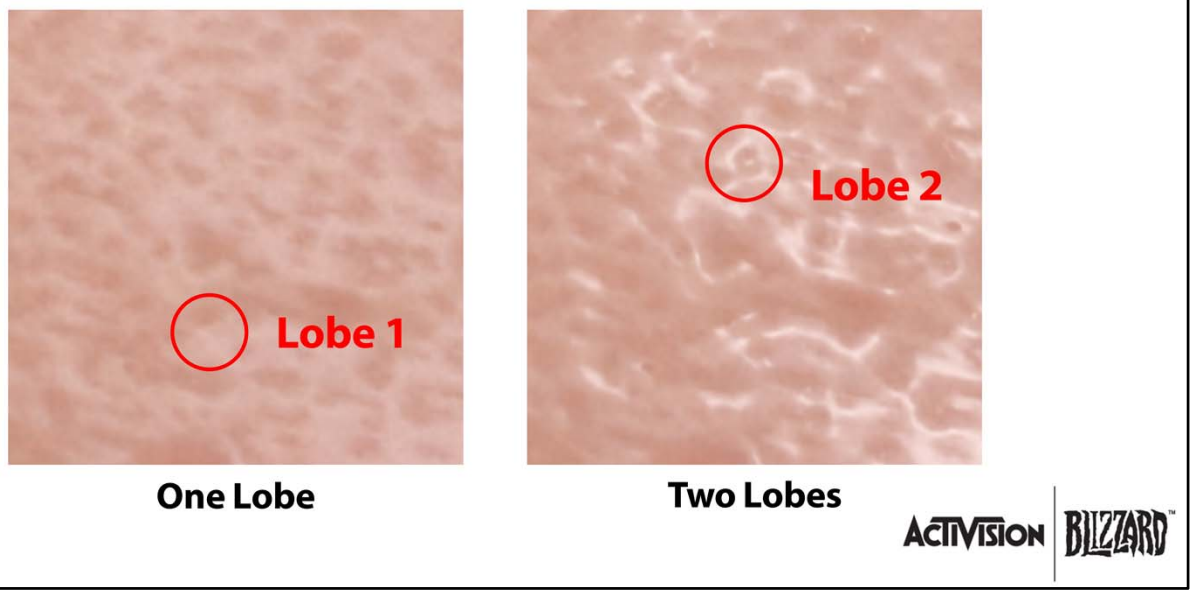
Two Lobes

ACTIVISION | BLIZZARD

…and as you can see in this comparison, the results are quite similar.

[Back and forth]

Two Lobes Rendering (Our Render)



Rendering with two specular lobes instead of a single one is **critical** for accurately rendering skin reflections.

Here you can see the difference of using one lobe…

…versus using two.

[Back and forth]

Implementation Details

- **Some implementation details:**
  - **We use a linear blend between the two specular lobes:**
    - **85% for lobe 1**
    - **15% for lobe 2**
  - **Specular power for lobe 2 derived from the first one (we wanted to avoid two gloss maps)**

In this slide you have some numbers we used for the two-lobe specular parameters.

Performance

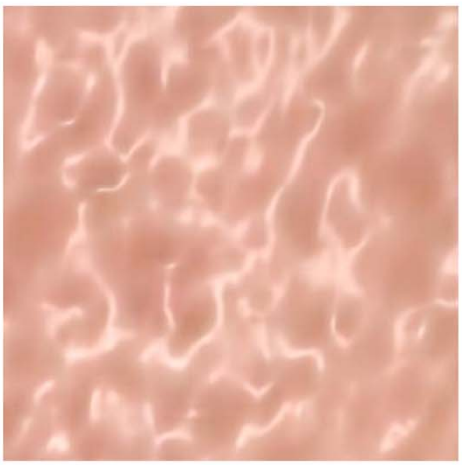
- **Proper vectorization of code allows us to implement the double lobe with only two extra instructions per light!**

ACTIVISION BLIZZARD

You may be thinking “this is going to be expensive!”, but I’ve got good news!

By using proper vectorization, this only adds two instructions per light.
(Only applies to vector-based architectures)

Microgeometry Render (ICT Reference)

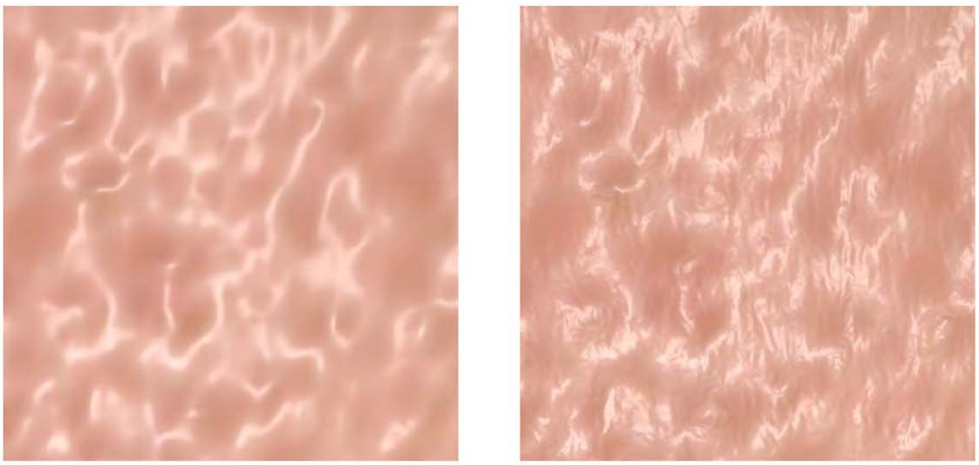


Microstructure BRDF Fit

From [Graham2013] ***Measurement-Based Synthesis of Facial Microgeometry***

Diving further into the technical details of the ICT paper, you can see that they not only fit the BRDF using captured microgeometry…

Microgeometry Render (ICT Reference)



Microstructure BRDF Fit | Microstructure Render

From [Graham2013] *Measurement-Based Synthesis of Facial Microgeometry*

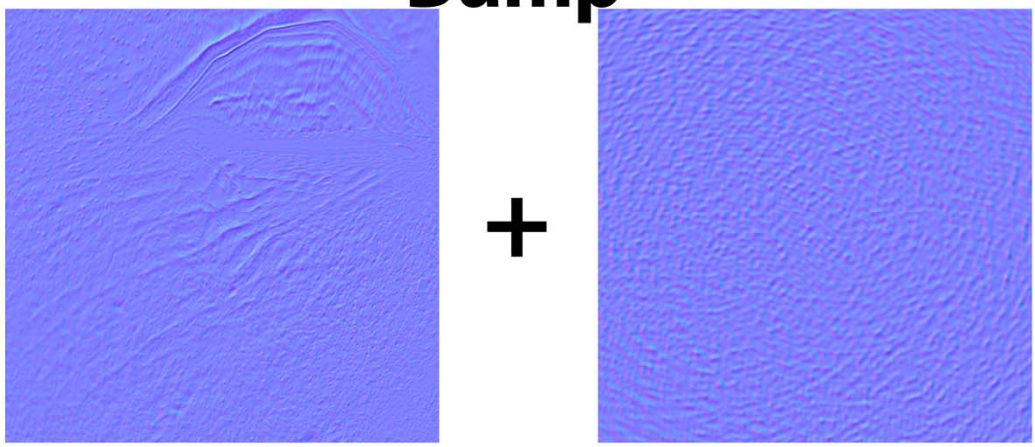
…but actually render with it.

Notice how the microstructure pattern breaks the synthetic appearance of the reflections.

Again, we wondered if we could find a plausible solution for real-time.

If you look closely at the microstructure distortions, you'll notice that they look pretty much sinusoidal.

Sine-Based Tiled Procedural Bump

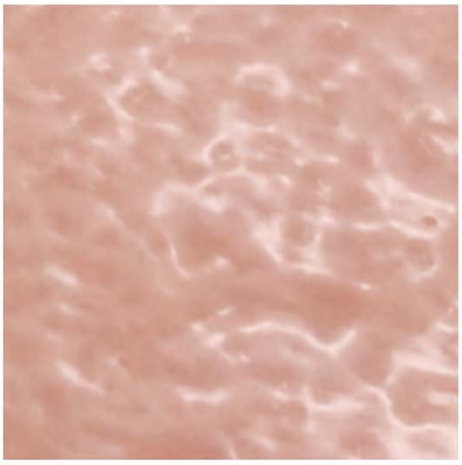


Base Map + Micro-Detail Map

Scale is exaggerated for illustration purposes

So, we thought: why not perturb the original base map with tiled sinusoidal noise?

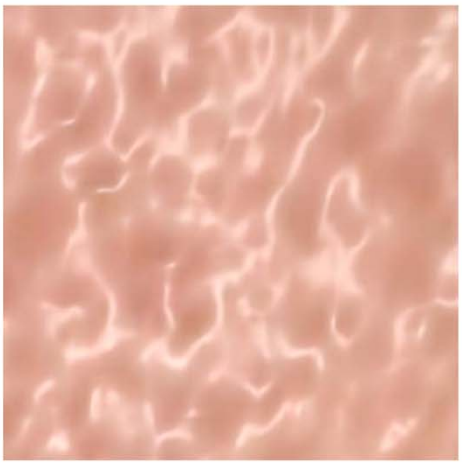
Microgeometry (Us)



ACTIVISION | BLIZZARD

We found this simple approach to yield very good results.

Microgeometry (ICT Reference)



From [Graham2013] *Measurement-Based Synthesis of Facial Microgeometry*

Here you have the ICT reference… [back and forth]

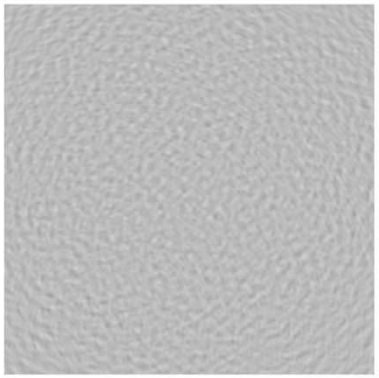
Again, the result is good enough for our purposes.

Here we have a render with two lobes…

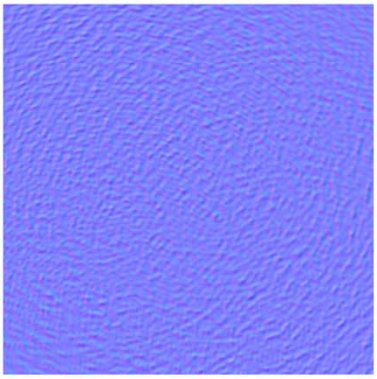
…and here with two lobes and microgeometry noise.

Sine-Based Tiled Procedural Bump

$$p_z = \frac{1}{100}\sum_{1}^{100} \sin(\sqrt{(p_x - rand())^2 + (p_y - rand())^2} \times 1/(2.08 + 5 \times rand()))$$



Height Map



Normal Map

ACTIVISION | BLIZZARD

For creating this noise map, we used 100 radial sinusoidal generators to produce a height map, which was then converted to a normal map using NVIDIA Texture Tools.

This same texture was also useful for the eyes, as we'll see later on.

How to Blend Them

- **We want the detail normal to perturb the base normal (not just to blend)**
- ***Reoriented Normal Mapping* (RNM) is perfect for that!**
- **http://blog.selfshadow.com/publications/blending-in-detail/**

Thanks to Stephen Hill and Colin Barré-Brisebois for the technique!

For blending, we modified the base normal using reoriented normal mapping, which gave very good results.

Skin Gloss Map

- **What should we put in here?**
- **A lot of approaches, most of them visually guided.**
- **Good enough for current gen. Possibly not enough for next gen!**

ACTIVISION | BLIZZARD

I'd like to talk a little bit about something related to this: what should we put into the gloss map for a head?

There are a lot of approaches, most of them visually guided.

This is acceptable within the constraints of current consoles, but probably not for next generation.

Skin Gloss Map

- What is glossiness?
- How does it relate to bumpiness?
  - Bumpiness, as defined by the normal map

ACTIVISION | BLIZZARD

Skin Gloss Map

- Glossiness is related to bumpiness at the microstructure level.
- If we model the microstructure bumpiness at high enough resolution, we can accurately obtain the glossiness
- But we'd need to model the skin at mm level

- We have actually modeled the microstructure with our tiled noise!

ACTIVISION | BLIZZARD™

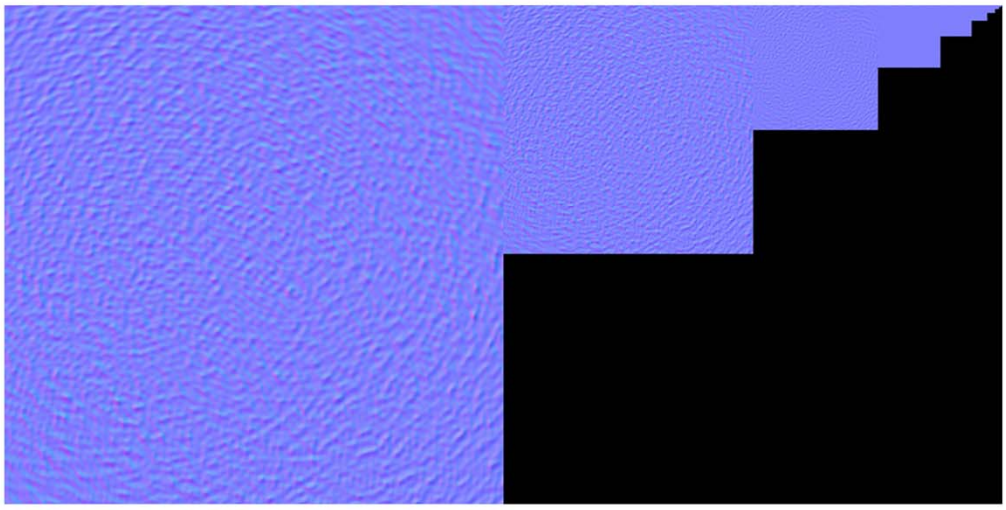
Skin Gloss Map

- **The microstructure tiled bumps convert to glossiness when we see the surface from far away**
  - **[Hill2012] *Rock-Solid Shading: Image Stability Without Sacrificing Detail***

ACTIVISION | BLIZZARD

So, what should happen is that our tiled bumps should convert to glossiness when we see the head from far away.

Tiled Bumps



Just to show it with an example, here we have the bump mip-maps.
As you can see, the normals flatten with distance.

Variance to Gloss

- **Estimate normal variance, and use it as glossiness**

$$\sigma^2 = \frac{1 - |N_a|}{|N_a|}$$

New power

$$p = \frac{s}{1 + \sigma^2 s}$$

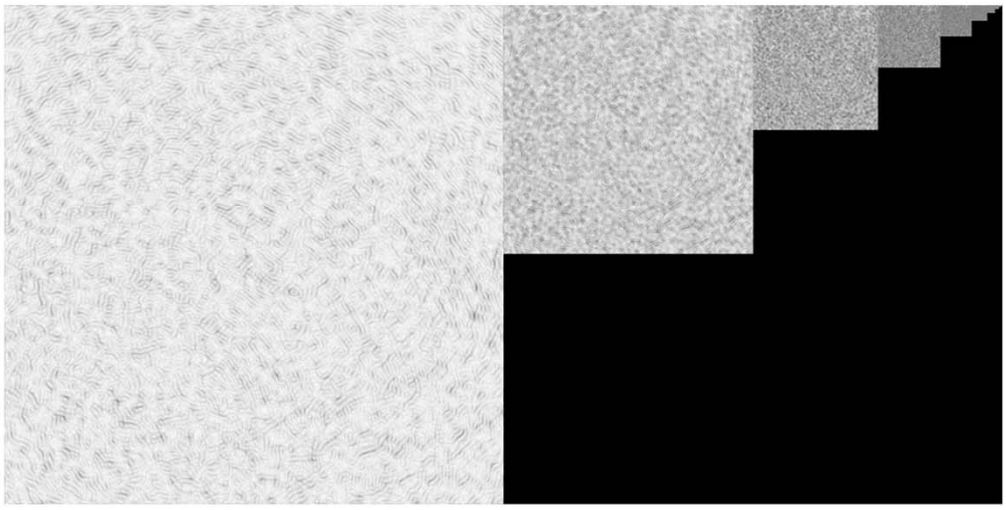
old power

From [Hill2012] *Rock-Solid Shading: Image Stability Without Sacrificing Detail*

We can convert this bump map into a gloss map by estimating normal variance…

Tiled Gloss

ACTIVISION | BLIZZARD



Thanks to Stephen Hill for the map!

…obtaining this result.

Notice how the gloss noise, unlike the bump one, is present even when far away from the model.

Dealing with Detail Maps

- **Combine base glossiness with detail variance:**

$$\frac{1}{s'} = \frac{1}{s} + \sigma^2$$

- **We assume that base glossiness already contains base normal map variance. You will get sweaty heads when you are far away otherwise!**
- **More details in [Hill2012] *Rock-Solid Shadi Stability Without Sacrificing Detail***

ACTIVISION | BLIZZARD

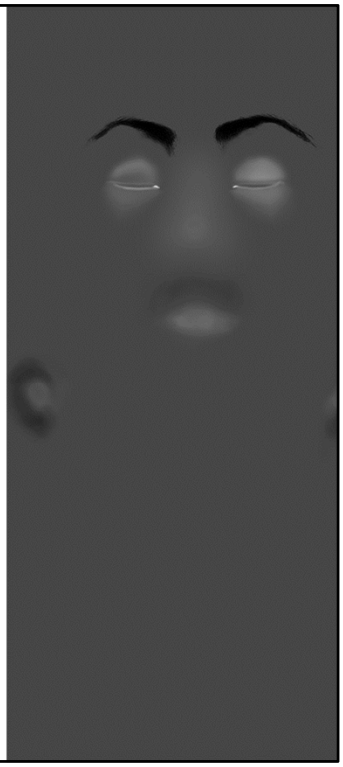
So, we have now two sources of glossiness: the artist authored gloss map and the gloss values obtained from the tiled noise.

For combining them, we need to store the noise variance alongside with the normal direction,
and combine it with the original artist authored gloss map.

We refer you to Stephen Hill and Dan Baker's presentation about specular antialiasing for more details.

Gloss Map Authoring

- **Only low-frequency details**
- **Bump/gloss tiled noise for micro details**
- **Normal map should contain meso details**
- **This improves workflow**



As a final comment on this, our recommendation is to author low-frequency details in the gloss map, and then add microdetails using tiled bump/gloss.

In the end, the noise texture allows for very intuitive map editing, as an artist only has to paint low-frequency gloss changes.

High-frequency details are automatically generated on the fly from the tiled microstructure texture.

(First Lobe) Gloss Map

	Spec. Power S	Gloss G (Scale=14)
Main	26.5	86
Lips	111.43	124
Upper Lip	18.37	76
Nose	38.49	96
Ear (Main)	10.07	61
Ear (Near the hole; due to wax)	61.1	108
Eye (Top)	44.22	100
Eye (Bottom)	80.63	115
Caruncle (very wet)	31288	255
Eyelid (Top; it is wet)	73.51	113
Eyelid (Bottom; wetter due to gravity)	512	164

$$S = 2^{(14\times\frac{G}{255})}$$

In this slide you have the values we used for authoring the gloss for the first lobe, both as specular power and log-encoded.
More details in [Lazarov2011] - *Physically-based lighting in Call of Duty: Black Ops*.

The second lobe should be either 2.0*G or S^2.0.

Using the “Main” gloss preset for the whole face works surprisingly well.

These values work well for regular skin, but not for old skin (for instance), which is usually more dry.

Some rationale behind it:

Ears have lower gloss in general as there seem to be less pores there (and sweat comes out from pores). However, there’s higher gloss near the earhole because there’s usually wax in there.

For the eye area, the gloss depends on the tear fluid of the eye dropping to nearby skin areas, so that's the reason for having more wet values in the eyelids/caruncle.

The bottom eyelid has a higher value because of gravity, and the caruncle setting is

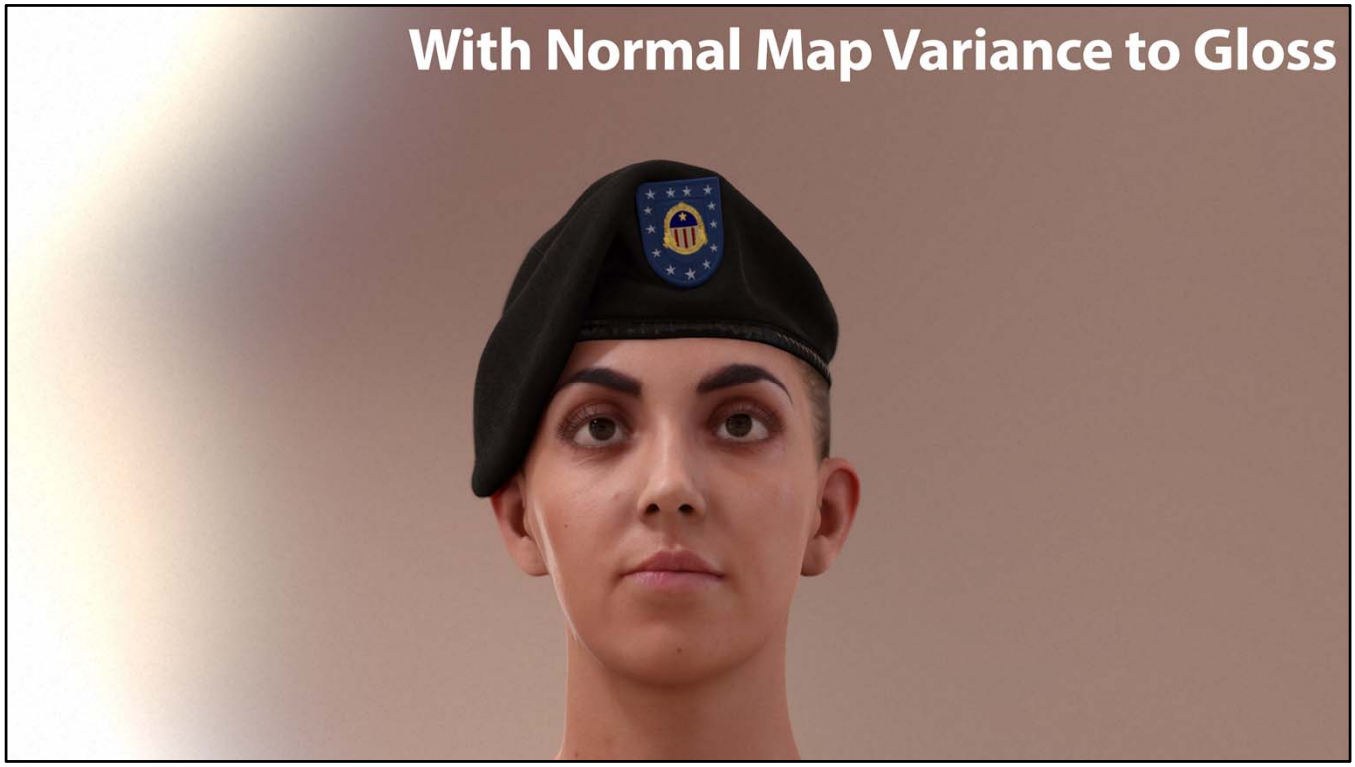
higher than both because it is where tear fluid sinks (it comes from the opposite side, on the top part of the eye).

Also note that this gloss map was painted for normal skin.

In case of sweat, the forehead, cheek and nose should get higher gloss values, since they contain larger pores than other facial areas, and thus sweat is more intensively produced in those zones.

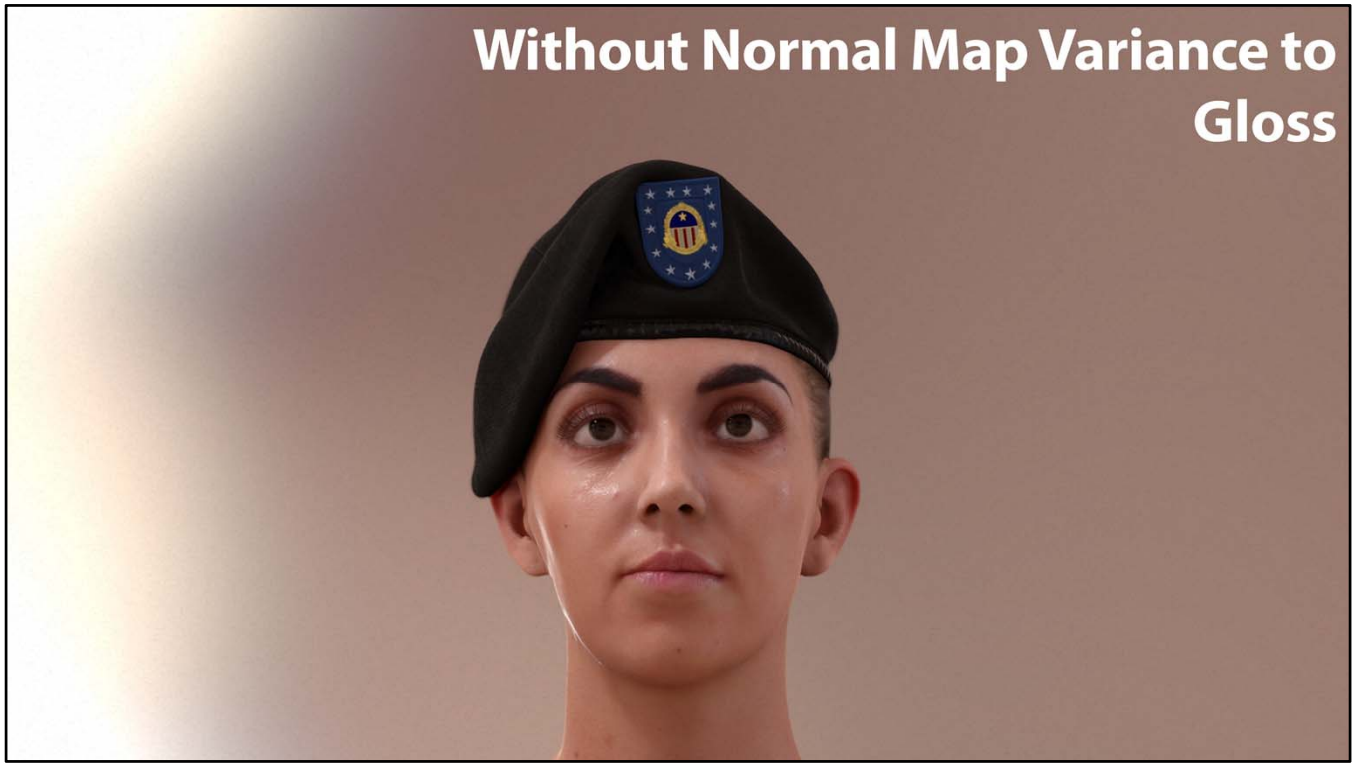
Some notes:

- The image colors won't match the values in the second column, as they were created for a gloss scale of 17)
- The values are intended for the Black Ops 2 BRDF specular model, so they may need to be adjusted for other models
- Gloss G has a range of 0 - 255



We want to highlight the extreme importance of taking specular antialiasing seriously.

If you don’t…



…you will end up with sweaty faces when you look at them from far away.

Even if they are not sweaty in a close up.

This is normal map variance in action.

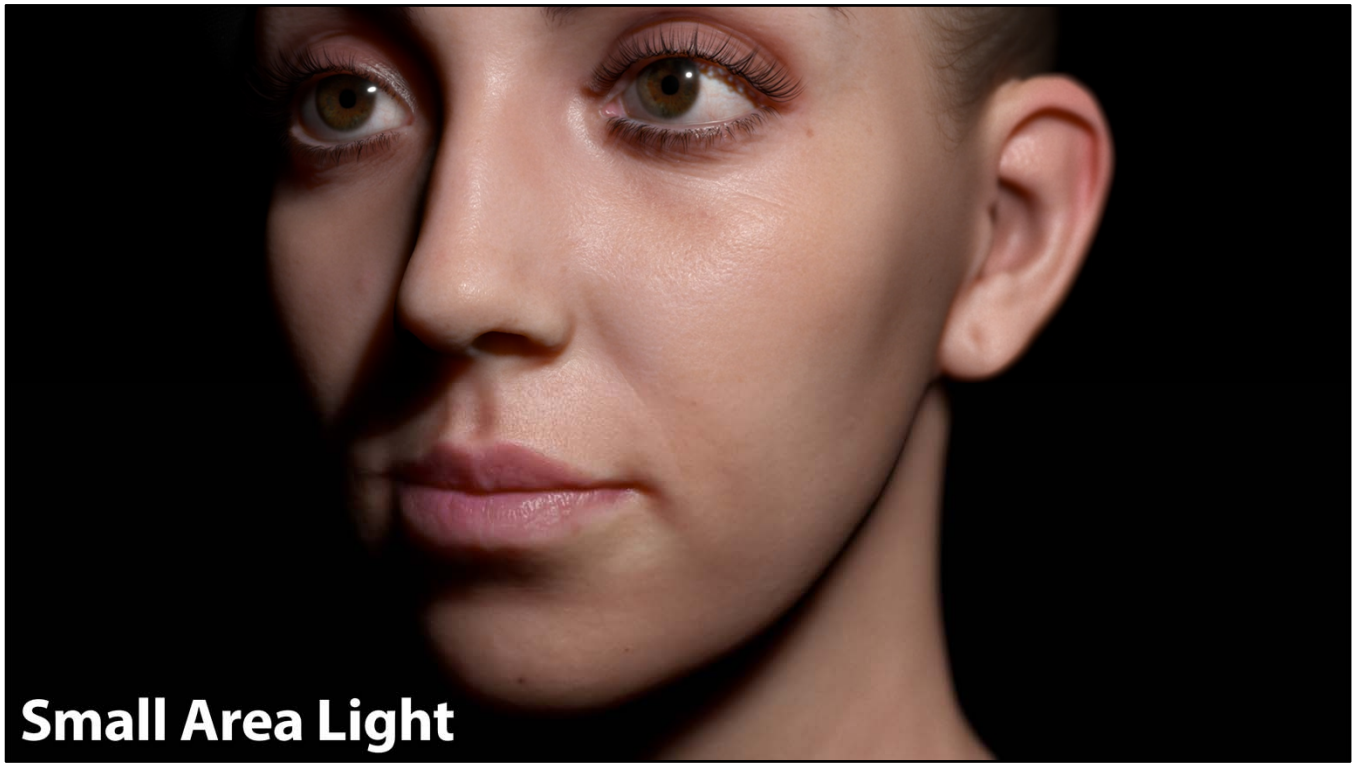
Geometry normal variance is important as well, especially for the eye wetness geometry.

(More on the geometry itself later on.)

Without it…

…you end with an unbearable amount of aliasing, given the high specular power used for this area.

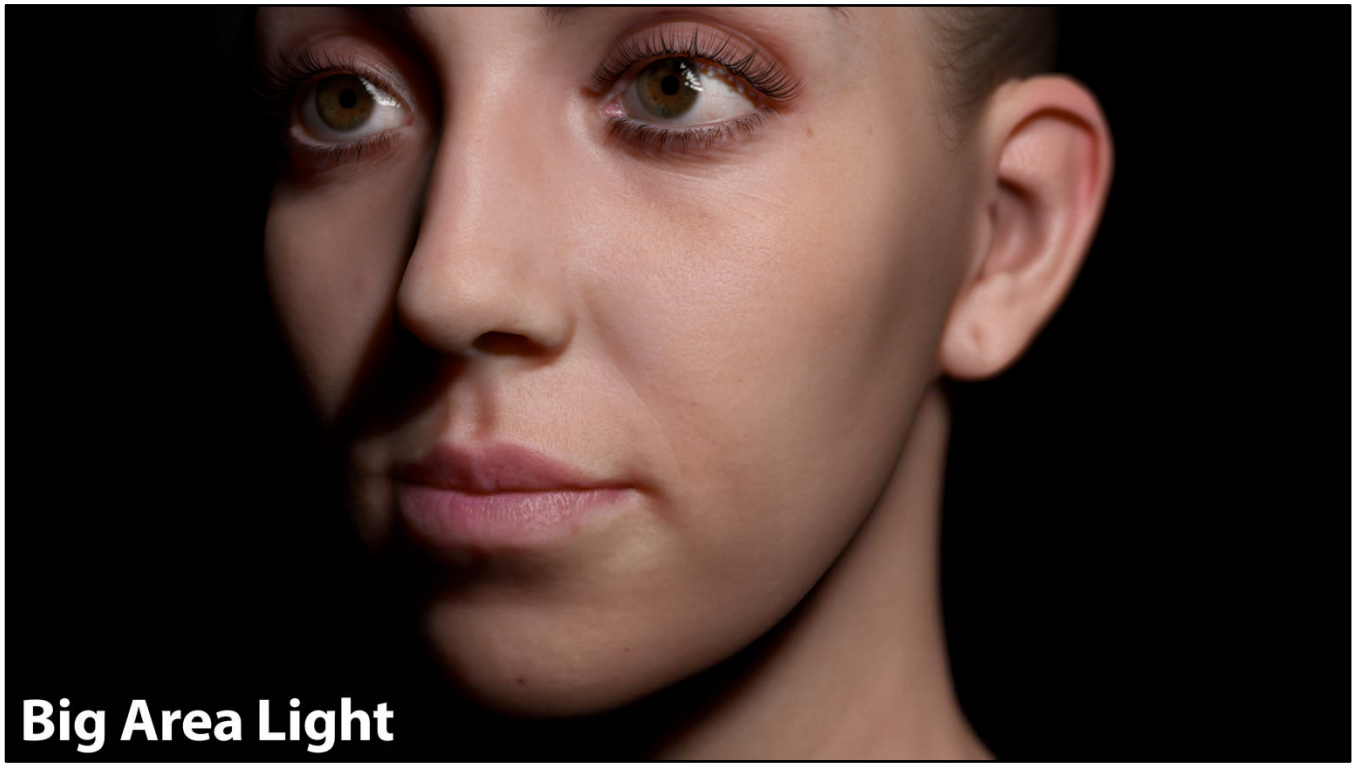
Again, we refer you to [Hill2012] *Rock-Solid Shading: Image Stability Without Sacrificing Detail* for information about Geometry AA.



I’d like to also highlight how hard authoring gloss can be, as it depends a lot on the lighting setup.

The size of the light, and its distance is something to take into account.

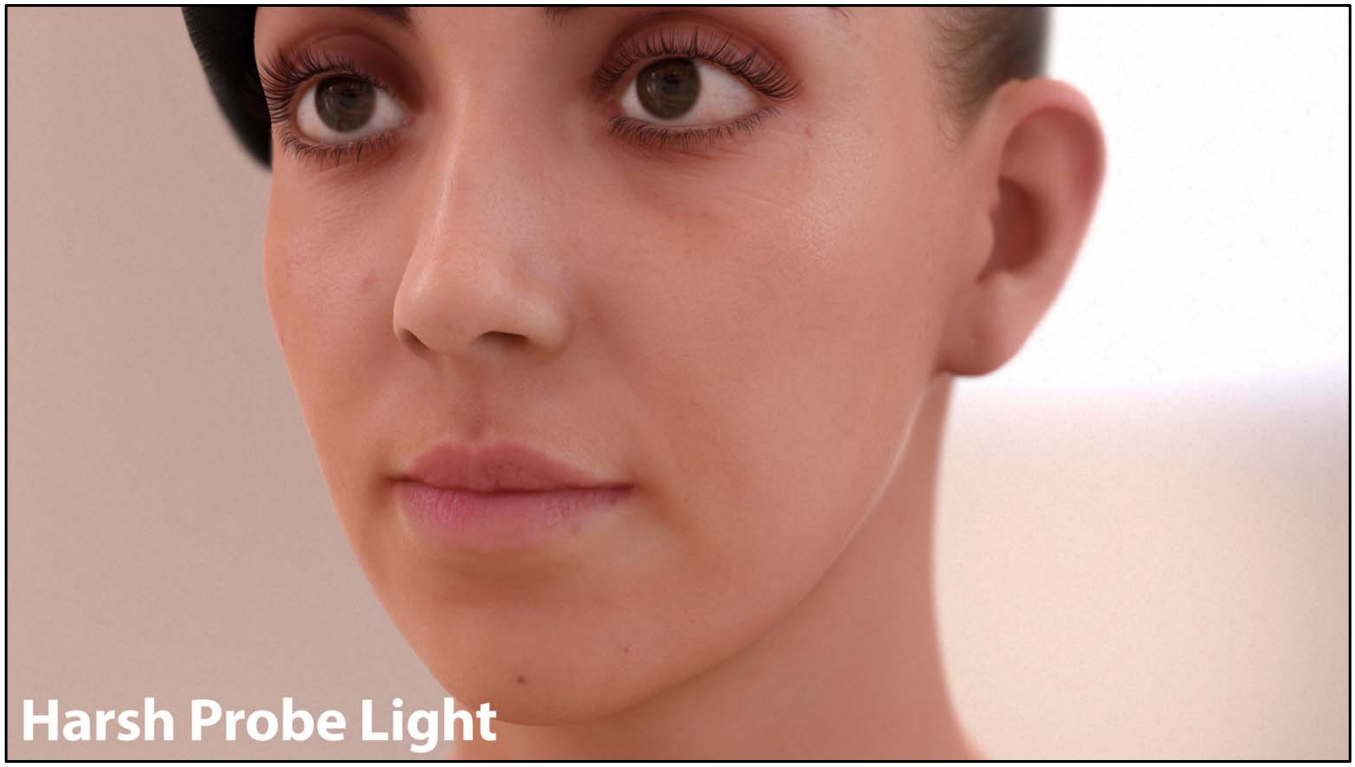
Here we have a point light source…



…and here a bigger area light.

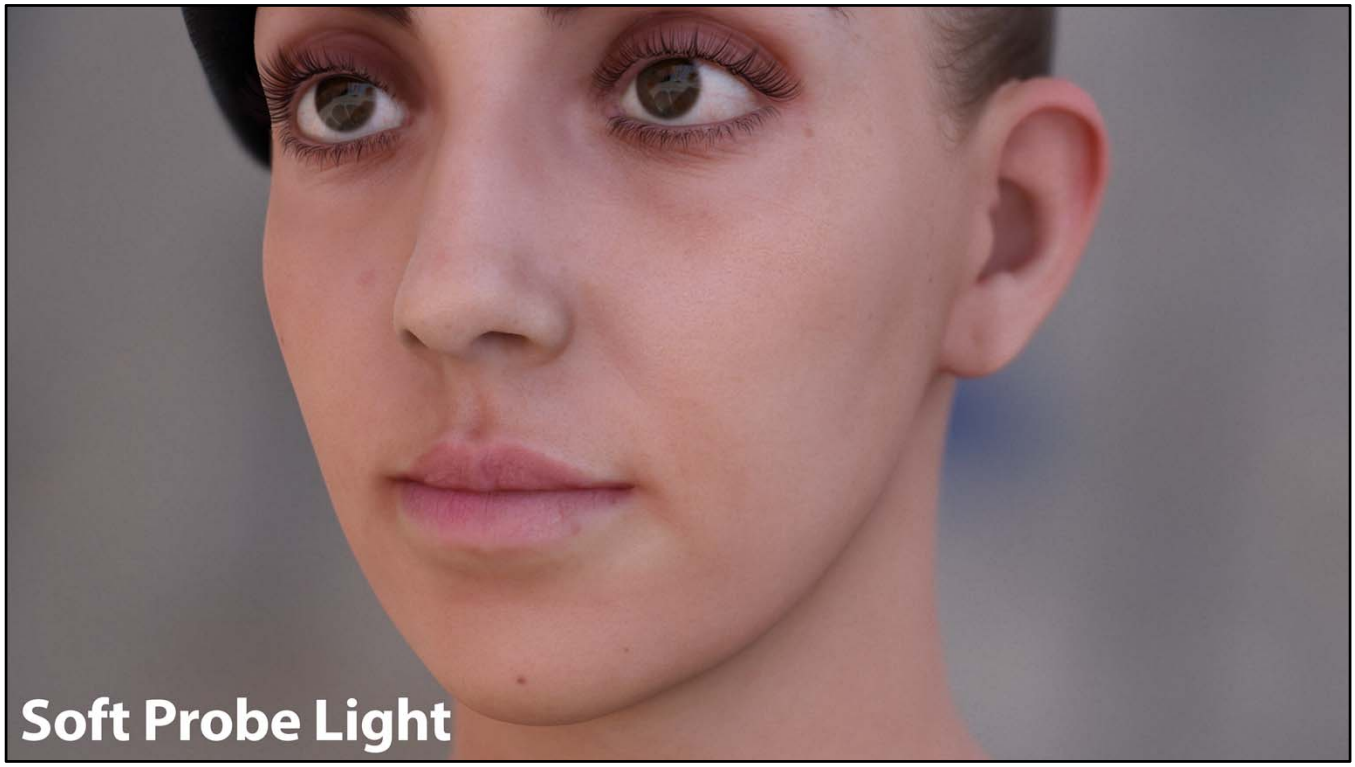
Notice how specular reflections have softened.

We like to think of gloss and the size of the light as two sides of the same coin, as they both blur/sharpen the specular highlights.



With probe lighting it can get even less predictable (and hence, we recommend authoring with procedural lighting).

This is a harsh probe lighting setup…



…and this is a softer one.

The degree to which the specular ‘pops’ depends on the input.

It’s very easy to adjust the gloss to make this happen all the time, but in reality this is wrong.

Specular highlights should only be visible if there is a very strong directional light source somewhere.
(Strong with respect to the rest of the environment.)

This makes specular more difficult to author in a robust way.

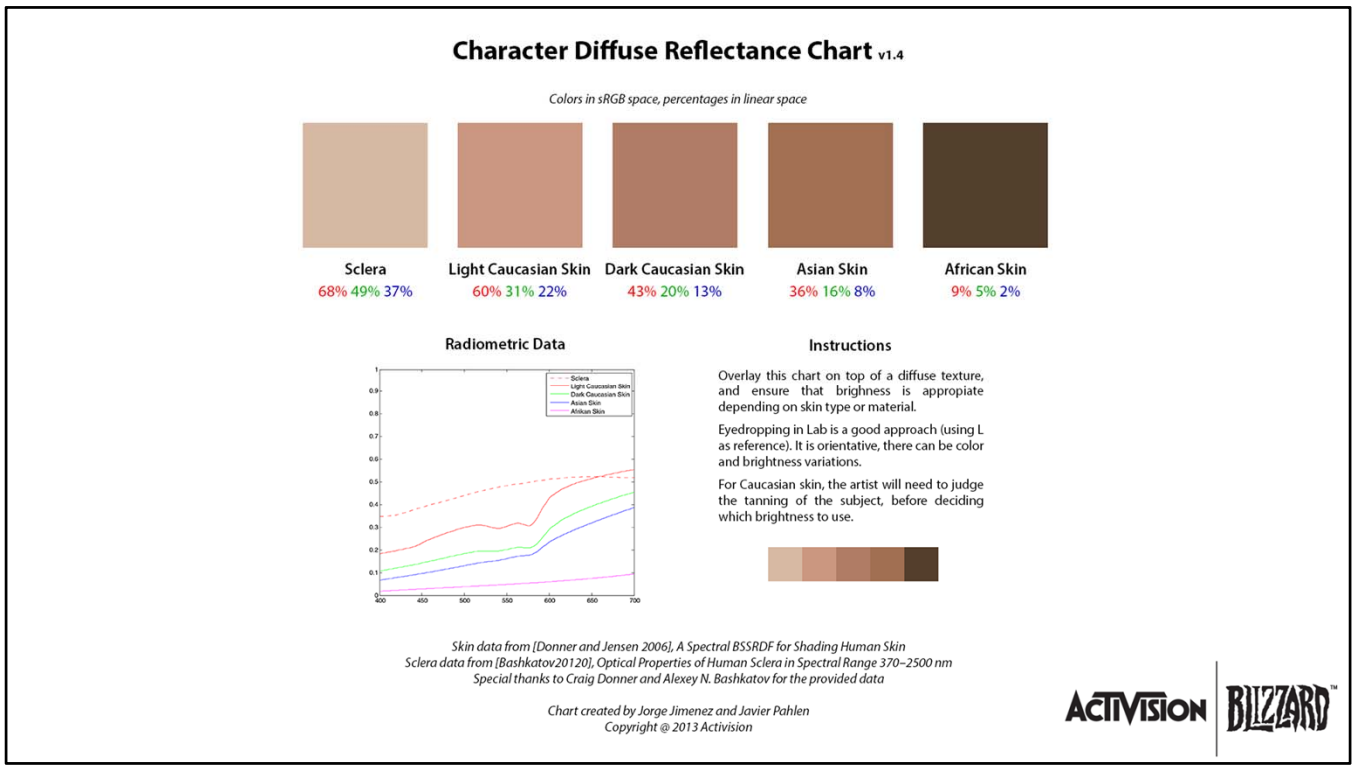
Combinatory Parameter Explosion

- **Diffuse Albedo**
- **Specular Gloss (with visual changes depending on light size, making it harder to author)**
- **Specular Albedo**
- **Ambient/Specular Occlusion**

- **Easy to find a solution for a single lighting configuration**
- **Extremely hard to find a solution that works all time**

For me, the takeaway is that there are too many parameters that control the appearance of a material.

It's easy to find a solution for a particular lighting configuration, but extremely hard to find something that works all time.



One of the aspects we worked on is the percentage of diffuse reflection for various character parts.

For this we created a reference chart for artists, using spectral measurements (which should be fairly accurate).

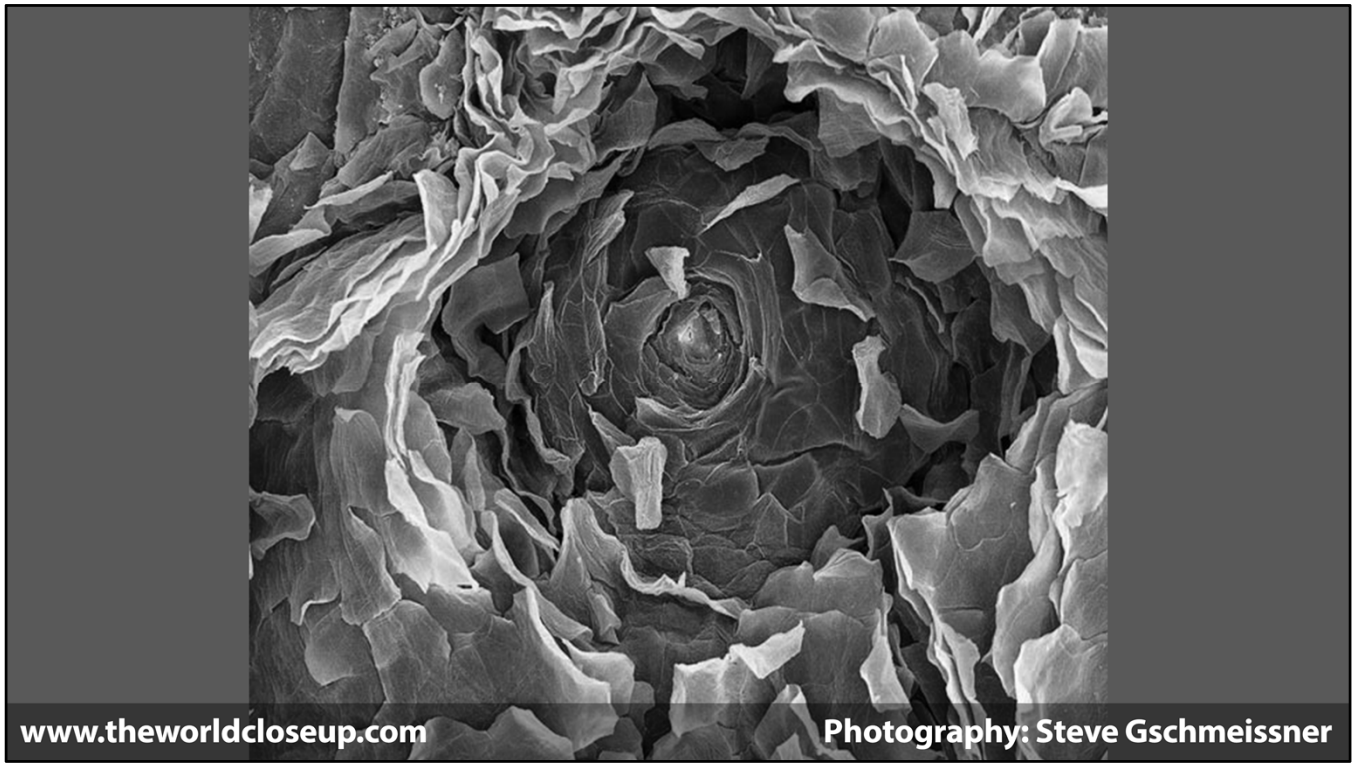
We think that when using a physically based render system, we should provide a range of good values to artists, as this is probably the main advantage of such a system.

(Note that sclera color is not correct. Unfortunately it's based on an in-vitro experiment, which means that the eye was dead tissue.)

Human Pores

ACTIVISION | BLIZZARD

Now, to the next topic, human pores.



This beautiful image is a scanning electron micrograph of a human pore, taken by Steve .

Notice that there is a lot occlusion happening there.

You may be thinking that this completely irrelevant for rendering. We thought the same…

(Photograph used with the permission of Steve

Comparing with Reality

Render

Photograph

From [Graham2013] *Measurement-Based Synthesis of Facial Microgeometry*

…but we were wrong!

If we compare a render with a photograph…

Comparing with Reality

Render

Photograph

From [Graham2013] *Measurement-Based Synthesis of Facial Microgeometry*

…you will notice that we're getting specular reflections inside of the pores.

In general this is not physically possible, as there is a hole in the pore.

Cavity Occlusion

ACTIVISION | BLIZZARD

So, to solve this, we thought a good idea would be to use a cavity map.

Here we have a comparison…

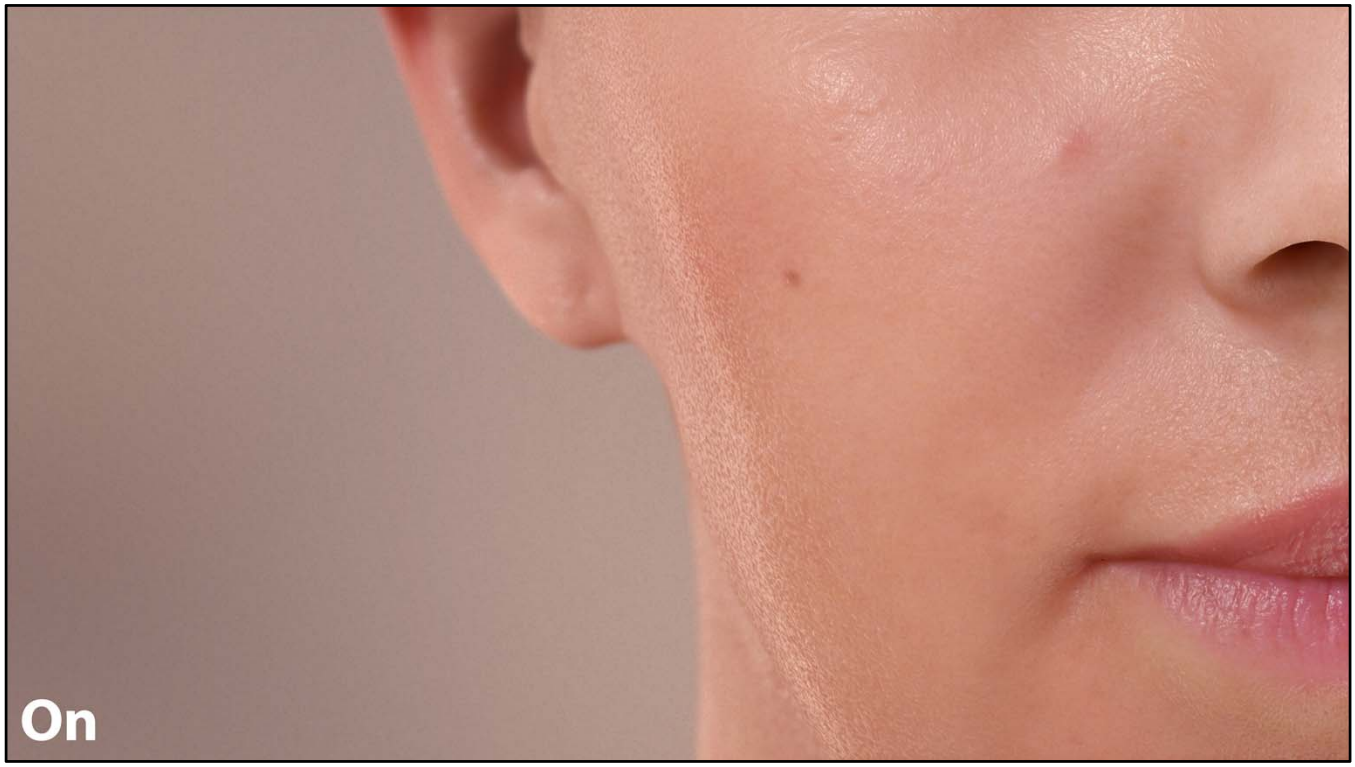
…and you can see how using the cavity map further breaks the specular pattern, adding more imperfections to the image.

This helps to add occlusion to the pores, which is important for ambient lighting…

…as you can see on this slide [back and forth].

We're using the default cavity map straight from ZBrush.

Note that:
- The effect is very reactive to view direction, this screenshot has been chosen to maximize the contrast.
- It's actually enhanced in motion.
- A character with a lot of bumpiness (versus someone with soft skin) will exhibit a stronger cavity effect.

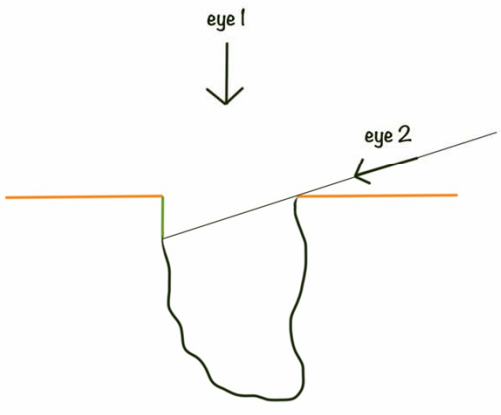


However it’s not that simple. If we turn the light to the side, we are in trouble.

Notice that the specular looks rather unnaturally contrasted. In this case we are using the cavity map as a specular occlusion factor,
effectively multiplying the specular by the cavity map.

What's Going On?

- **You cannot see the pore's holes from the side!**
- **So, we should remove the cavity occlusion in that case**
- **In theory the cavity is an occlusion value. So it should be:**
  - **cavity * Specular(gloss) * Fresnel(reflectance)**
- **However we can very cheaply remove it at grazing angles by using this hack:**
  - **Specular(gloss) * Fresnel(cavity * reflectance)**
- **Which also means we can use the same map used for reflectance (if any)**



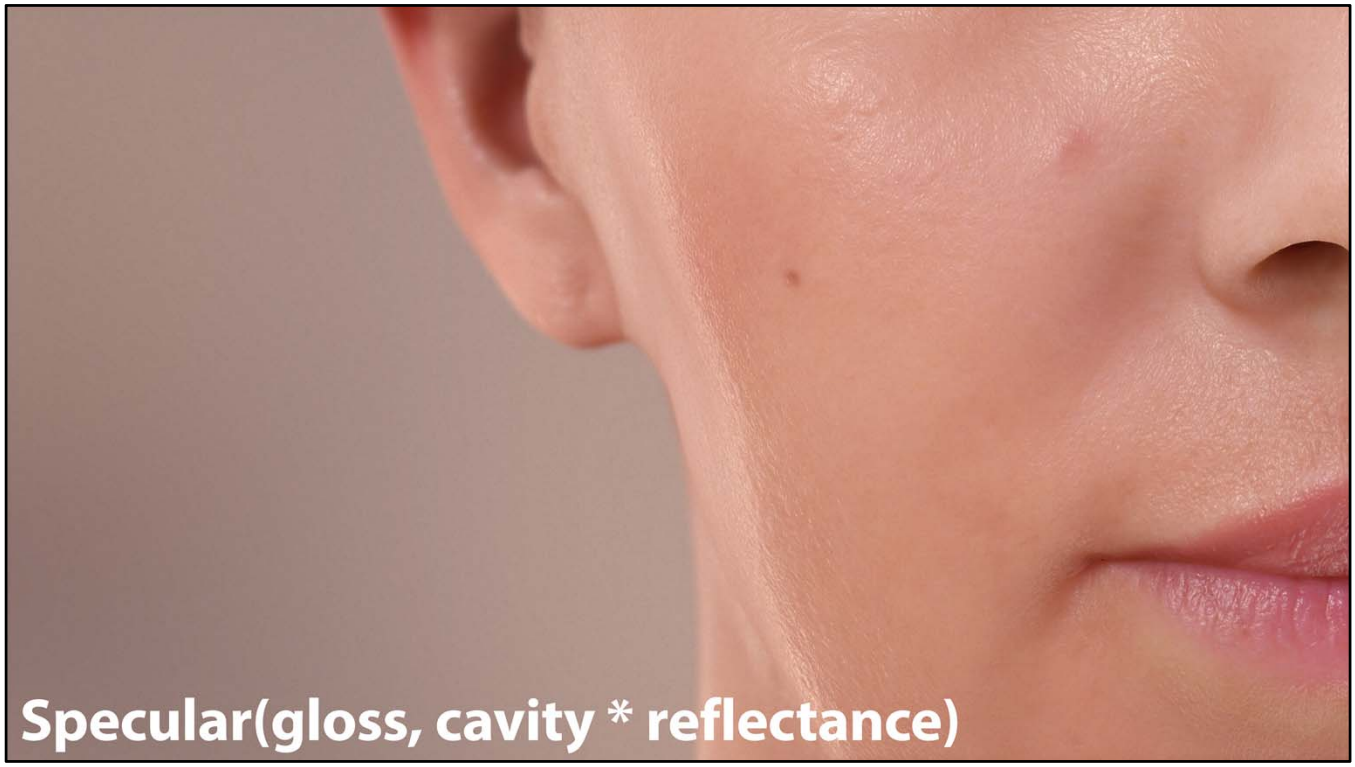
ACTIVISION | BLIZZARD

And the problem is that you don't see holes from the side.

But fortunately the solution is simple: just remove the cavity map at grazing angles.

And for that we can just multiply the reflectance inside of the Fresnel function, instead of multiplying the resulting specular.

This is not good…



But this is better!

(Faking) Global Illumination

ACTIVISION | BLIZZARD

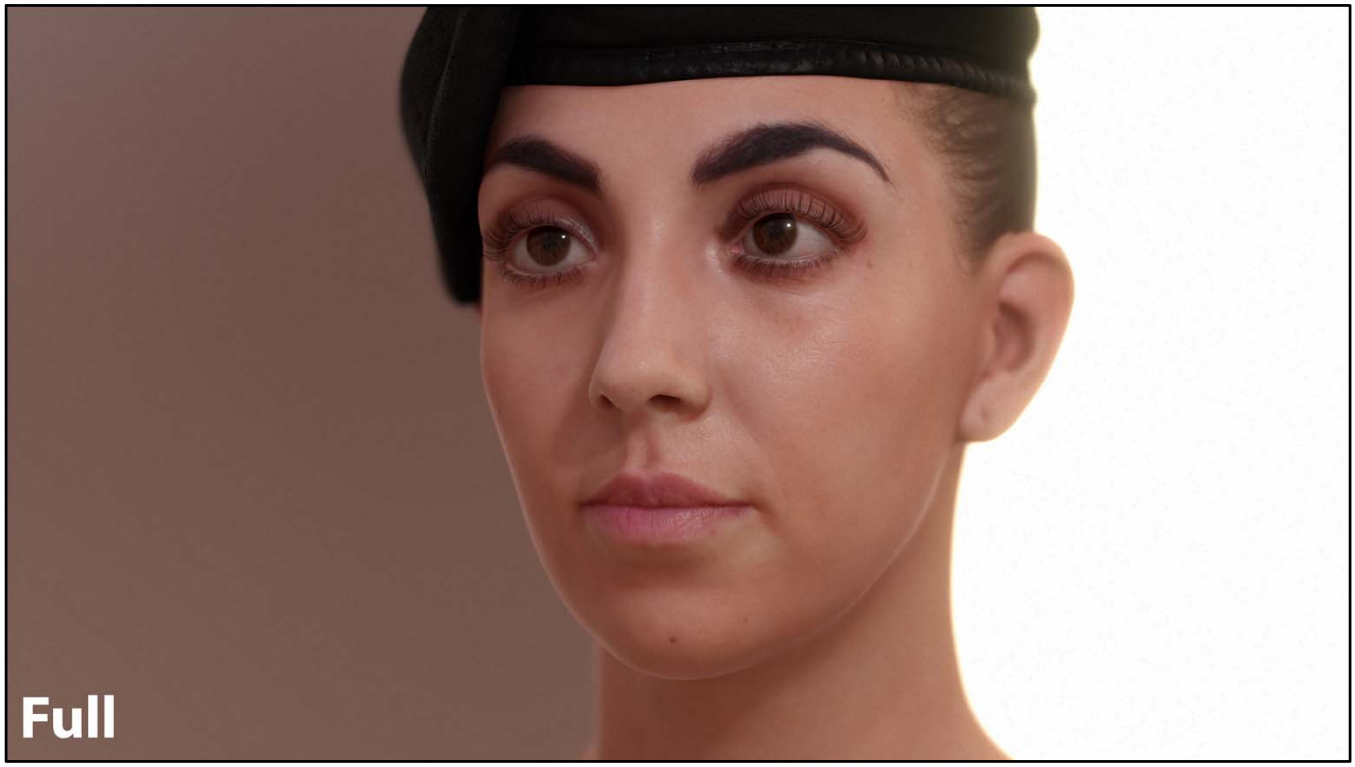
The next topic related to light transport is global illumination.

Or more accurately: faking it.

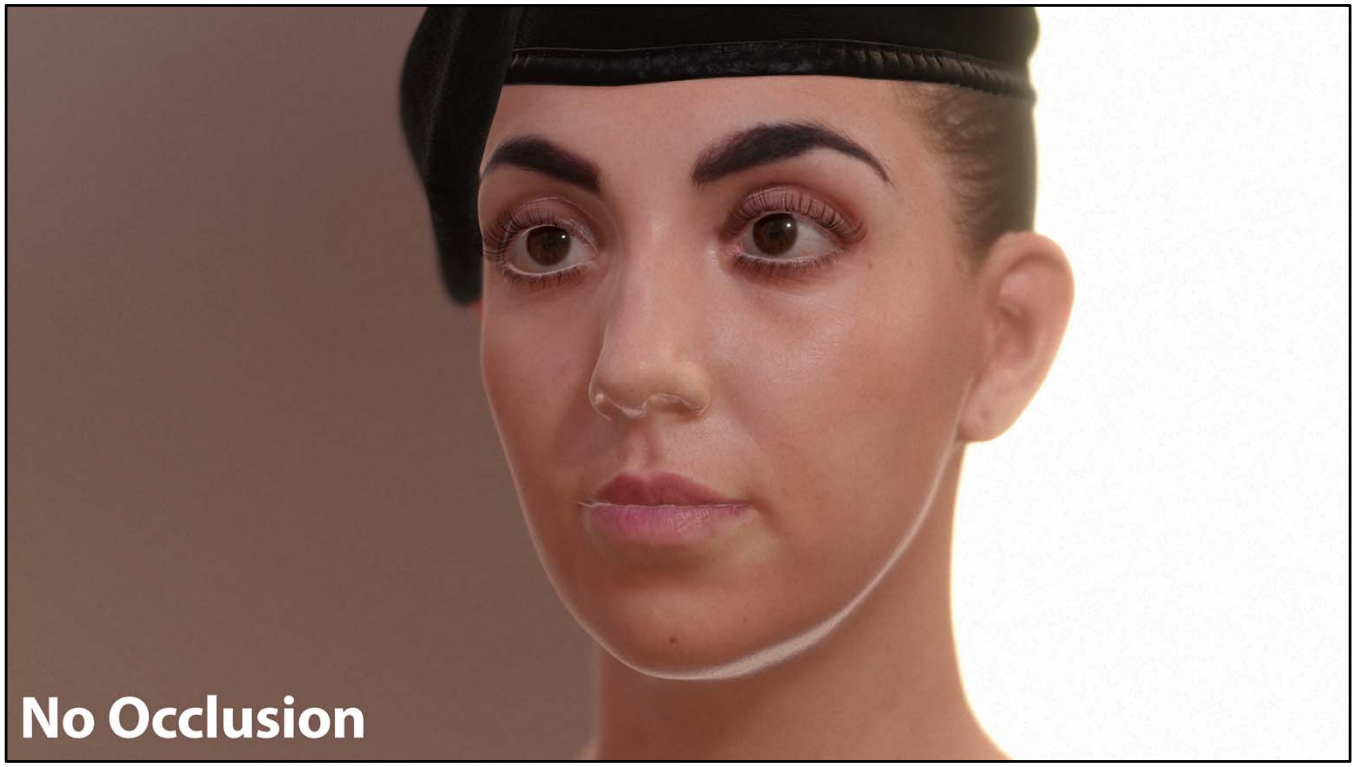
Faking Global Illumination

- Occlusion is crucial if (physically based) HDR light probes are used for lighting
- Both for the eyes and the skin

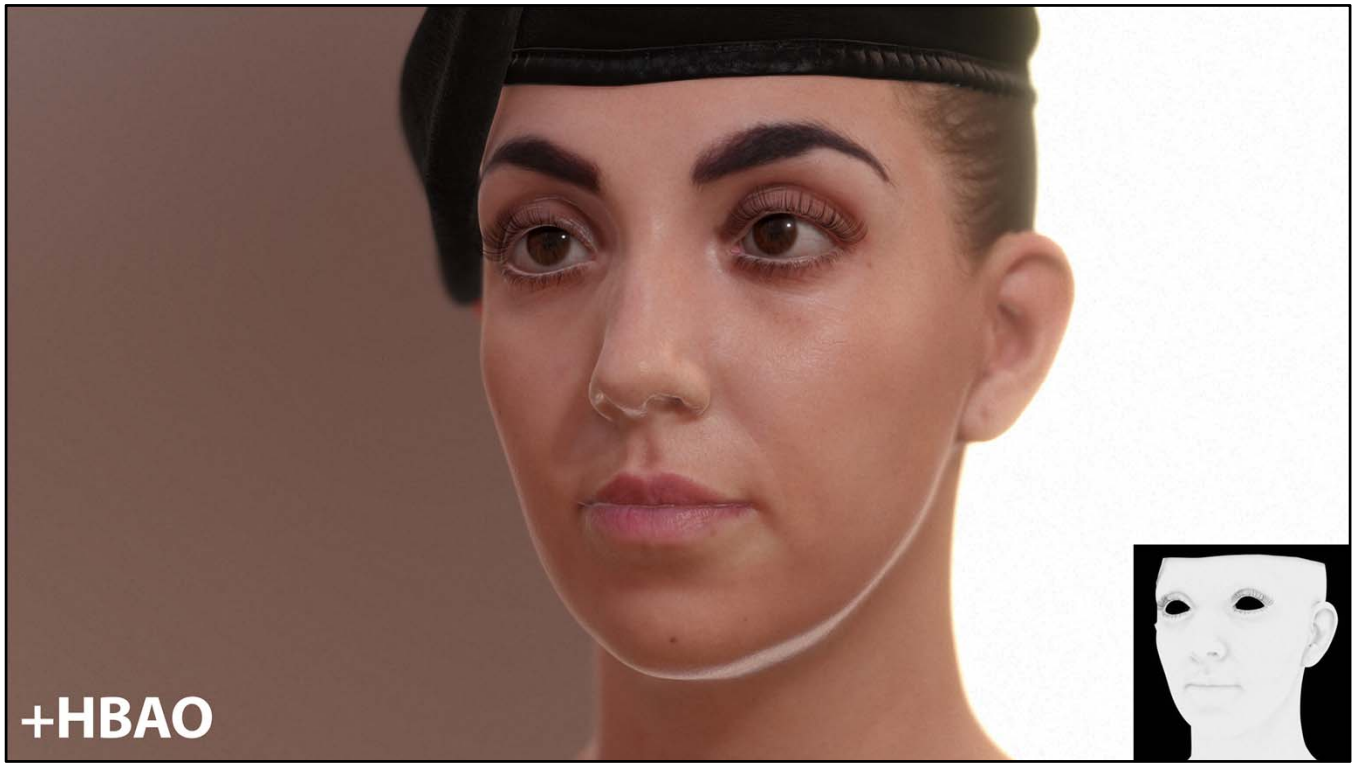
ACTIVISION | BLIZZARD



This is a shot using our full occlusion solution…

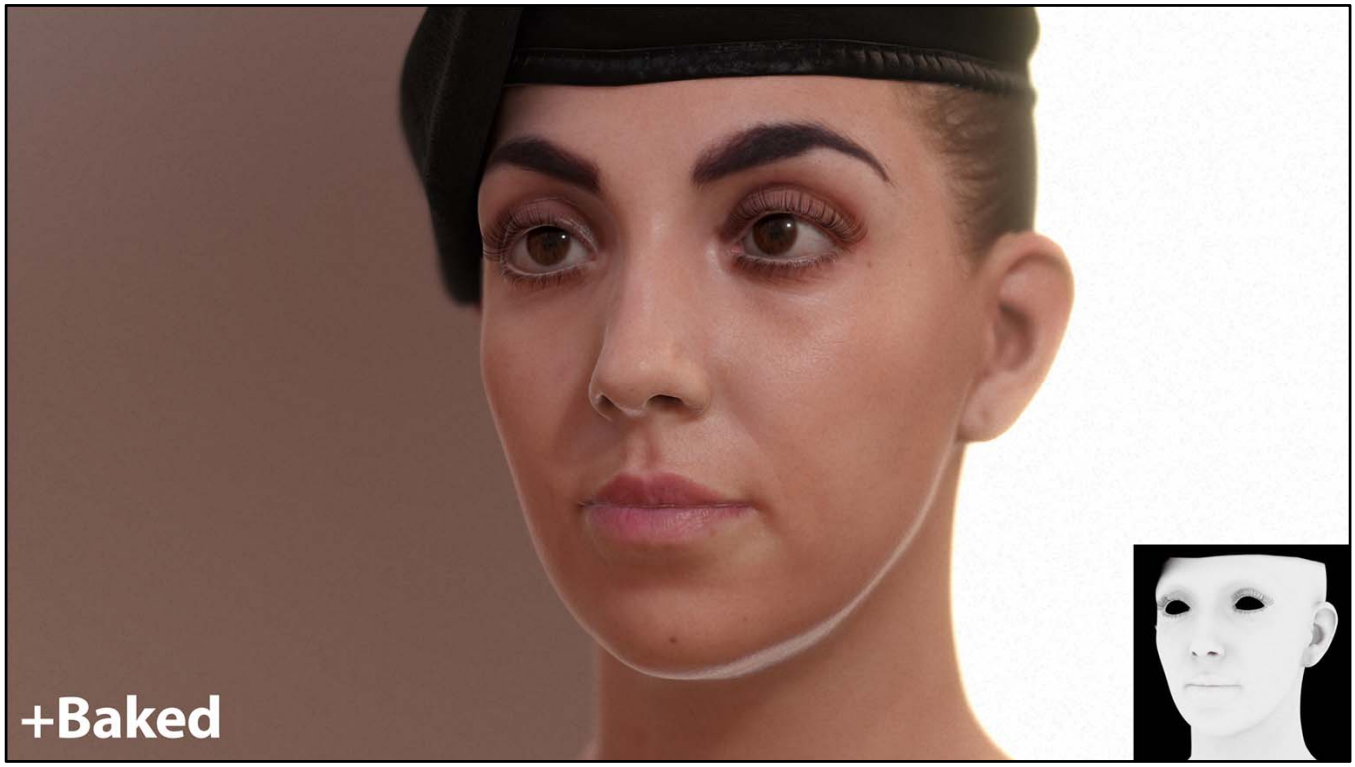


…and this is where we started.



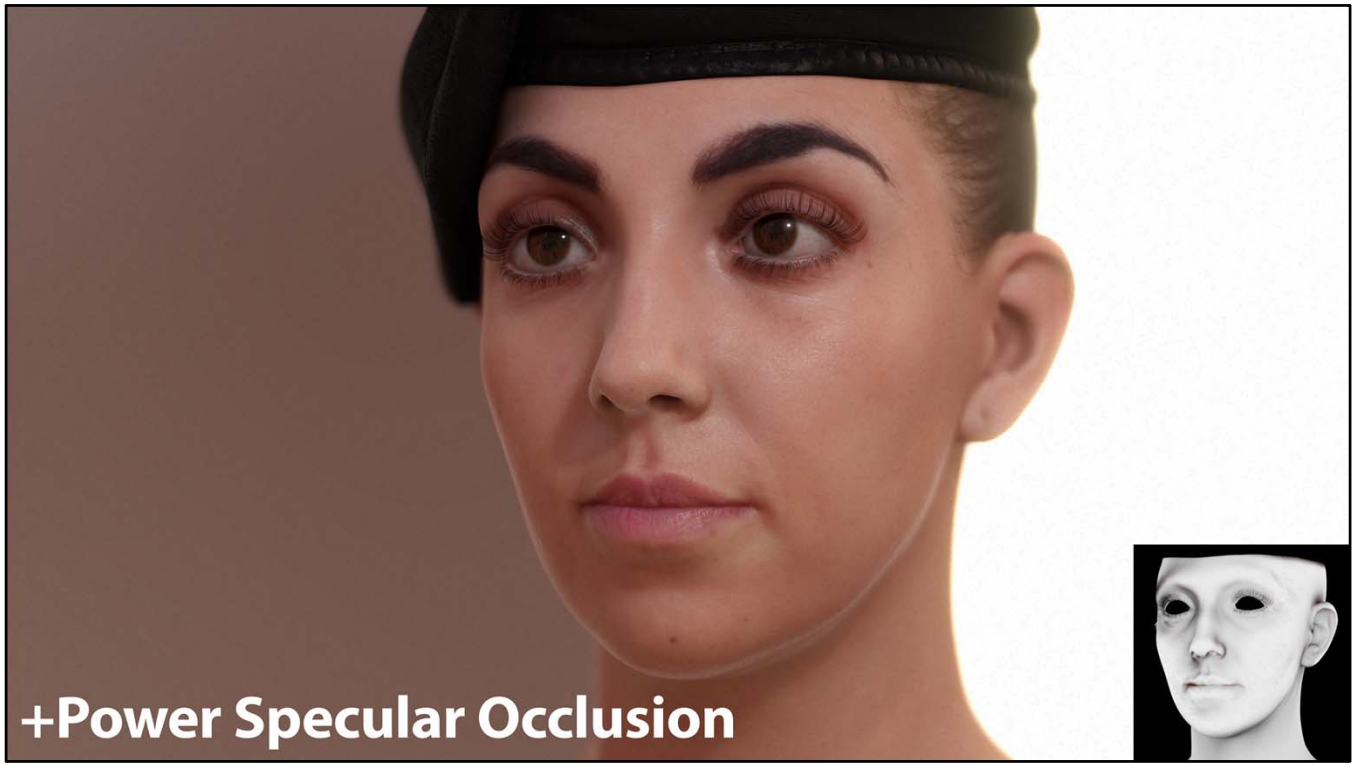
We can first add screen-space ambient occlusion.

It improves the situation, especially for the eyes, but it is still not enough.



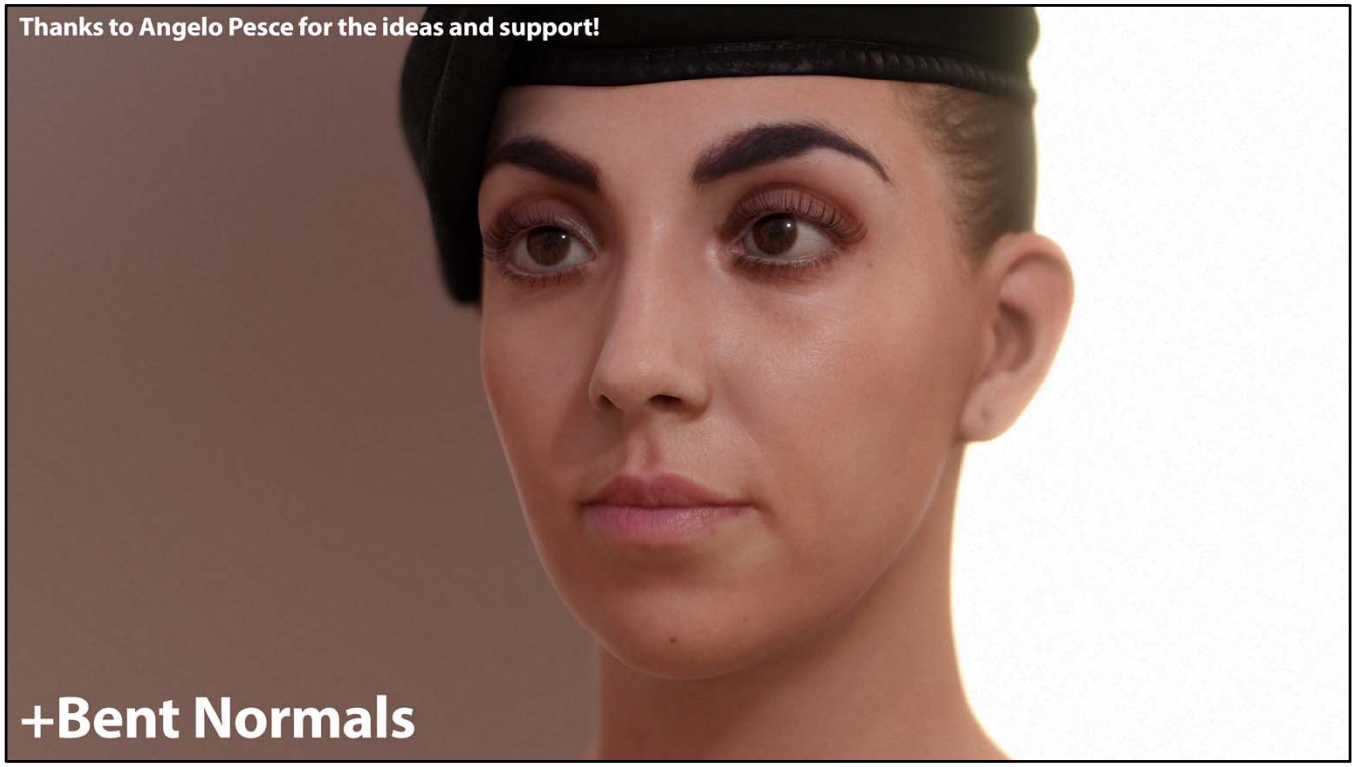
So, we added low-frequency baked occlusion for “static” facial features.

It’s starting to look better, but it’s still not there, especially because the Fresnel is producing very high specularity on the silhouette of the character.



Adding stronger specular occlusion helps a little bit with that…

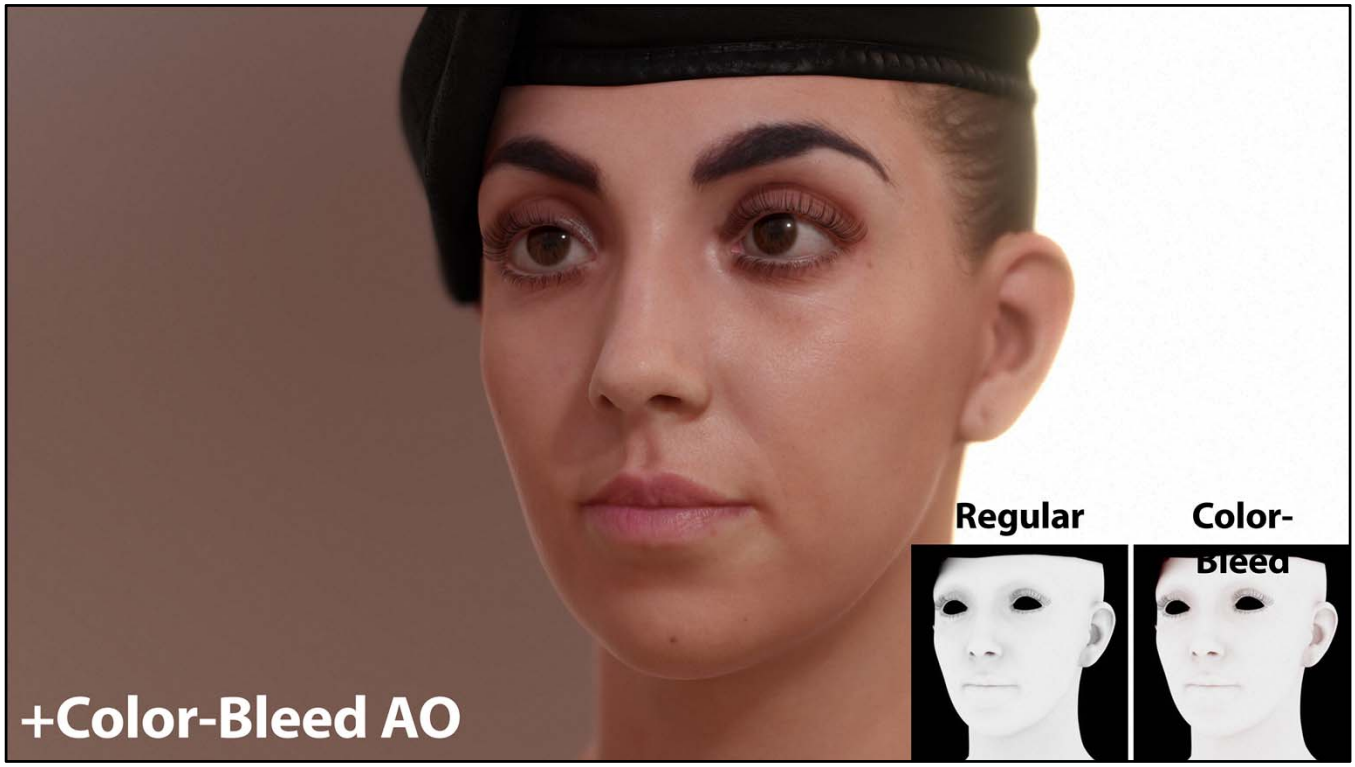
(More about this later on)



…and using bent normals completely eliminates the problem.

Bent normals are vectors that point to the direction of minimum occlusion (http://renderman.pixar.com/view/production-ready-global-illumination).

(More about this later on)



Finally, using “color-bleed” ambient occlusion enables colored shadowing and reflections, resulting in more coherent lighting on the face.

Combining SSAO with Baked AO

- **We use them both:**
  - **Baked AO takes care of long range, low frequency occlusion**
  - **SSAO takes care of mobile parts (which, coincidentally, are short-range for faces)**
- **Simple:**

```
finalAO = min(BakedAO, SSAO);
```

- **If animating, ensure that baked AO contains the minimum occlusion in the animation, so that SSAO can kick-in when mobile parts get more occlusion**

ACTIVISION BLIZZARD

Screen-space ambient occlusion didn't deliver the quality we needed, but mobile parts of the face
are difficult to represent using baked ambient occlusion.

So, we decided to use both solutions at the same time, to leverage their particular strengths.

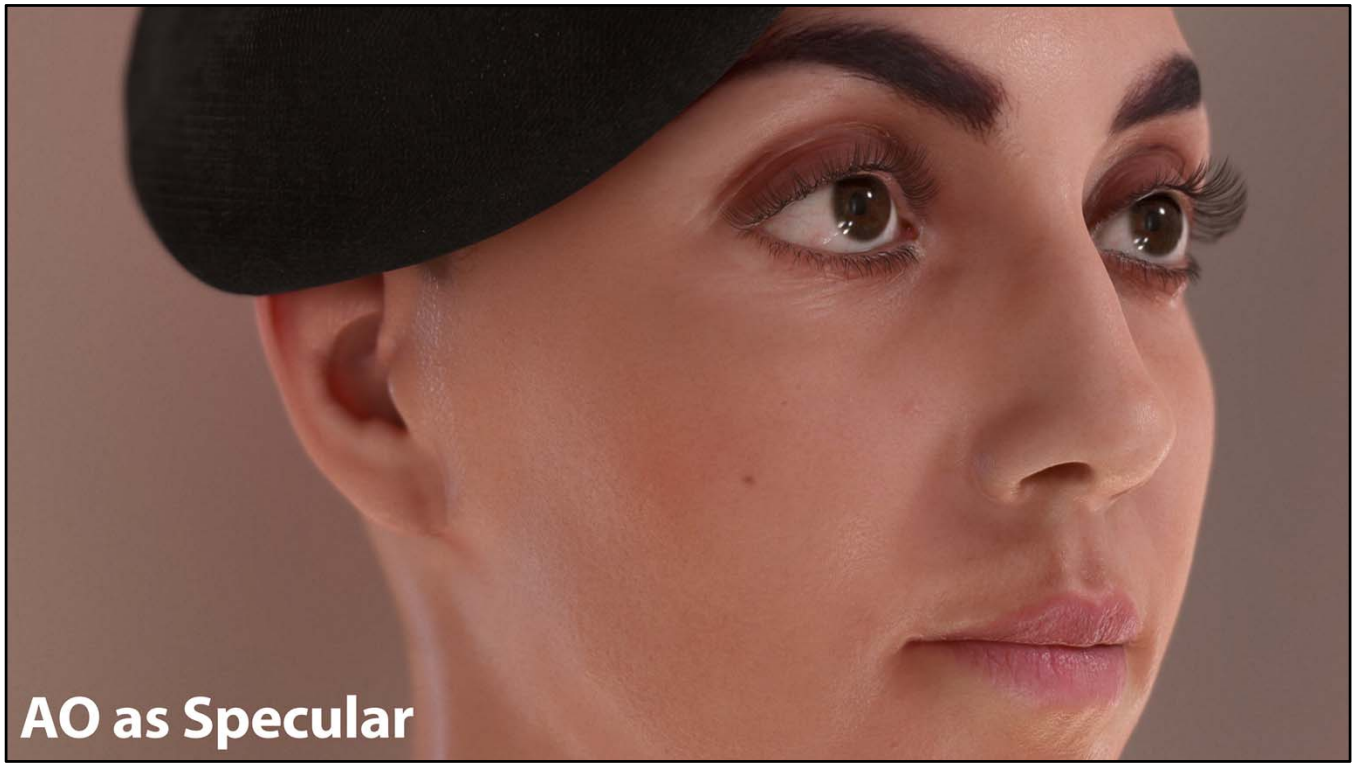
Power Specular AO

- **Can be empirically derived from ambient occlusion (see [Gotanda2011] *Real-time Physically Based Rendering - Implementation*)**
- **We tried to find a better solution for our assets**

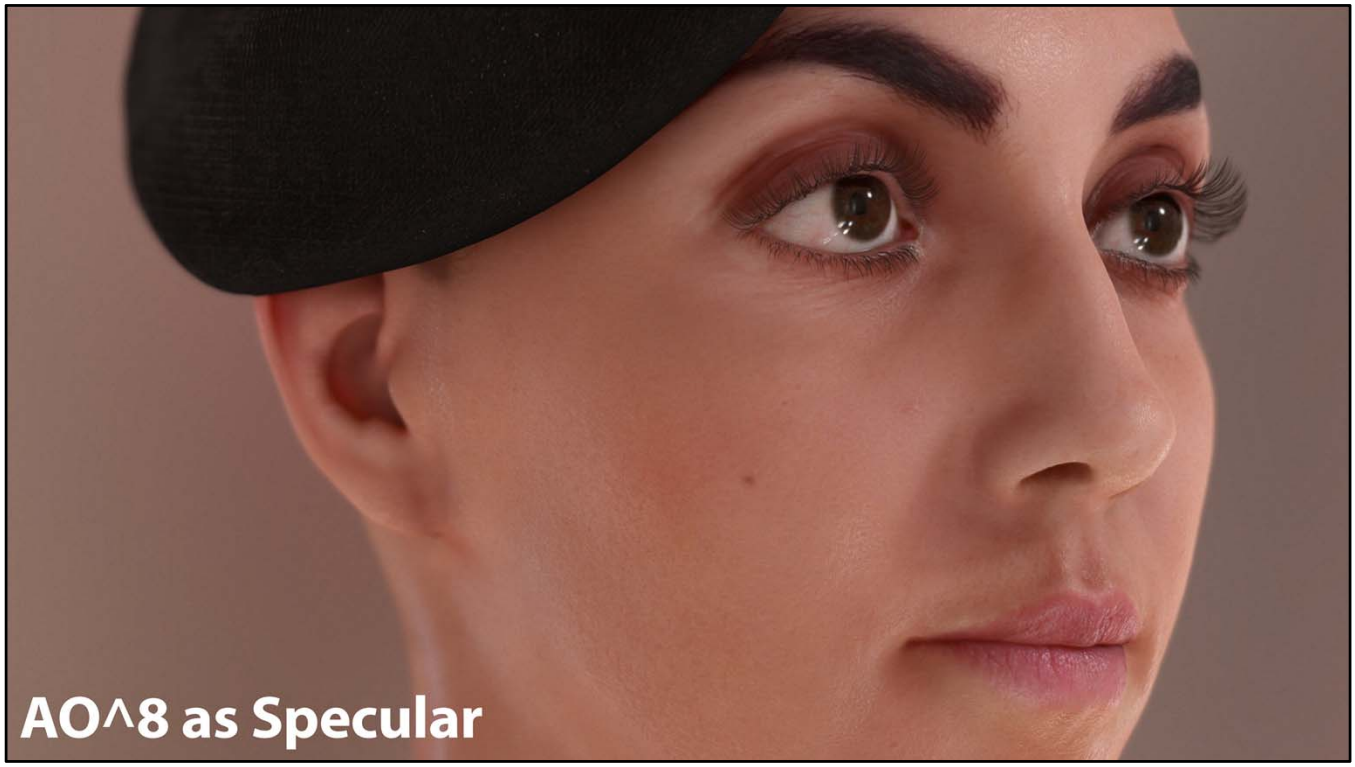
ACTIVISION BLIZZARD

There are existing empirical approaches that derive specular ambient occlusion from diffuse ambient occlusion.

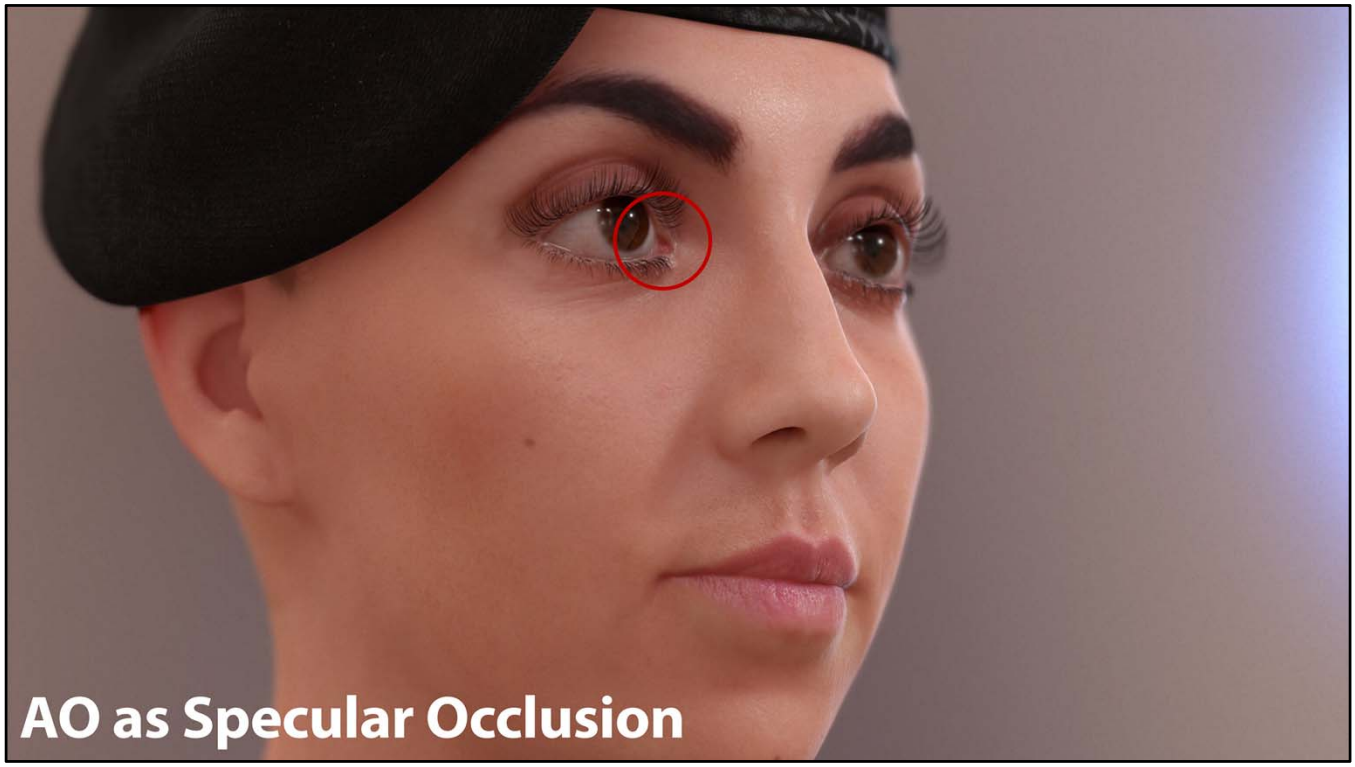
However, we wanted to develop a solution that worked better for our assets.



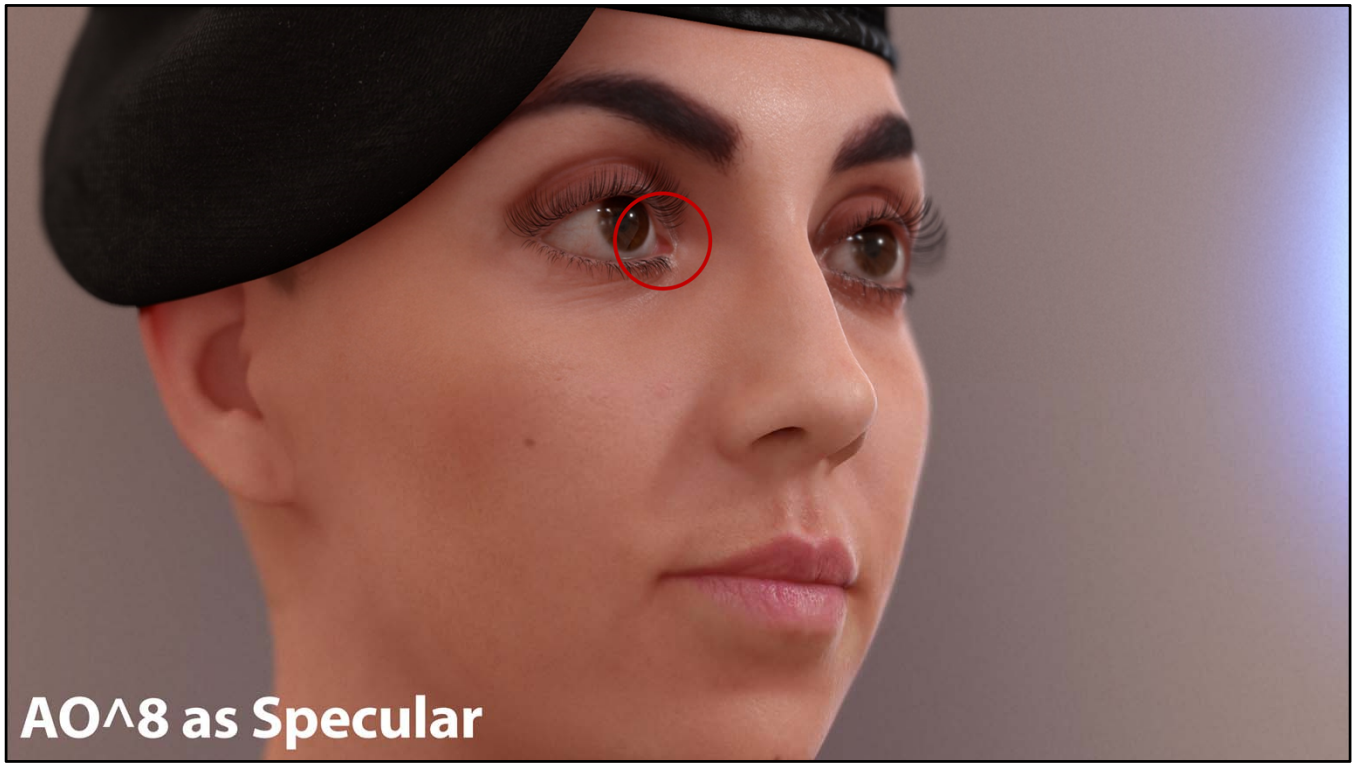
Here we see that simply using ambient occlusion as specular occlusion, doesn’t deliver.



Using ambient occlusion raised to a power works better…



…but if you look at the specular reflections at the caruncle in this other shot…



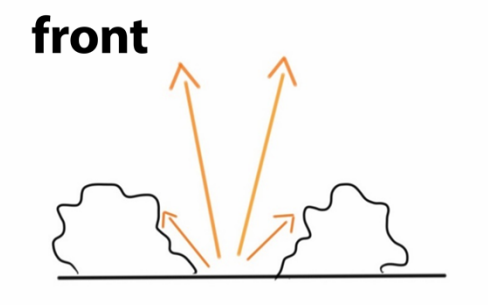
…you will notice that we have completely removed them using the power version of the occlusion.

And this happens for any view angle.

Specular Occlusion

Rationale

- **It should only occlude at grazing angles, not when viewed from the front**



Shader

```
const float SpecularPow = 8.0;
float NdotV = dot(normal, view);
float s = saturate(-0.3 + NdotV * NdotV);
return lerp(pow(ambientOcclusion, SpecularPow),
            1.0, s);
```

ACTIVISION | BLIZZARD

So, following Gotanda observations (see [Gotanda2011] *Real-time Physically Based Rendering - Implementation*),
it can be seen that there should only be specular occlusion
when looking at the surface from the side, but not when you look at it from the front.

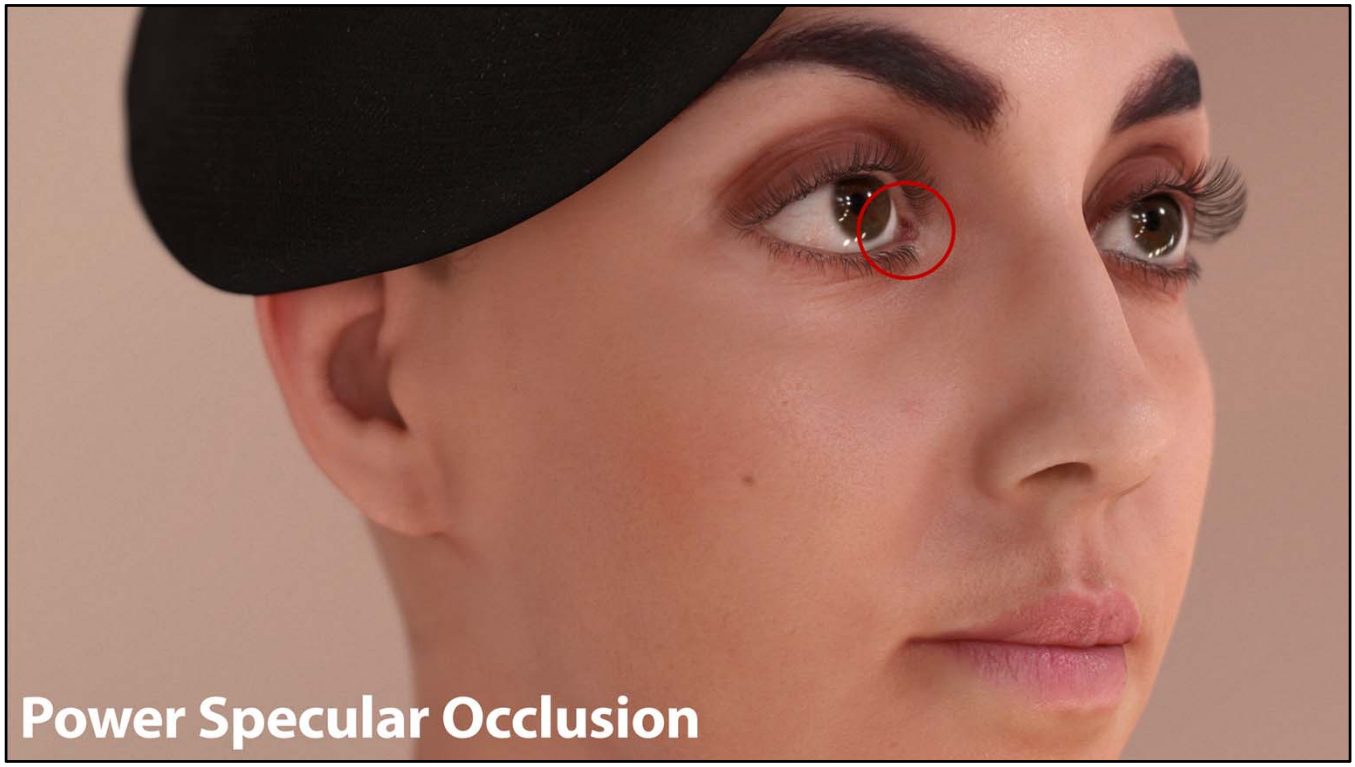
So, what we did is to apply occlusion only at grazing angles, using an empirical approach, that worked well for our assets.

This seems to contradict the previous observation that the cavity map should be removed at grazing angles.

However, note that in that case what we are doing is to take into account the fact that the pore is completely
flat in our models, and the occlusion that you would see from the side would be less than what you would see from
the front (where you can actually see the deeper part of the pore).

In other words, with the specular occlusion we want to model geometric occlusion, while in the cavity map case we
were trying to fix the fact that we are using flat textures for modeling pores.

Here you have an image using the simple pow approach…



…and here applying occlusion only at grazing angles.

Notice how the caruncle reflections are back.

Bent Normals

- **But unfortunately this is not enough for all of the lighting configurations**
- **To improve specular occlusion further, we baked bent normals per vertex**

Thanks to Angelo Pesce for the ideas and the support!

To improve things further, we also bake bent normals per vertex using Autodesk Turtle.

Then, for ambient lighting, we just replace the vertex normal N by the bent normal N', and use that to construct the TBN frame* (with an optional reorthonormalization).

Another possibility is to transform the normal map using the bent normals as a preprocessing step,
and use that for both ambient and procedural lighting. This is free at run-time, but not as accurate for procedural lighting.
(We tried this approach and found that sometimes procedural lights didn't look as good as using the regular normal map.)

* This is the matrix that's used to convert tangent-space normals to world space (for lighting).

The first thing we see if we apply regular ambient occlusion on skin is that it looks too gray.

This is because areas of occlusion are expected to receive indirect lighting from the occluding geometry,
which is obviously not gray, but red.

Saturated AO

- **Saturated AO [Driancourt2012] :**
  - **ao * pow(color, (1.0 + saturation) - saturation * ao);**
- **Regular AO:**
  - **ao * color**

ACTIVISION BLIZZARD

Artists are well aware of this, and have been using this saturation effect while authoring faces for quite some time.

But Square-Enix took this idea to the next level, and applied it directly in real-time.

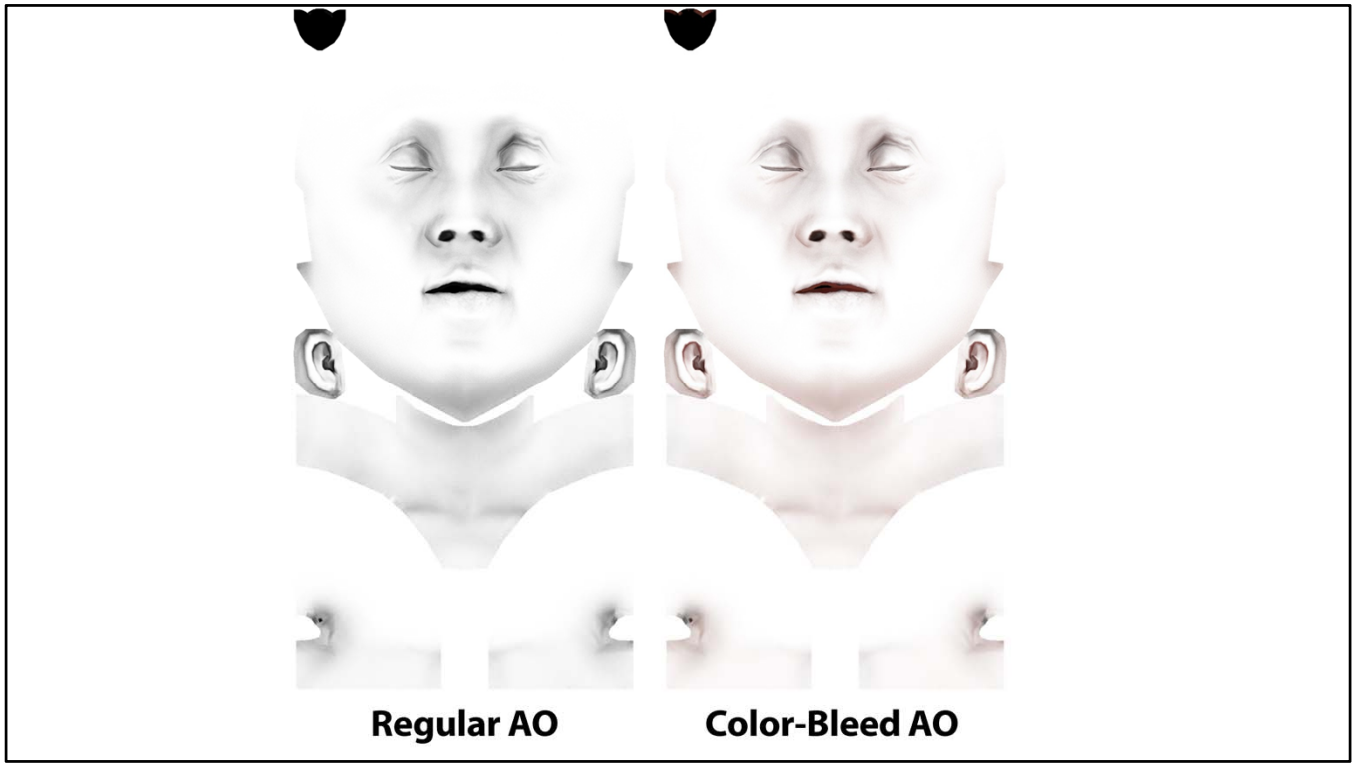
(=> http://game.watch.impress.co.jp/docs/news/20121129_575412.html)

Color-Bleed AO

- **Etienne Danvoye developed a multi-pass ambient occlusion renderer that takes the color of blockers into account**
- **Rays are weighted:**
  - **When a ray hits the sky:**
    - **On all passes: float3(1.0, 1.0, 1.0)**
  - **When a ray hits the geometry:**
    - **On first pass: float3(0.0, 0.0, 0.0)**
    - **On following passes: diffuse * previousAO**
- **Calculations and maps are in linear space**

ACTIVISION | BLIZZARD

We wanted to take this a step further, so we developed a multi-pass ambient occlusion renderer
that takes color bleeding into account.



In this slide you have a comparison between regular and color-bleed AO.

Areas of occlusion get colored by a reddish tone.

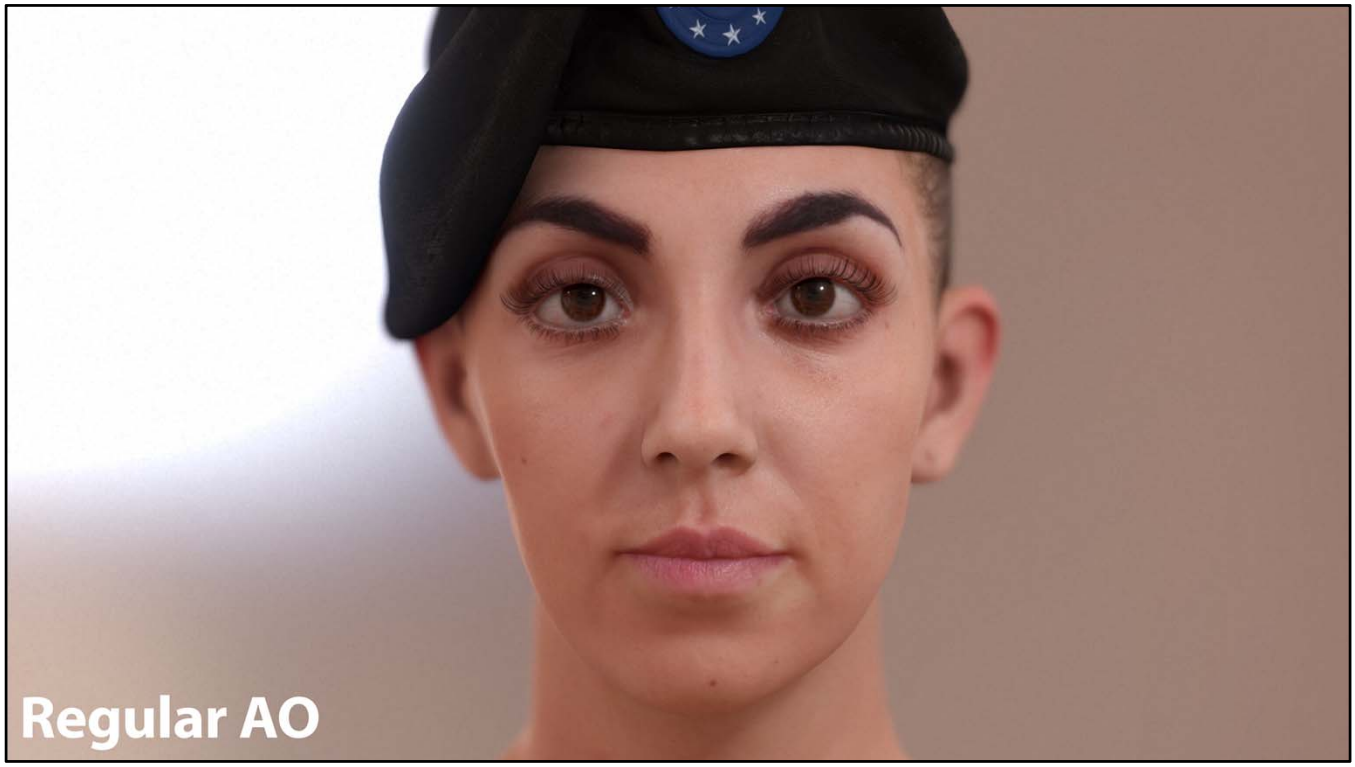
Color-Bleed AO

- **For character heads, the difference is subtle enough to be expressed with a gamma correction:**
  - **colorBleedAO = exp(regularAO,1.0 - strength)**
- **For Lauren:**
  - **strength = float3(0.4,0.15,0.13)**

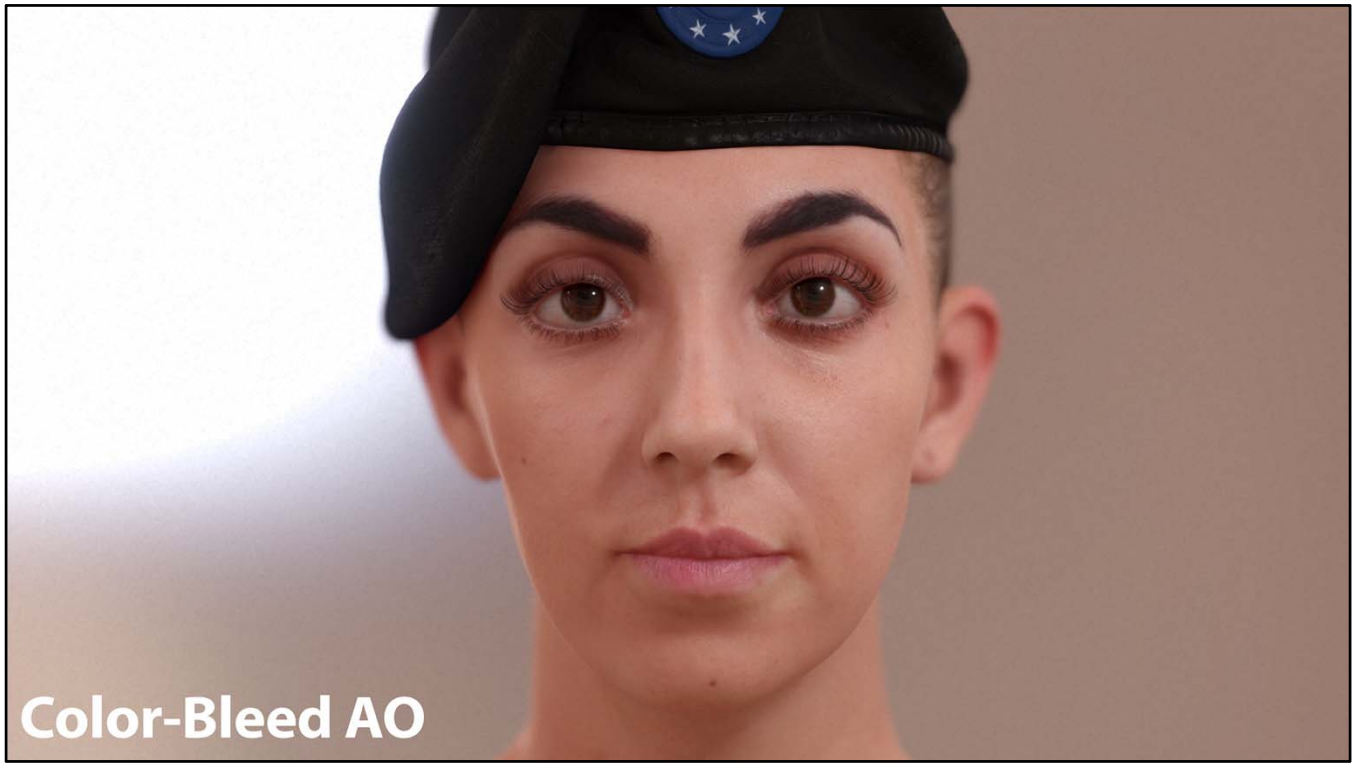
ACTIVISION | BLIZZARD

We found the difference to be subtle enough to be expressed with a gamma correction.

This means that we can avoid the lengthy multi-pass ambient occlusion calculations, as well as the possible additional memory cost of storing colored AO.



Here you have a render with regular AO…



…and here with color-bleed AO.

Notice how the lighting looks more natural, as we are now taking global illumination bounces into account.

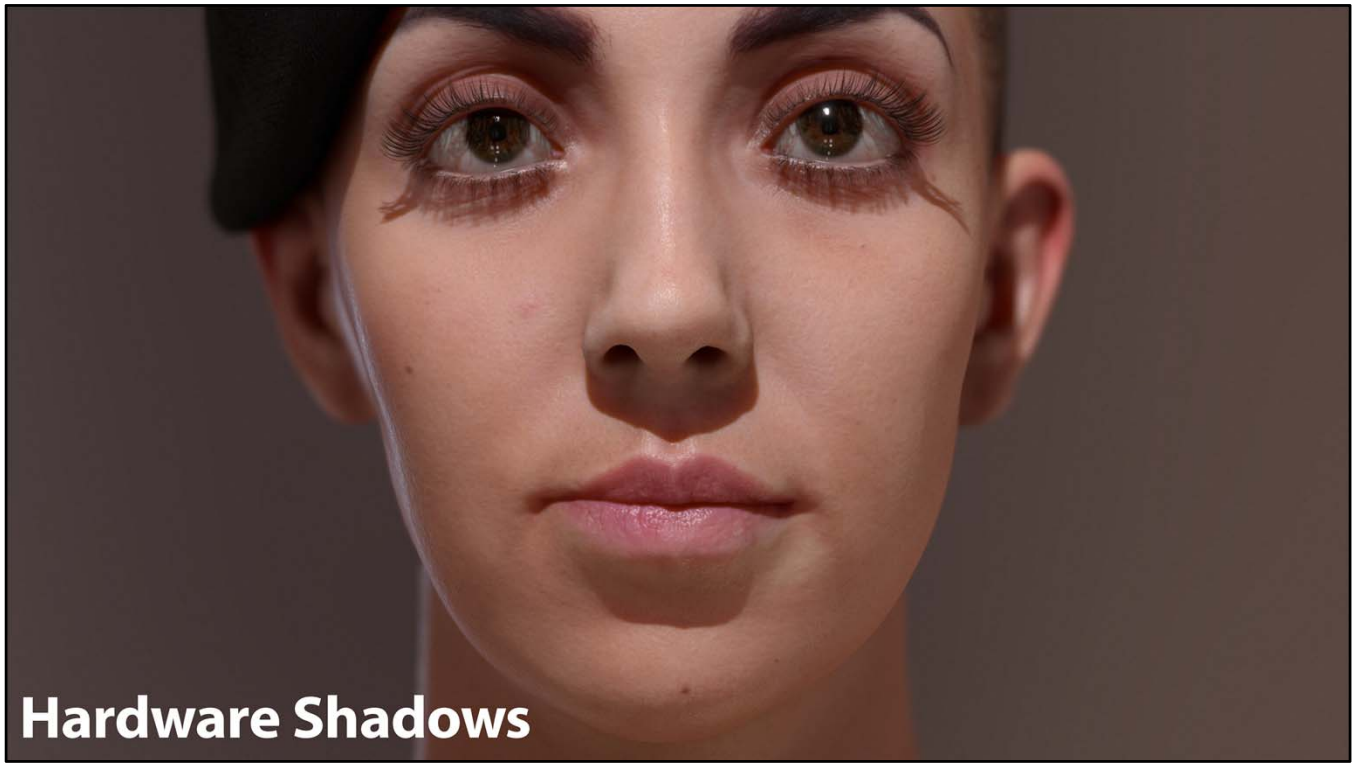
In the case of specular AO, this effect can help in faking secondary reflections.

Variable-Penumbra Shadow Maps

- **High quality shadows are relevant for character rendering**

ACTIVISION | BLIZZARD

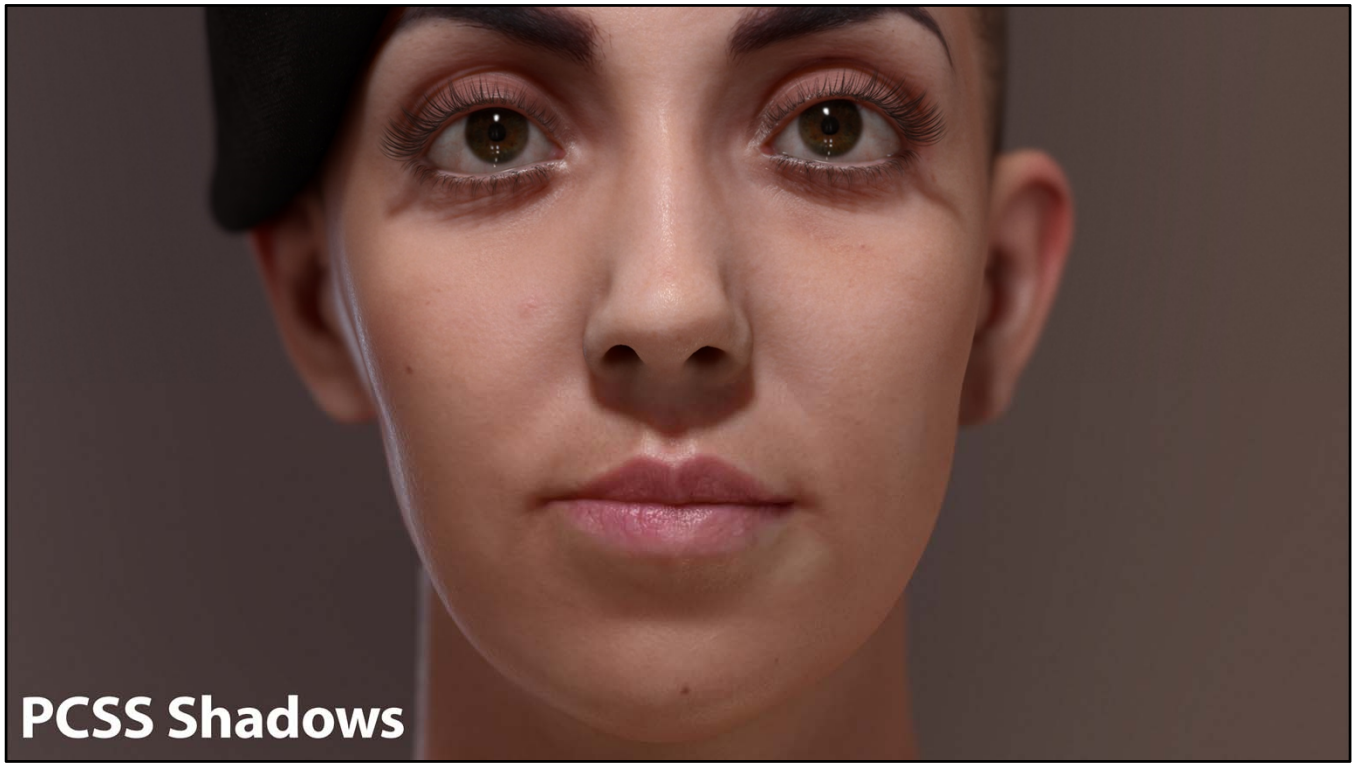
On a related topic, I've prepared a comparison to highlight the importance of proper shadowing.



Here we have regular shadow mapping.

Notice how unnatural the shadows look, especially for the eyelashes.

Using soft shadows improves the situation, but it almost completely removes contact shadows.



Notice how using a variable penumbra (via PCSS) enables soft shadows, while still rendering
proper contact shadows. This is especially important for eyelashes.

The main drawback is that you need really high-resolution shadow maps to achieve this quality,
as faraway lights can alias considerably (since the filtering kernel will be small).
So, you may have to limit this technique to cinematic sequences.

Subsurface Scattering
Reflectance Part

ACTIVISION BLIZZARD

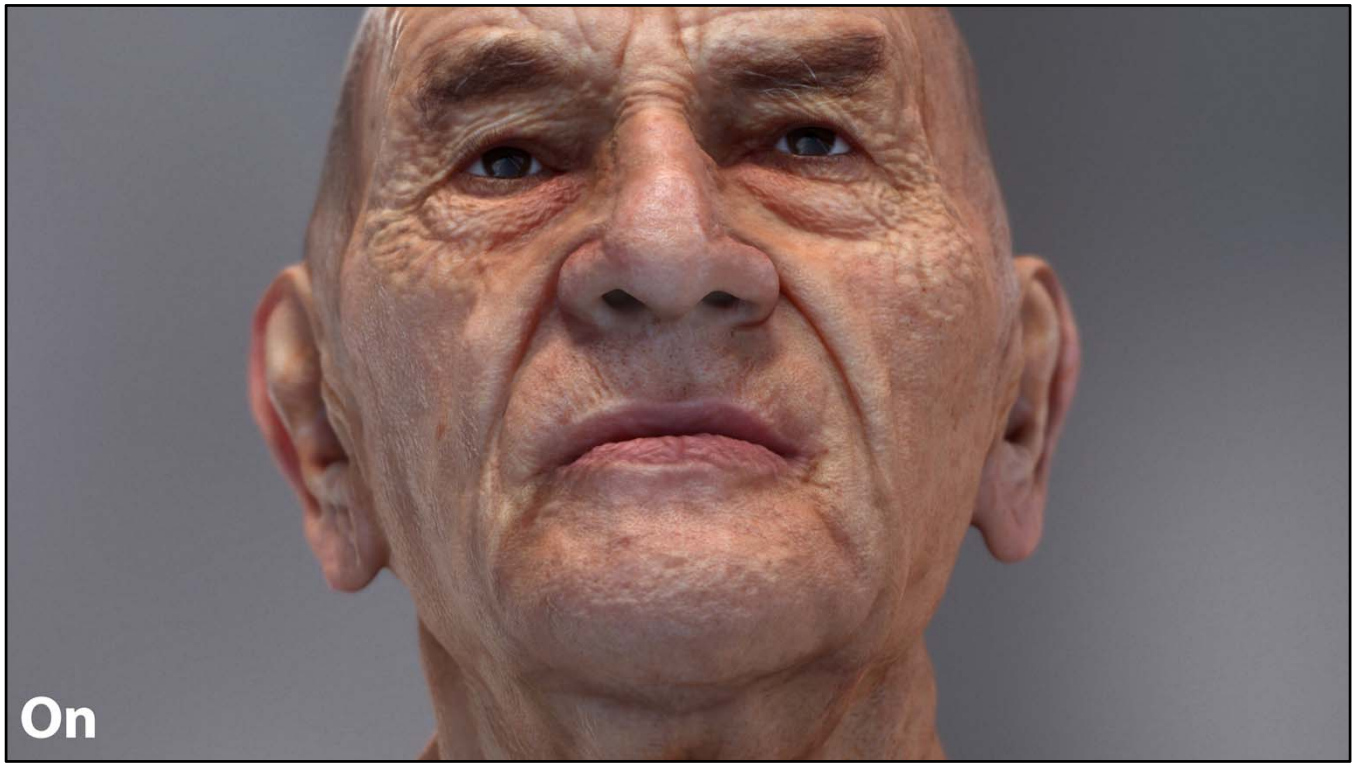
So, now that we've finished with specular reflections, I'll dive into subsurface scattering…

We can divide this part into two sections.

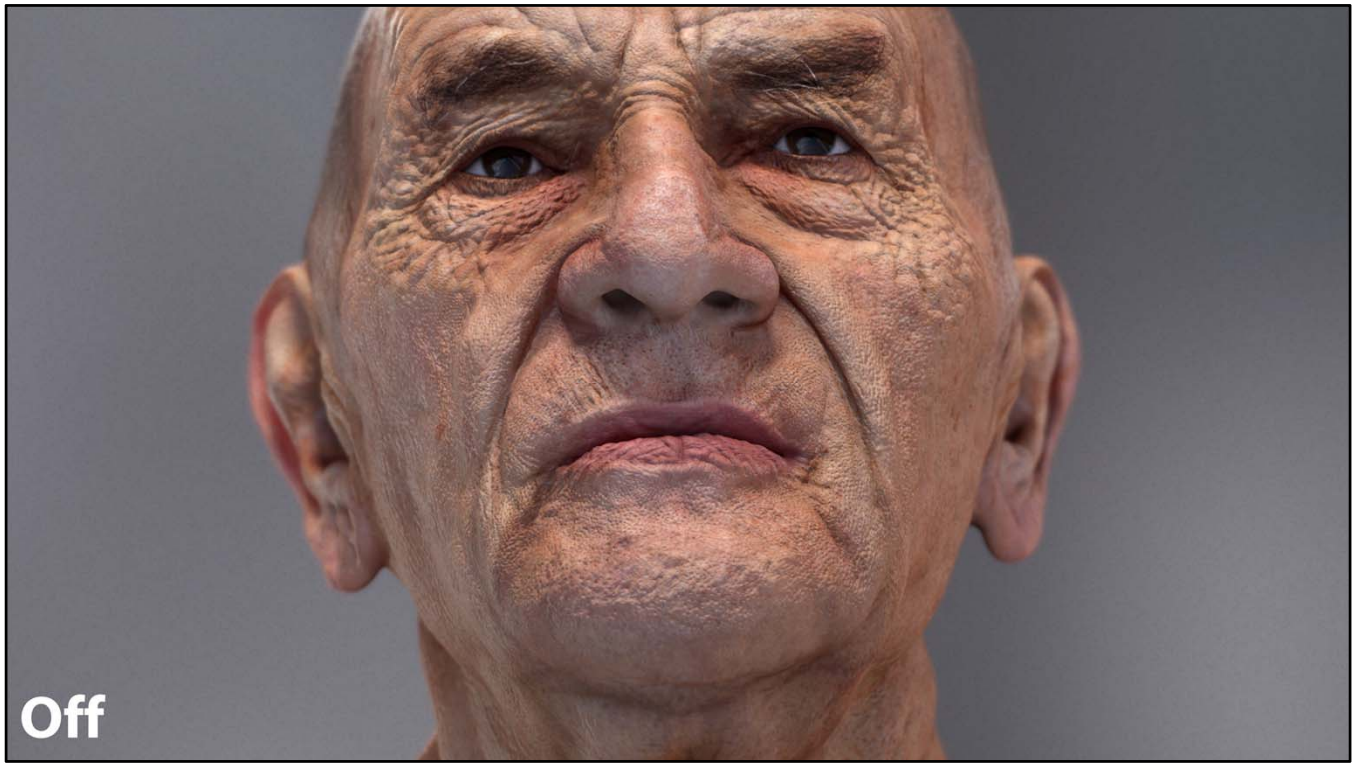
Reflectance subsurface scattering, which smoothes the lighting on the surface of the skin.

And transmittance subsurface scattering, which allows light to travel through thin slabs like ears or nostrils.

First we will look at the reflectance smoothing part.

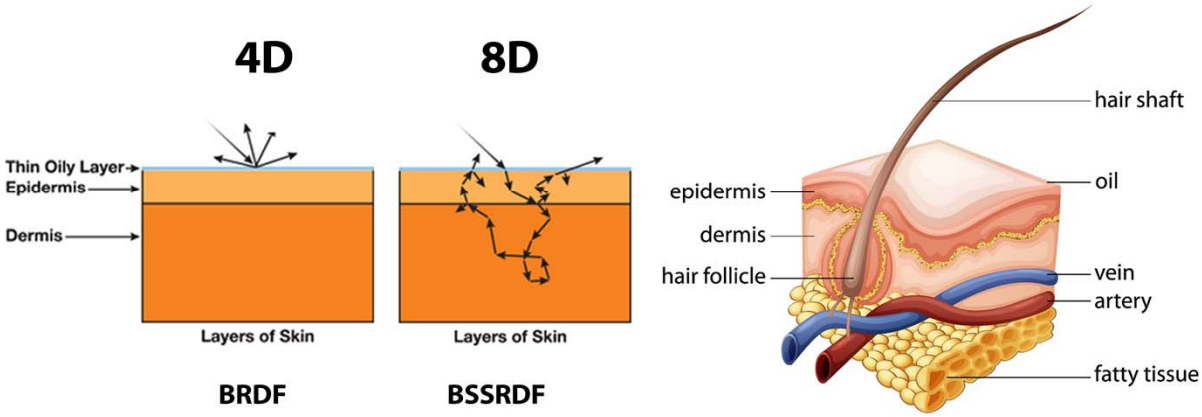


…which is, in my opinion, one the most important parts for rendering realistic faces.



Without it, rendering hard or wrinkled faces is extremely difficult.

BRDF vs. BSSRDF



From [Eon2007] *Advanced Techniques for Realistic Real-Time Skin Rendering*

A lot of you will already know this, but I'd like to give it a quick review.

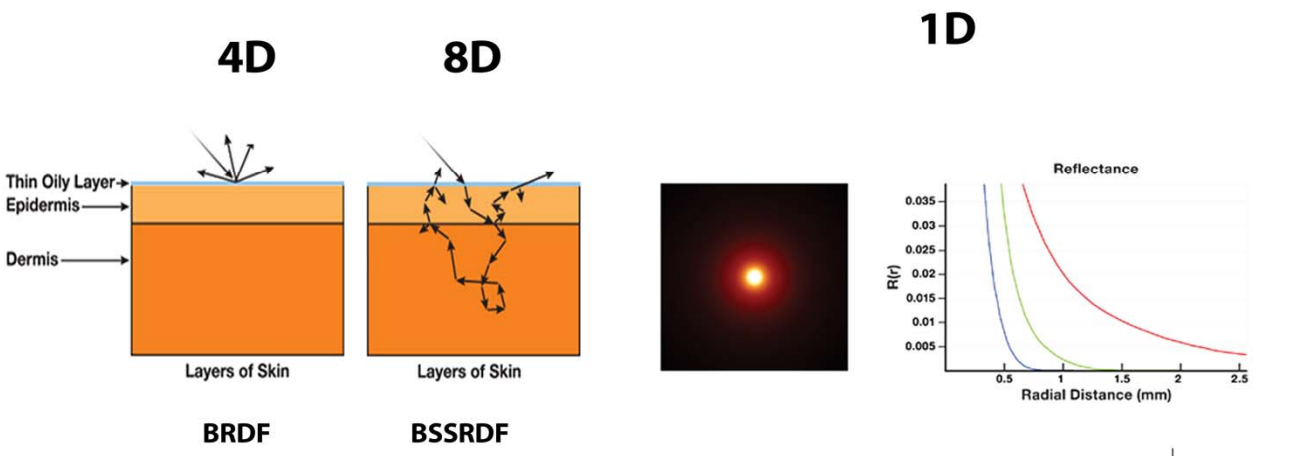
When rendering skin using conventional techniques we make the assumption that light scatters at the surface,
so it exits at the same point that it enters.

This model, called a BRDF, works really well for simple materials like stone, or plastic.

However, when rendering complex translucent materials like skin, we have to use the B**SS**RDF model,
where light penetrates the surface, travels beneath the object and exits at adjacent points around the incident point.

Notice that the BRDF is a four dimensional function, while the BSSRDF is an eight dimensional one, which is obviously very costly.

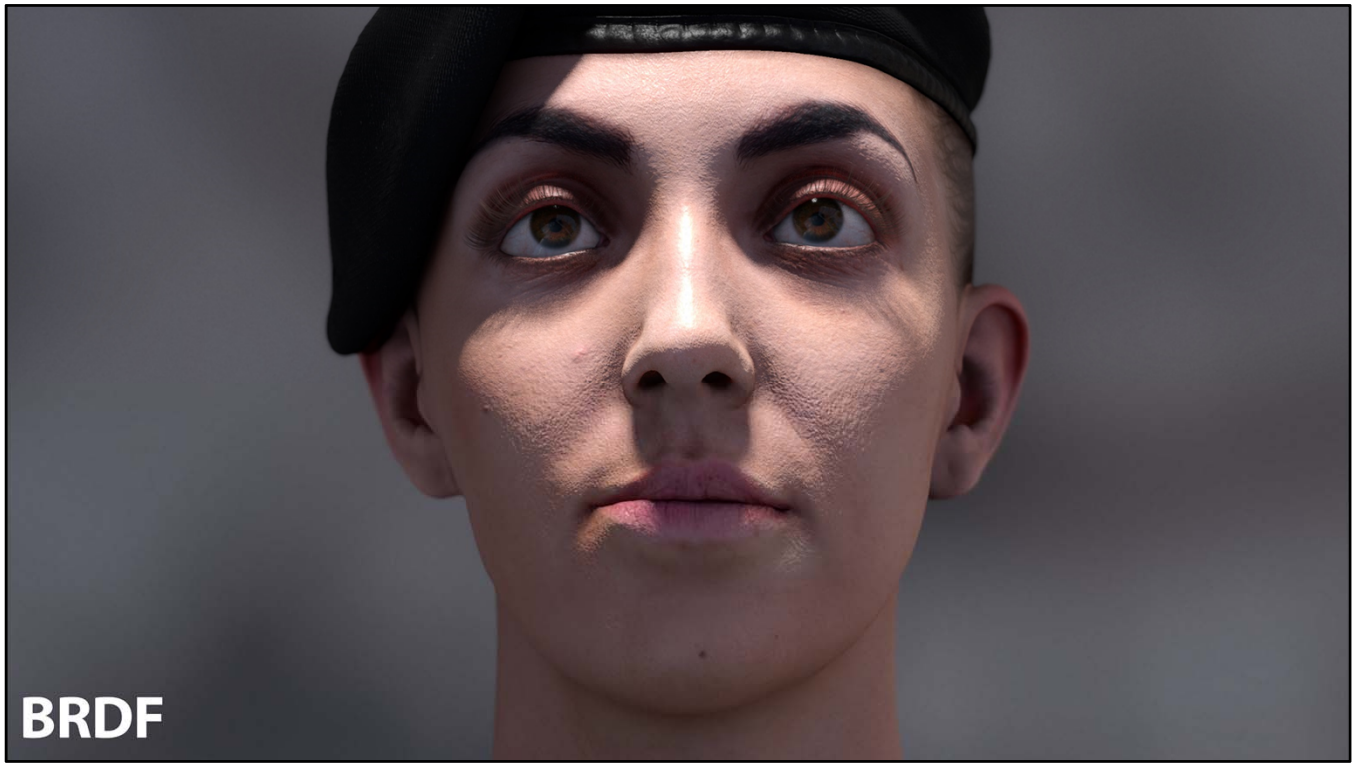
Diffusion profiles



From [Eon2007] *Advanced Techniques for Realistic Real-Time Skin Rendering*

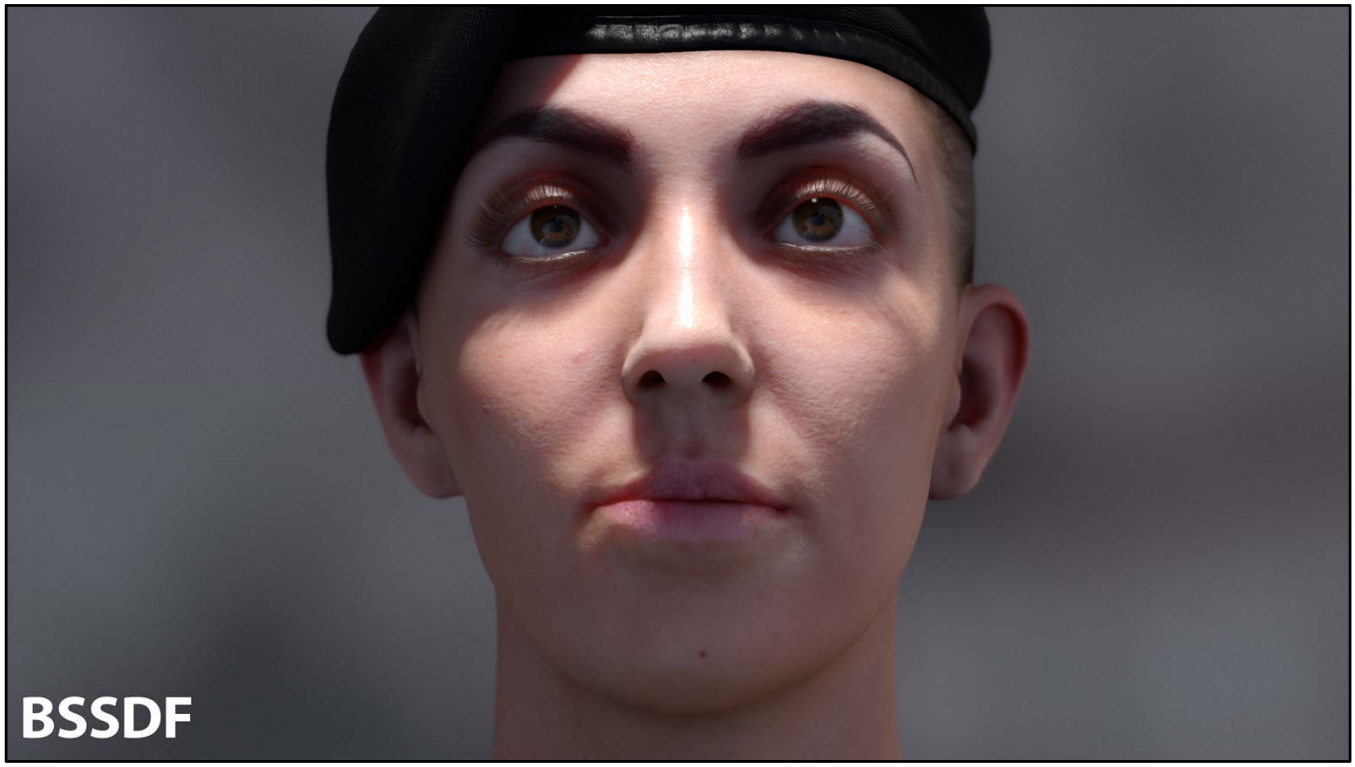
Since 8D sounds like too many dimensions, the problem is usually simplified by using the diffusion approximation,
which reduces the description to one single parameter: the distance to the incident point.

Notice how light intensity obviously decays with distance.



Here we have another comparison using a harsh lighting setup.

Notice the importance of subsurface scattering…



…for bringing the character to life.

Note that it’s in these harsh lighting situations that subsurface scattering is truly important.

Some Previous Work

- **Original NVIDIA SSS technique worked in texture space, with 5 Gaussian blurs:**
  - **[Eon2007] *Advanced Techniques for Realistic Real-Time Skin Rendering***
- **We shifted it to screen-space:**
  - **[Jimenez2009] *Screen-Space Perceptual Rendering of Human Skin***
- **Then, we reduced it to 2 blurs, in our separable SSS technique:**
  - **[Jimenez2012] *Separable Subsurface Scattering and Photorealistic Eyes Rendering***

ACTIVISION | BLIZZARD

For me, subsurface scattering in real-time started with the awesome work that Eugene d'Eon did for NVIDIA, back in 2007.

Unfortunately, it took all the processing power of the fastest machine at the time to render a single head.
So, we shifted the calculations to screen space, and found the differences to be relatively small.
Finally, in 2012, we found a way to reduce the number of Gaussians to 2 separable blurs, while still yielding realistic results.

For more details, we refer you to [Jimenez2012] Separable Subsurface Scattering and Photorealistic Eyes Rendering.
(Search for the slides here: http://advances.realtimerendering.com/s2012/index.html)

Improving the Performance

```
float4 colorM = colorTex.Sample(PointSampler, texcoord);

if (initStencil)
    if (colorM.a == 0.0) discard;

float depthM = depthTex.Sample(PointSampler, texcoord);
float2 finalStep = colorM.a * step / depthM;

float4 colorBlurred = colorM;
colorBlurred.rgb *= kernel[0].rgb;

[unroll]
for (int i = 1; i < N_SAMPLES; i++) {
    float2 offset = texcoord + kernel[i].a * finalStep;
    float4 color = colorTex.Sample(LinearSampler, offset);

    #if FOLLOW_SURFACE == 1
    float depth = depthTex.Sample(LinearSampler, offset);
    float s = saturate(0.5 * width * abs(depthM - depth));
    color.rgb = lerp(color.rgb, colorM.rgb, s);
    #endif

    colorBlurred.rgb += kernel[i].rgb * color.rgb;
}

return colorBlurred;
```

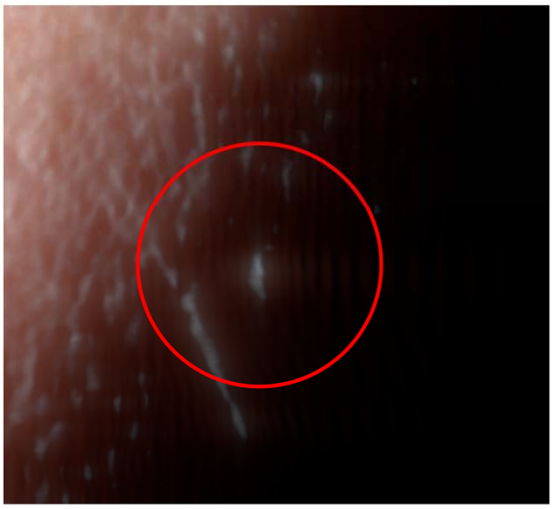
From [Jimenez2012] *Separable Subsurface Scattering and Photorealistic Eyes Rendering*

Here we have the separable subsurface scattering technique we presented past year.

The first improvement we have done is to store 1/depth in the alpha channel of the color buffer,
which helps to reduce memory bandwidth (lines in orange can then merge to a single one).

Improving the Quality

- **In the separable SSS technique, we're using a non-symmetrical separable kernel to approximate a radially symmetric non-separable kernel**
- **This can lead to artifacts**



The core idea behind the separable SSS technique is to approximate a radially symmetric non-separable filter kernel
using a non-symmetrical separable filter kernel.

In general, it looks good, but sometimes it can lead to artefacts, or degrade the quality.

You can see vertical/horizontal patterns in the image on the right, which appear as result of using a non-symmetrical filter.

Per Pixel Random Filter Axis

- **In [Huang2011] *Separable Approximation of Ambient Occlusion*, a per-pixel jitter is applied to rotate the axis for SSAO searches**

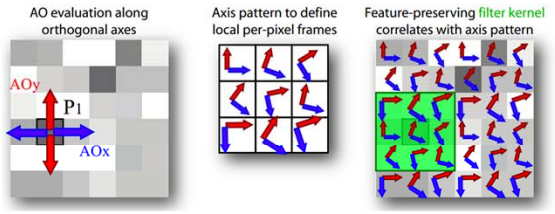
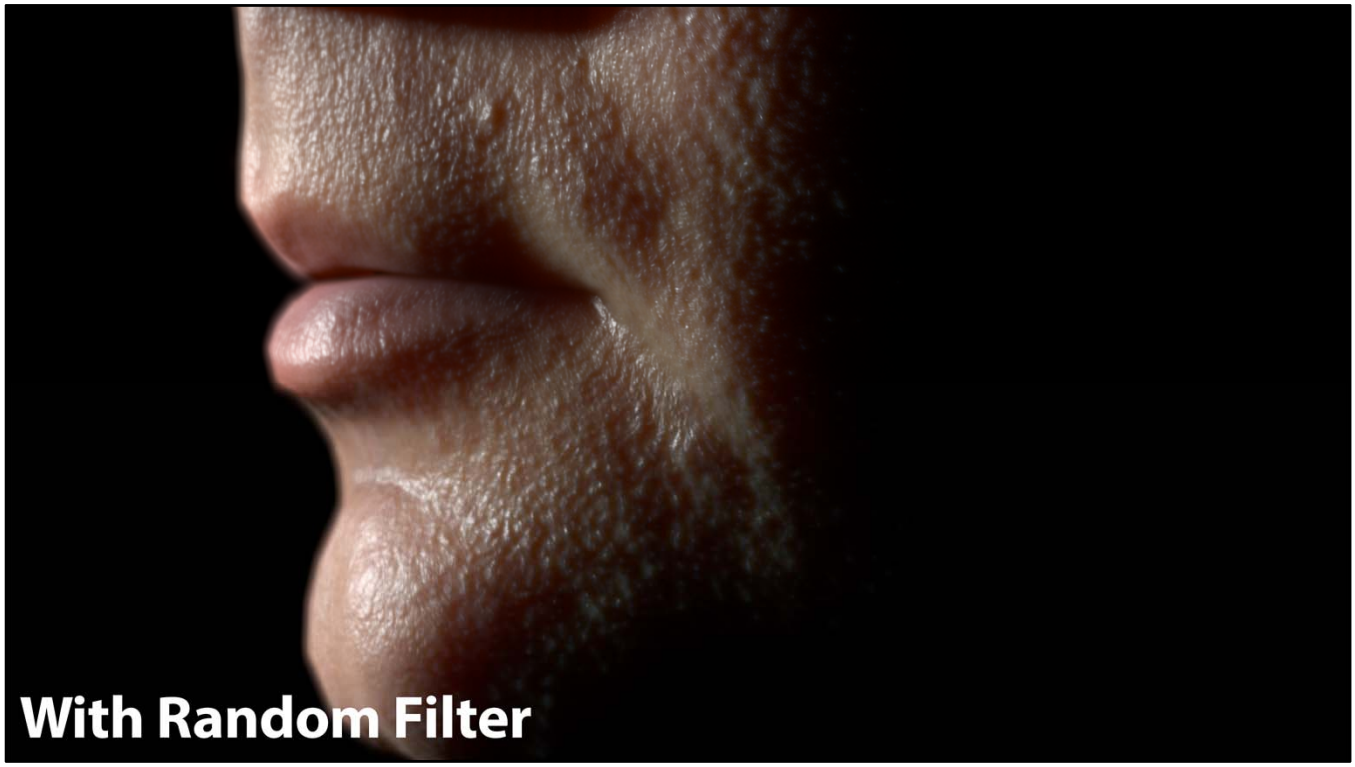


Figure 1: ***Principle:*** *our separable approximation combines two 1D evaluations in orthogonal screen directions (e.g. x,y axis, left). By changing coordinate frames per pixel, we can derive a stochastically valid approximation of the 2D occlusion by combining the result of neighboring samples.*

In the Separable Approximation of Ambient Occlusion paper, a solution is presented to address a very similar problem – in this case for AO sampling.

They do two blocker searches in two orthogonal axes which are rotated per pixel.
We can use the same idea for screen-space subsurface scattering.

Here you have our old separable approach…



…and here the jittered one, for the same sample count.

Both images are using exactly the same parameters (aside from the jitter).

(But keep in mind that the comparison isn't completely fair, as the non-jittered approach could be artistically tweaked to achieve better results.)

Performance

- **Depends on screen-space coverage of skin pixels**
- **Very fast if far way, potentially very slow in tight macro shots (at high resolutions)**
  - **Probably due to poor texture cache utilization, caused by the per-pixel jittering and the potentially big filter size**

ACTIVISION | BLIZZARD

With regard to performance, it is still very fast in the general case, just a little bit slower than our older separable SSS technique.

However, randomly sampling per pixel can have a huge performance impact for cinematic shots at high resolutions.

Hybrid Jittered/Separable Blur

```
...

float2 jitter = samplingNoiseTex.Sample(WrapSampler, svPosition * (1.0 / 64.0));
float2x2 rotationMatrix = float2x2(jitter.x, jitter.y, -jitter.y, jitter.x);
float2x2 identityMatrix = float2x2(1.0, 0.0, 0.0, 1.0);

for (int i = 1; i < SSSS_BLUR_SAMPLE_COUNT; i++) {
    float2 offset = kernel[i].a * finalStep;

    // We jitter the kernel axes randomly near the center tap:
    float dist = saturate(SSSS_BLUR_JITTERED_TO_SEPARABLE_FACTOR * abs(kernel[i].a));
    offset = mul(offset, lerp(rotationMatrix, identityMatrix, dist * dist));

    float4 color = colorTex.Sample(LinearSampler, texcoord +
                                                  float2(aspectRatio, 1.0) * offset);

    ...

    colorBlurred.rgb += kernel[i].rgb * color.rgb;
}
```

ACTIVISION BLIZZARD

To prevent performance problems, we lerp between a randomly-rotated axes and the original one using the distance to the filter center,
which helps to reduce cache problems.

Another option could be to create a mip-map chain and access the levels depending on the distance to the center of the filter,
which can be a good approach if you can leverage the chain for other uses.

Performance

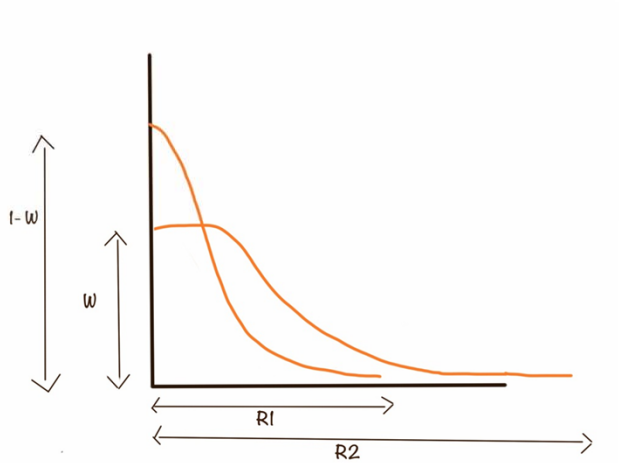
- **Compared with the original SSS technique, for a portrait shot at 1080p:**
  - **The pure jittered approach is 47% slower**
  - **The hybrid approach is 14% slower**
- **For far away setups, both the pure and hybrid jitter approaches have negligible performance slowdown**

ACTIVISION | BLIZZARD

Note that because the kernel is jittered, we can also lower the sample count more aggressively and still get visually appealing results.

User Interface

- We wanted a customizable SSS filter
- So, we tried to break the original kernel into pieces:
  - A narrow Gaussian
  - A wide Gaussian
  - A blend weight between them



ACTIVISION | BLIZZARD

We also created a better user interface for modeling subsurface scattering, using the concept of near and far scattering.

Basically, it is based on separating the lighting that scatters near the incident point, and also far away from it.

Coincidentally, there is a symmetry between this and the two lobe specular model:

They both contain a narrow and a wide shape, and a blend is performed between them.
It could be that this is something that happens often in nature.

User Interface

- **Three parameters:**
  - **Wide Gaussian RGB radius in world space**
  - **Narrow Gaussian radius in world space**
  - **RGB blend weight between the two Gaussians**
- **Visually tweaking these parameters, we were able to match the three-layer multipole SSS profile ([Eon2007] *Advanced Techniques for Realistic Real-Time Skin Rendering*)**
- **It is simple:**

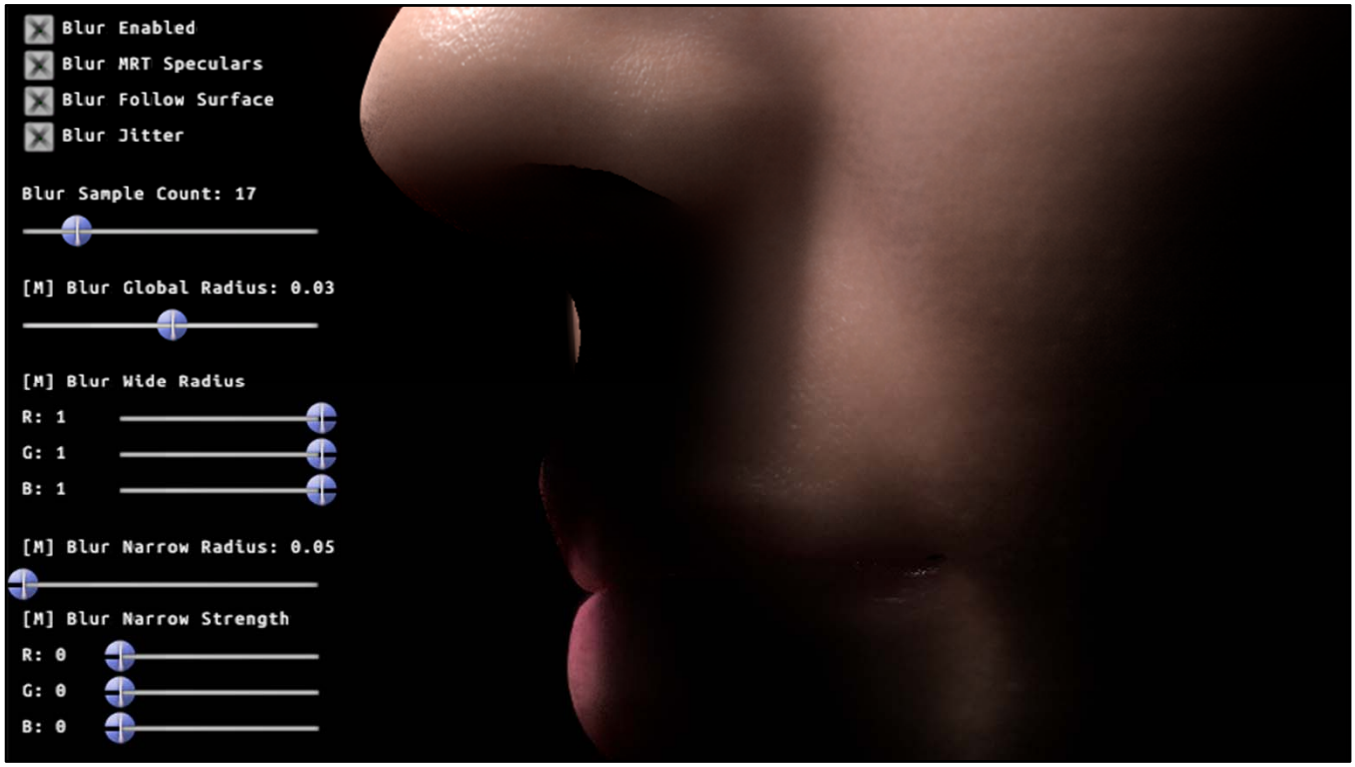
```
D3DXVECTOR3 SeparableSSS::profile(const D3DXVECTOR3 &radiusRGB,
                                  float narrowRadius,
                                  const D3DXVECTOR3 &narrowStrength,
                                  float r) {
    D3DXVECTOR3 wideVariance = radiusRGB * radiusRGB;
    D3DXVECTOR3 narrowVariance = (narrowRadius * radiusRGB) * (narrowRadius * radiusRGB);
    return gaussian(wideVariance,   r) * (D3DXVECTOR3(1.0f, 1.0f, 1.0f) - narrowStrength) +
           gaussian(narrowVariance, r) * narrowStrength;
}
```

The main controls are:
- The wide Gaussian radius in world space, for each color channel.
- The narrow Gaussian radius, also in world space.
- And finally, a color that specifies the blend weight between both Gaussians.

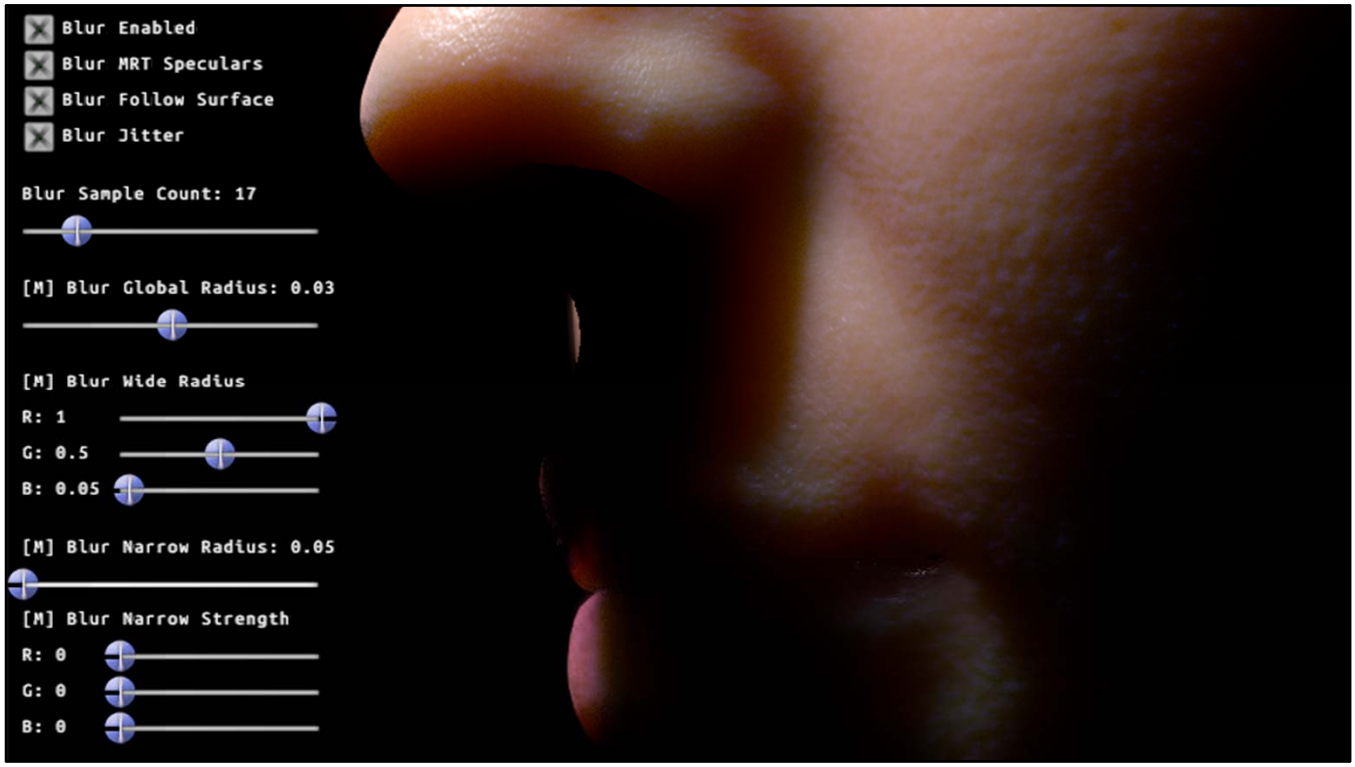
By manually tweaking these parameters, we matched the three-layer multipole SSS profile described in the original NVIDIA work.

You may want to compare the code with the profile used in our older approach (released during my PhD):
https://github.com/iryoku/separable-sss/blob/master/Demo/Code/SeparableSSS.cpp

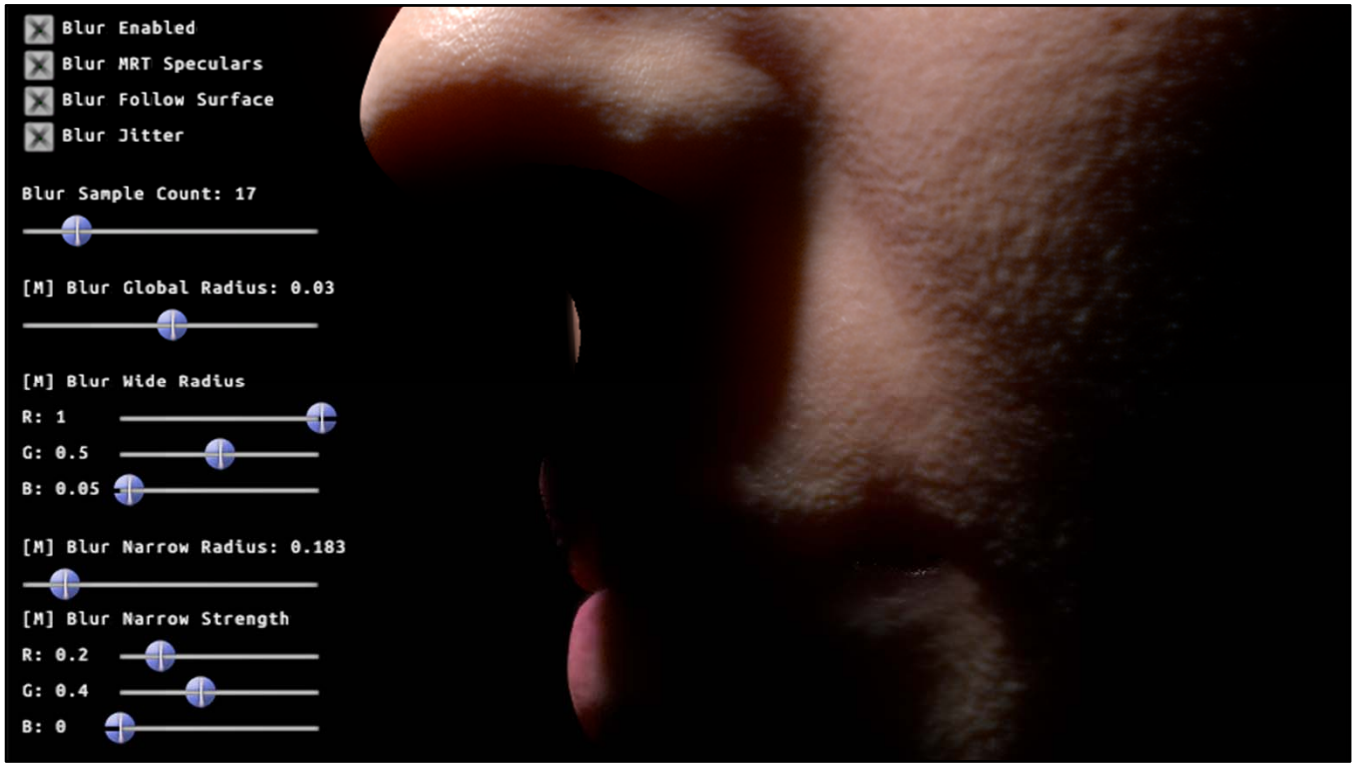
To create a subsurface scattering profile, you first adjust how far the light reaches in world space…



…then you change the color of the far scattering…



…and finally the color and radius of near scattering…

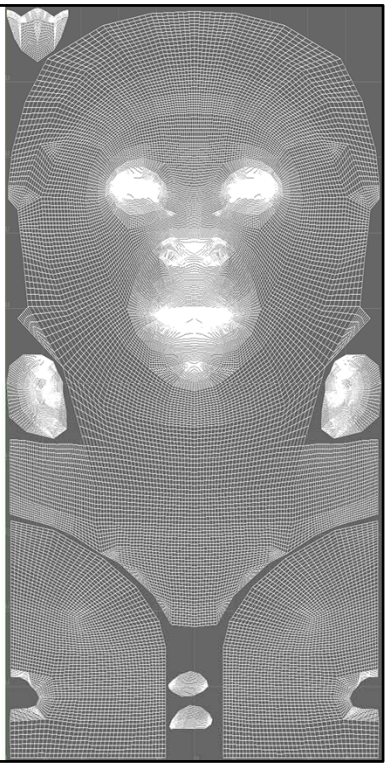


…which allows us to maintain bump detail.

In particular, we found the parameters shown in this screenshot to work well for various types of skin.

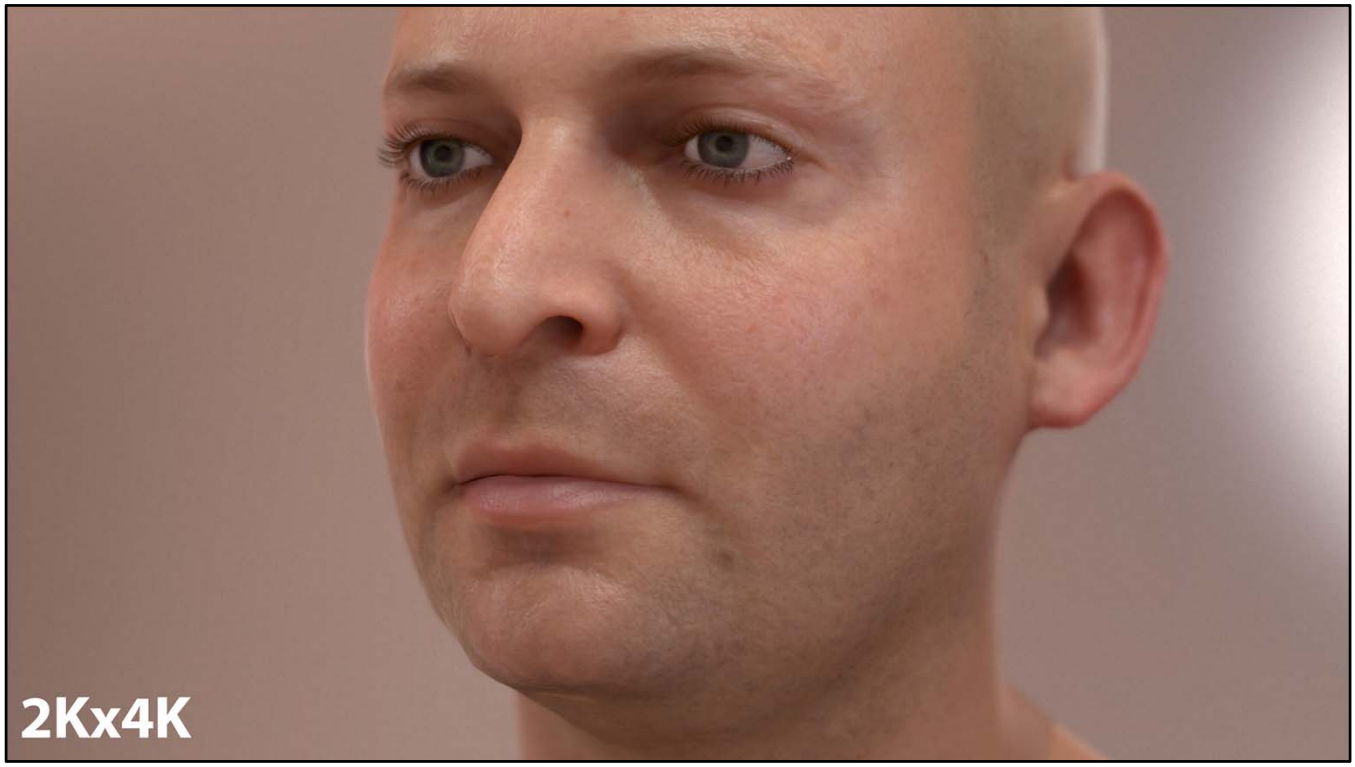
Texture Size

- **The question always asked**
- **Scary: we're using 2k x 4K textures**
- **But Bernardo Antoniazzi developed an optimized UV layout that manages to keep most of the detail in 1K x 2K**

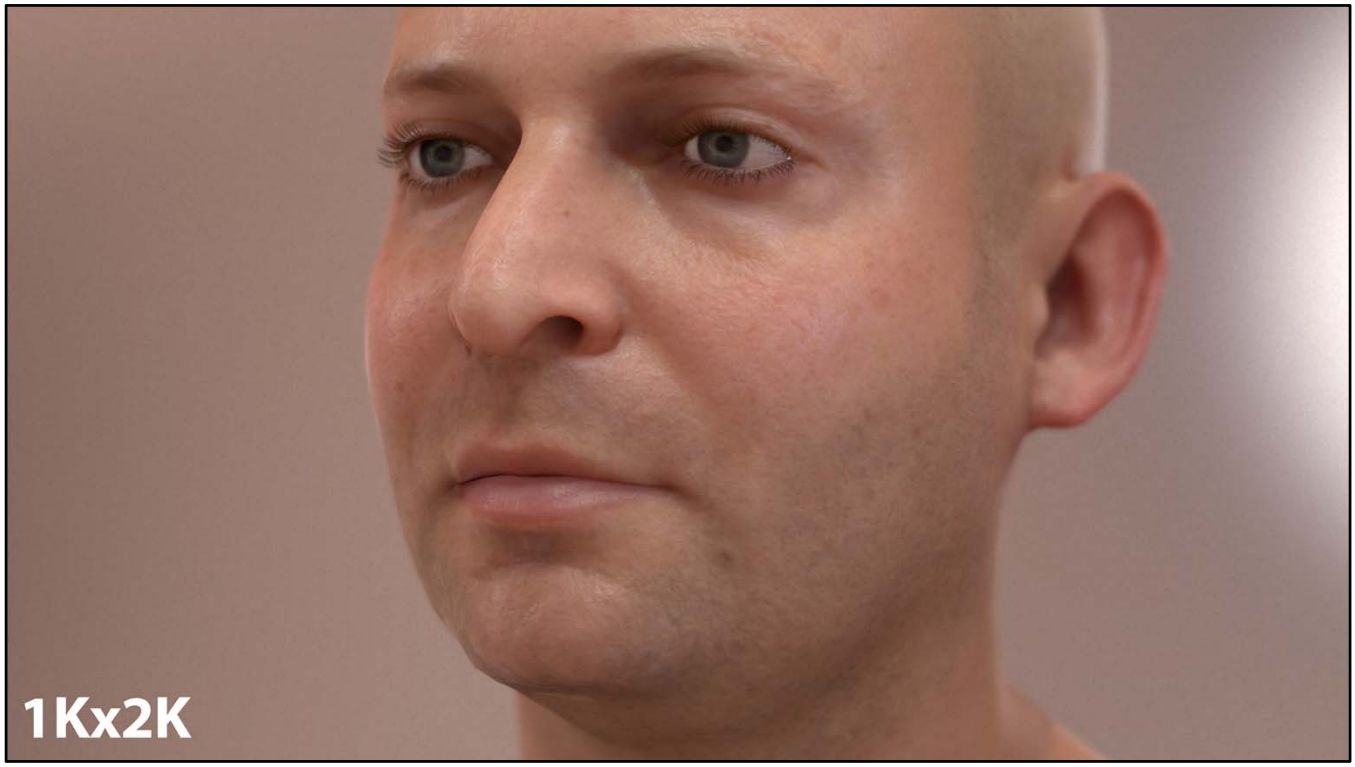


A possible concern is the texture size we're using, which is quite big: 4K.

However, Bernardo Antoniazzi developed a UV layout that allows us to keep most of the detail in just 1K.



Here you can see the 2Kx4K textures (using Bernardo’s layout)…



…and here 1Kx2K. There’s some degradation in quality, but it still looks good.

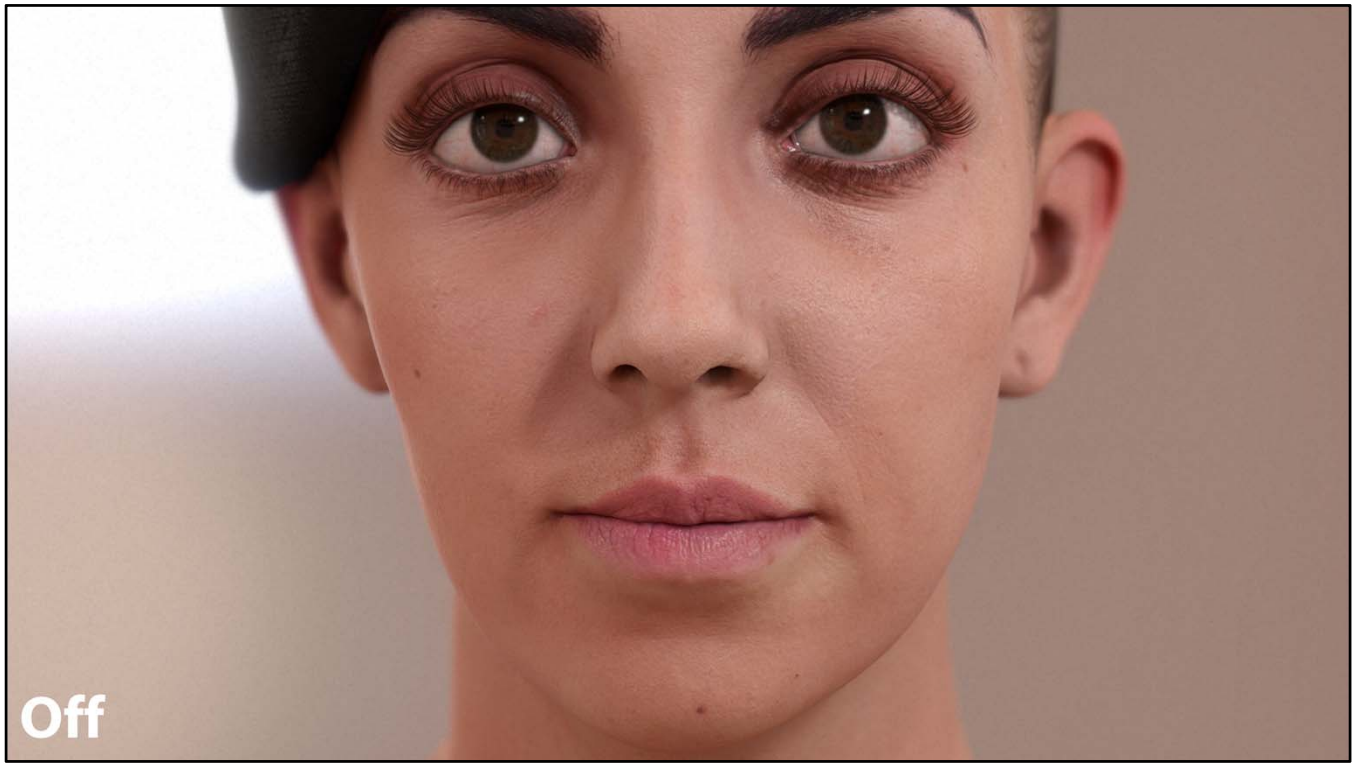
You can reduce the diffuse even further, and still achieve plausible results.

Where SSS Matters

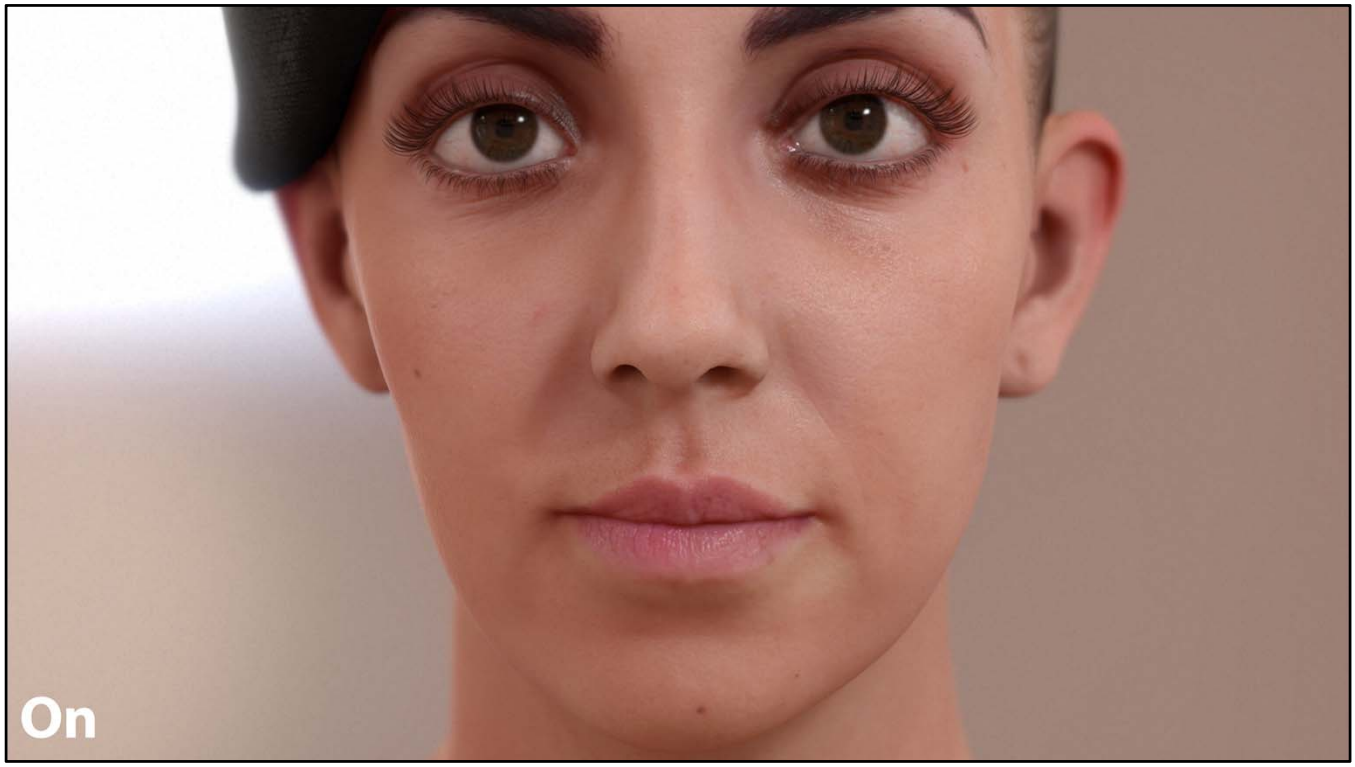
- **Not essential:**
  - **Low resolutions**
  - **Far camera setups**
  - **Ambient-like lighting**
  - **Smooth skin**
- **Crucial for:**
  - **High resolutions**
  - **Cinematic shots**
  - **High-contrast lighting**
  - **Wrinkled characters**
  - **Softening material boundaries (eyes and skin)**

ACTIVISION | BLIZZARD

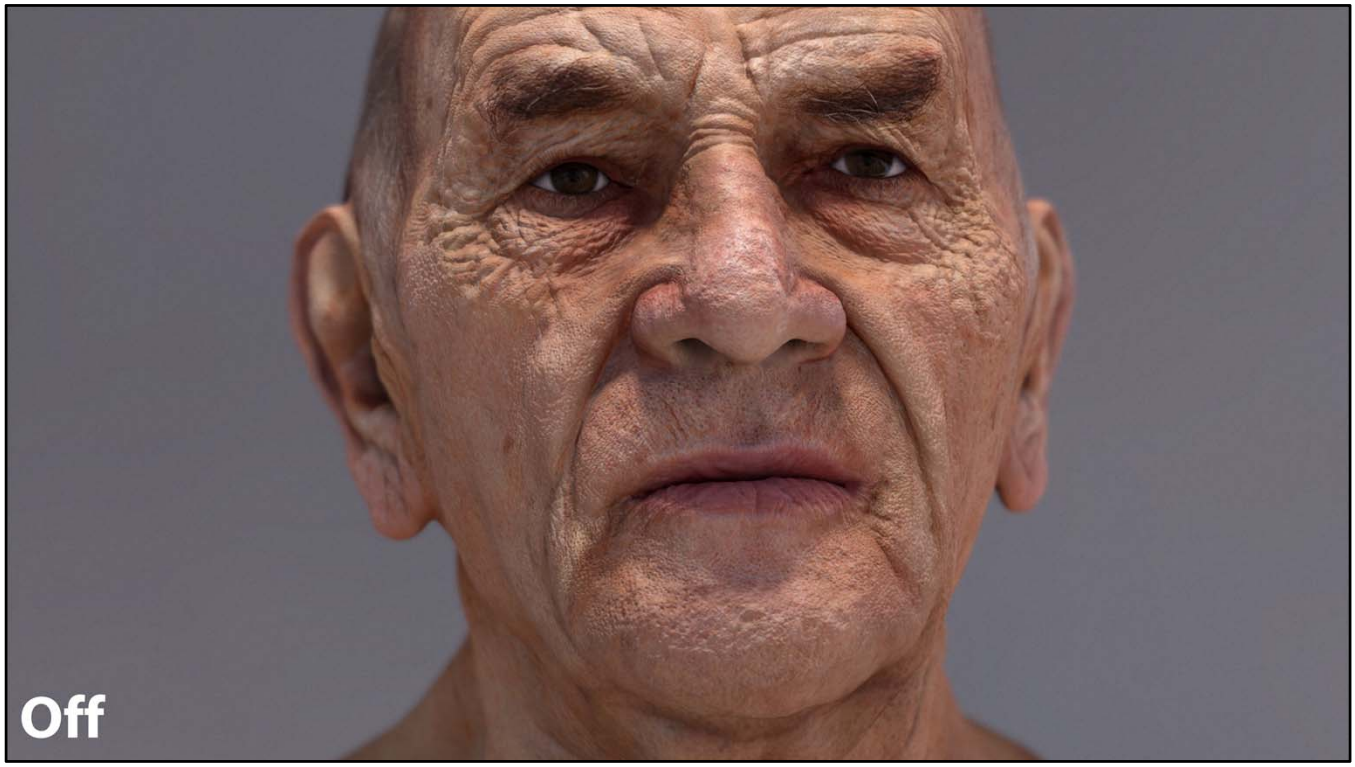
The importance of subsurface scattering depends on a lot of variables…



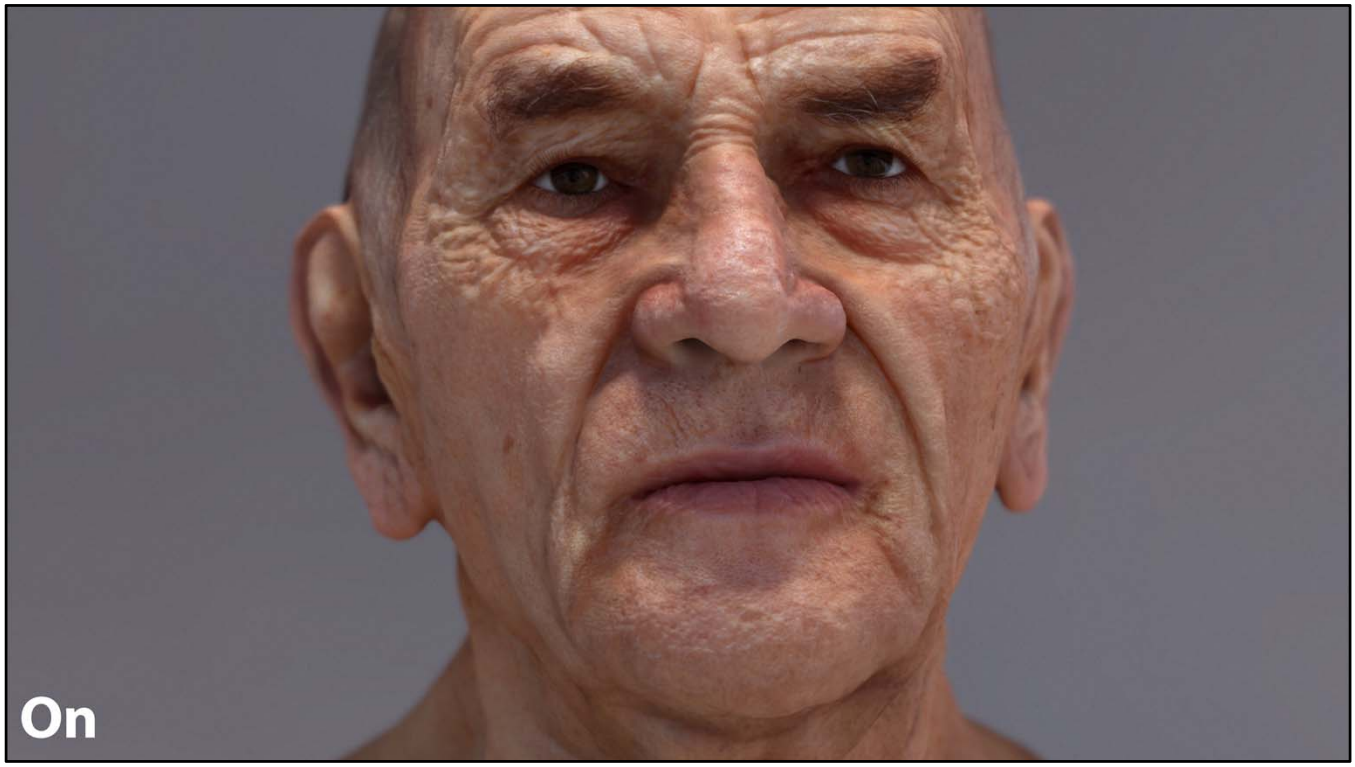
…for a character with smooth skin in an ambient lighting setup…



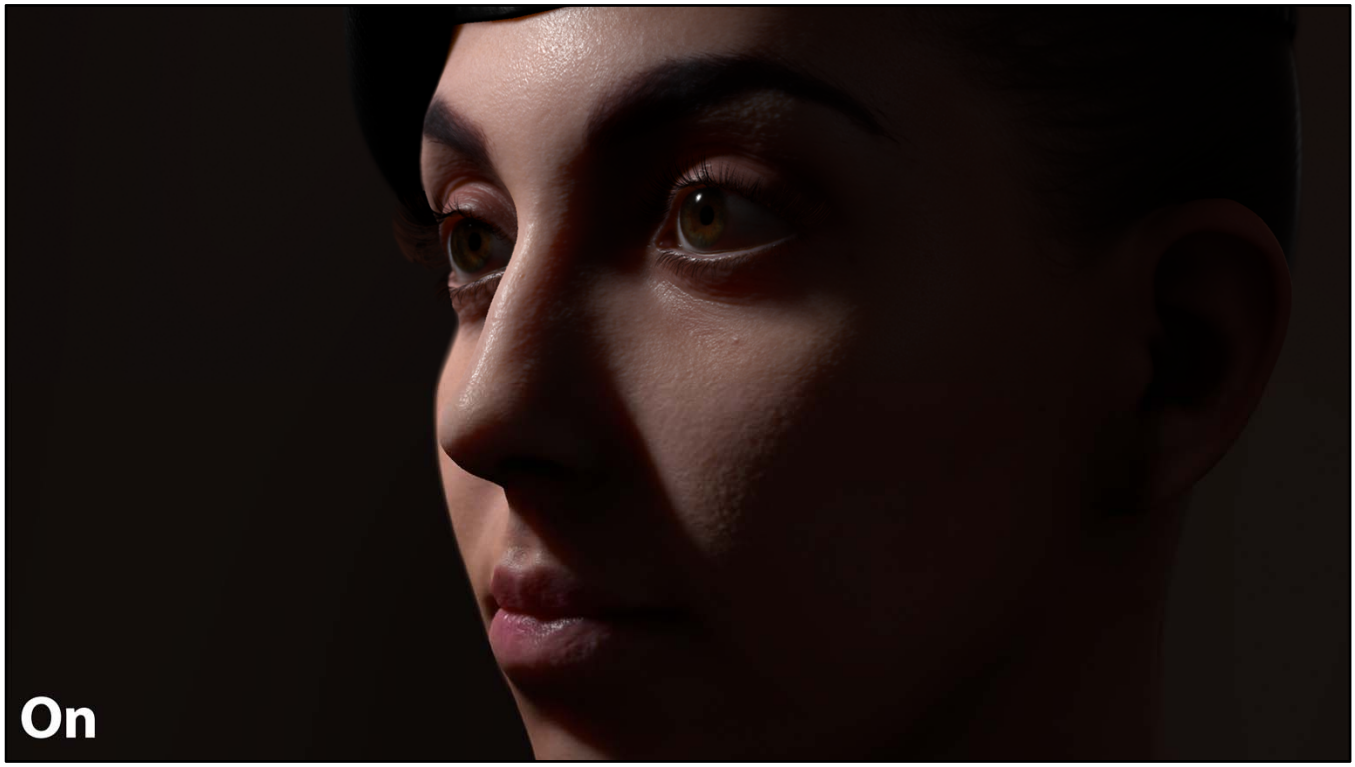
…it is probably not crucial.

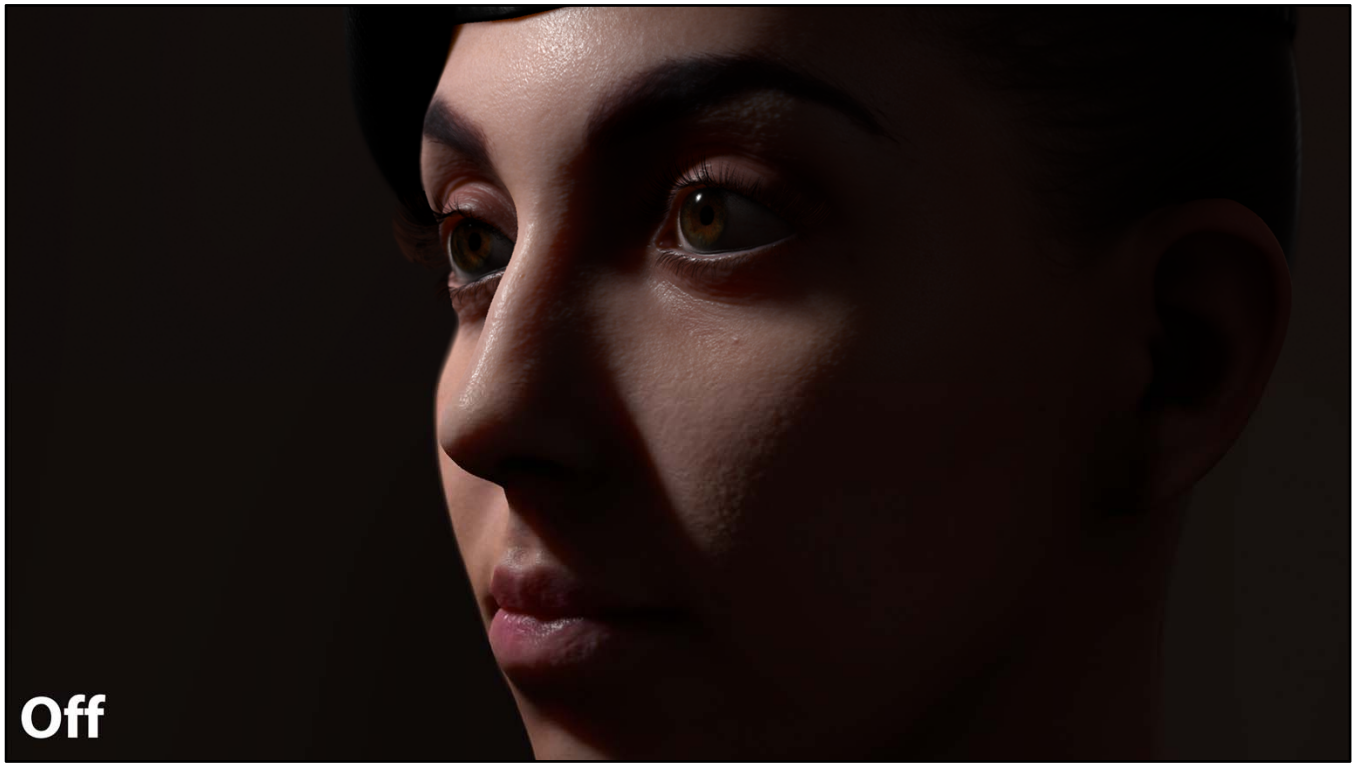


But if you have a wrinkled surface or harsh lighting…



…it’s extremely hard to achieve realistic results, without using subsurface scattering.


On


Off

Integration Issues

- **For best results you need to MRT skin objects:**
  - **Diffuse lighting**
  - **Specular lighting**
  - **Albedo**
- **SSS should be applied only to the diffuse lighting (recall: specular doesn't enter the skin)**
- **Also, applying it before texturing helps to preserve texture details**

(See pre/post scattering texturing in [Eon2007] *Advanced Techniques for Realistic Real-Time Skin Rendering*)

ACTIVISION | BLIZZARD

Regarding possible integration issues, I'd like to mention that for optimal results, you need to use MRT for skin objects.
The diffuse, the specular and the albedo should go into separate buffers.

Subsurface scattering should only be applied to the diffuse component and for best results, applied either before texturing, or after applying the "sqrt" of the texturing. (See pre/post scattering texturing in [Eon2007] *Advanced Techniques for Realistic Real-Time Skin Rendering*)

Integration Issues

- **Deferred engines:**
  - **Only calculate the diffuse for all lights**
  - **Do SSS and combine with the albedo**
  - **Apply dominant specular lighting**
- **Forward engines:**
  - **For less specular blurring:**
    - **We store 1/depth in the alpha channel of skin objects, which in the end determines the blur radius in screen space**
    - **Decrease this factor where specular is high**
  - **For less texture blurring:**
    - **Smart sharpen diffuse map details in Photoshop**

ACTIVISION | BLIZZARD

For deferred engines, an option for skin is to only calculate the diffuse in the regular lighting pass,
then apply SSS, combining the results with the albedo.

Finally, apply another pass for the dominant specular lighting.

For forward engines, we can decrease the specular blurring on skin by modifying the depth depending on the level
of specular lighting for a given pixel.

While blurring, the depth of each tap is compared with the center depth.

This means that if specular pixels have different depth values than their neighbors, they won't bleed into them while filtering.

Corrected SSS Scale Factor

```
// Calculate the specular term:
float specularTerm = dot(specularLighting, specularLightingSharpness);

// Scale correction factor depending on distance:
float scale = saturate(1.0 - invDepth * specularTerm);

// invDepth can be thought as the width of the SSS filter. So, we're
// effectively reducing it as specular lighting increases:
colorSVT.a = scale * invDepth;
```

ACTIVISION | BLIZZARD

This shader snippet shows how to apply this idea.

Remember that the alpha channel of the color buffer stores 1/depth, which will be used for SSS filtering.

Note that this should only be used in the case of outputting the diffuse and the specular lighting into the same buffer.

Subsurface Scattering
Transmittance Part

ACTIVISION | BLIZZARD

The other component of subsurface scattering is light transmittance or what we usually refer to as translucency,
a phenomenon commonly seen with ears and nostrils.

Our Previous Work

- [Jimenez2010] *Real-Time Realistic Skin Translucency*
  - http://www.iryoku.com/translucency/
- [Jimenez2012] *Separable Subsurface Scattering and Photorealistic Eyes Rendering*:
  - http://advances.realtimerendering.com/s2012/index.html

Again, we refer you to previous work for a better understanding of this part.

Introduction

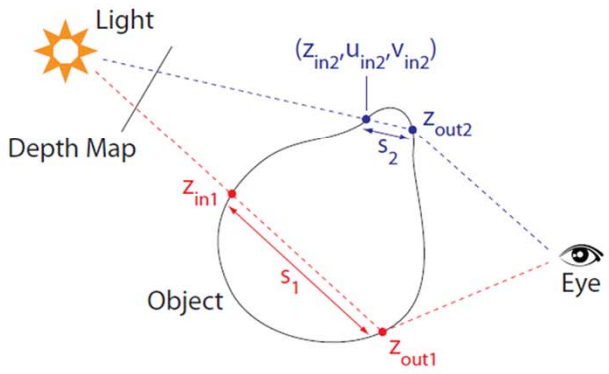
- **To avoid confusion these are the approaches we will talk about in this section:**
  - **Single-Sample Transmittance: [Jimenez2010] *Real-Time Realistic Skin Translucency***
  - **Multi-Sample Transmittance: the natural evolution of [Jimenez2010].**
  - **Spherical Harmonic Transmittance: Multi-Sample Transmittance encoded using SH4.**

First of all, I'd like to mention all the techniques we'll talk about in this section, to avoid any confusion.

Single-Sample Transmittance is our latest previous work in this regard, and we will present two new methods we have tried:

Multi-Sample Transmittance and Spherical Harmonic Transmittance.

Previous Work

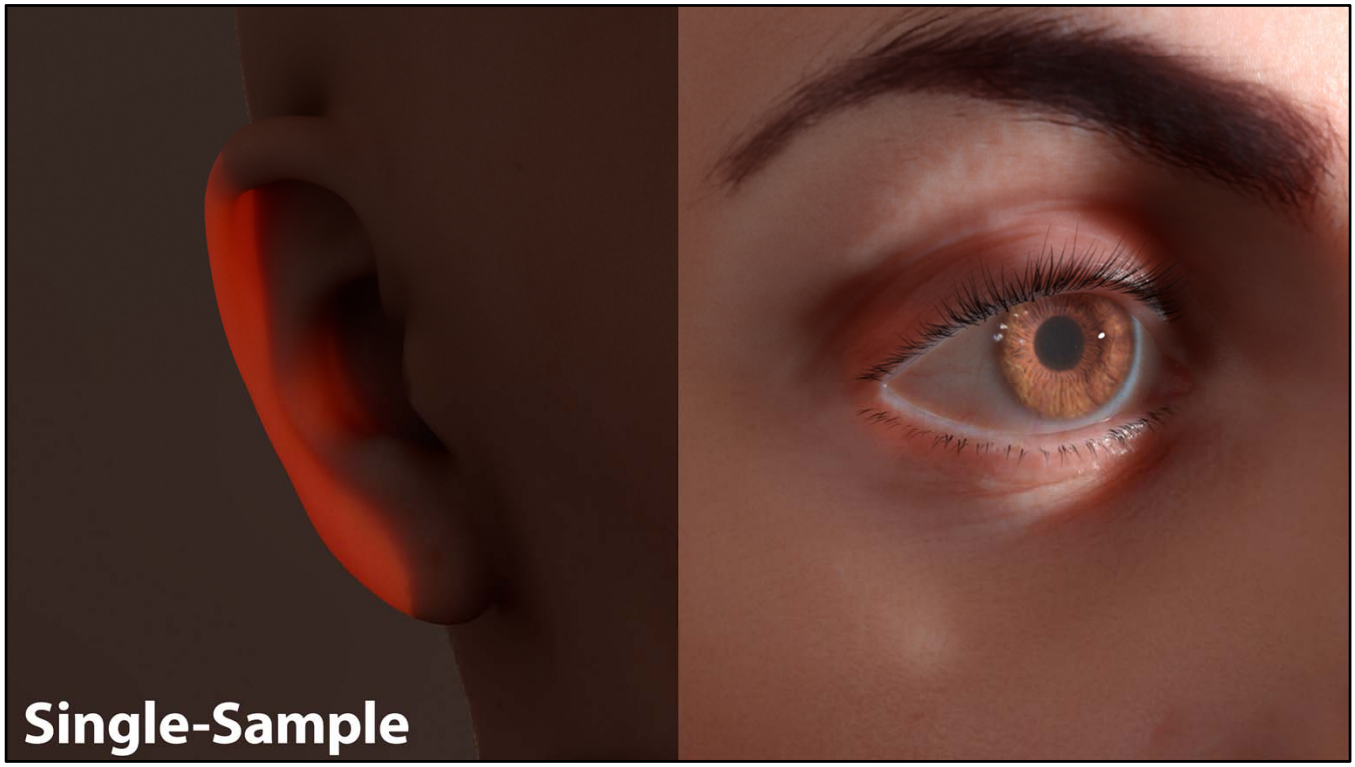


To summarize: in our previous work, we calculate the distance light travels through an object using shadow mapping,
comparing current shader position z_out1 with shadow map position z_in1.

As light travels inside of the skin, it scatters more and more (as opposed to water in a glass).

So, in the end, rays don't travel in a straight line. This means that light is diffused.

However, as we use a single sample, light won't be diffused as it travels inside of an object, which is something highly undesirable.



This is the result of our previous shadow mapping approach.

Notice how you can clearly see the shape of what is behind the ear – something that is not so obvious in the real world.

Our old, simple approach, also produces artifacts for complex areas such as the eyes.

Multi-Sampled Transmittance

- **Skin is not glass: transmitted light cannot be, in general, accurately represented by a single sample**
- **In certain configurations, this results in artifacts**
- **Solution: take more samples!**

ACTIVISION | BLIZZARD

So, how to solve this?

We can blur the shapes behind a translucent object (like the ear) by using multiple samples from the shadow map.

Just as a fixed-size soft shadow mapping technique fades away shadows of distant objects instead of introducing aliasing,
taking multiple samples for SSS will have a similar effect, making the technique more robust.

When the light is far away from the character, instead of introducing artifacts, transmittance will just progressively fade out.

Multi-Sampled Transmittance

- **Problem: taking multiple samples is expensive**
- **Solution: jitter them spatially, like soft shadow techniques**
  - **Globally, using uniform registers with Poisson disk samples**
  - **Per-pixel, by rotating randomly via a jitter texture**
- **This solution doesn't pretend to be accurate; it's just to improve the results visually**

ACTIVISION | BLIZZARD

However, I've got bad news: multisampling translucency is very expensive.

Bur fear not, as we can leverage random sampling to improve the performance.

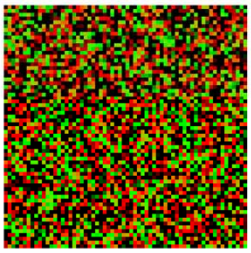
By jittering the translucency offsets using typical Poisson disk random samples and a tiled random rotation texture
(as is commonly done for soft shadows), we were able to dramatically reduce the number of samples, while still
yielding good visuals.

Multi-Sampled Transmittance

Uniform registers

```
// Poisson disk random samples:
static const float2
SSSSPoissonOffset[] = {
    // 1
    float2( 0.402211,  0.126575),
    // 2
    float2( 0.297056,  0.616830),
    float2( 0.298156, -0.001704),
    // 4
    float2( 0.019369,  0.395482),
    float2(-0.066918, -0.367739),
    float2(-0.955010,  0.372377),
    float2( 0.800057,  0.120602),
    ...
}
```

Jitter Texture



```
// Calculate position in light coordinates:
float4 shadowPosition = mul(float4(positionW, 1.0), lightViewProjection);
shadowPosition.xy /= shadowPosition.ww;
float shadowDepthNormalized = shadowPosition.z / lightFarPlane;

// Fetch per-pixel jitter:
float2 jitter = SSSSSampleWrapL0(samplingNoiseTex, svPosition * (1.0 / 64.0));
float2x2 rotationMatrix = {  jitter.x, jitter.y,
                            -jitter.y, jitter.x };

// Do shadows...

// Calculate transmittance:
float2x2 transmittanceRotationMatrix = transmittanceBlurScale * rotationMatrix;
[unroll]
for (int j = TransmittanceStart; j < TransmittanceStart + transmittanceSampleCount; j++) {
    float2 pos = shadowPosition.xy + mul(SSSSPoissonOffset[j], transmittanceRotationMatrix);

    float blocker = SSSSSamplePointL0(shadowTex, pos);
    float thickness = transmittanceScale * max(lightFarPlane *
                                               abs(shadowDepthNormalized - blocker),
                                               minTransmittance);
    float light = exp(-thickness * thickness);

    shadow.rgb += transmittanceColor * light;
}
shadow.rgb *= 1.0 / transmittanceSampleCount;

// At grazing angles there should not be any transmittance contribution:
shadow.rgb *= saturate(0.3 + cosineAngle); // dot(light, -normal)
```

ACTIVISION BLIZZARD

Here you can see the uniform registers holding the Poisson offsets for different sample counts (for 1 sample, 2 samples, 4 samples, etc.),
and the jitter texture to be tiled in screen space.

It contains [cos(x), sin(x)] in the RG channels, which will be used for per-pixel rotation of the Poisson disk samples.

If you happen to use jittered soft shadows, then you may be able to reuse a lot of code, which will make the transmittance shader faster.

The amount of blur is determined by `transmittanceBlurScale`.

`To be accurate, this parameter should be linked to transmittanceScale, but this interface gives more artistic control.`

With regards to the shadow map, we can think of `transmittanceBlurScale as acting on the XY plane, and transmittanceScale in the Z plane.`

Jittered Soft Shadows and our Multi-Sampled Transmittance has a lot to share!

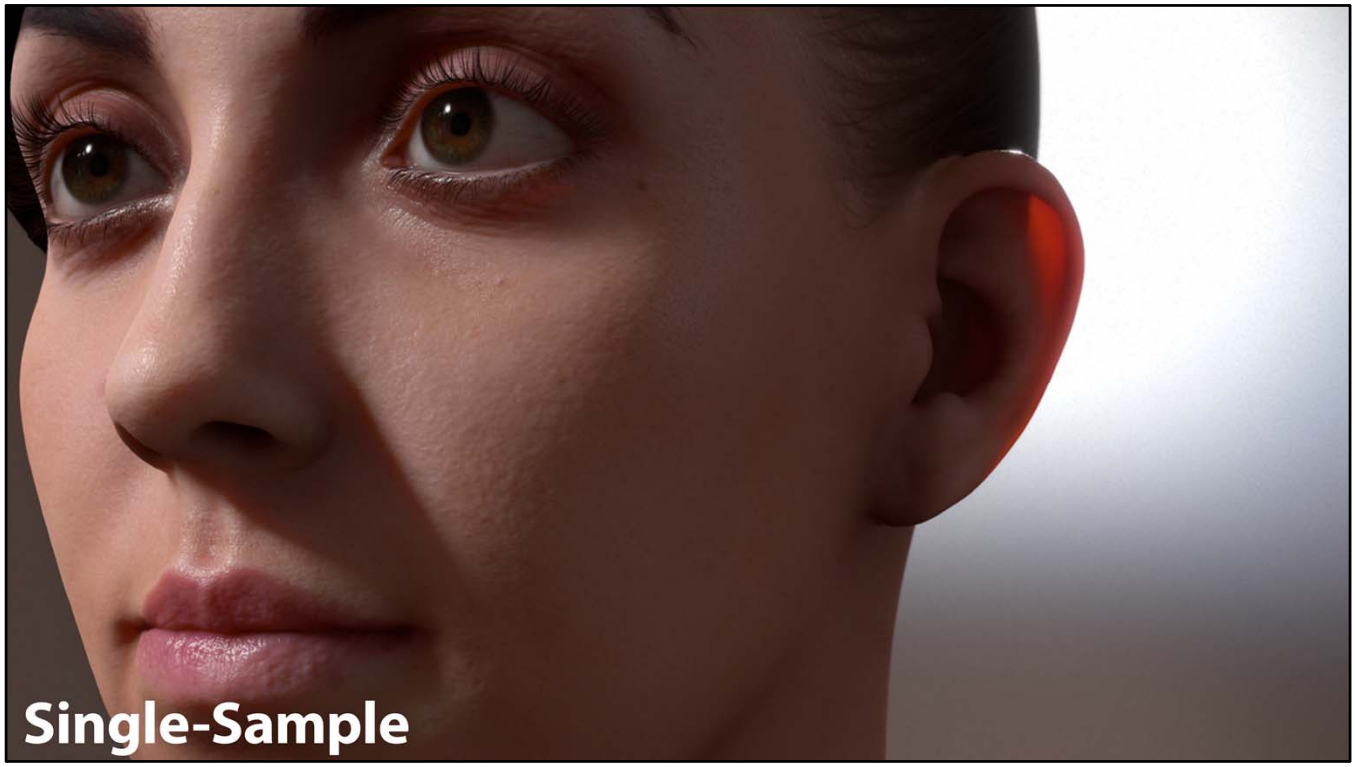
Multi-Sampled Transmittance

- 16 shadow samples per pixel still too expensive
- Sampling two times leads to noisy results
- But SSS Blur will take care of the noise!

ACTIVISION | BLIZZARD

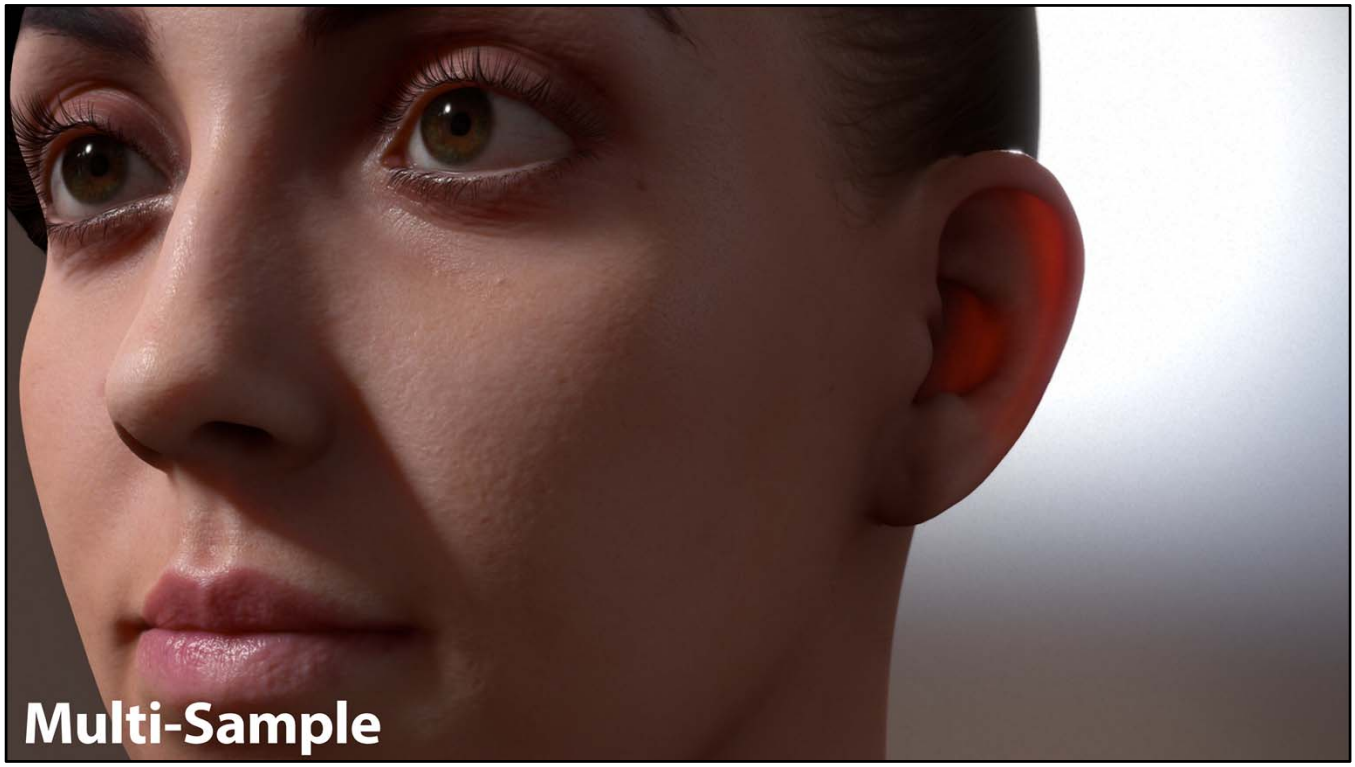
Here you can see how sampling two times yields noisy results…

…however the screen-space subsurface scattering diffusion will do a good job of removing the noise, for free!



So, how does this compare with our old single-sample approach?

(remember, our old approach is [Jimenez2010] *Real-Time Realistic Skin Translucency)*



Notice how the light that travels through the ear is blurred, with a less clear hint of what is behind it.

Here we have a comparison movie.

Notice how the annoying artifacts of the translucency around the eye are gone, and how the hard line in the ear is now blurred,
leading to more natural looking results.

(Note: this was a work-in-progress capture, hence the lack of hair)

User Interface

- **Using the multipole diffusion profile [Eon2007] :**
  - **Is expensive**
  - **Lacks customization possibilities**

```
float thickness_2 = -thickness * thickness;
float3 transmittance = float3(0.233, 0.455, 0.649) * exp(thickness_2 / 0.0064) +
                       float3(0.1,   0.336, 0.344) * exp(thickness_2 / 0.0484) +
                       float3(0.118, 0.198, 0.0)   * exp(thickness_2 / 0.187)  +
                       float3(0.113, 0.007, 0.007) * exp(thickness_2 / 0.567)  +
                       float3(0.358, 0.004, 0.0)   * exp(thickness_2 / 1.99)
                       float3(0.078, 0.0,   0.0)   * exp(thickness_2 / 7.41);
```

From [Jimenez2010] *Real-Time Realistic Skin Translucency*

ACTIVISION BLIZZARD

We also wanted to deal with the user interface, in a similar way to how we handled the subsurface scattering blur.

Our older approach ([Jimenez2010] Real-Time Realistic Skin Translucency) was not customizable,
and also expensive given it used 6 Gaussian filters for the calculations.

Improving the Interface

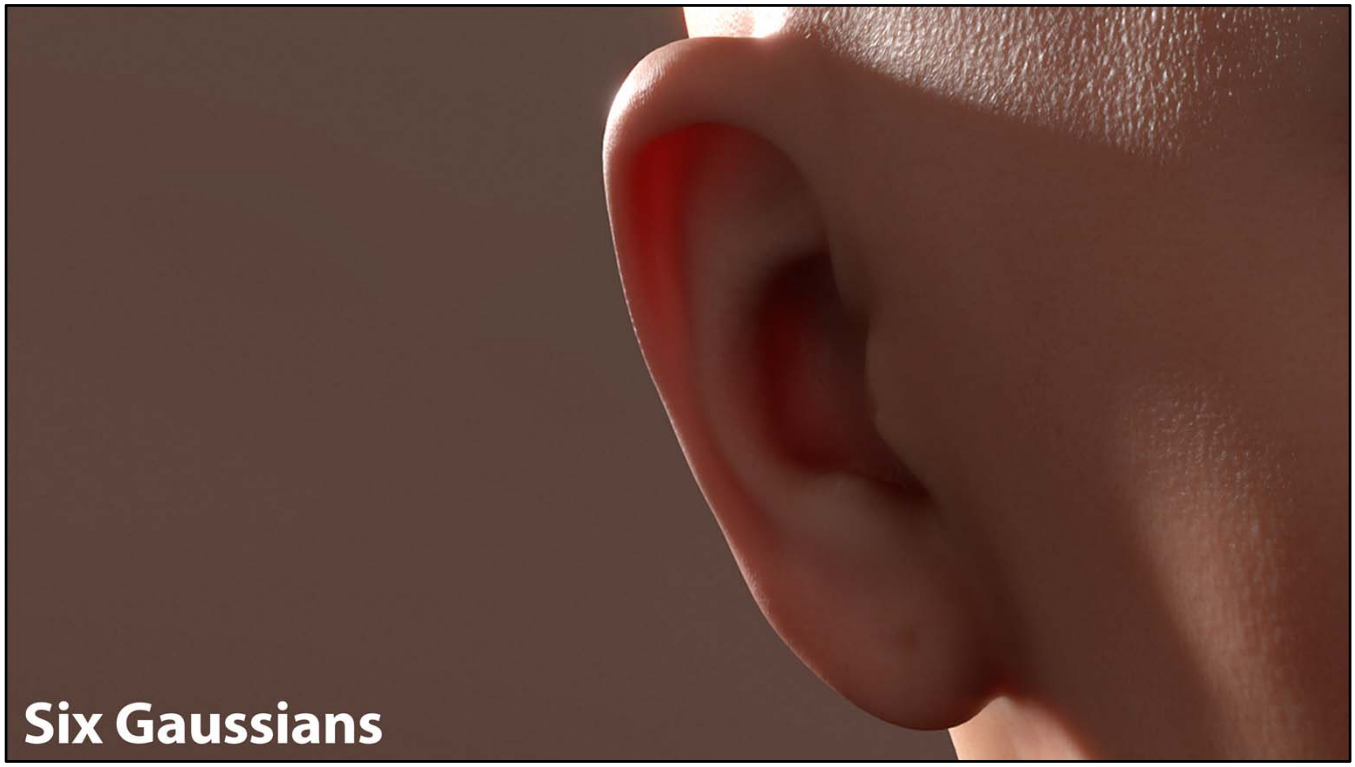
- **Too many Gaussians to control**
  (interface problem)
- **Too many Gaussians to calculate**
  (performance problem)
- **Both problems have the same solution:**
  Use one!

ACTIVISION | BLIZZARD

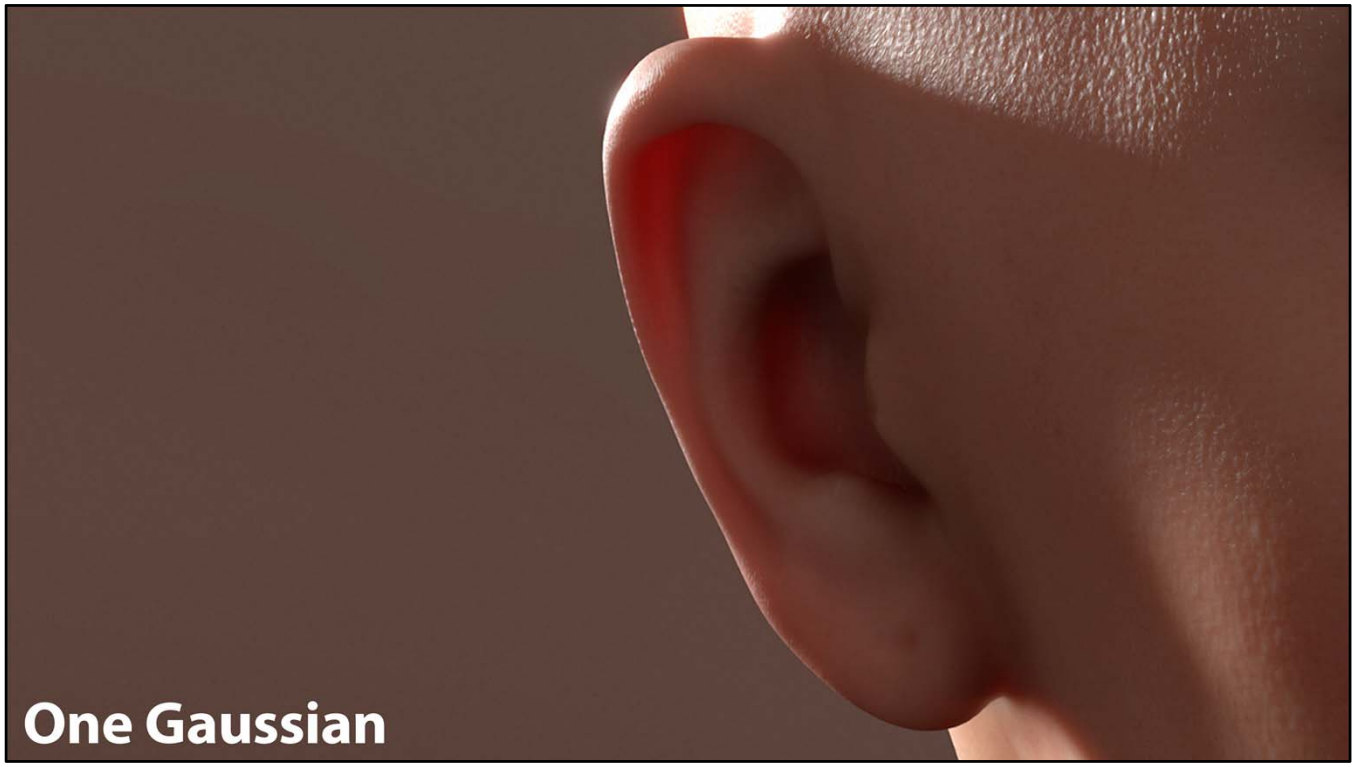
So, basically, we had two problems.

We had too many Gaussians to control and too many Gaussians to calculate.

But both problems have the same solution: use only one!



Visually, It doesn’t seem to have a big impact…



…as can be seen on this slide.

It was very important to use all of them (or a suitable representation like our separable approach) for reflectance blurring, but for transmittance – unless the slab is very thin – we're fine with one Gaussian.

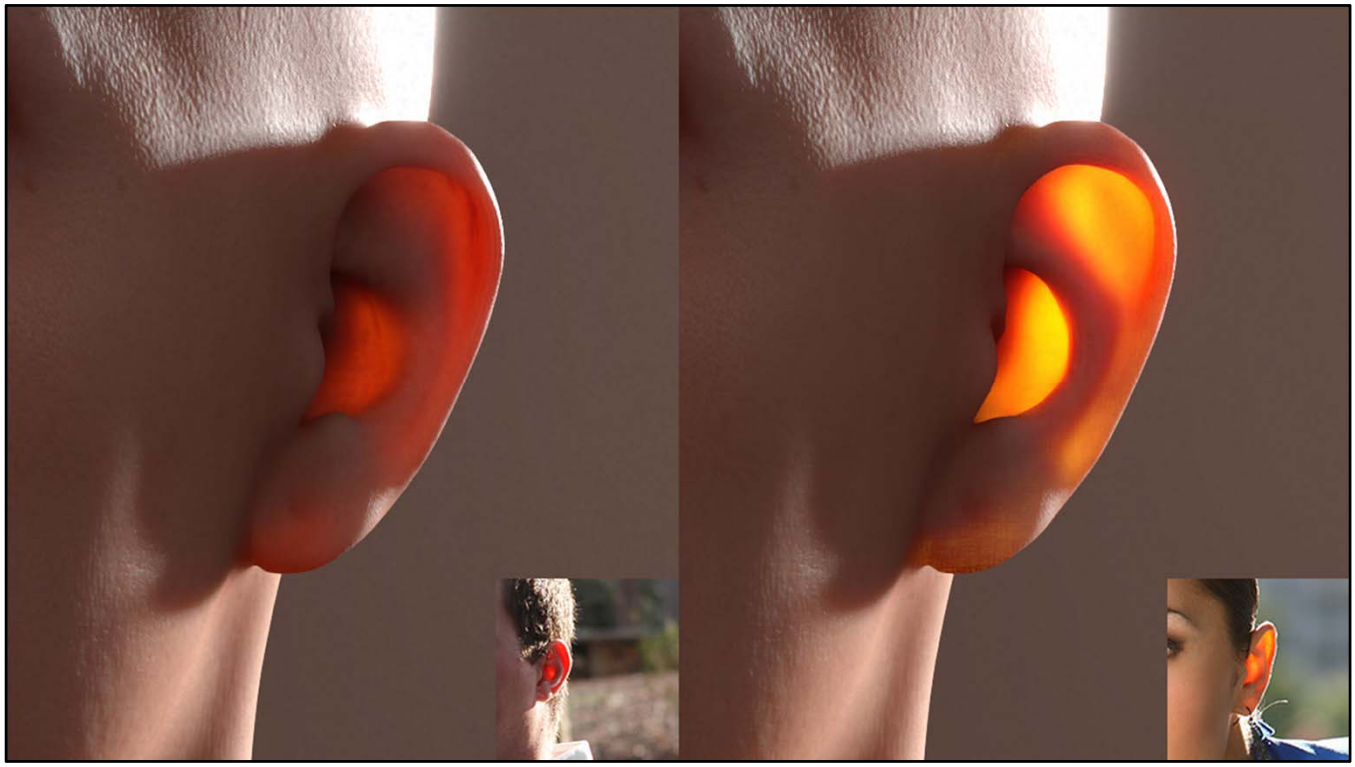
Improving the Interface

- **The interface is simple:**
  - **Transmittance scale**
  - **Transmittance color**
- **And the code as well:**

```
thickness *= transmittanceScale;
float3 transmittance = transmittanceColor * exp(-thickness * thickness);
```

ACTIVISION | BLIZZARD

This simplifies the interface and code by a large margin, as can be seen in this slide.



A limitation of the technique is that it cannot model the yellow to red gradients produced by very thin ears.

However, we found that rendering thick ears with only red gradients to be enough for our purposes,
as they also seem to be the most common type of ears out there.

Limitations

- **We could use two Gaussians to model the yellow-to-red gradients found in thin ears**
- **But not in practice**

ACTIVISION | BLIZZARD

You might think that we could use two Gaussians to model this, but unfortunately we cannot.

We cheated

- **We didn't explain a little piece of code which has important implications**
- **Silhouettes of thin objects produce an incredible amount of artifacts**
- **In order to properly solve this, you need to sample at least 16 times**
- **Or you can just wipe small thickness values:**

```
float thickness = max(thickness, minTransmittance);
```

ACTIVISION | BLIZZARD

We didn't explain the nasty and hacky details.

In the silhouettes of objects (with regards to the light position),
small thickness values will produce a lot of artifacts (similar to shadow mapping).

To solve this properly, you need a lot of samples (>16), so our solution is to
clamp small thickness values instead.

Limitations

- **We don't have enough precision in thin slabs – they will all be equal to** `minTransmittance`
- **Do we care?**
  - **Probably not: red ears look realistic anyway**

ACTIVISION BLIZZARD

By doing that, we lose resolution in very thin slabs, as all the areas will have the clamped value.

This leads to more flat translucency for very thin slabs.

Should we care? Probably not: red ears look good anyway! (and more natural)

Spherical Harmonics (SH4) Transmittance

- Also a good approach
- But, light leaks because of transmittance lighting lagging behind the light direction when using 2 bands (SH4)
  - Light may be on the back of the head, but the transmittance in the nostril may still be in effect
- Stored in a uncompressed 8-bit RGBA texture.

ACTIVISION | BLIZZARD

Using spherical harmonics for transmittance is also a plausible approach.

We baked our Multi-Sample Transmittance approach into SH4…

And here you have a comparison of the multisample approach with spherical harmonics.

We begin with spherical harmonics.

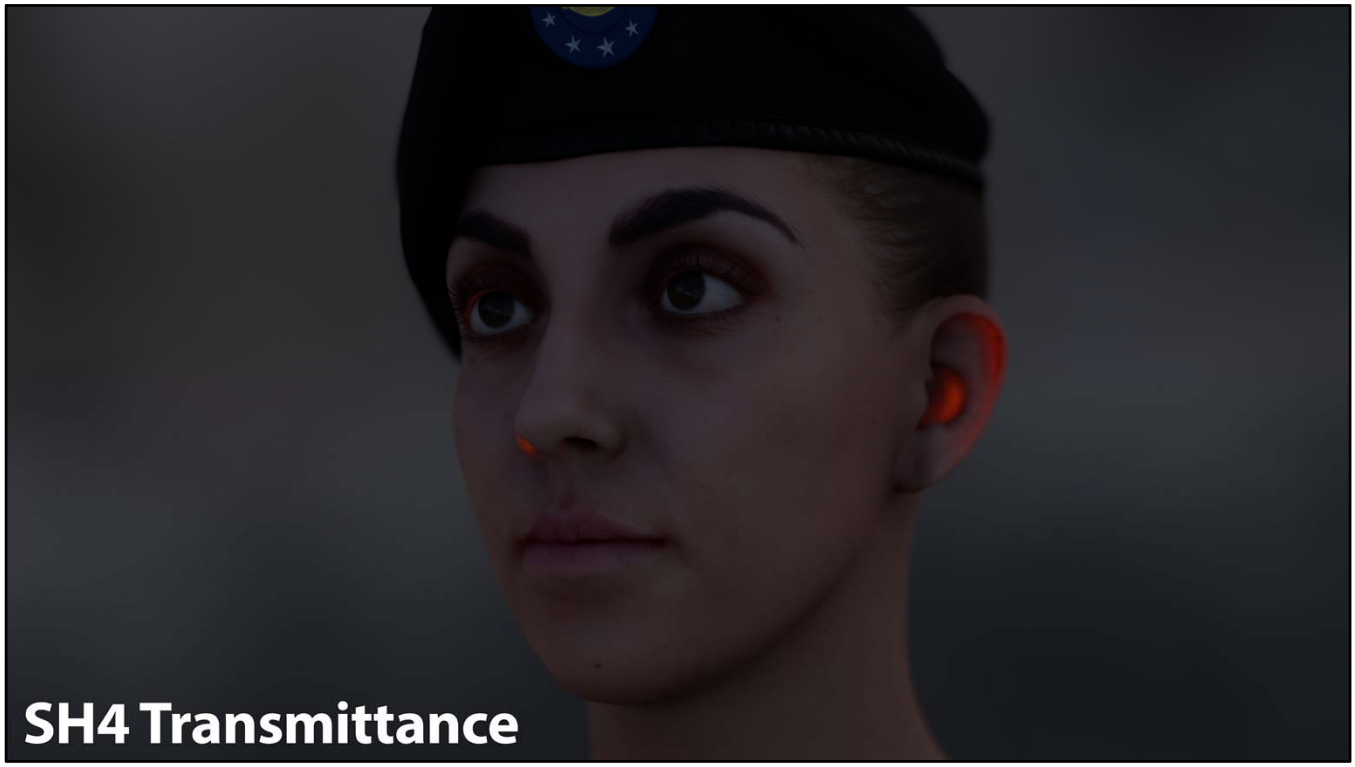
With a weak light the results are reasonably good. However, notice how the light lags, something especially noticeable with strong lights.

Finally, note how when we switch to Multi-Sample Transmittance (when we switch from “Direct SH4” to “Direct Shadow” in the demo),
the light leaks disappear.

On the other hand, it introduces more noise.

Expending to SH9 should reduce this problem, at the expense of using more memory to encode transmittance.

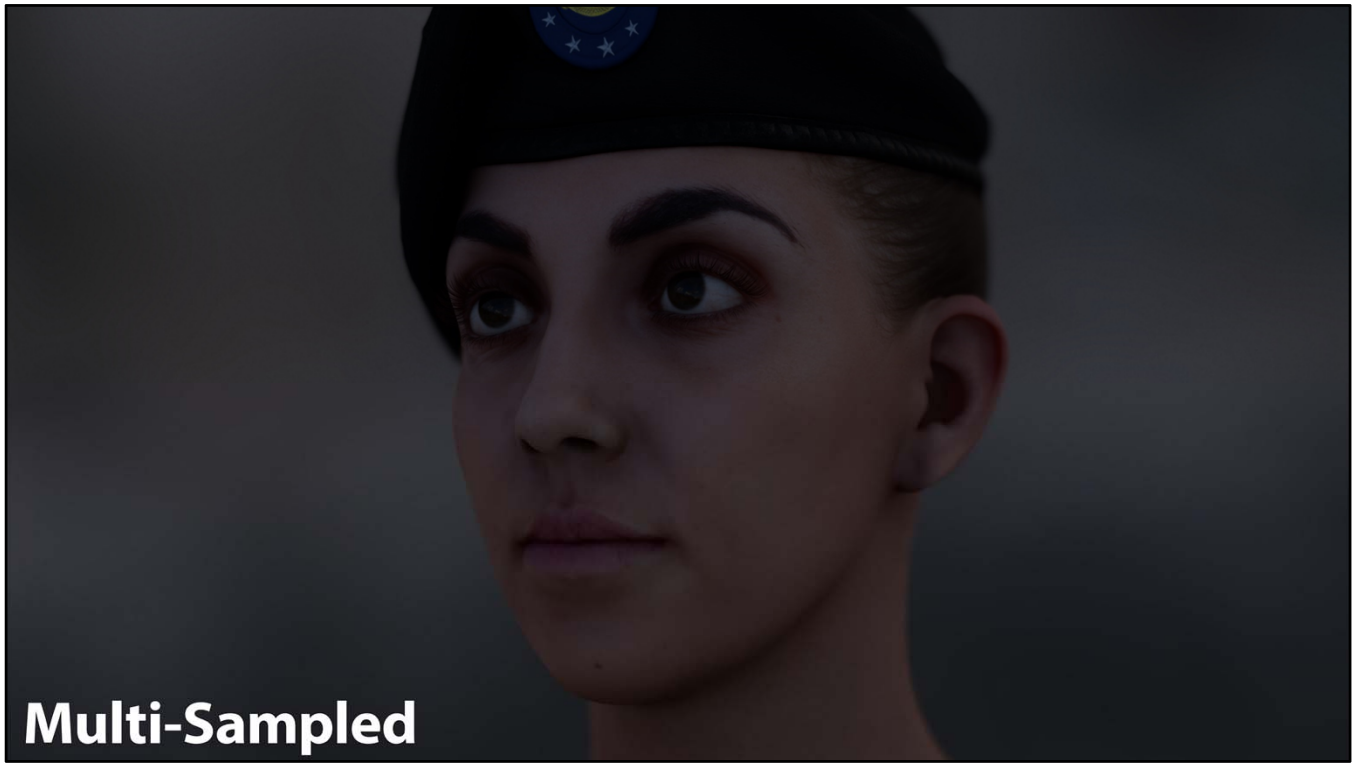
Compressing the SH4 transmittance texture with the DXT5 format produces even more noticeable light leaks.



Here you can see how, sometimes, the light lag can be extremely noticeable when using SH4.

The light in the nostril is not caused by any light coming from the left.
A similar problem happens in the right ear.

Note that the problem is less noticeable when the lighting has more ambient than directional lighting.



Multi-Sample approach, on the other hand, don’t have this problem.

Single-Sample	Multi-Sample	SH4
Pros	**Pros**	**Pros**
• Simple runtime	• Very high quality • Scalable	• Simple runtime
Cons	**Cons**	**Cons**
- Artifacts - Hard Shapes	- *Complex* runtime - Noise	- *Complex* offline - Light leaks - Textures - No shadow occ.

This slide compares the three techniques.

The simple single-sample approach is fast and delivers something that, while not perfect, is better than nothing.

The multi-sampled approach is more complex, but has very nice properties like the high quality of the visuals it produces,
and the fact that you can just use 2 samples for in-game, and 16 samples for cinematics, with the same runtime.

Finally, the Spherical Harmonics approach with 4 terms (order 2) is good if you are already using SH,
and you don't mind using additional textures and having light leak problems.

Ambient Subsurface Scattering

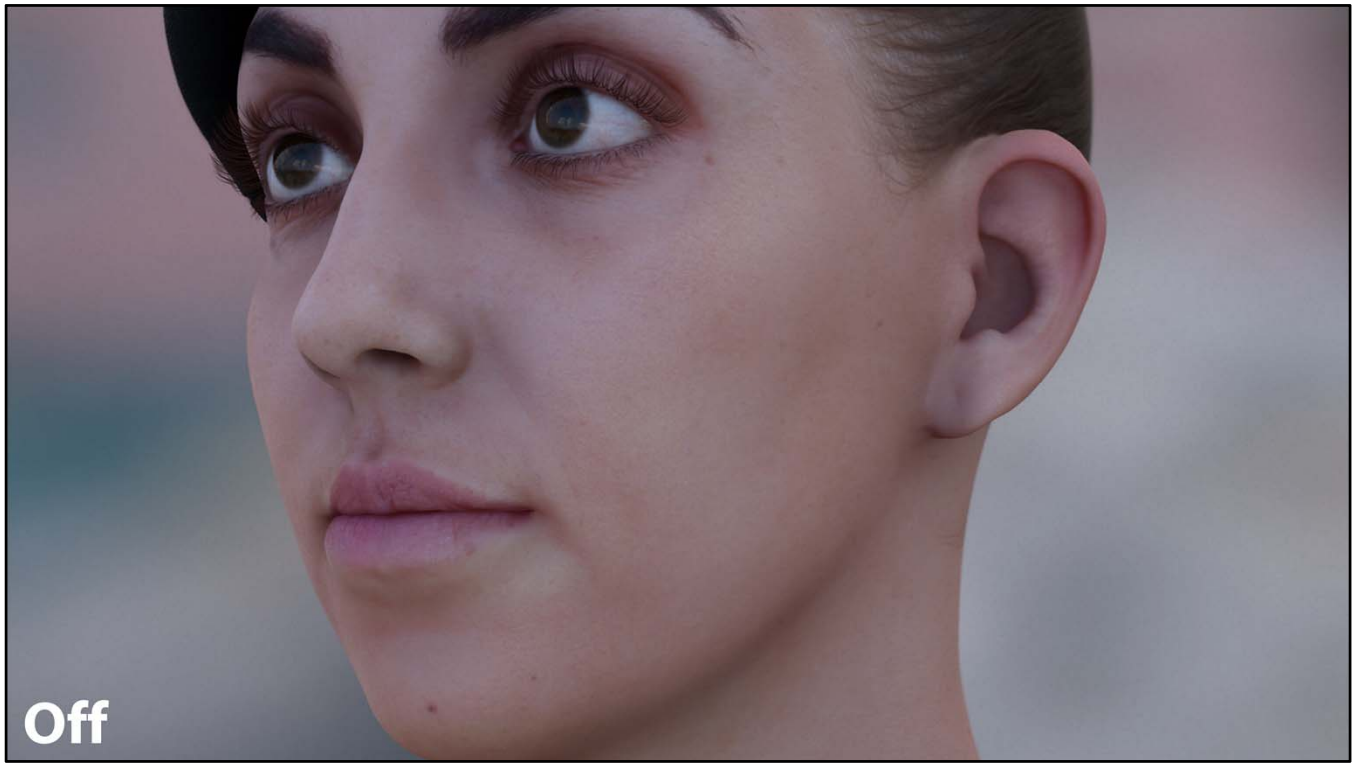
- **We can use SH4**
- **If the lighting is not very directional, SH1 is more than enough (just taking the ambient term of spherical harmonics)**

ACTIVISION | BLIZZARD

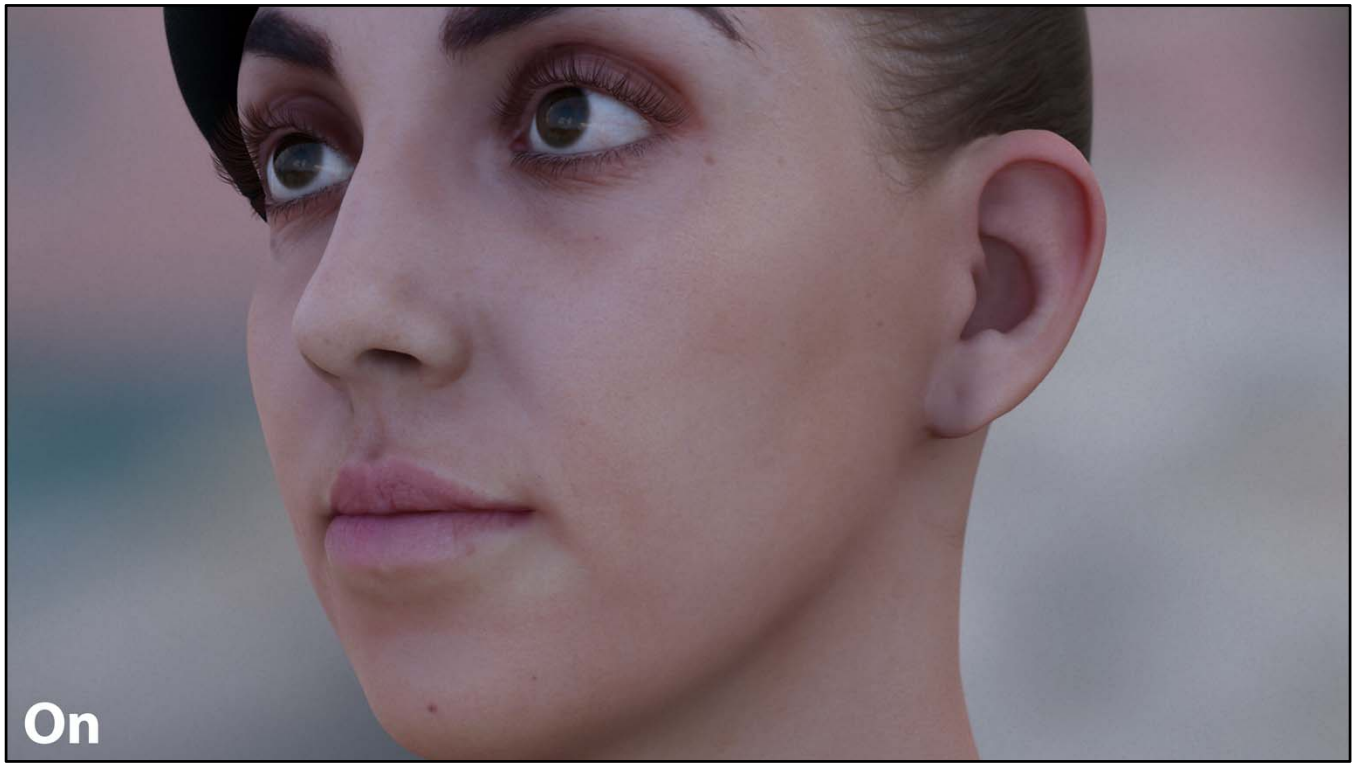
So far, we have just dealt with procedural light transmittance. But what happens with transmittance due to light probes?

For this we can use SH4 together with the SH4 representation of the light probe,
or if the light is not very directional, use the ambient term (SH1) for subsurface scattering.

(So as not to confuse anyone that happened to attend previous presentations,
SH1 is similar in spirit to the approach we presented in
[Jimenez2012] *Separable Subsurface Scattering and Photorealistic Eyes Rendering* past year,
but better thought out)



Here we have ambient subsurface scattering turned off...



…and here turned on.

The difference is very subtle for more uniform lighting environments (which is often the case for light probes).

Transmittance can only be seen when the light on the back of the face is much stronger than the light from the front.

So, being practical, it’s probably best is to capture (using photographs), or paint a little bit of ambient translucency into the diffuse map.

Eye Rendering

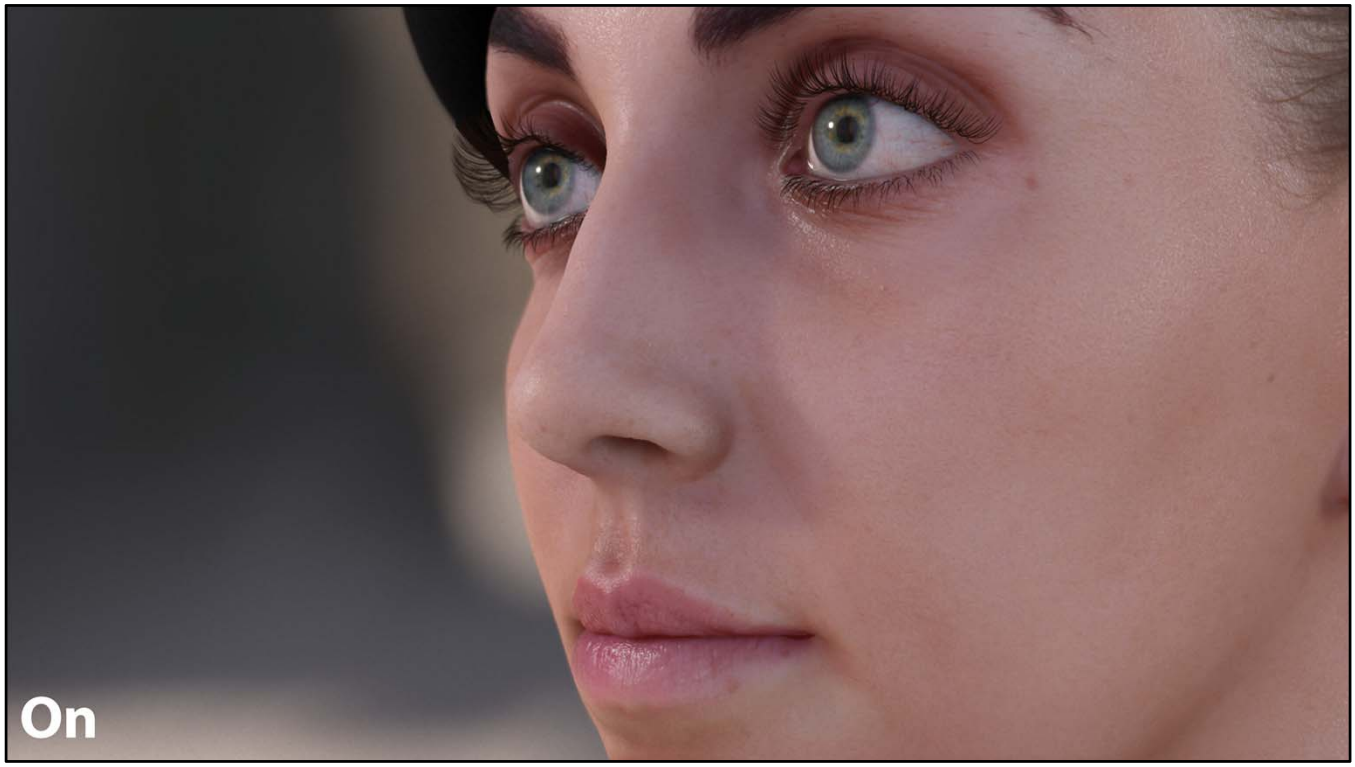
ACTIVISION | BLIZZARD

We’re now switching to the next big topic: eye rendering.

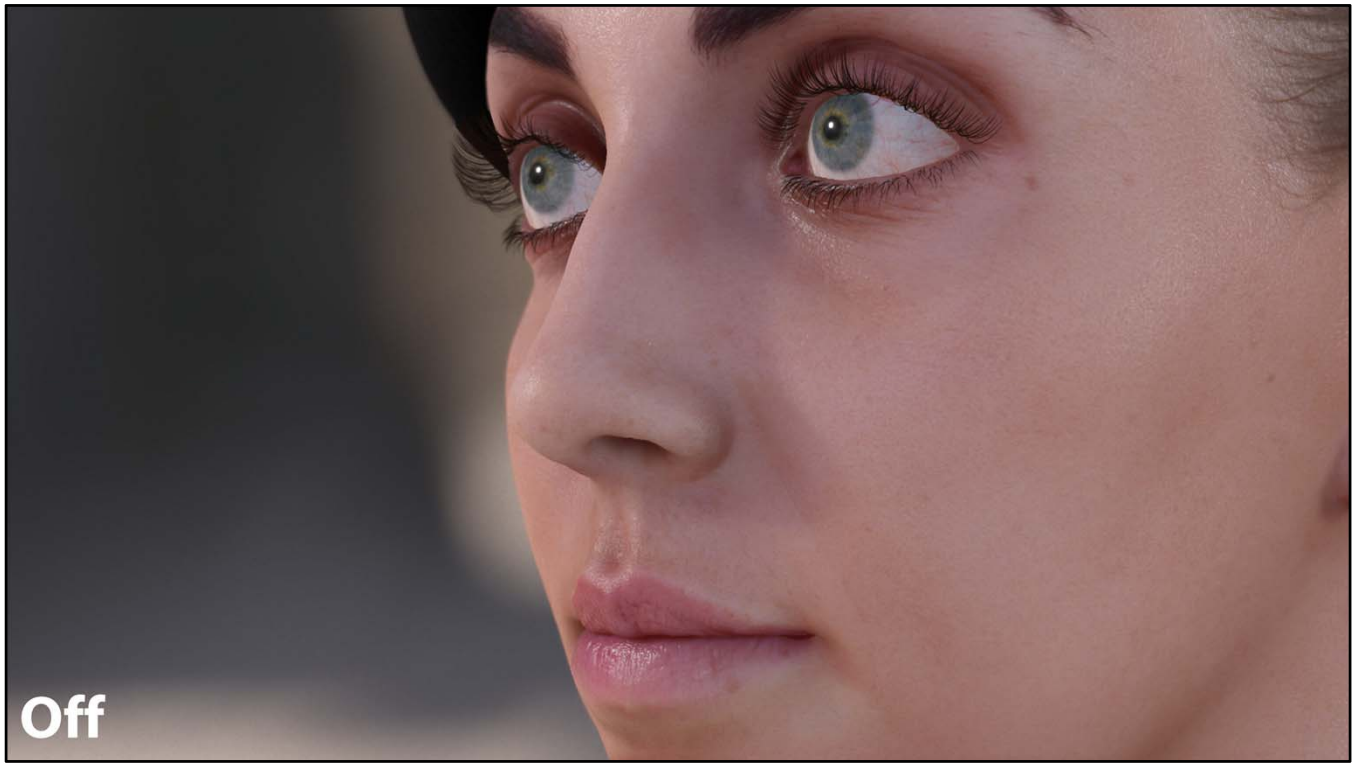
I’d like to start with some shots that highlight the importance of properly rendering eyes.

Notice the caustics, the refraction, the correct shadowing, the wetness on the eyelids…

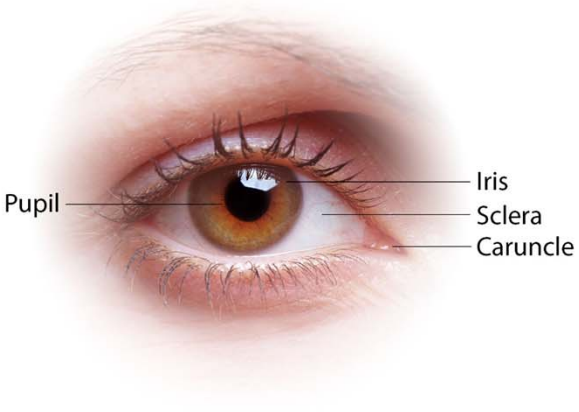

Off



Another shot, where refraction is more prominent.


Off

Eye Anatomy



Render Features

- Reflections
- Wetness
- Reflection Occlusion
- Screen-Space Reflections
- View Refraction
- Light Refraction
- Two-Layer Eye Shading
- Eye Redness Shading
- Ambient Occlusion

ACTIVISION | BLIZZARD

A lot of you will already know this, but let me quickly review the most important parts of the eye.

The sclera is the white part of the eye, the iris is the colored part, and the pupil the black part inside of the iris.

The cornea is a thin transparent layer that covers all these elements, and the origin of the refraction phenomena happening in the eye.

Regarding the eye rendering topics,

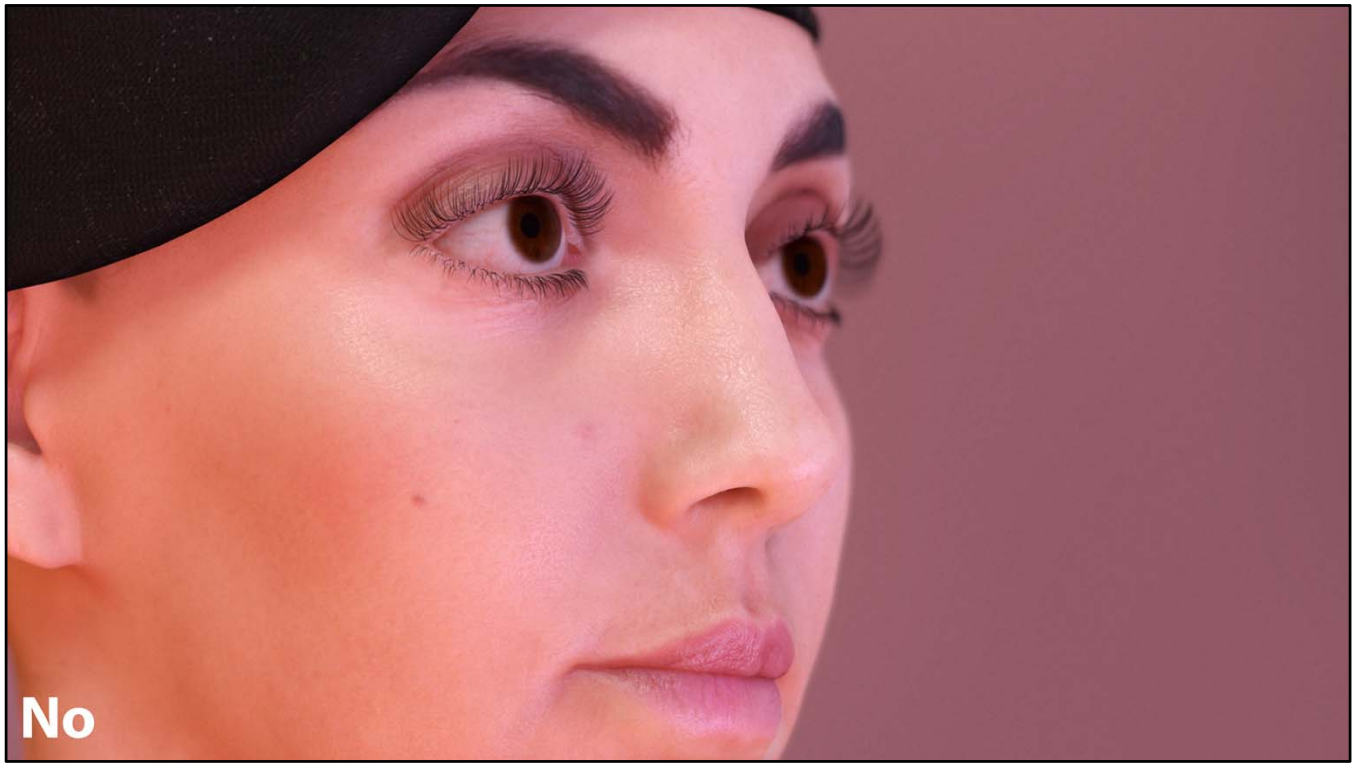
I'll talk about the features of the eye shader one by one, first motivating the problem,

then showing what happens in our real-time demo, and finally explaining the idea behind our solution.

Reflections (Motivation)

This is the first feature I want to talk about.

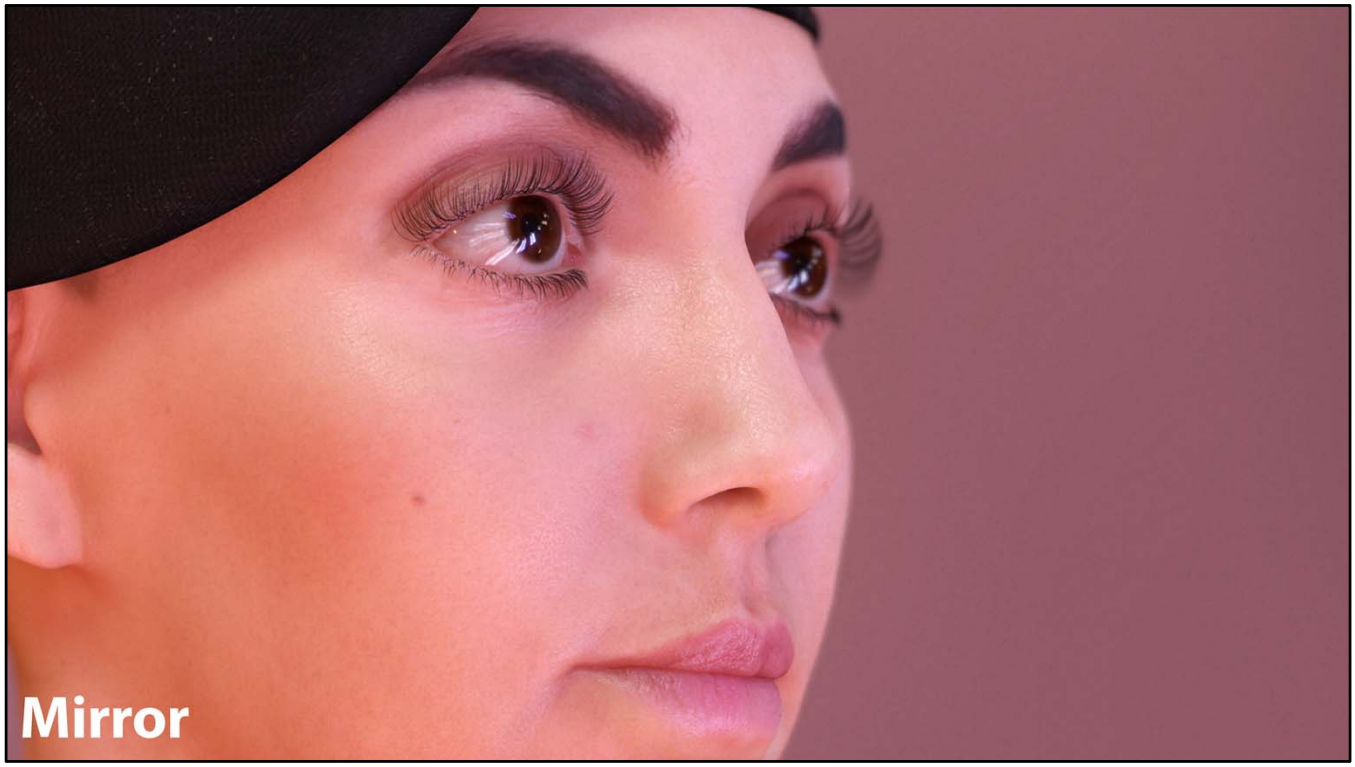
Reflections on the eyeball, which are probably one of the most important things you have to take into account.



Without them, you actually kill the character.

Direct light reflections are usually not enough for most lighting configurations.

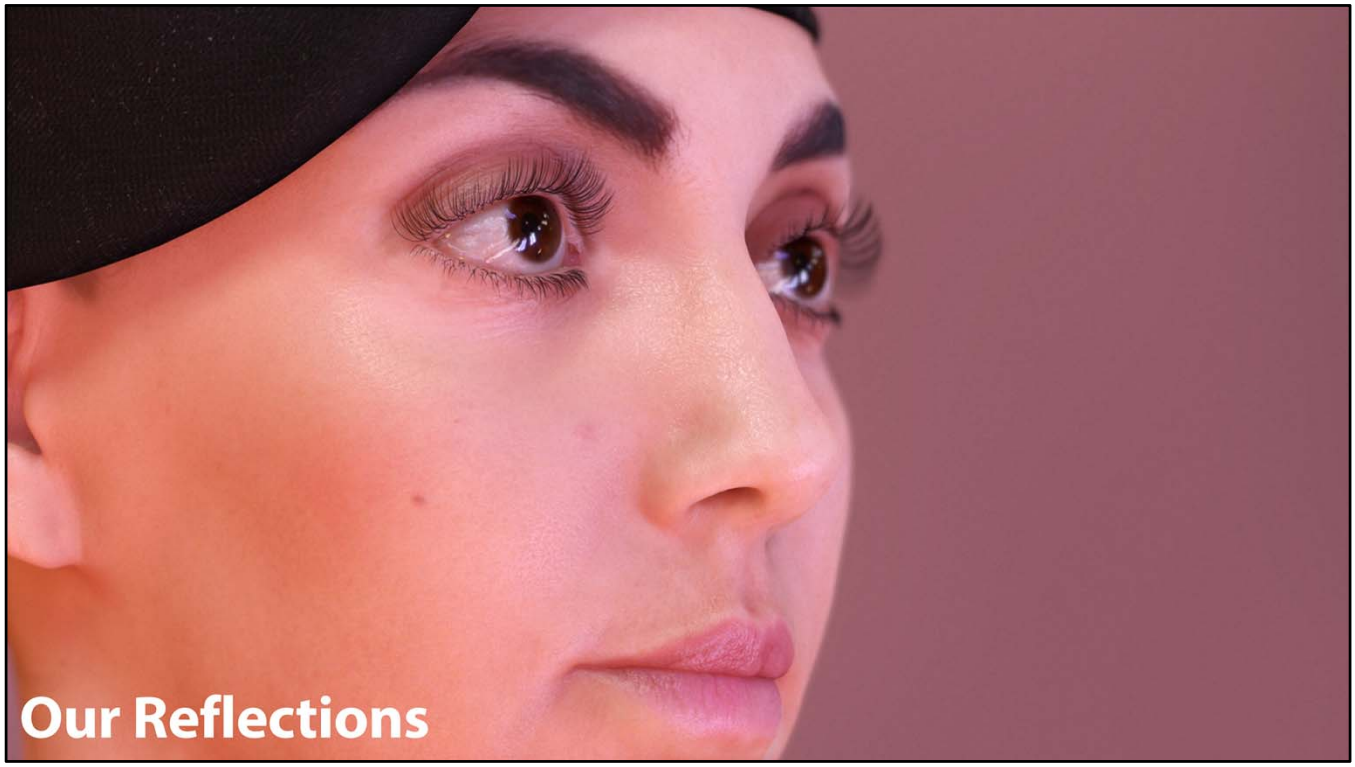
So, we tried regular environment map reflections…



…and obtained this result.

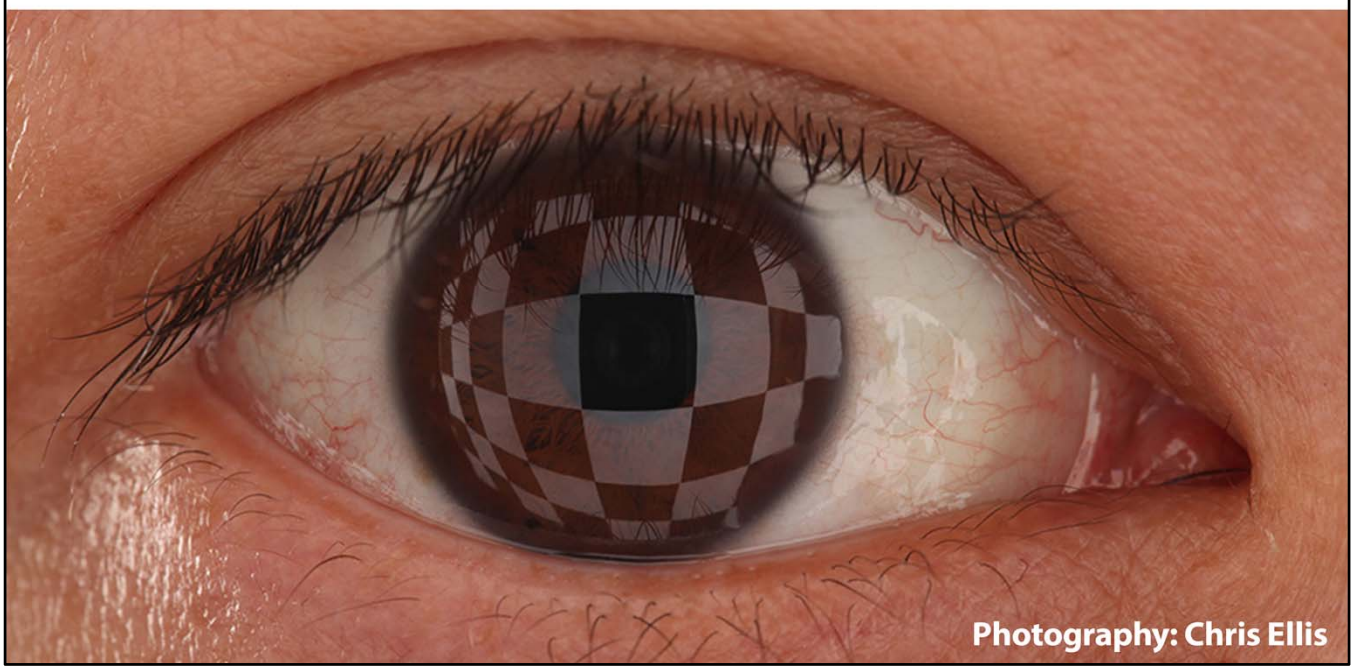
It actually looks like a mirror.

So, the challenge is in how to turn this mirror-like appearance…



…into something that feels more organic and wet.

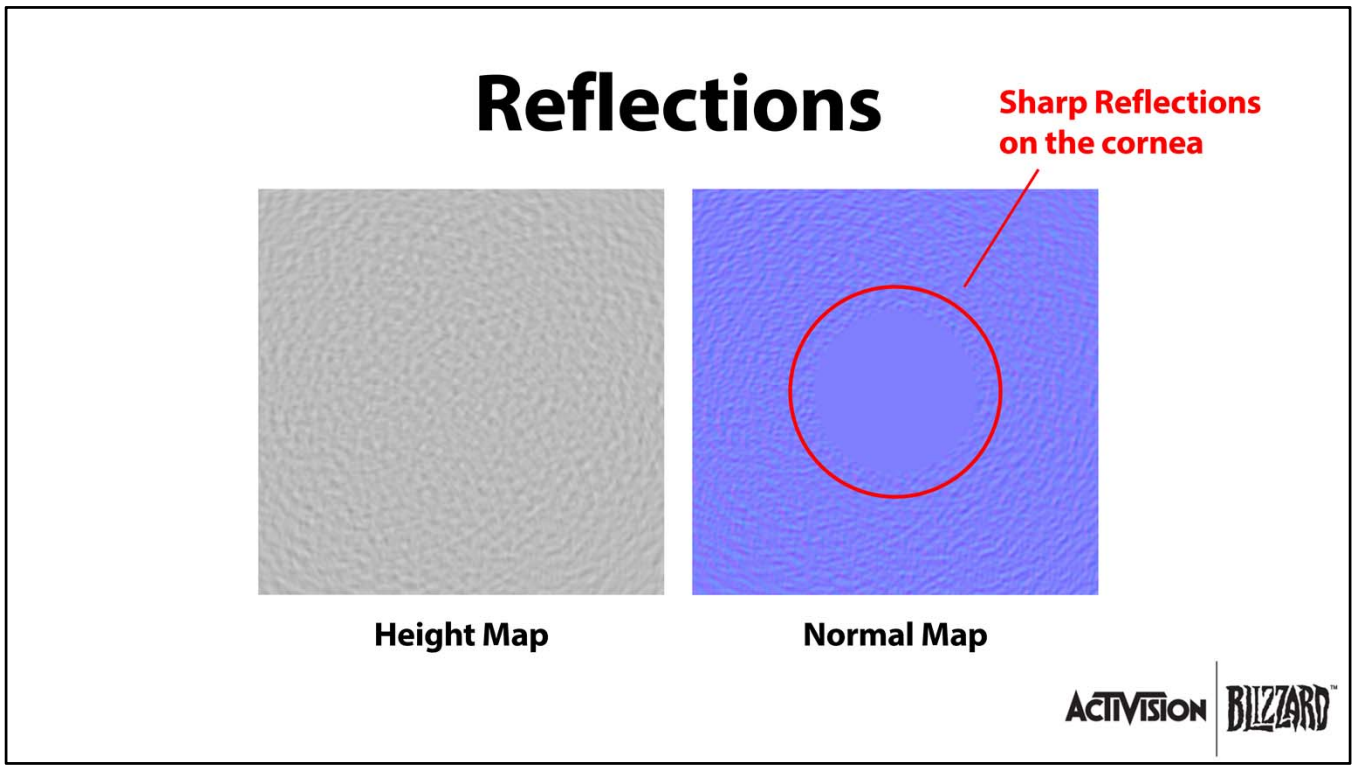
Reflections (Motivation)



Looking at real eyes, we realized that you can see strong sine-like distortions, caused by low-frequency changes
in the normals of the fluid surface.

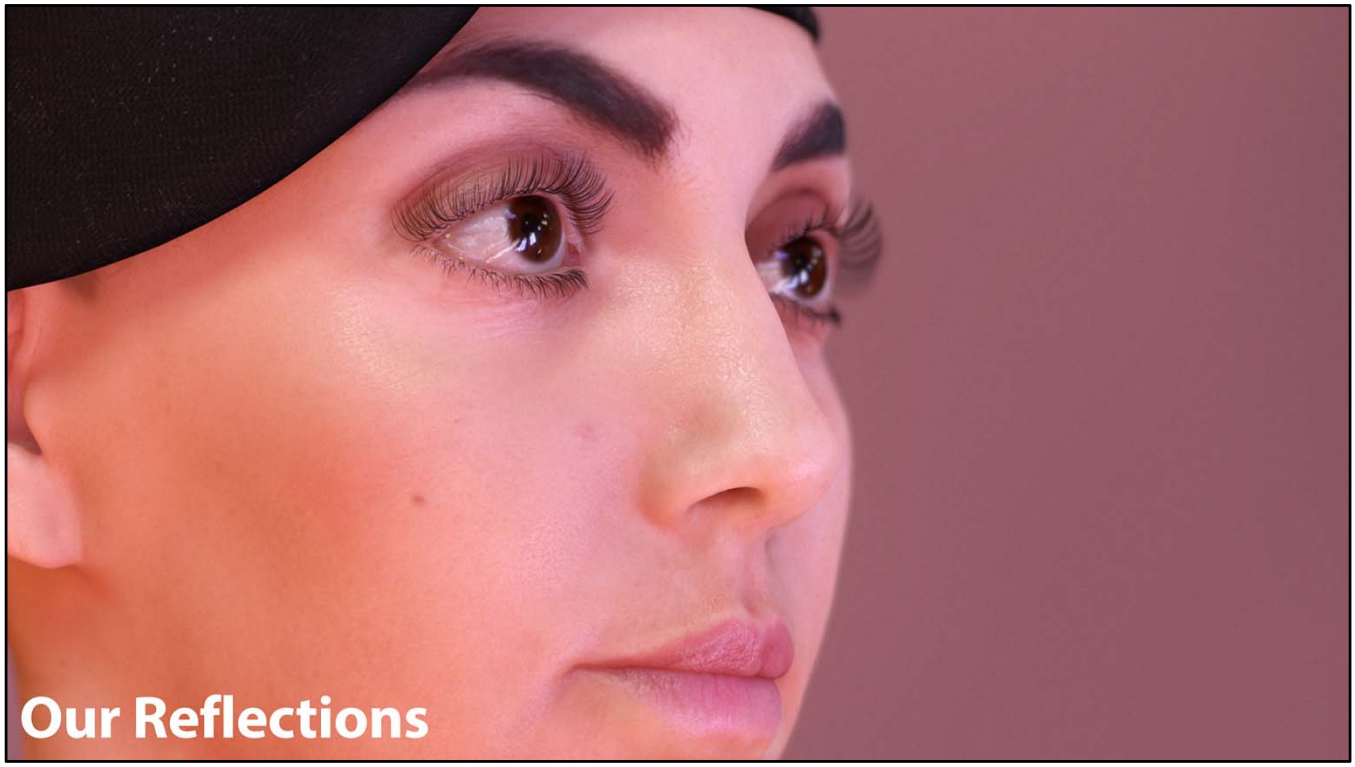
They only happen on the sclera, whereas reflections on the cornea are perfectly sharp.

(Probably because the cornea has a different internal structure that enables fluid to sit flat on top of it, to avoid distortions)



So, we created a height map using sine functions, and then generated a normal map from it.

(Note, the approach we presented on [Jimenez2012] *Separable Subsurface Scattering and Photorealistic Eyes Rendering* is not completely correct. We no longer see any reason to have an authored gloss map for the eyes, as we suggested in the presentation.
There is always tear fluid covering the eye, even when it is dry; so, there is no reason to have puddles of tear fluid in the eyes.)



This little change improved the quality of the reflections significantly.

But we are halfway done.

Wetness

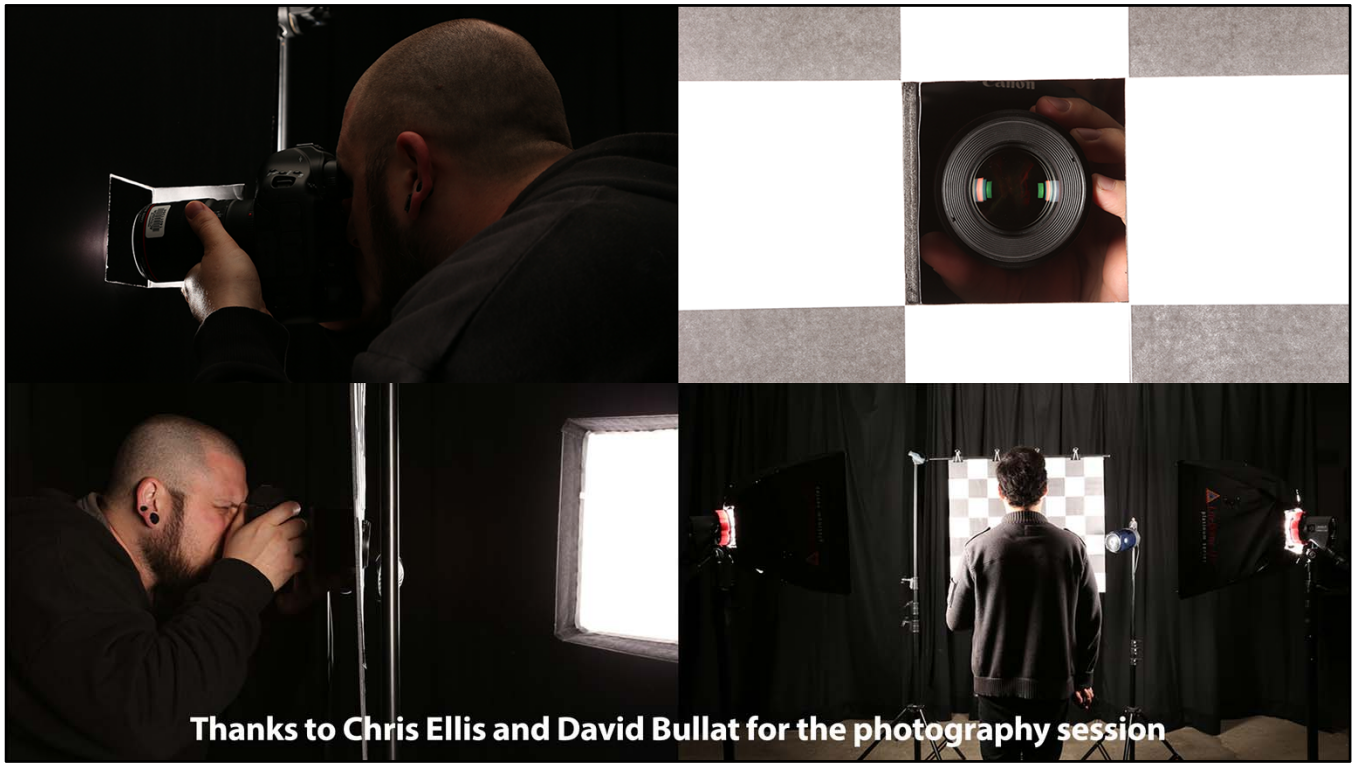
- **The next question we asked ourselves:**
  "What causes an eye to look more wet?"

- **To discover that, we captured:**
  - **Eye in normal conditions**
  - **Eye after blowing air onto it for a minute (dry)**
  - **Eye after we stop blowing air (wet)**

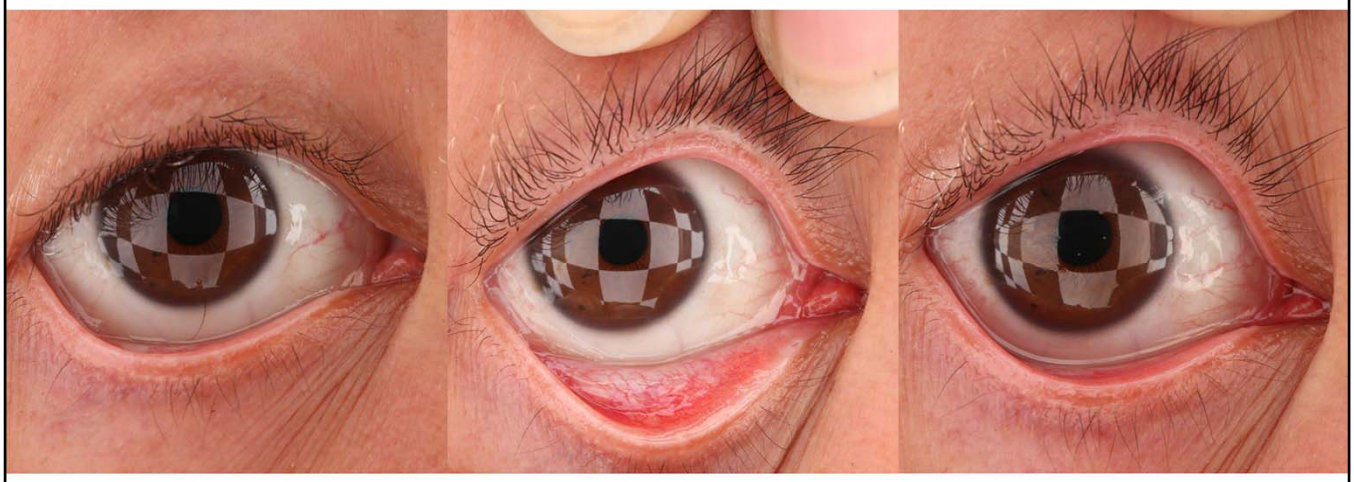
ACTIVISION | BLIZZARD

We tried to discover the visual clues that cause an eye to look wet.

For that, we shot eyes under different conditions…



...using a checkboard to have predictable reflections.



The main different we found between a dry and a wet eye,
as you will see in this sequence is the increase in surface bumpiness
as the eye wetness increases.

Wetness

- **Low frequency bump height seems to become higher**
- **To render this effect:**
  - **Add a low frequency wetness bump layer to the previous normal map**
  - **Blend strength depends on wetness**

ACTIVISION | BLIZZARD

Adding a low frequency normal map layer can give very important clues about the wetness of the eye.

Here you have various renderings for low…

…mid…

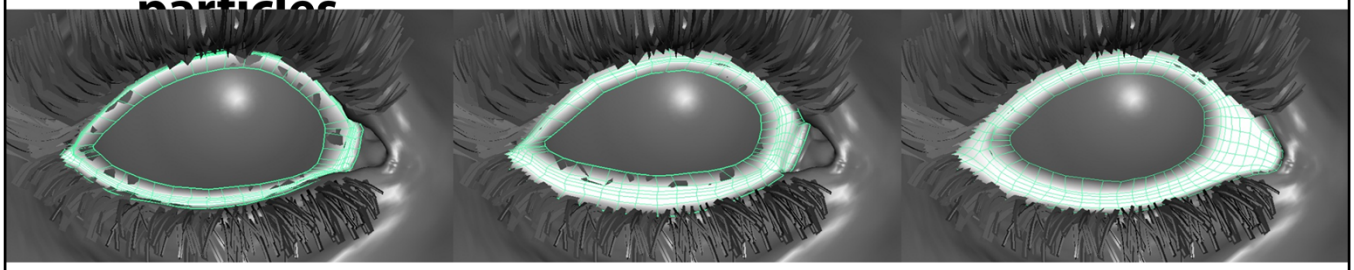
…and high wetness.

In these images, we are lowering the frequency of the wetness normal map layer as we increase the wetness value.

Similarly, we increase its strength as we increase the wetness value.

Wetness Geometry Blendshape

- We created three different geometries for wetness
- We "blend shape" and render them using soft particles

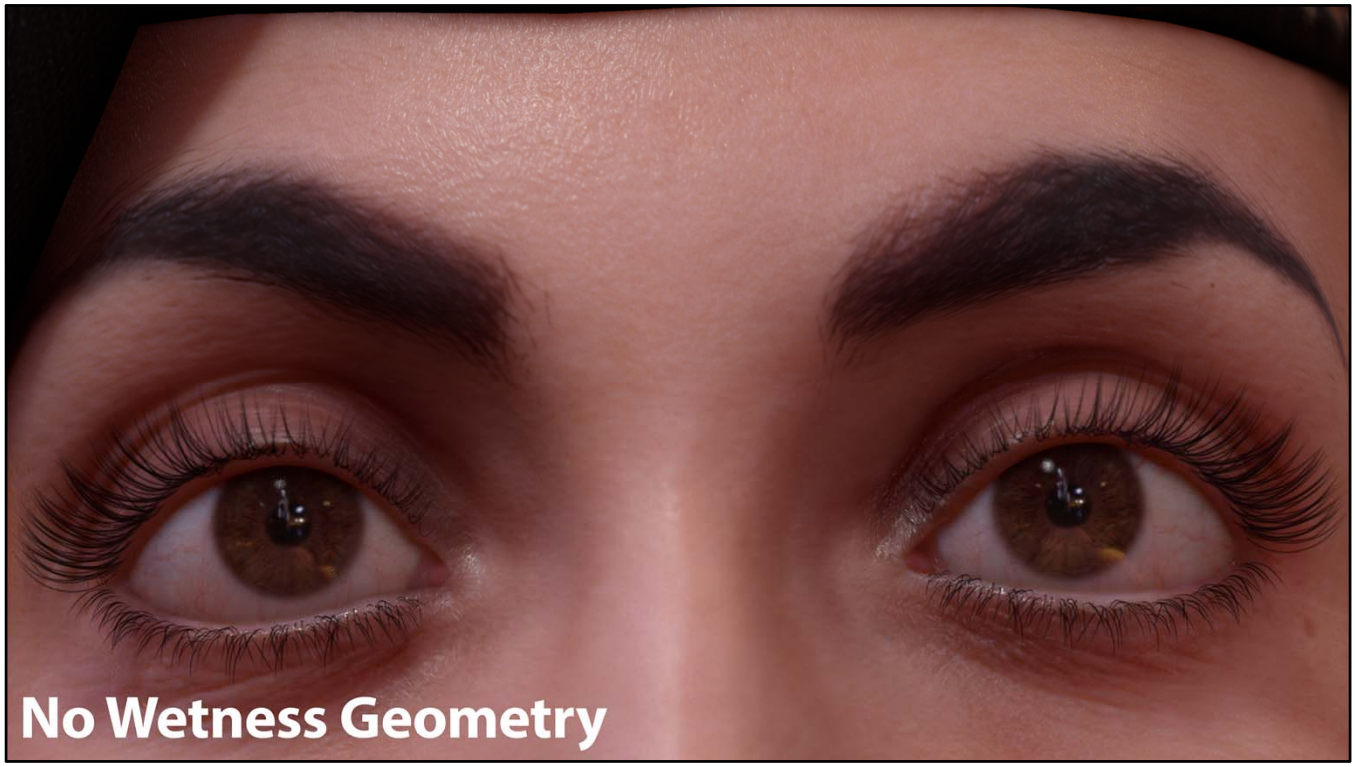


Low Mid Hi

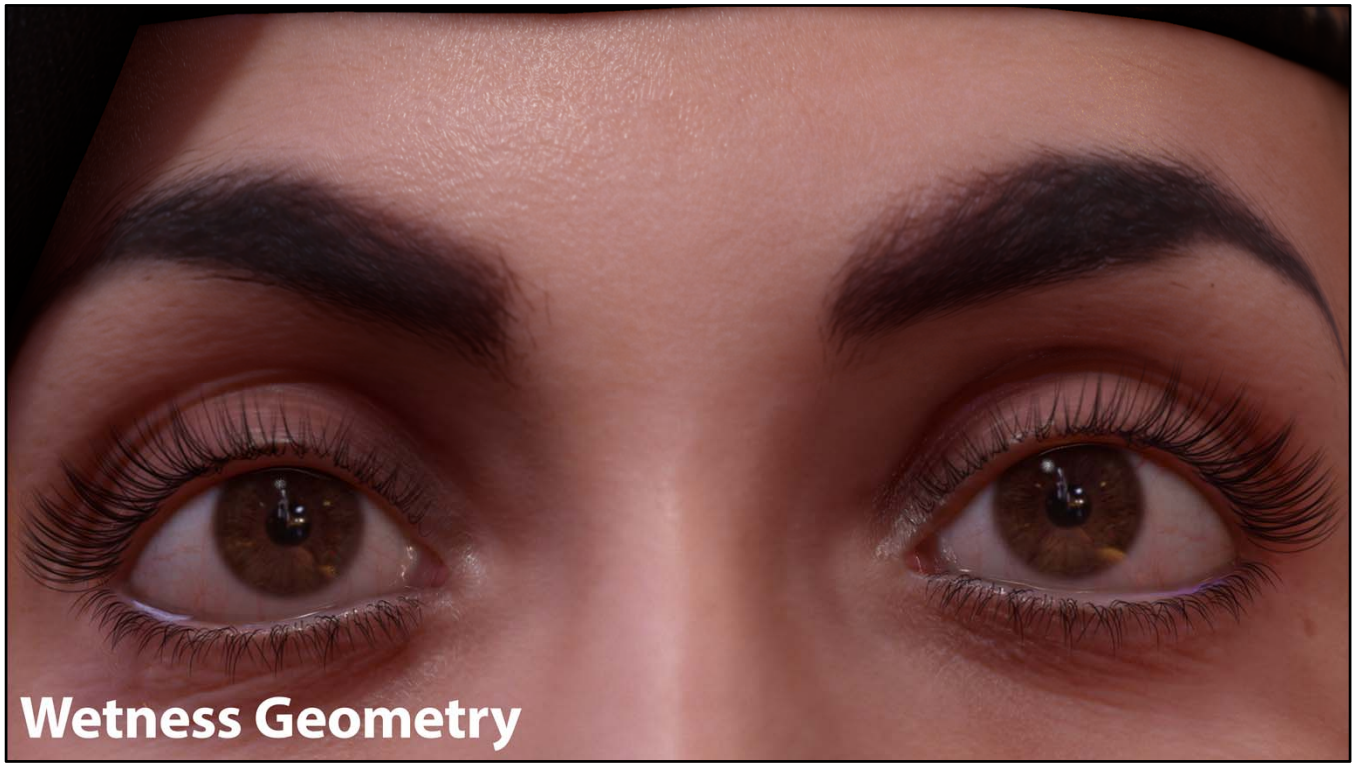
The other visual cue we found to be important are the specular lines in the eyelids.

To model this, we created different geometries for simulating different wetness conditions, and just blend shape them.

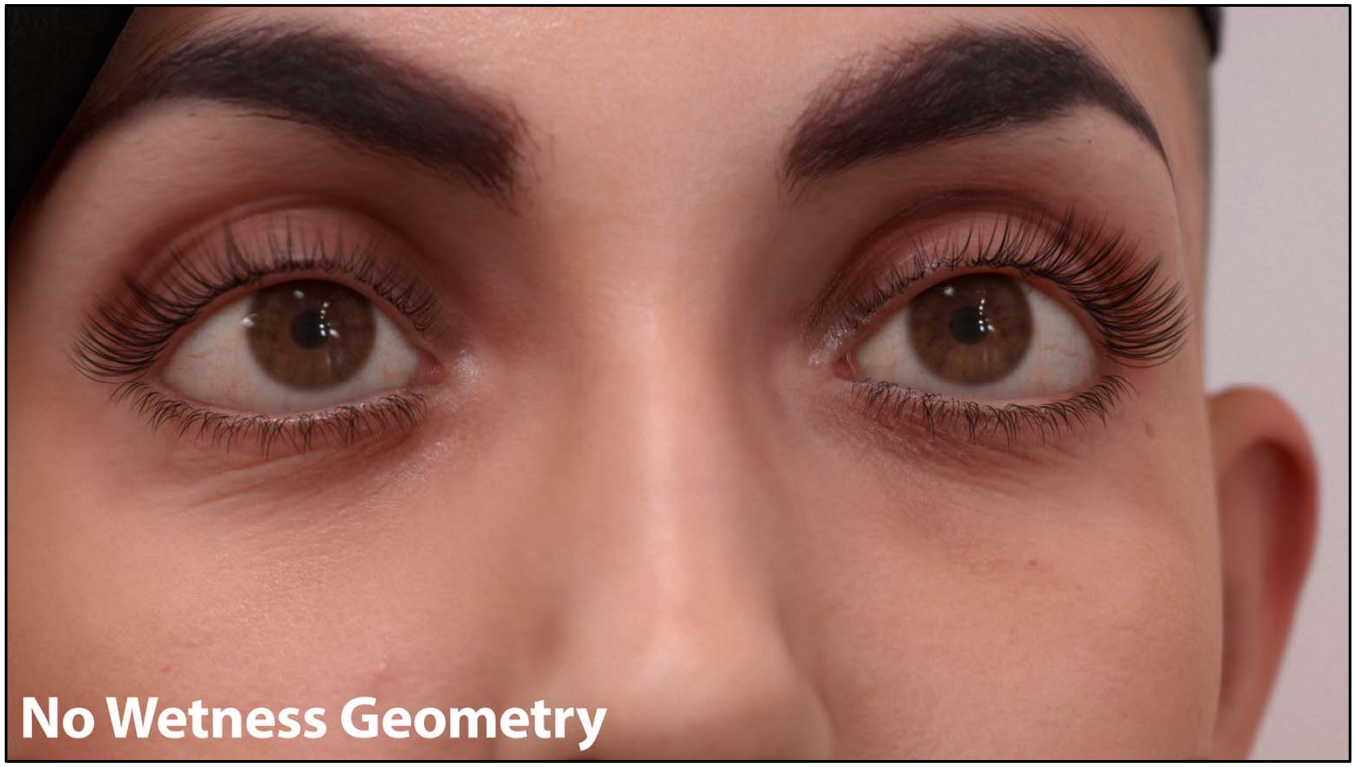
Note that the top part of the geometry is not as important as the bottom one, so it could be omitted for performance/authoring time reasons.



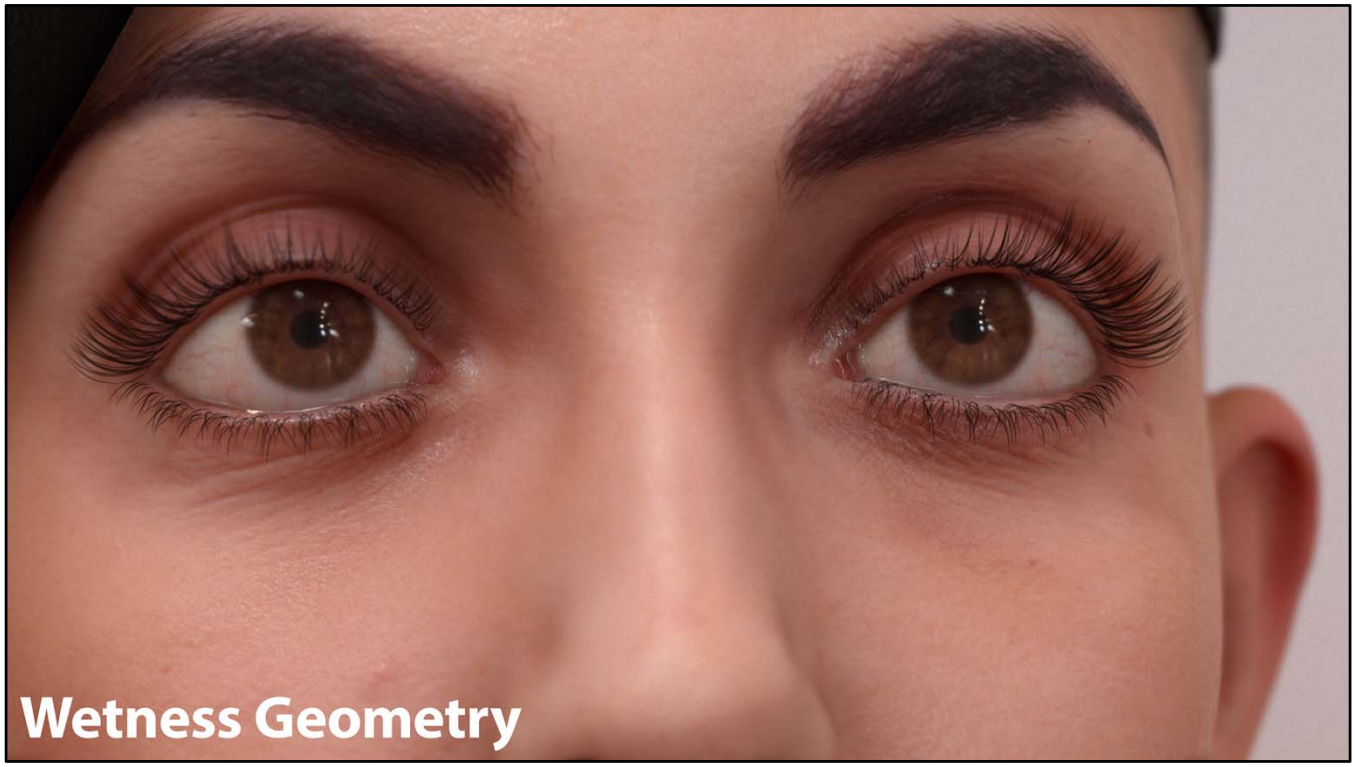
Here you have a shot without wetness geometry…



…and here with it.



Just another comparison…


Wetness Geometry

For believable macro shots we also found it important…

…to blur the details behind the wetness geometry

We implemented this blurring in the runtime,

using a screen-space blur when rendering the wetness geometry.

This also softens the (usually hard) line between the eyeball and the skin.

This is one of those little things that adds realism without you noticing its existence.

The Sparkle of the Eye

- **We have covered just two aspects that give clues about eye wetness:**
  - **Tear fluid distortions**
  - **Tear fluid accumulation on eyelids**
- **But there's more to it:**
  - **There is evidence that reflection intensity changes depending on wetness:**
    - **[Goto2011]** ***The Sparkle of the Eye: The Impact of Ocular Surface Wetness on Corneal Light Reflection***
  - **However the reason is not yet known to science**
    - **Gloss change due to different tear composition?**
    - **Reflectance change due to a multilayer thin film?**
    - **Thin film interference?**

Thanks to Sébastien Lagarde for the countless conversations about the topic!

We think we are still far away from the complete answer to eye wetness.

Wetness also produces changes in reflection intensity. You will probably have heard about the sparkle of the eyes,
which can happen when a person is extremely happy, extremely sad… or even jealous.

You may be thinking: it's wet, of course it has more intensity.

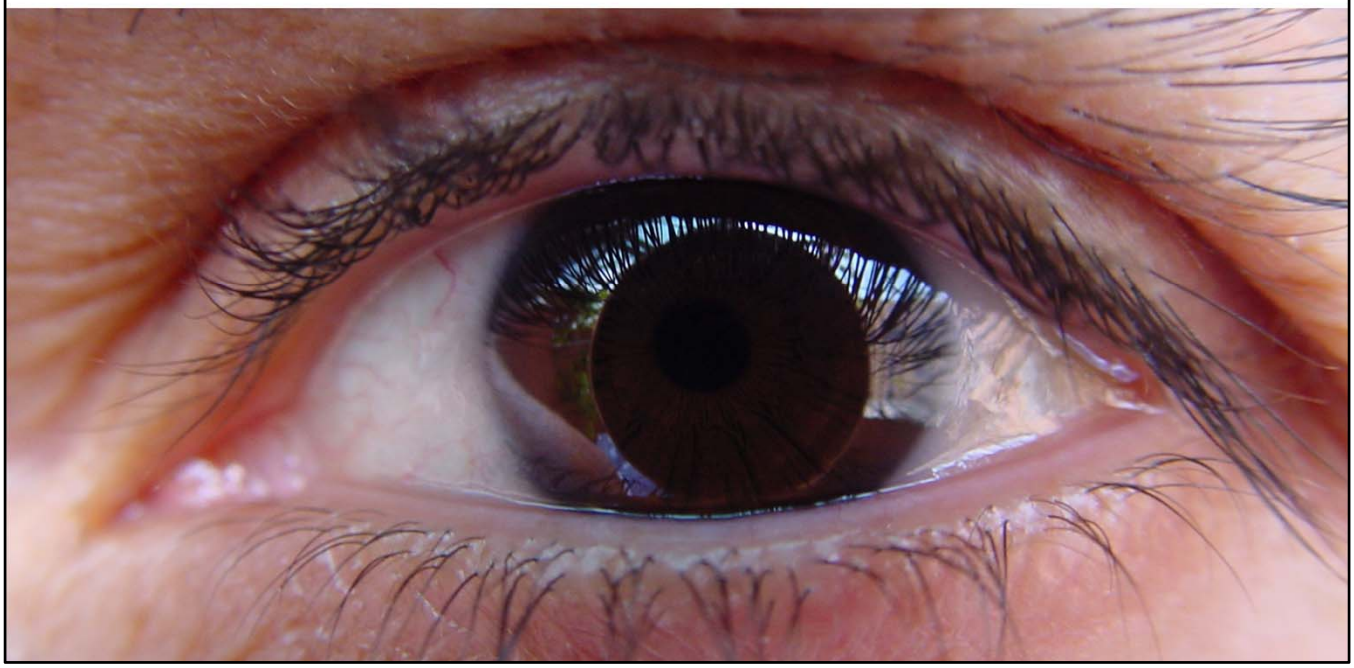
But the eyeball is always covered by tear fluid, even when it is dry. So, the reflection intensity should always be
that of the tear fluid, regardless of how wet it is.

This means it is not a gloss change due to tear fluid filling the microstructure of the eye as it becomes wetter.

It could be a gloss change due to a change of tear composition, maybe due to reflectance changes in the tear multilayer structure,
or even because of thin-film interference.

The reason is still unknown as of today.

Reflections Occlusion (Motivation)



Ok so, we now have better looking reflections, but what happens with the eyelids and eyelashes?

They should be blocking the light from the environment.

But they don’t.

They are important as they help make the eye feel really integrated into the face.

Baked Reflection Occlusion

Reflection Occlusion Map

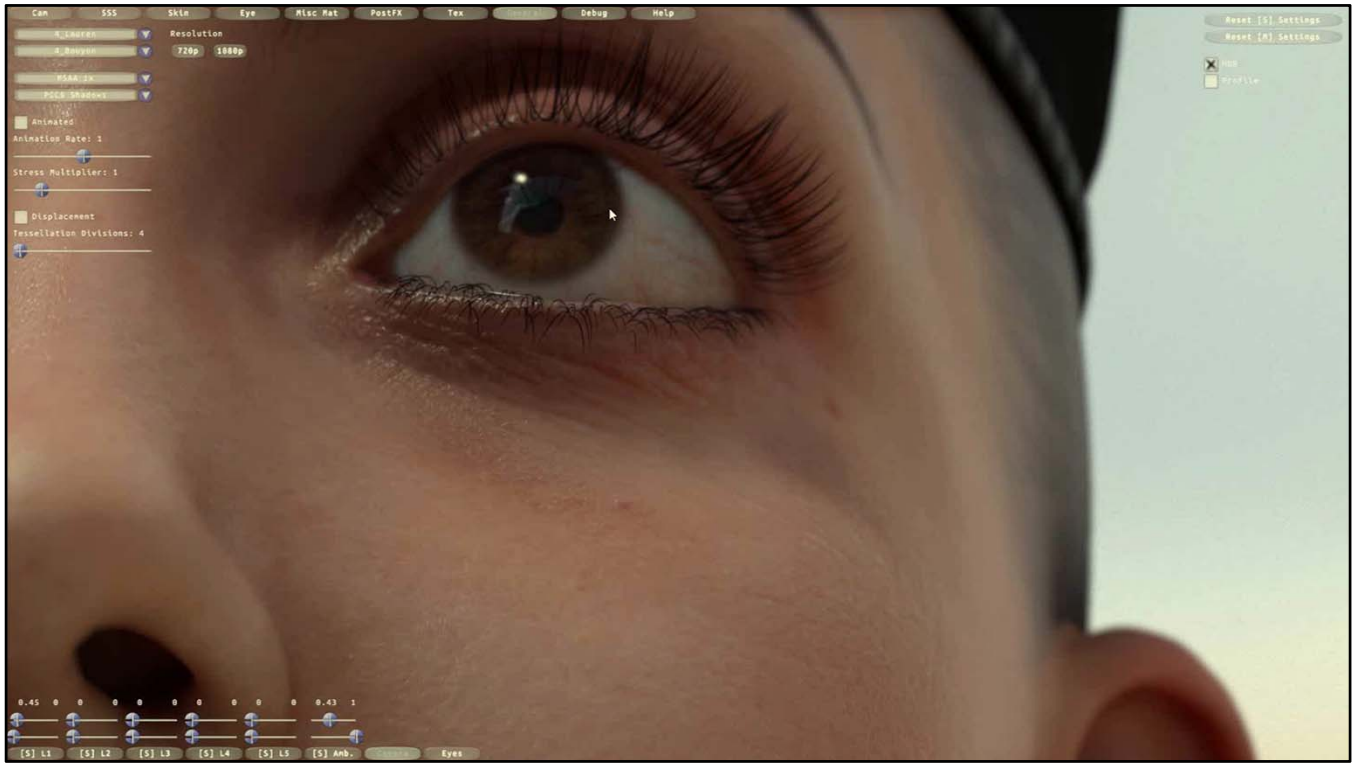
Shader

```
#if EYE_REFLECTION_OCCLUSION_PARALLAX == 1
viewW.y = -viewW.y;
texcoord -= 0.14 * viewW.xy;
texcoord -= 0.01 * normalT;
#endif
return aoTex.Sample(AnisotropicSampler, coord).g;
```

ACTIVISION | BLIZZARD

For this we simply use a baked reflection occlusion map using Turtle

and apply a little bit of parallax depending on the view situation.



And these are the results.

Notice how the reflection of the eyelids changes as the camera moves.

Screen-Space Reflections (Motivation)

Occluding the environment is not enough in some scenarios.

Notice in this shot how the nose color is being reflected in the eye.

Screen-Space Reflections (Motivation)

Screen-Space Reflections

- **In harsh lighting configurations, local reflections are important**
- **We explored the possibility of using screen-space reflections for the eyes (see [Kasyan2011] *Secrets of CryENGINE 3 Graphics Technology*)**

Thanks to Xian-Chun Wu for the implementation!

So, we also explored the idea of using screen-space reflections for the eyes, as an alternative or complement
to the baked reflection occlusion approach.

For occlusion, and for actually reflecting nearby objects (not just blocking the environment).

And we found them to add coherency to the lighting of the eyes with respect to the face.

Notice the reflection of the nose onto the eyes and how this helps to make the eyes respond to the surrounding lighting better.

And perceptually, it makes the eye look truly wet.

Not a deal-breaker, but if you already have screen-space reflections in your engine, it's worth activating the effect for the eyes.

Important tricks to make screen-space reflections look good for eyes:
- Use a Fresnel function (like Schlick's approximation)
- Attenuate the effect in the distance, to avoid a sharp, noisy reflection on the eyes. Note that this isn't physically accurate, but we found it to yield better visual results.

The usage of the screen-space reflections is two fold:
- To occlude the environment map if we got a hit in the ray march (like the reflection occlusion parallax technique).
- To add reflections of nearby objects.

This solution is in general visually better than using baked reflection occlusion, as it avoids authoring steps and also takes into account reflection of the face onto the eyes. However it cannot reflect the eyelashes, has more aliasing, and is way more expensive.

In our demo, we use them both: baked reflection occlusion for blocking light from the

environment (and keeping the nice baked reflections of the eyelashes), and screen-space reflections for dynamic effects like the nose reflection.

View Refraction (Motivation)

The next feature we want to show is view refraction.

This phenomenon makes the iris look like a very 3D structure, especially when you look from the side.

Failing to take this effect into account leads to a “cow-like” stare like this.

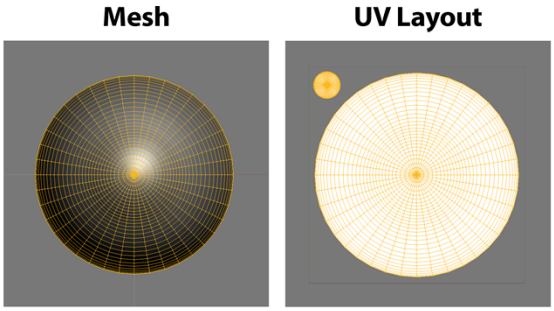
Whereas we would prefer to have something more like…

…this.

Notice how it has a much more natural look.

View Refraction

- **Parallax mapping**
- **Physically based refraction**



ACTIVISION BLIZZARD

For solving this we used two different approaches.

The first one involves using parallax mapping, and the second one using accurate refraction calculations.

To ease the implementation we require the eyeball UV layout to be its frontal projection (both the layout itself and the size).

In other words, we use an eye mesh with:
- XY coordinates in 0..1 range
- UV matches XY coordinates

This allows us to convert from world space to texture space for refraction tricks.

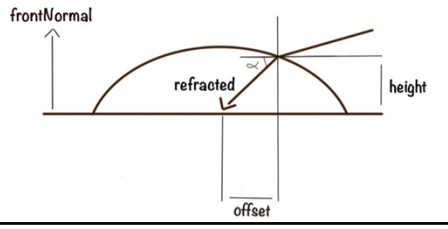
Parallax Refraction

Parallax Refraction

```
// Calculate distance from iris to the cornea:
// (Alternatively, use a displacement map)
float height = max(-positionL.z – eyeIrisDepth,
0.0);
float2 viewL = mul(viewW, (float3x2) worldInverse);
float2 offset = height * viewL;
offset.y = -offset.y;
texcoord -= parallaxScale * offset;
```

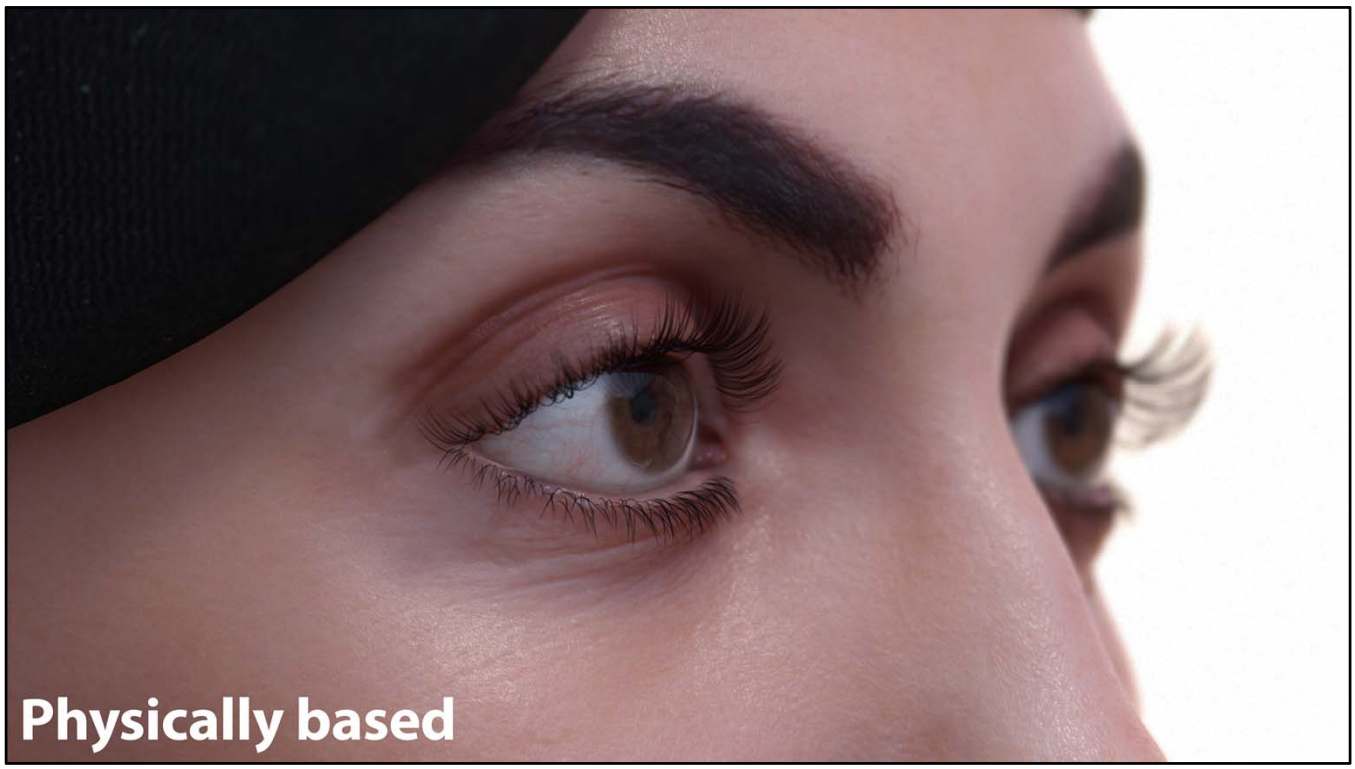
Physically based Refraction

```
// (For refractedW, check out Real-Time Rendering
//  book, section 9.5. Refractions)
// (Note the W in heightW, it should be in world
//  space in this case)
float cosAlpha = dot(frontNormalW, -refractedW);
float dist = heightW / cosAlpha;
float3 offsetW = dist * refractedW;
float2 offsetL = mul(offsetW, (float3x2)
worldInverse);
texcoord += float2(mask, -mask) * offsetL;
```



The parallax mapping hack is very simple, as you can see in this shader code.

The parallax refraction approach is able to produce pleasing results, comparing well with…

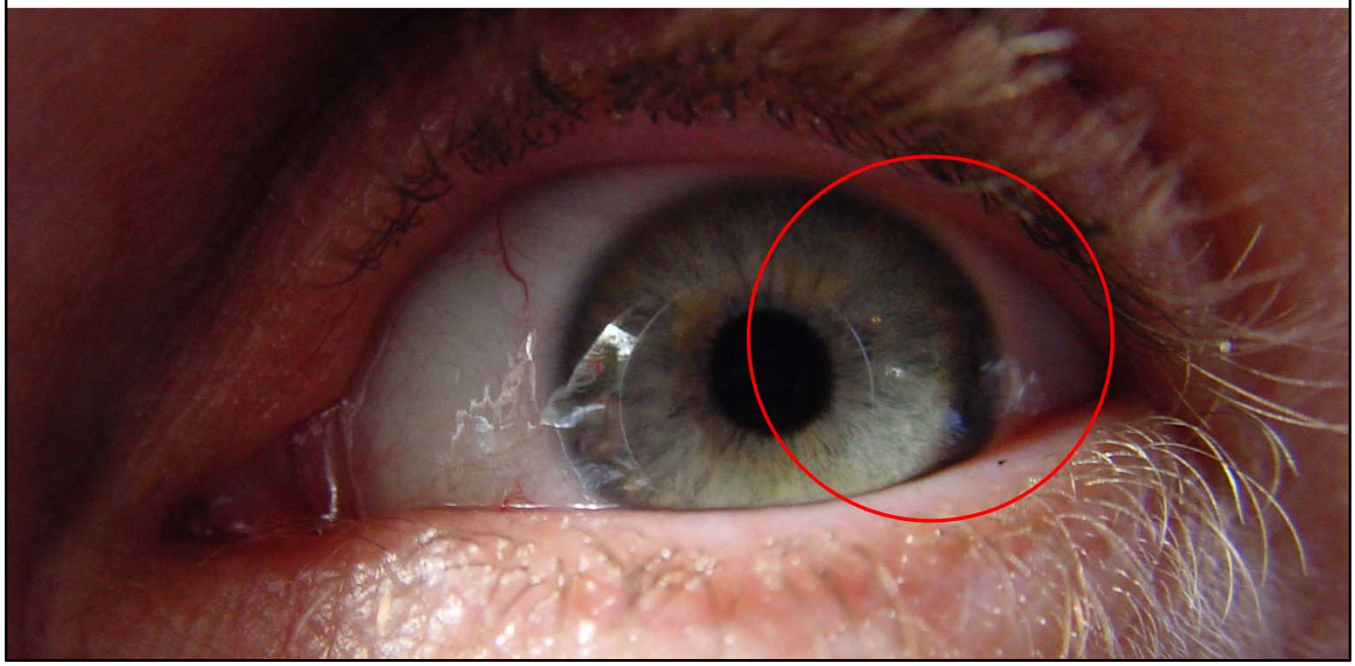


…the accurate approach for some view angles.

However, as you turn the camera to very grazing angles, the illusion starts to break.

Notice how using accurate calculations leads to much better results in this case.

Light Refraction (Motivation)



So, we've got reflections and view refraction, but light also refracts when it travels through the cornea.

Notice the shadowing effect on the right. This comes from refraction, which changes the light direction in such a way,
that this area is totally unreachable from this specific angle.

So, this is the starting point.

Here we don’t have light refraction, which leads to a very flat and unnatural appearance.

And this is with light refraction. Notice the shadowing on the iris.

Then, increasing the light strength and moving the light to a grazing angle produces caustics on the eye.

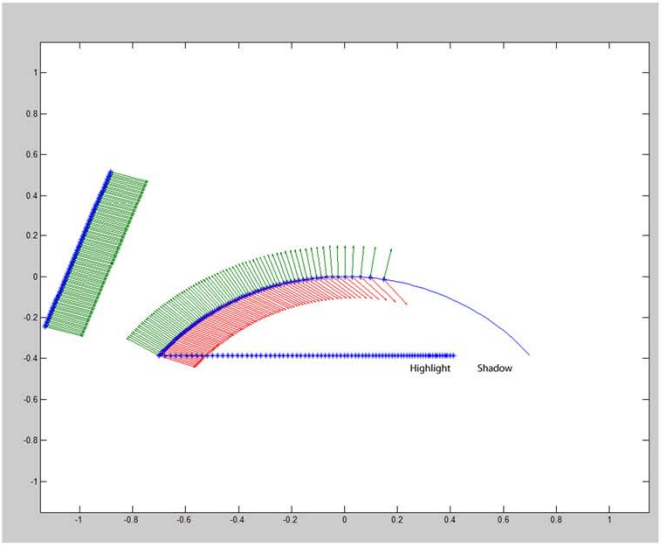
And again, if we disable this feature…

…the eye totally loses its 3D appearance.

As you know, the usual thing is to bake this shadowing into a diffuse texture.

But we wanted to go a step further and have this effect be fully dynamic (and also cheap).

Schematic View



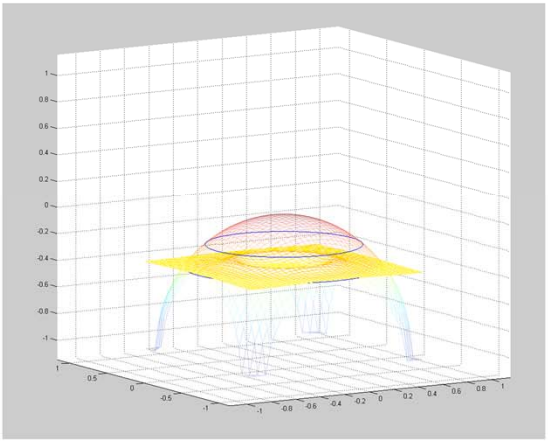
ACTIVISION | BLIZZARD

This figure shows light refraction in the eye.

Notice how the rays coming from the area light on the left change direction in the cornea, generating a highlight, or caustic, and an area in shadow.

Precomputed Light Refraction

- **[Francois09] *Image Based Modeling and Rendering of the Human Eye***
- **Analytical eye model**
- **Baked using photon mapping**



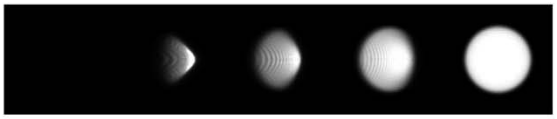
ACTIVISION BLIZZARD

So, we worked on the idea of precomputing the irradiance for all possible incoming angles.

It's an idea that has already been explored in previous work.

Precomputed Light Refraction

- **Precompute using photon mapping into a 3D texture**
- **Integrate all directions for ambient lighting**



←Inclination angle→

ACTIVISION | BLIZZARD

The results are stored into a 3D texture, which encodes, in each slice, the irradiance of the iris for a specific inclination angle.

For probe lighting we integrate all possible directions into a single map.

A much better approach would be to encode this into spherical harmonics, as it would be able to provide some degree of directionality.

Shadow Mapping Tips

- **Light refraction generates lit surfaces at angles where shadow mapping is completely blocking light**
- **We found no trivial way to refract shadow rays using shadow mapping**
- **Fix: don't render eyeballs into the shadow maps, so that they don't block the caustics**

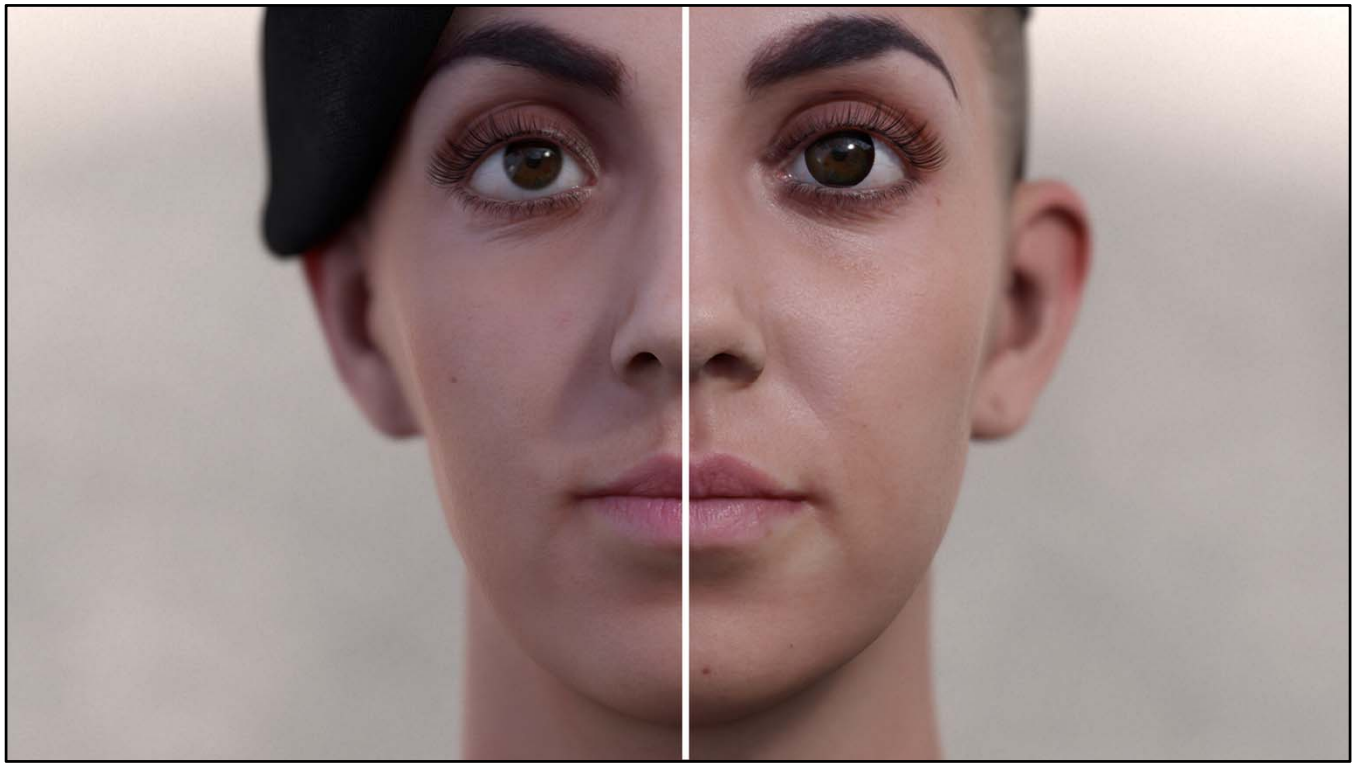
ACTIVISION | BLIZZARD

Shadow mapping is a big foe of light refraction in the eyes.

For the angles where caustics start to appear, the caustic area is completely in shadow as the eyeball is self-shadowing
(take into account that light bends in the air/cornea interface).

We would need to refract the shadow rays before accessing the shadow map, but we've not found any trivial way to do it.

The fix for us is to not render the eyeballs into the shadow maps. This way they won't block the caustics!

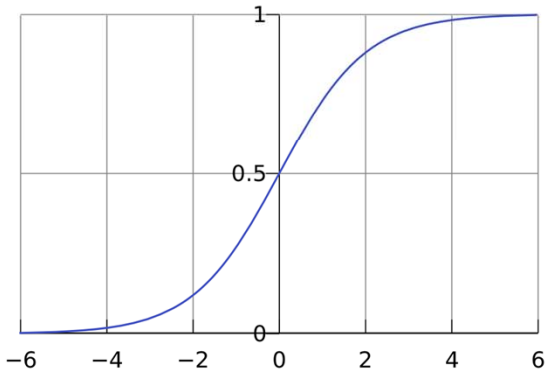


Something we want to point out: we render everything in a single mesh using two layers: one for the sclera, and other for the iris.

On the right you can see the two layers separated by a binary interface (It's either sclera or iris).

Limbus Rendering

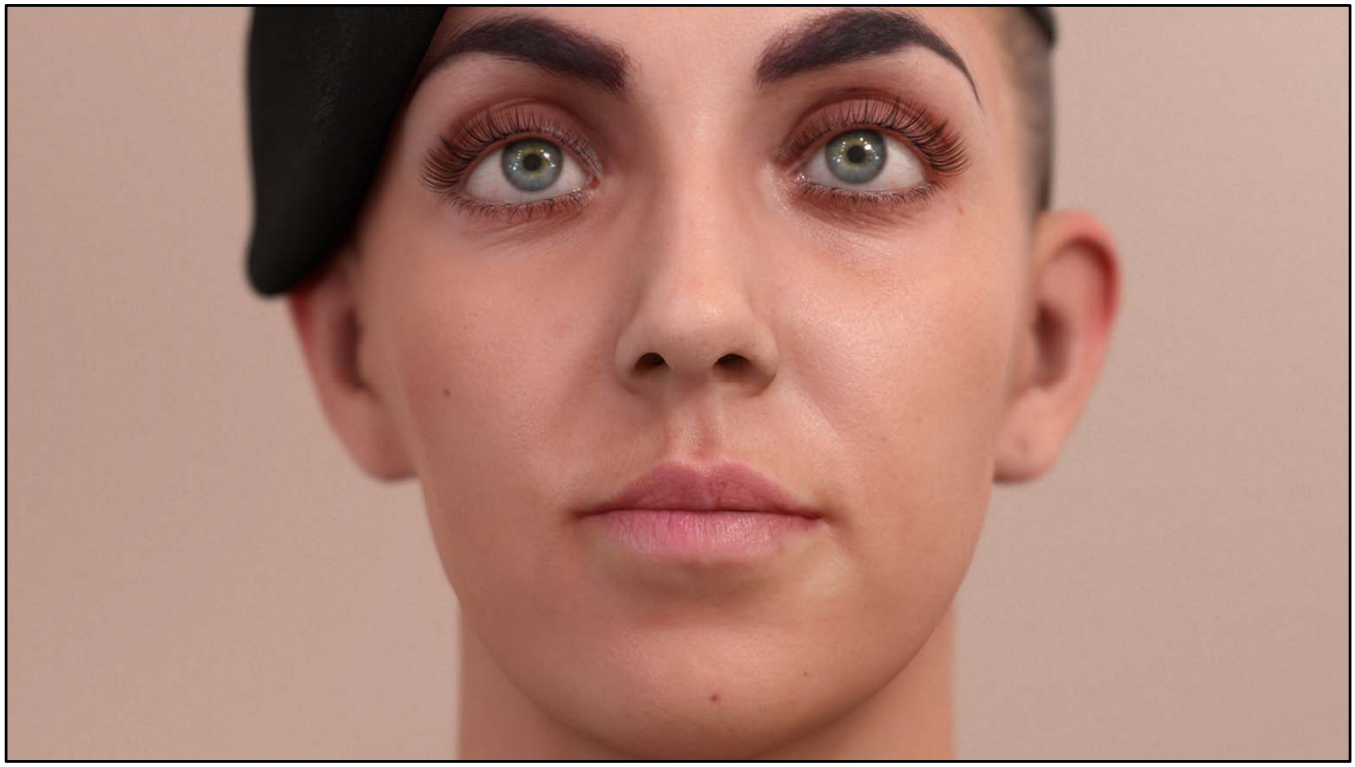
- **References point to the use of sigmoid functions**
- **[Iskander2006] A Parametric Approach to Measuring Limbus Corneae From Digital Images**



ACTIVISION BLIZZARD

We found references that point to the use of sigmoid functions for limbus size recognition, which is the inverse of the problem we're interested in.

So, we decided to use a sigmoid function to interpolate between the iris and sclera layers, which allowed for better results than a simple linear ramp.



Note that rendering two distinct layers actually enables the limbal ring (the dark area around the iris)…

…to disappear as we move the camera to the side, as happens in the real world, giving a more three-dimensional appearance.

(We encourage you to try this with a colleague with light eyes!)

Sclera Wrap

- Linear lighting produces very hard transitions in spherical objects
- This accentuates the fact that we're not taking sclera SSS transmittance into account
- Using energy-conserving wrapped lighting helps with it:
  - http://blog.stevemcauley.com/2011/12/03/energy-conserving-wrapped-diffuse/

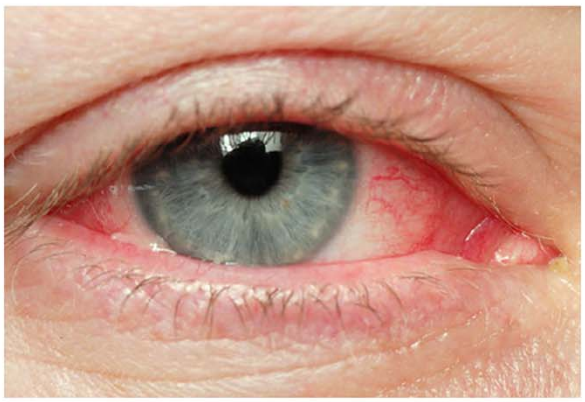
Thanks to Steve McAuley for developing the math and sharing the knowledge!

This is a shot using regular lighting…

…and this using energy-conserving wrapped lighting.

Notice how the smoother transitions between light and shadows allows for a more natural result.

Redness Shading (Motivation)



ACTIVISION | BLIZZARD

The final feature we will show is eye redness, which can be caused by several things, such as dry eyes, or just crying.

Note how pink eyes have sometimes a uniform colour,

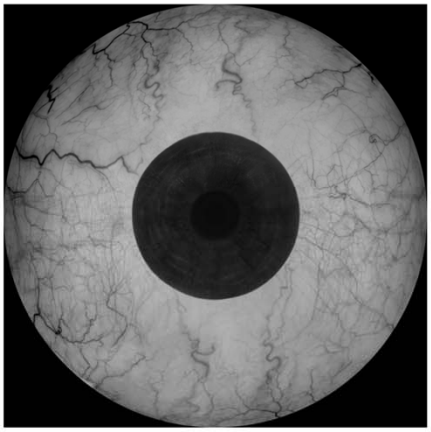
while in other cases only the veins become more reddish.

Here we have a render of normal eyes…

…and here with some redness applied.

Redness Shading

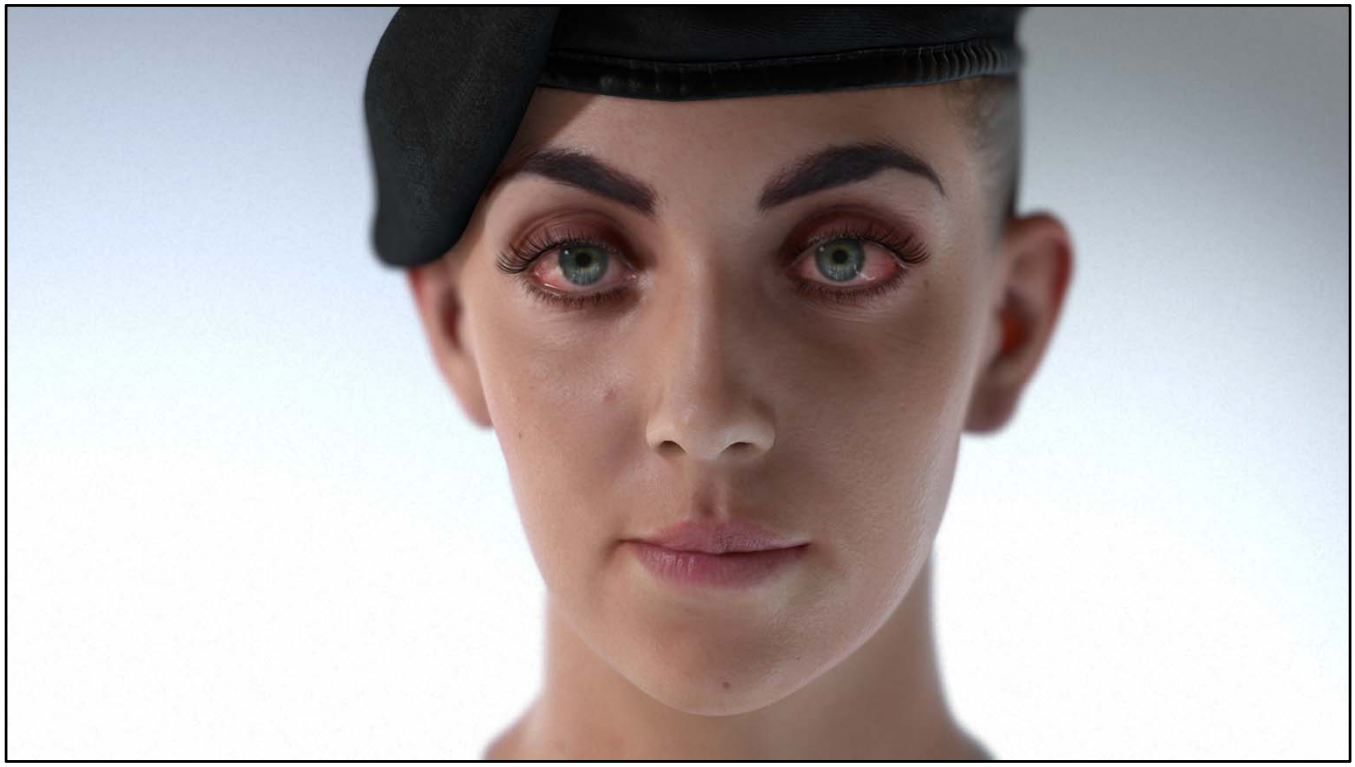
- **Independent control for veins and sclera is desirable**
- **Coincidentally, the blue channel of the sclera looks like a veins mask**



ACTIVISION | BLIZZARD

For the effect to be fully dynamic, we use an on-the-fly generated veins map to separate the veins from the sclera.

Using it we separately lerp the sclera and veins color to a more reddish tone in the runtime.



This can be coupled with the wetness simulation, to produce convincing crying effects.

Post FX

ACTIVISION BLIZZARD

Finally, post FX.

Full HD

- **The higher the resolution**
- **The harder the challenge**
- **At 720p/1080p render looks extremely synthetic/perfect**

ACTIVISION | BLIZZARD

In the past few years, not only have we increased the amount of bump detail in character modeling, but also the screen resolution.

And, as you can imagine, the higher the resolution, the harder the challenge. An image rendered at 1080p looks extremely synthetic.

Extremely perfect.

Post FX

- **This is where post effects come to the rescue:**
- **Probably the most important purpose of post effects is making the image less predictable**
  - **It's incredibly hard to reverse-engineer a photograph**
  - **And extremely easy to reverse-engineer most CG images**

This is where post effects come to the rescue!

You may be thinking: "But what do post effects have to do with character rendering?"

Stop that! It's actually more important than anything else.

It will help to hide technical problems and increase the photorealism by masking the synthetic nature of the image.

They help to make the image more fuzzy, and harder to reverse-engineer by our trained eyes.

When you look at a photograph, it's extremely hard to see what's going on.

That is not the case for CG, generally speaking.

To motivate the idea with images: if you look at an image from film, like this shot from Blade Runner,
you will see that it’s anything but perfect.

Notice how the depth of field and grain soften the image, and make it less predictable.

In this one, notice the atmosphere of the shot and how depth of field helps to add three-dimensionality to the character,
and a sense of distance.

Just to show the impact of these effects on rendering, I'll show some images from our demo.

Here you have an image rendered with depth of field.

Here without.

Notice how the lack of depth of field reveals the fact that we don’t have facial hair on the character,
showing its perfect and synthetic silhouette.

On the other hand, we all grew up with and are used to the presence of grain in film.

And depending on the film stock, it can be quite noticeable.

Notice that these kinds of imperfections make the image more believable…

…without them, the perfection of a render – especially in low lighting conditions – can quickly ruin the realism.

In low light, less photons arrive at the sensor of a camera, making the random nature of light transport more apparent,
given the fact that less photons will be temporally averaged.

Pipeline Highlights

- **Hiding jaggies & increasing image stability:**
  - **Antialiasing/Temporal Supersampling**
- **Making the image more natural:**
  - **Depth of Field**
  - **Bloom**
  - **Film Grain**

ACTIVISION | BLIZZARD

Post effects often look unnatural. From exaggerated bloom, to depth of field that looks more like bloom than a defocusing effect.

Temporal stability of these effects is also more important than you may initially think. They can easily flicker, or even break antialiasing if incorrectly applied. In these cases, post effects can do more harm than good to photorealism.

I'll present a natural-looking and stable post-effect pipeline.

Antialiasing

- **We are using SMAA T2x (Subpixel Morphological Antialiasing):**
  - **Morphological AA for the jaggies**
  - **Temporal AA for supersampling and image stability**
- **More details in [Jimenez2011] *Enhanced Subpixel Morphological Antialiasing*:**
  - **http://www.iryoku.com/smaa**

ACTIVISION | BLIZZARD

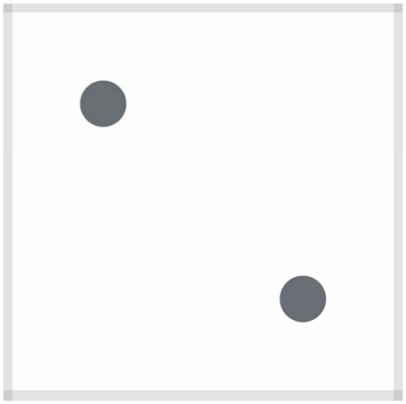
One of the main features of our post-effect pipeline that enhances the natural appearance of the image is our antialiasing solution.

We are using SMAA T2x subpixel morphological antialiasing. It basically deals with the jaggies by using a improved version of morphological antialiasing, and with subpixel features and temporal stability by using temporal supersampling.

(Check out the SMAA paper for more details about the technique)

Temporal Antialiasing

- **A quick explanation for two samples:**
  - **Jitter the scene at subpixel level over alternate frames**
  - **Blend current image with previous one**
  - **Avoid ghosting by using reprojection/velocity weighting**



ACTIVISION | BLIZZARD

As a brief recap, temporal supersampling can be described as a smart blend of two consecutive frames.

Supersampling Jittered Techniques

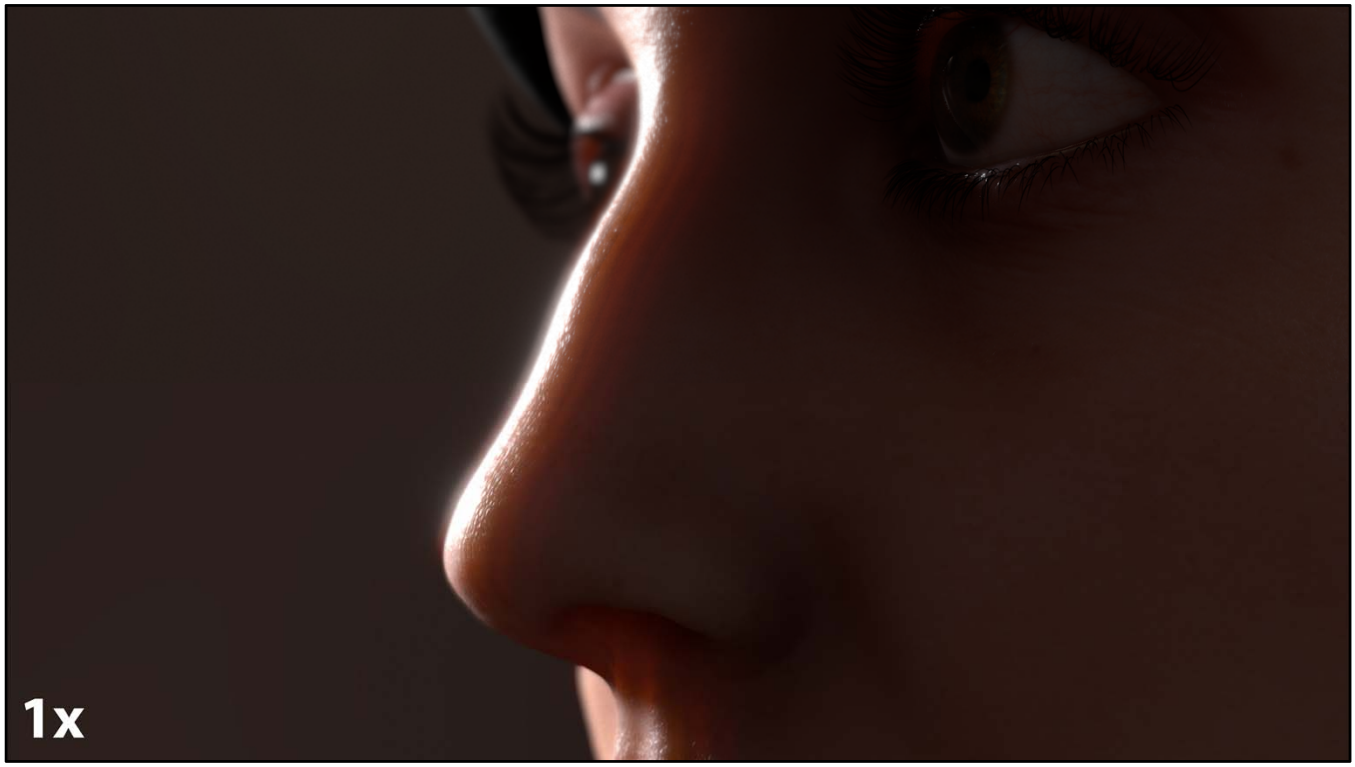
- **Probably the biggest advantage is that it enables supersampling jittered techniques:**
  - **Subsurface Scattering Reflectance Blur**
  - **Subsurface Scattering Transmittance**
  - **Shadow Mapping**
  - **Ambient Occlusion**
  - **Screen-Space Reflections**
- **For SMAA T2x, this effectively doubles the sample count for free!**

ACTIVISION | BLIZZARD

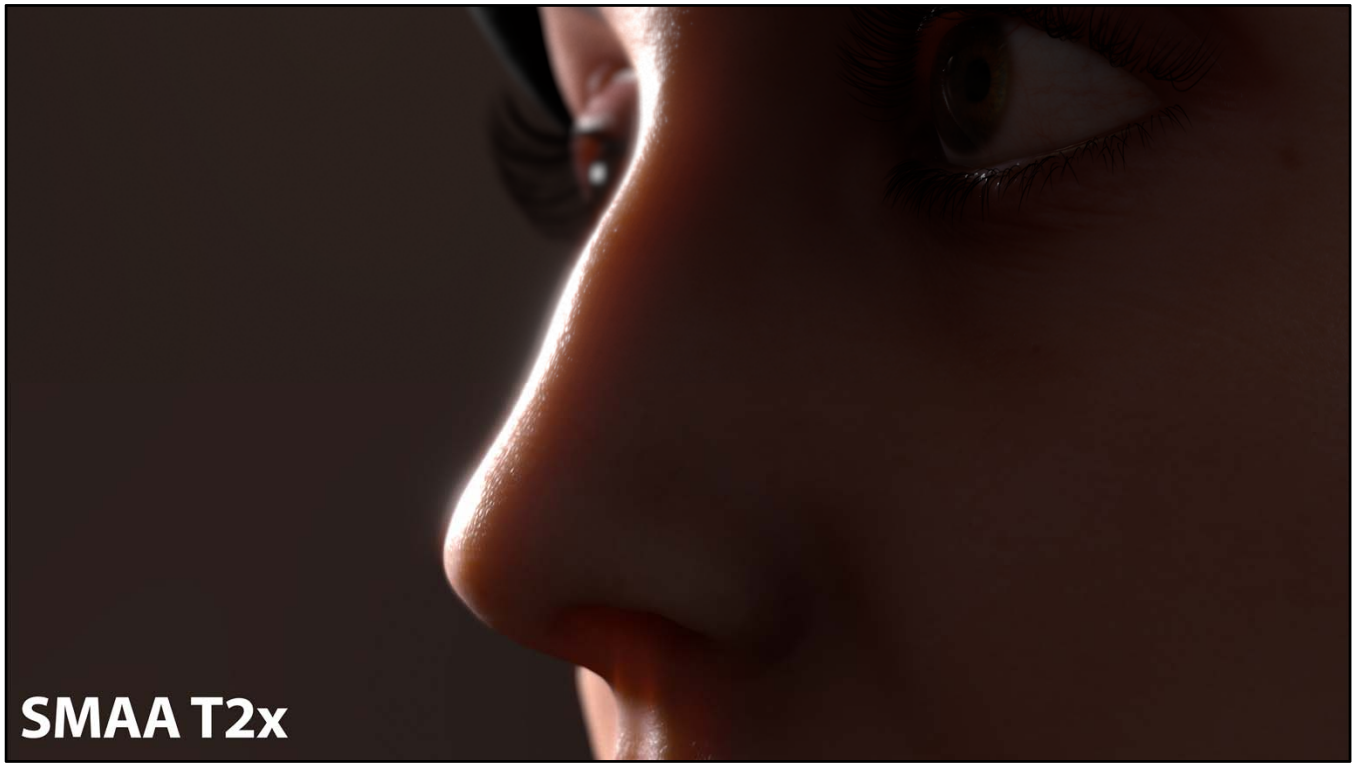
Temporal antialiasing is usually seen as a way to fight the jaggies, or shading aliasing.

Something not usually taken into account is that all jittered techniques like subsurface scattering,
shadow mapping, ambient occlusion and so on, will double their sample counts for free.

For that, we just have to swap the jitter offsets and/or the jitter textures per frame.



Here you have a comparison for subsurface scattering…



…where you can see the reduced banding in the case of using SMAA T2x.

SMAA Improvements

- **Merged neighborhood blending pass with the temporal resolve**
  - **Better performance**
  - **Can anti-alias everything in the same pass:**
    - **Color: for next post-effect stages**
    - **Depth: for next post-effect stages**
    - **Velocity: for immediate use (temporal resolve)**

We also improved the reference implementation of SMAA with two small modifications.

First, we merged the neighborhood blending pass with the temporal resolve.

This yields better performance, and we no longer need to store velocity in the alpha channel in the main render pass – as in the reference implementation – leaving it free for other purposes. (More on this later on.)

SMAA Improvements

- **SMAA Velocity Weighting:**
  - **w = SMAA_VELOCITY_WEIGHT * sqrt(abs(currentVelocity - previousVelocity))**
  - **0.5 - 0.5 * SMAASaturate(w)**
- **When in full effect, it disables blending, which leads to flickering**
- **SMAA morphological component alleviates it, but can still be apparent in some situations**

Second, we improved the velocity weighting.

SMAA T2x velocity weighting allows us to remove temporal blending if a given pixel has differing velocities, in order to to prevent ghosting.

When velocity weighting is in full effect, subsamples will alternate each frame, sometimes creating distracting effects (especially at 30 or lower FPS).

Here you have a movie that shows the problem.

Note that it’s actually not that terrible, but we have chosen a case where it’s very apparent (lots of U shapes, low FPS, and low resolution).

Also, the (hidden) background is rotating at a very high velocity, which helps to increase the effect of the artefacts.

SMAA Improvements

- **Solutions:**
  - **Disable it for slow-moving objects:**

```
w *= SMAA_ENHANCED_REPROJECTION_WEIGHT * currentVelocity;
```

  - **Remove SMAA jitter in the vertex shader depending on velocity weighting (better, but more intrusive)**

We found two solutions for this issue:

- Disabling velocity weighting when objects are moving slowly.

We're more sensitive to flickering when the jitter is big compared to object movement. Removing it in these cases helps with the flickering, at the expense of slightly more ghosting.

- Or: removing the SMAA jitter during the main render pass, depending on velocity weighting.

This completely removes the problem, but involves fetching previous-frame velocity in the vertex shader of each rendered object.

These jitter artefacts are mostly caused by either sub-pixel features or U morphological patterns.
As an alternative solution, it might also be possible to find a better U shape filter that's temporally stable.

Pipeline Order

• Z Pre Pass	HDR
• SSAO	HDR
• Forward Pass	HDR
• Screen-space SSS	HDR
• Transparency Pass	HDR
• Tone Map	HDR
• SMAA	HDR
• Bloom	HDR
• Depth of Field	LDR
• Film Grain	LDR

Filter-based Post FX

ACTIVISION | BLIZZARD

The other main characteristic of our post FX pipeline is the effect order.

Sorting the post-effect passes is, in a sense, similar to resolving a puzzle that probably doesn't have a perfect solution!

We apply antialiasing before filter-based post effects…

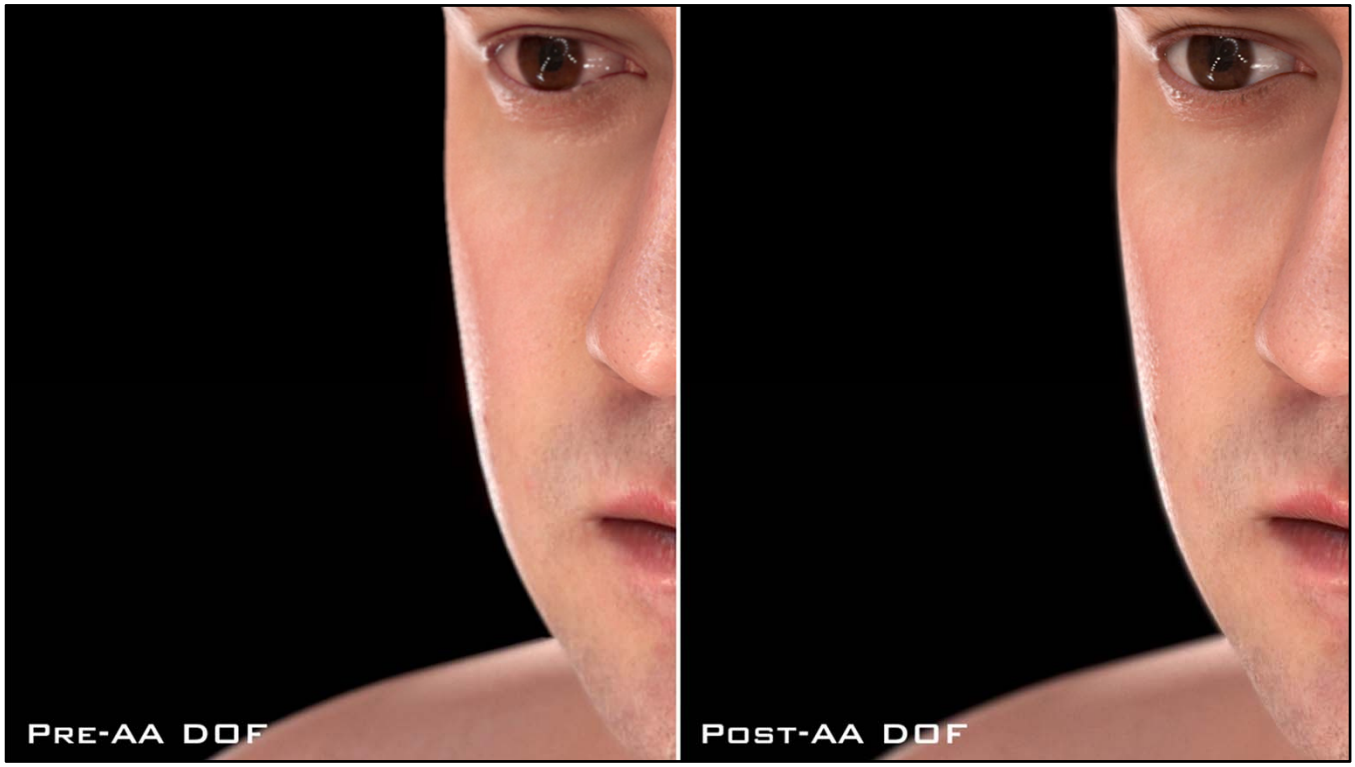
Antialiasing before Post FX

- **Increases image stability:**
  - **Removes high frequency jaggies for slightly better bloom filtering**
  - **Enables alias-free depth of field**
  - **Enables better edge detection**

ACTIVISION | BLIZZARD

…which enables better filtering in the bloom and depth of field effects,
as the high-frequency jaggies are removed from the image before the blurring required for these effects.

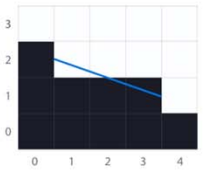
Edge detection also performs better, as the input image is sharper.

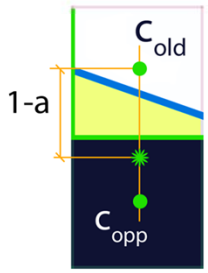


Here you can see how this pipeline order improves depth of field stability, producing more natural results under motion.

Antialiasing Data Buffers

- **Antialiasing data buffers is important for Post FX stability:**
  - **For color:**
    - **C = lerp(C_old, C_opp, a)**
  - **For depth (for depth of field):**
    - **D = min(D_old, D_op)**
  - **For velocity (for reprojection):**
    - **V = D_old < D_op? V_old, V_op**





ACTIVISION BLIZZARD

In order to not break morphological antialiasing approaches (part of SMAA) when applying post effects,
a useful trick is to also anti-alias data buffers (like depth or velocity) when anti-aliasing the color buffer.

For that, instead of blending values – as is done for color – we take the value of the pixel closest to the camera.

As we will show later in the presentation, we carry 1.0/depth in the alpha channel of the color buffer,
so we can apply antialiasing to both the color and depth buffer at the same time without extra texture accesses.

(See [Jimenez2011] *Subpixel Morphological Antialiasing* for more details about this blending operation)

Pipeline Order

• Z Pre Pass	HDR	
• SSAO	HDR	
• Forward Pass		HDR
• Screen-space SSS	HDR	
• Transparency Pass	HDR	
• Depth of Field	HDR	
• Bloom	HDR	
• Film Grain	LDR	
• SMAA?	HDR	
• Tone Map?	HDR	

ACTIVISION | BLIZZARD

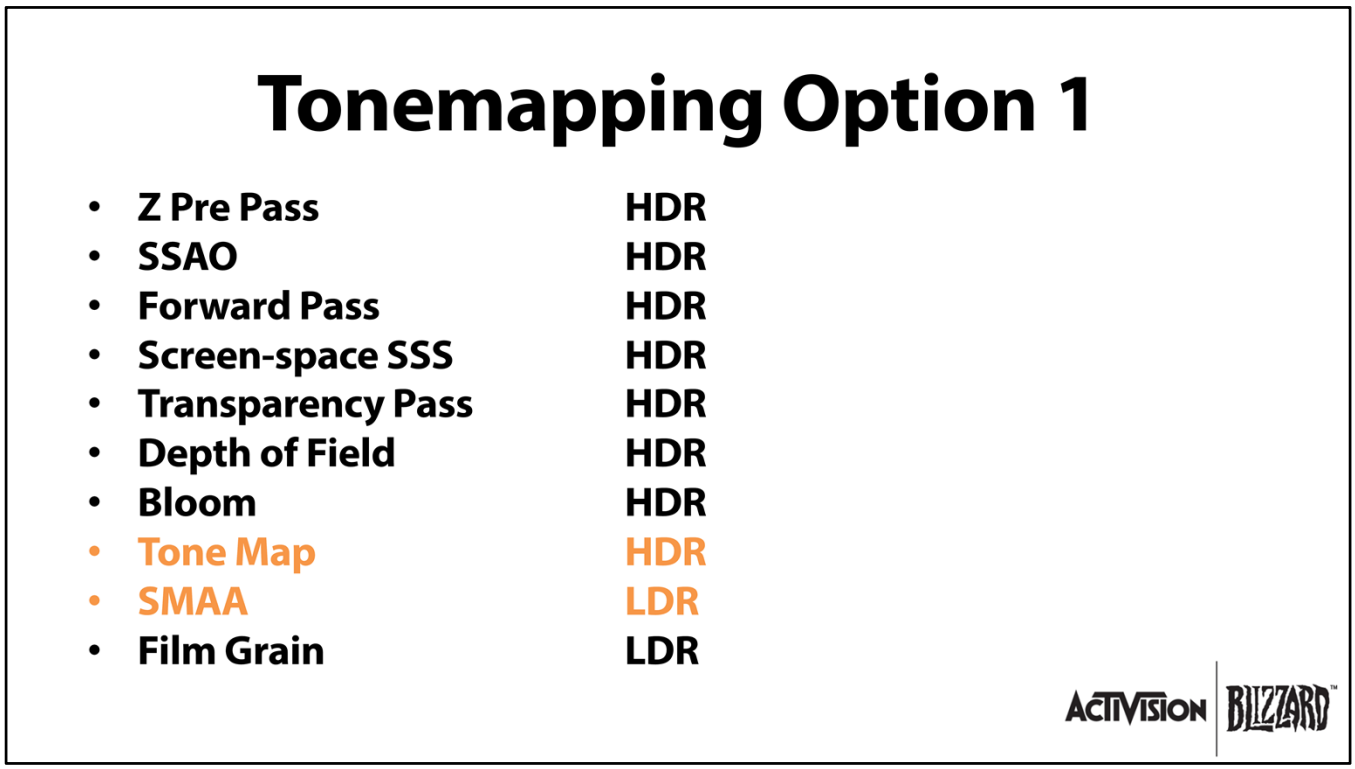
Something open to discussion is where to place the tone-mapping step.

This is where post effect-ordering really starts to turn into a puzzle.

In principle, bloom happens as the photons reaching the sensor start to saturate the maximum capacity of a photodiode. In this situation, photons start to overflow into adjacent device structures. This overflow happens in linear (photon) space.

After blooming, tone mapping (or the camera response curve) in the analog-to-digital converter takes place.

This means that for physical correctness, tone mapping should happen after bloom.



We considered three different options.

The first one is to do depth of field, bloom, tonemap then antialiasing.

If we do this, we have two problems:

- We can potentially break antialiasing (see [Persson2008] *Post-tonemapping resolve for high quality HDR antialiasing in D3D10*).

- We cannot leverage antialiasing for more stable bloom and depth of field.

Tonemapping Option 2

• Z Pre Pass	HDR
• SSAO	HDR
• Forward Pass	HDR
• Screen-space SSS	HDR
• Transparency Pass	HDR
• SMAA	HDR
• Depth of Field	HDR
• Bloom	HDR
• Tone Map	HDR
• Film Grain	LDR

ACTIVISION BLIZZARD

The next option consists of performing antialiasing, depth of field and bloom, followed by tone mapping.

In this case:

- We can potentially break antialiasing (see [Persson2008] *Post-tonemapping resolve for high quality HDR antialiasing in D3D10*).

From a physical standpoint, this is the most correct approach.

Furthermore, aliasing problems when tonemapping after antialiasing can be reduced by using a wide-filter resolve (http://mynameismjp.wordpress.com/2012/10/28/msaa-resolve-filters/).

Tonemapping Option 3

• Z Pre Pass	HDR
• SSAO	HDR
• Forward Pass	HDR
• Screen-space SSS	HDR
• Transparency Pass	HDR
• Tone Map	HDR
• SMAA	HDR
• Depth of Field	HDR
• Bloom	HDR
• Film Grain	LDR

ACTIVISION | BLIZZARD

Our final and preferred option consists of performing tone mapping, antialiasing, depth of field, then bloom.

In this case:

- Bloom happens after tonemapping, which is not physically correct.

Even if it is not the most correct approach, it lead to the most stable results in our test environment.

In order to keep bloom and depth of field working in linear space, the inverse of the tone map function can be applied after SMAA. Tonemapping is then applied again at the end of the pipeline.

In our case, we have not found the need to do so.

Depth of Field

- **We use a simple gather approach, with the blur width modulated by the pixel's circle of confusion**
- **To avoid bleeding we can:**
  - **Compare depth, to avoid bleeding of unfocused pixels in the background onto the focused foreground**
  - **Weight each sample by its circle of confusion, to avoid bleeding of focused pixels in the foreground onto the background**

  Similar to [Scheuermann2004] *Advanced Depth of Field*.
- **But this is not enough!**

For depth of field, we use a simple gather approach, with the blur width modulated by the pixel's circle of confusion.

We can compare depth values to avoid bleeding of unfocused pixels in the background onto the focused foreground.

Also, we can weight each sample by its circle of confusion, to avoid bleeding of focused pixels in the foreground onto the background.

But this is not enough!

Depth of Field

- **How can depth of field can break photorealism?**
  - **When it looks like bloom, something that happens often!**
- **This happens because object edges don't dilate realistically according to circle of confusion**
- **How can we improve the Depth of Field *blooming*?**
  - **Weight taps by `saturate(tapCoc - offset)`**
  - **Instead of simply weighting by `tapCoc`**

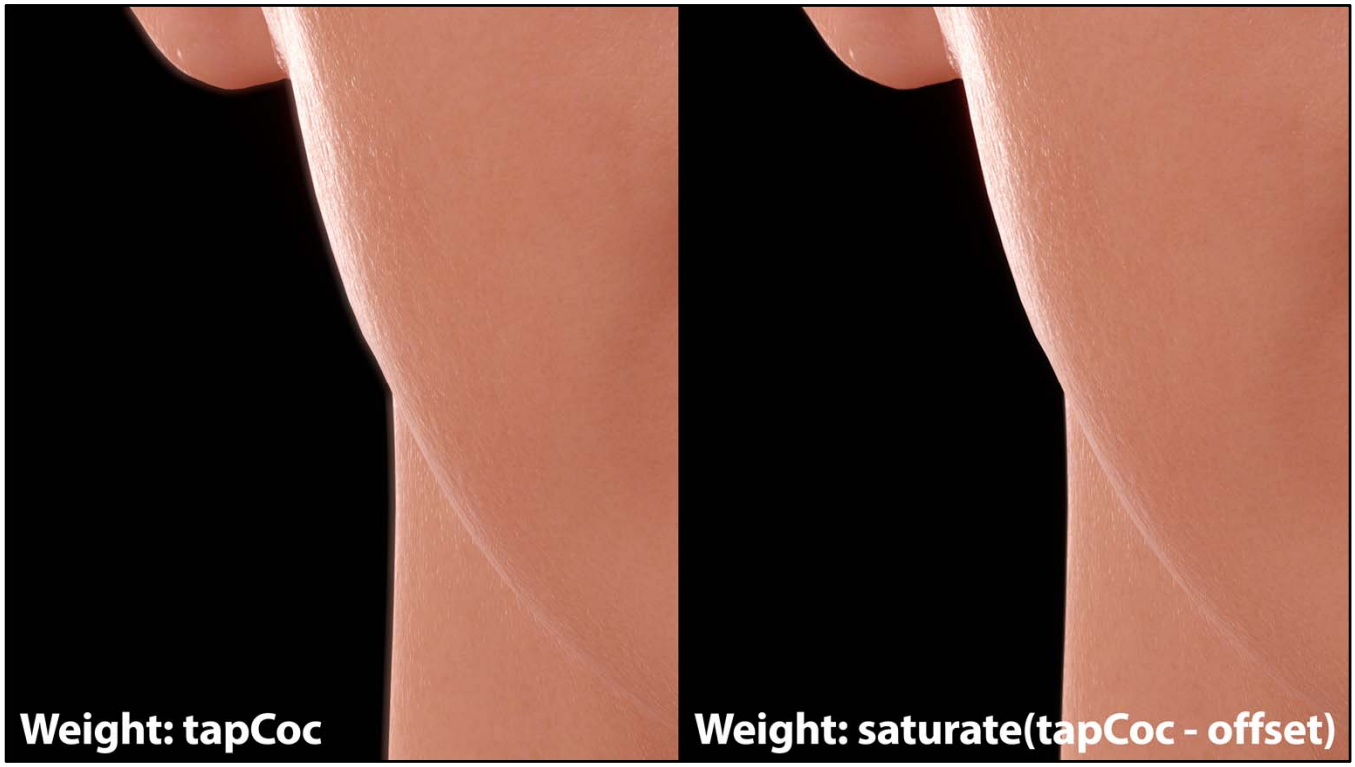
Depth of field often resembles blooming more than defocusing.

This happens because edges of objects don't dilate depending on the circle of confusion, but simply fade, creating very unrealistic halos.

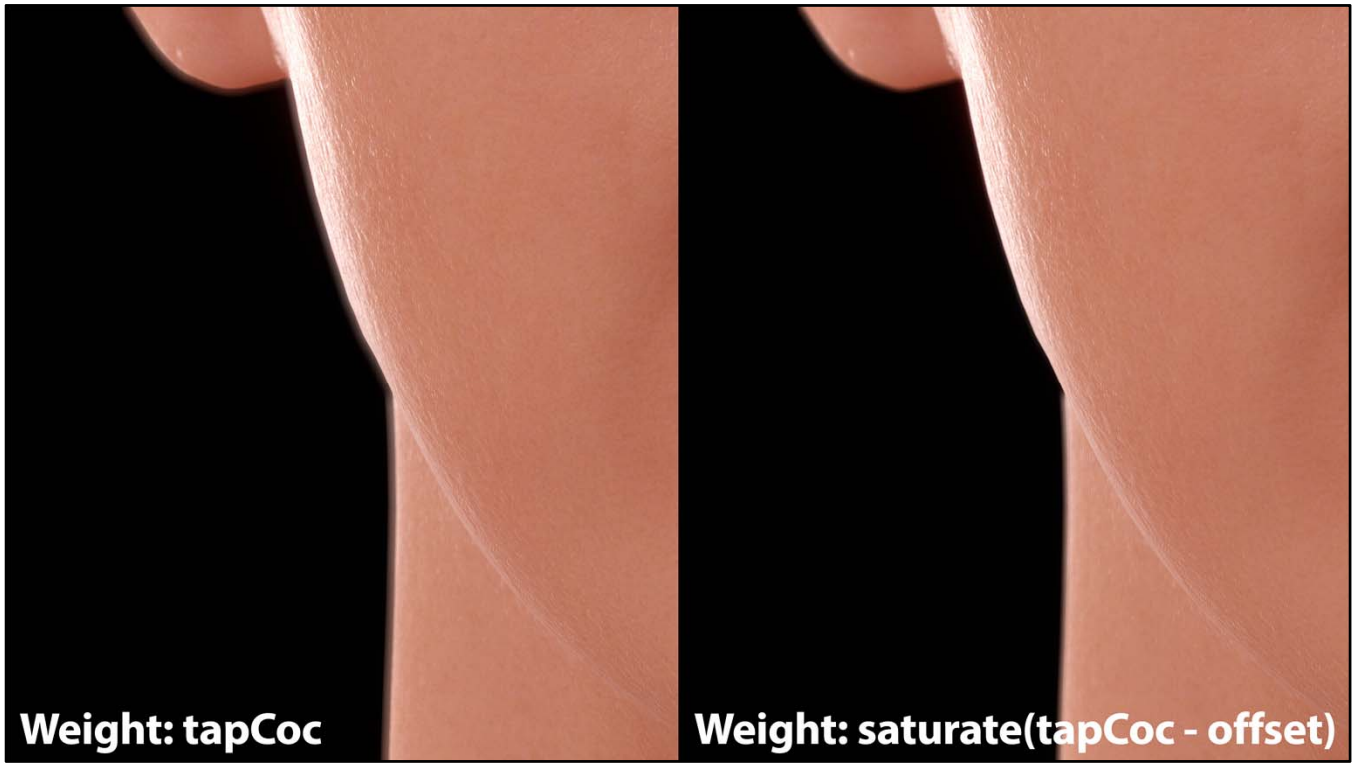
To fix this artifact, we compare the tap's circle of confusion with the offset used to fetch the tap.

Ideally, this should be a binary check, and only include taps whose circle of confusion actually includes our current pixel: tapCoc > offset.

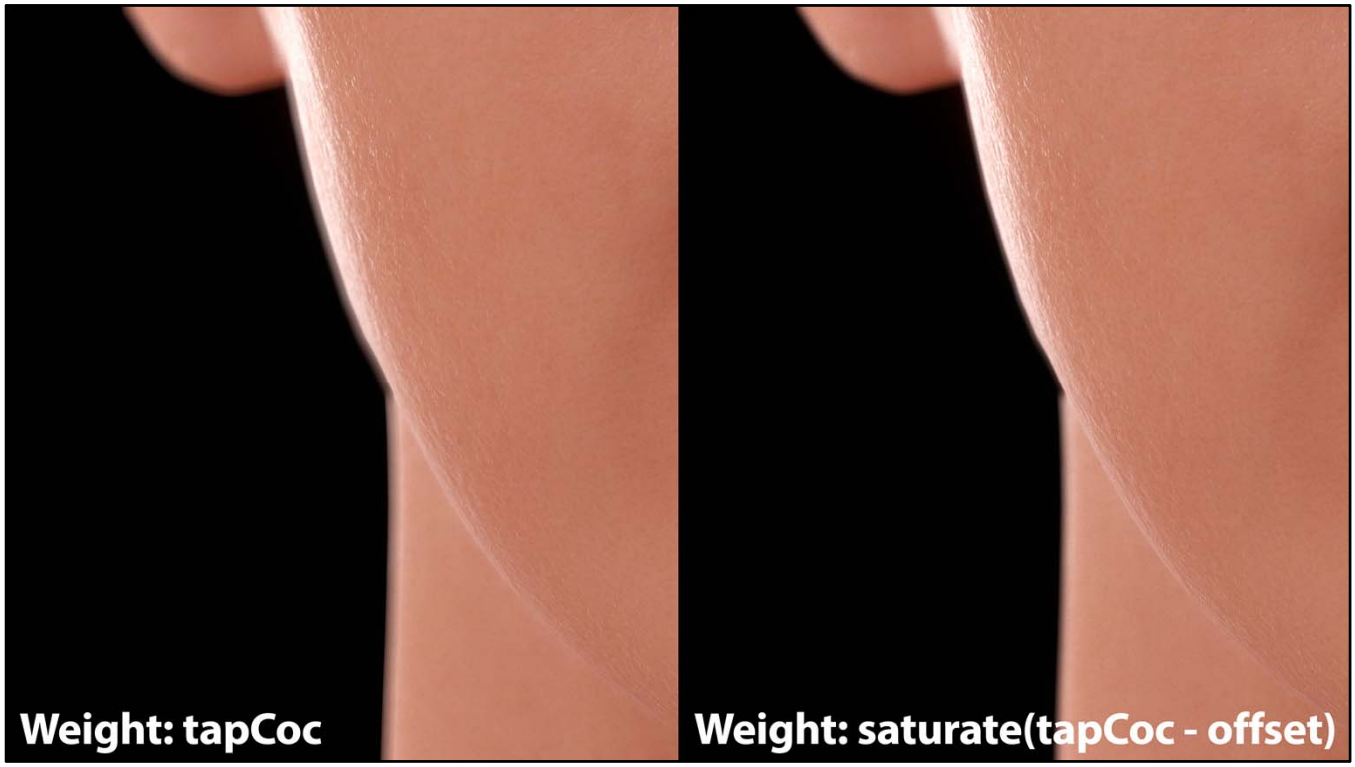
However, this would require many taps to work correctly, so we use a progressive check instead: saturate(tapCoc – offset).



Here you have a sequence…



…which shows how the edge dilates better…



…in the case of using saturate(tapCoc – offset).

Store 1/depth in alpha

- **We found a lot of post effects use depth and color:**
  - **Subsurface Scattering**
  - **SMAA (for an additional edge detection measure and for *antialiasing* depth)**
  - **Depth of field**
- **Storing 1/depth in the alpha channel gave us a noticeable speedup**

ACTIVISION | BLIZZARD

As briefly mentioned before, a useful trick we use is storing 1/depth in the alpha channel of the main render target.

A lot of post effects use both of them at the same time. So, we can leverage this to reduce the memory bandwidth of these shaders.

Storing the inverse is especially useful when using LDR render targets, as most post effects don't really need depth but a filter width (which usually is the inverse of the depth). Expressed in this way, it could potentially be compressed in a 8-bit render target.

Summary

Pros

- More stable depth of field and bloom
- Natural-looking
- Doubles sample count of jittered techniques (with temporal AA)
- Less memory accesses (1/depth is in alpha)

Cons

- Depth of field can show subpixel temporal instabilities (unless we temporally resolve depth)
- Requires extra memory
- Convoluted

ACTIVISION | BLIZZARD

In this slide you will find some of the pros and cons of our pipeline.

Final Comments on Rendering

- **Our technology runs at 180 FPS on a GeForce GTX 680 (@720p)**
- **When the face is covering most of the screen, why not invest all possible resources for delivering the best possible character graphics, and LOD it with distance?**
- **If you cannot afford it, prerecord the cinematics (at 1080p if possible – it makes a big difference for faces)**

ACTIVISION | BLIZZARD

To end my part, we would like to remark that this technology runs at 180 FPS on a GeForce GTX 680.

When the face is covering the screen we believe that investing extra GPU resources to render beautiful characters is a good idea.
And, when it's far away, the shaders could potentially be replaced by simpler ones.

Finally, if you cannot afford this, prerecording the cinematics can be an alternative solution.

Animation

ACTIVISION | BLIZZARD

Animation

1. Data Capture and Tracking: Institute of Creative Technologies of USC

- **30 High Resolution Scans ~ 4M pts.**
- **Mid Resolution Tracking ~ 30K pts.**

2. Fitting Deformation and Animation to Game Rig

- **Fitting Deformation to Bone Animation and Bone weights.**
- **Normals, Displacement and Diffuse Maps**

3. Game Data Driving

- **Standard Bone driven animation**
- **Stress blended Normal, Displacement and Albedo.**

ACTIVISION BLIZZARD

The rendered data looked great, the next step was to make this move.

So how did we animate this?

There are 3 sections to animation. The capture, which was done at Paul Debevec's graphic lab at ICT; fitting the deformation and animation to a standard game rig, and driving this fitted data in our game rendering environment.

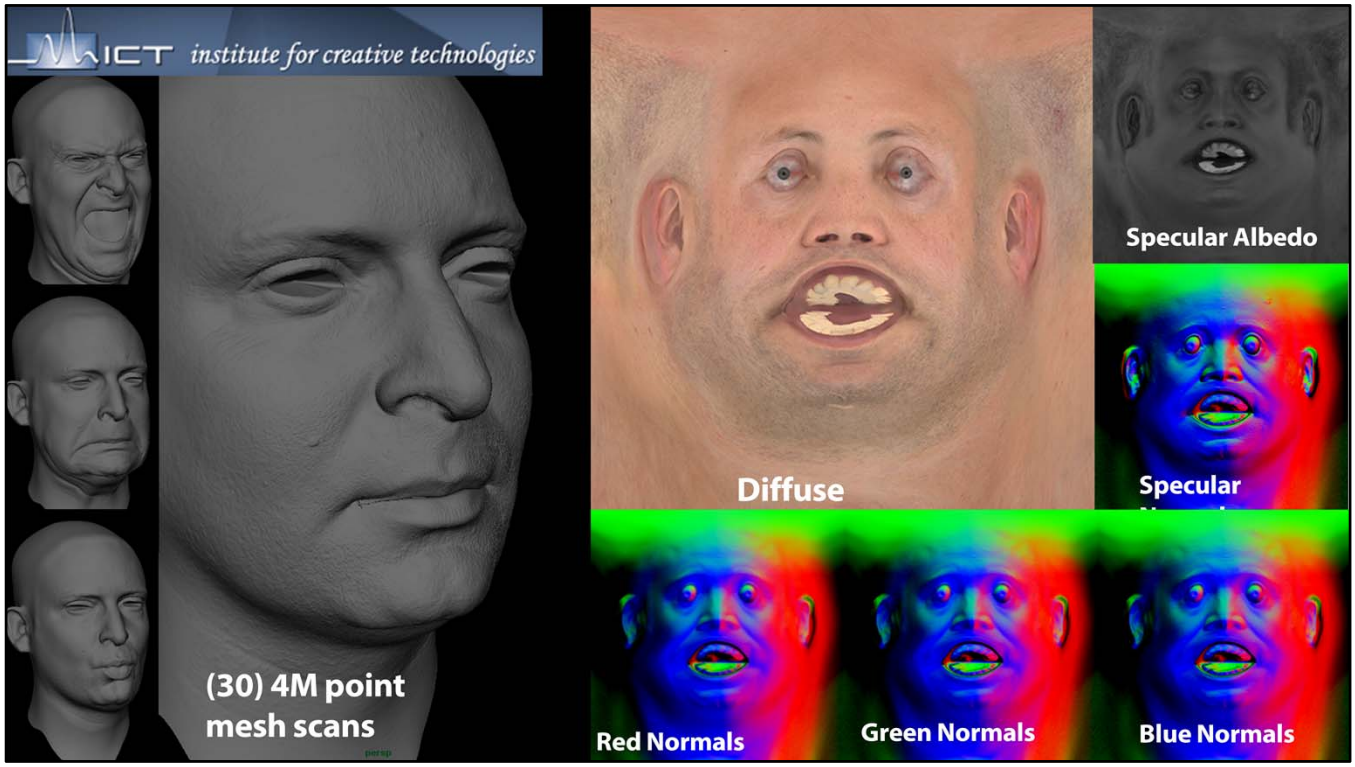
Source Data Capture

Data was scanned and tracked by Graphics Lab of the Institute of Creative Technologies of USC using Lightstage X

- **30 high resolution expressions scans @~ 4M pts + textures**
- **Mid resolution performance tracking @ ~ 20K pts**

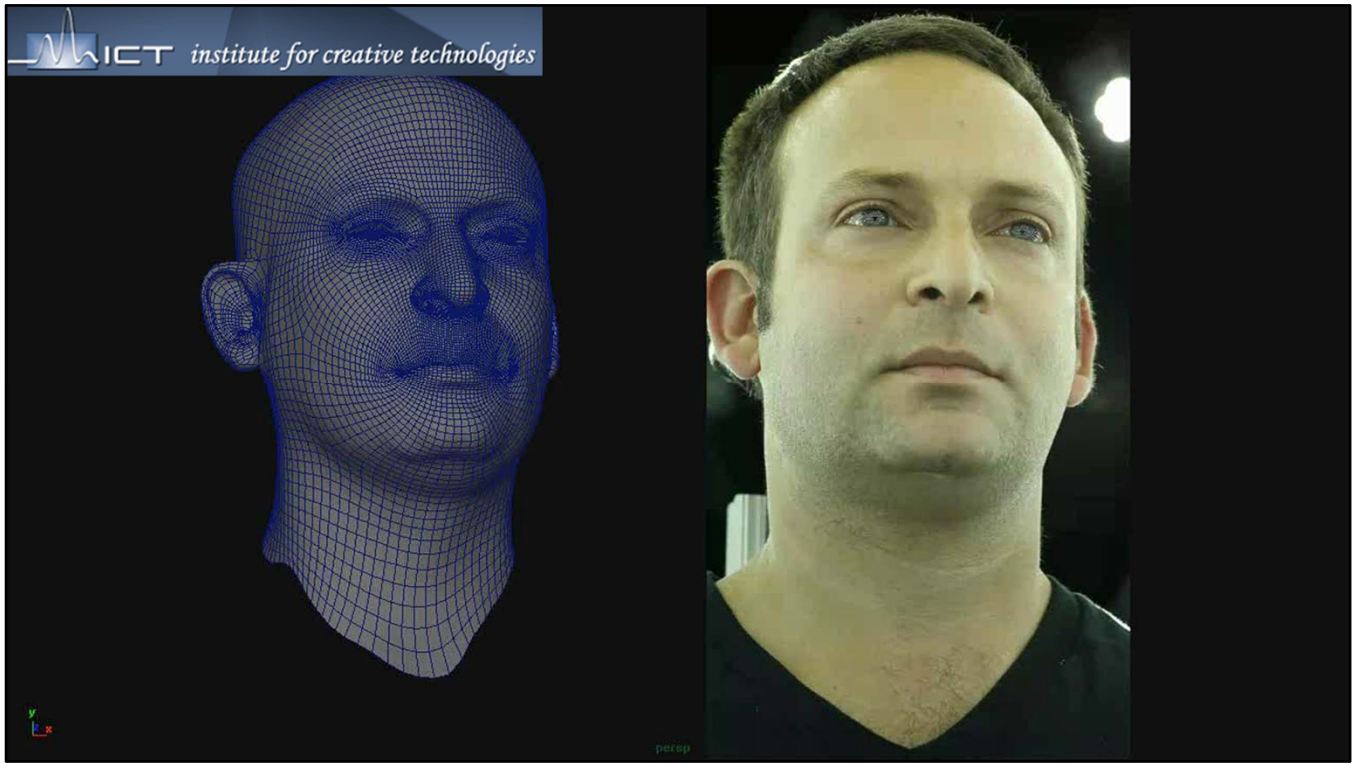
Oleg Alexander, Graham Fyffe, Jay Busch, Ryosuke Ichikari, Abhijeet Ghosh, Andrew Jones, Paul Graham, Svetlana Akim, Xueming Yu, Koki Nagano, Borom Tunwattanapong, Valerie Dauphin, Ari Shapiro, Kathleen Haase, Paul Debevec

Our source data was captured by the graphics lab at ICT. They provided 2 pieces of data, a series of 30 high resolution – 4.5M points scans, and a tracked mesh at approximately 20K points.



The high resolution scans were accompanied with RGB Normals, Specular normals, Specular Intensity and a Diffuse albedo. This data was captured with the Lightstage X. For this work we only utilized the high resolution scans and the diffuse albedo maps.

An example of the resolution of the meshes we started with.



The tracking, is a novel method developed by Graham Fyffe of the ICT Graphics Lab.

Digital Ira face is built from eight light stage scans of Ari Shapiro, each accurate to 0.1mm, each with diffuse texture, specular intensity, surface normals, and displacement.

The scans are corresponded together to the pixel level into a blendshape facial model , which is then driven by markerless video performance capture.

But this data is still too dense for us to run in game compatible environment.

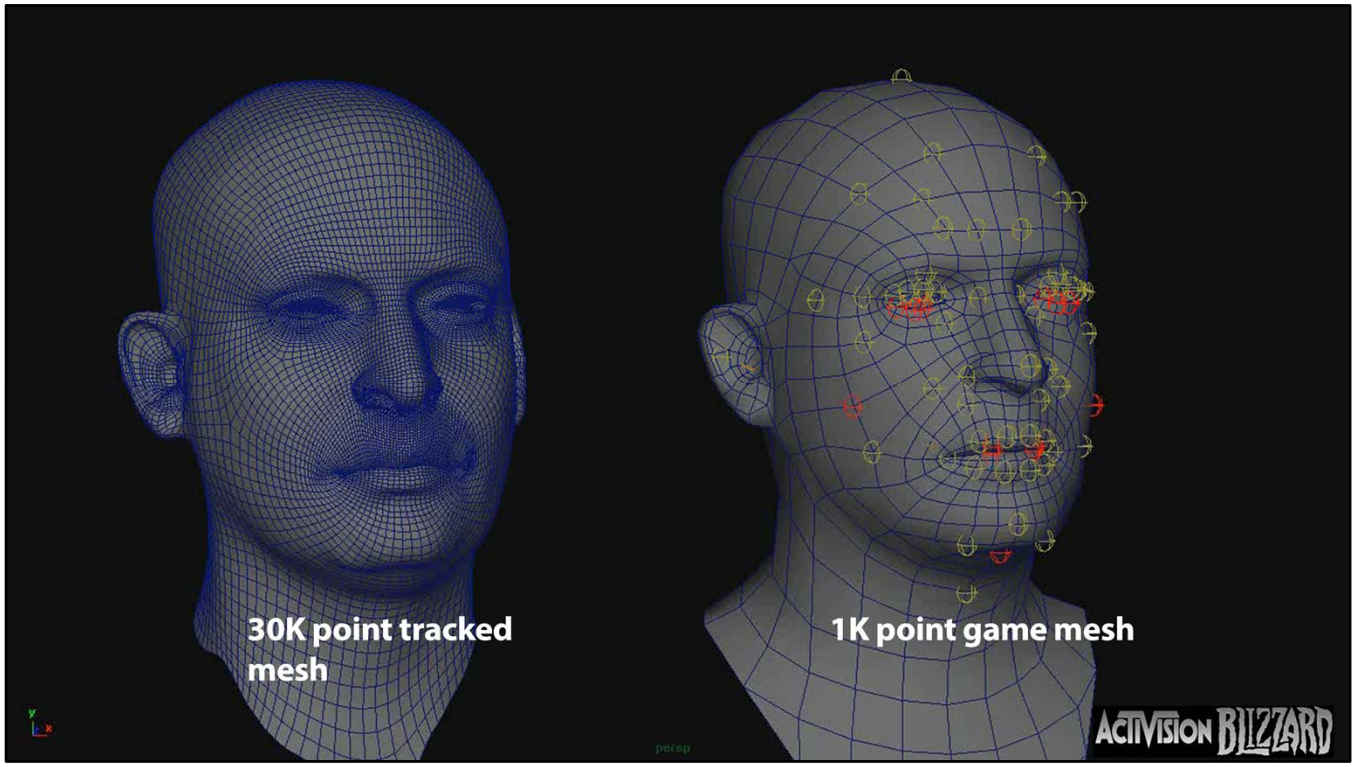
Fitting Deformation and Animation to Game Rig

1. **Low Frequency Deformation:**
   Fitting Deformation to Bone Animation and Bone weights

2. **High Frequency Deformation:**
   Extraction of Normals, Displacement and Diffuse Maps

ACTIVISION BLIZZARD

To compress the data further we divided the data into two, high and low frequency deformation.



Starting with the low frequency deformation we interpret the denser deformation as a series of animated bone and bone weights.

Animation Compression
1. Fitting Deformation to Bones

Given a reference deformation or vertex array stream, we want to represent it with standard skinning deformer at a lower resolution mesh.

a. Transfer animation to a game resolution mesh

b. Optimize Skinning Weights

c. Optimize Bone Animation

Iterate(b, c)

ACTIVISION BLIZZARD

To fit the deformation to bones, we have a 3 step process.

First we find, for every point in the game mesh, an equivalent point in the high resolution mesh, that becomes our target point.

We optimize skin weights and then the bone animation, we iterate this process a few times.

Weight Optimization

- **Given**
  - **an initial bone animation which tracks the closest point on the high-resolution mesh and**
  - **initial weights simply computed by distance from a larger set of bones (8)**
- **For each vertex, we want to find the 4 bone weights to minimize the error.**
- **We use NNLS (Non-negative Least Squares) to solve:**

$$\min_{\omega} \sum_{c=0}^{2} \sum_{f}^{frames} \sum_{i}^{verts} \left| \left(\sum_{b}^{weights} \omega_{i,b} * B_{b,f} * L_{i,b} \right)[c] - A_{i,f}[c] \right|^2$$

$c = [x, y, z]$

ACTIVISION BLIZZARD

We initialize the animation by loading a bone skeleton and extracting the bone animation by constructing a matrix transform from the closest point and normal on the source mesh. We initialize the bone weighting by standard closest distance to a larger set of bones. In the case of this test we used 8 bones.

Once we have our initial bone and skinning, we minimize the error to find the best weight combination of four bones. This is simply done by finding the square distance between the skinned point and the goal point for all verts for all frames for all axes and minimizing this using NNLS.

Weight Optimization

Related work

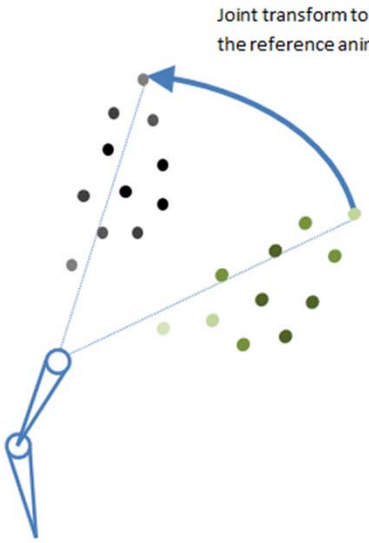
- ***Softimage Face Robot 1.5 Game Export, 2006.***
- ***Poisson-based Weight Reduction of Animated Meshes, Landreneau & Schaefer 2009.***

ACTIVISION BLIZZARD

There is prior related work on skinning optimization. The work we did on Face Robot did some of this, but instead of NNLS this work requires a good initial skinning and iterates towards the result. The major difference on our current work is that the algorithm we are using is very efficient and it is greedy, it can be used in any number of configurations, it is not restricted to faces.

The paper of Landreneau and Schaefer presents an interesting variation, in there the authors optimize for Laplacian instead of position. This means that there will be some more error on the point position and less on the derivatives of the surface. This is particularly interesting if the meshes are very dense, in case of our meshes we use very sparse data so the error in the Laplacian is less important as the error in position.

Bone Animation Optimization



For each frame, for all bones, we want to find the best transform such that the set of points weighed to the bone have the minimum mean squared distance to the reference animation.

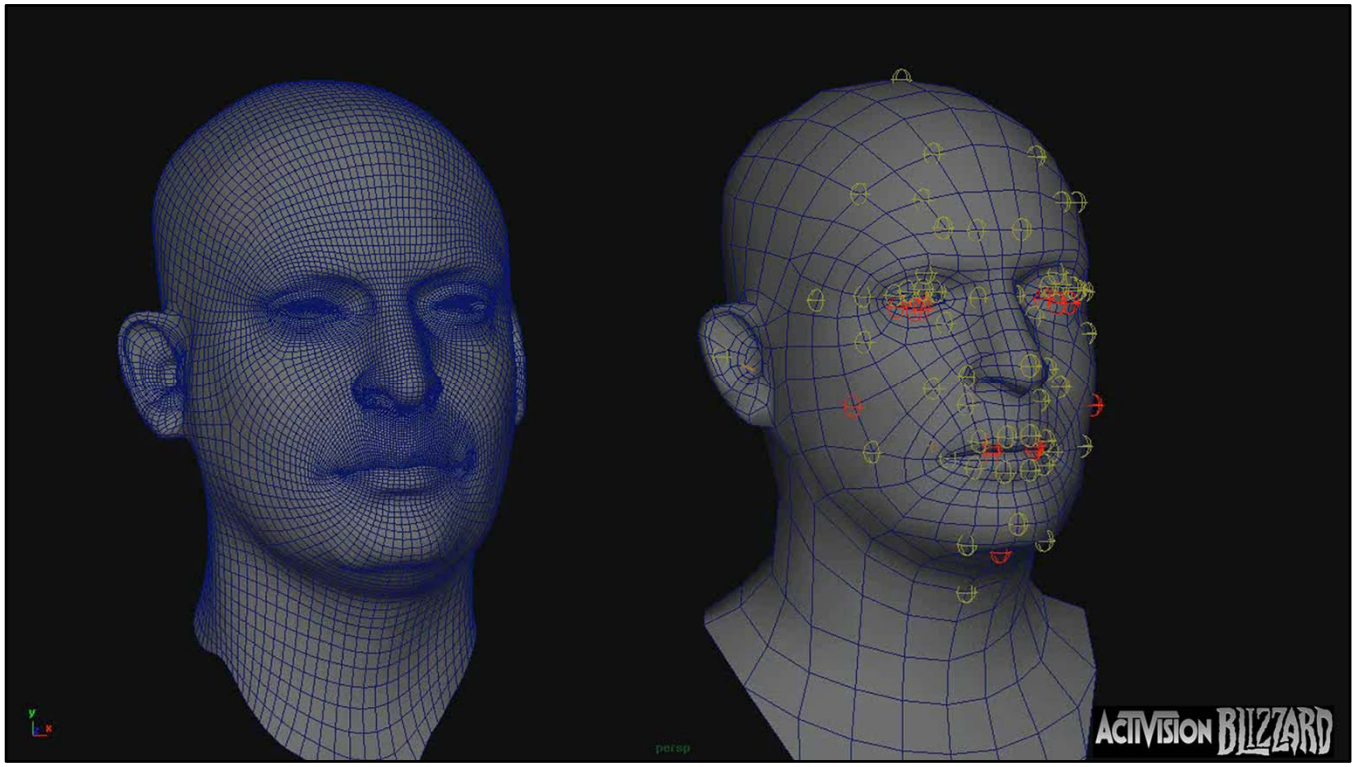
For each bone, we compute the weighted 3D point registration, and we iterate a few times through all bones.

Reference: *Dual quaternion method, based on "Estimating 3-D location Parameters Using Dual Number Quaternions", Walker et al 1991*

ACTIVISION BLIZZARD

In contrast to the prior work we realize that the error can be further minimized if we also optimize the animation. For this, we are using simple weighed 3d point registration. We used walker et al 3d point registration since it describes how to do weighted point registration.

Once the animation has been optimized per frames, we iterate and reoptimize the bones getting much closer to the original result.



This process results in very faithful, low frequency deformation.

In our work we used 70 bones, and 2 different resolution meshes, the one here is 1000 points.

High Frequency Extraction of Normal Displacement and Diffuse maps

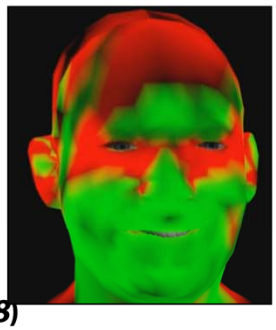
We need to find the minimum set of maps to closely match the high resolution.

Our strategy is to utilize the same technique to extract the maps that we will later use to blend them: vertex stress.

ACTIVISION BLIZZARD

We take the same fitting approach to derive the high frequency deformation. To have the complex subtle deformation in game we limited ourselves to very close to current gen number and size of maps. To create the data we first think how this data will be driven, then create the set of maps that will best represent the source. Since we use vertex stress to blend the different textures, we use vertex stress to derive these maps.

Vertex Stress Techniques



Displacement

1D Stress, Strain: normalized ratio of area/area at rest.

2D Stress, Strain projected on UV axes

5D Stress, 2d Strain and Displacement (*Wa Chun Ma et al. 2008*)
Taking PCA of 5D stress axes and projecting major 2 Axes on UV space can be used to determine axes of strain, deformation flow of a surface.

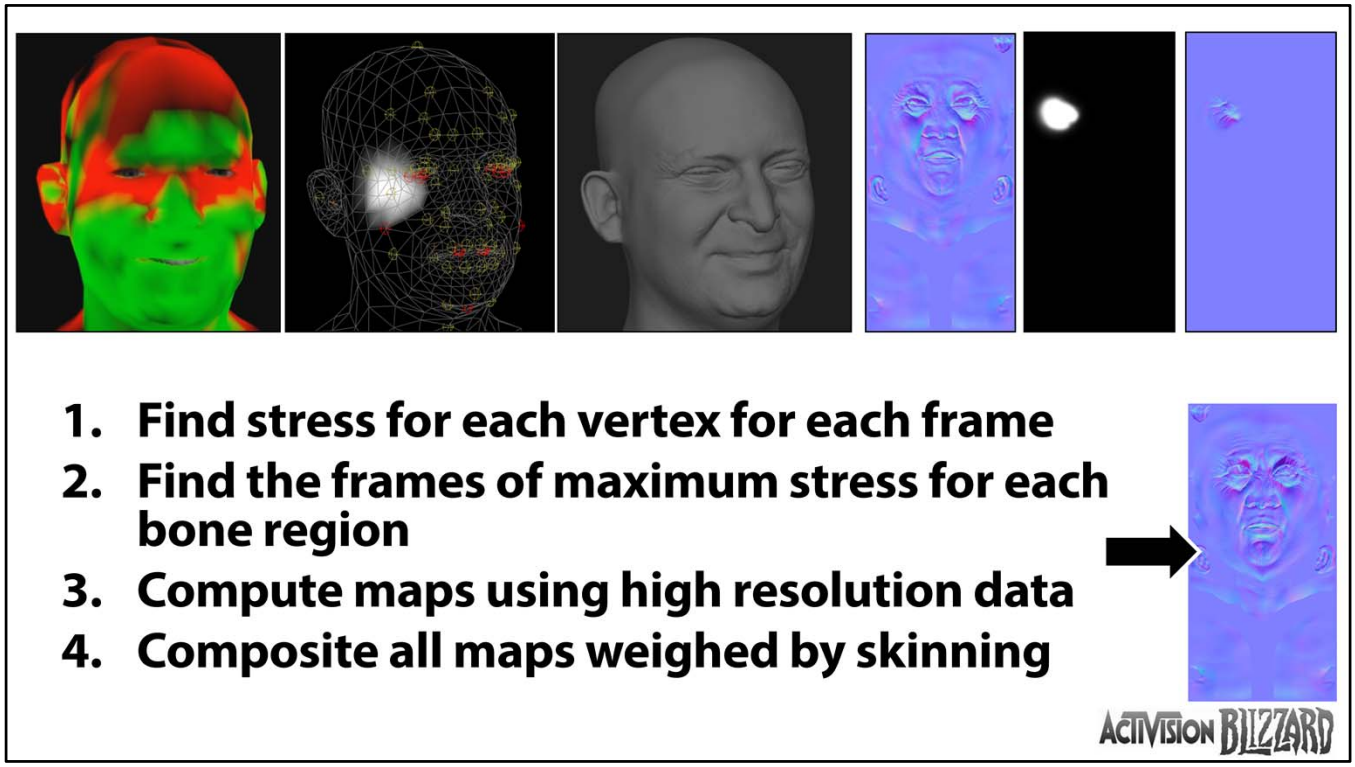
Strain can be computed by average edge length or using standard deviation of point neighborhood

ACTIVISION BLIZZARD

We looked at several versions of defining stress on a vertex.
The simplest is displacement. In this method, the stress of the vertex is determined by how far it travels in head space away from the rest pose.
1D stress or strain based stress considers the ratio between the animated and the rest area for every given vertex. This is a commonly used technique.
2D stress projects the 1D stress on UV axes
5D stress is 2d strain and displacement. Alex Ma describes this measure of stress in his paper on Polynomial Displacement Maps. The paper describes taking PCA of the 5d axes we can find the main 2. If we project these on the texture space we have the major axes of deformation of the skin which could serve as orientation to the 2d Stress.

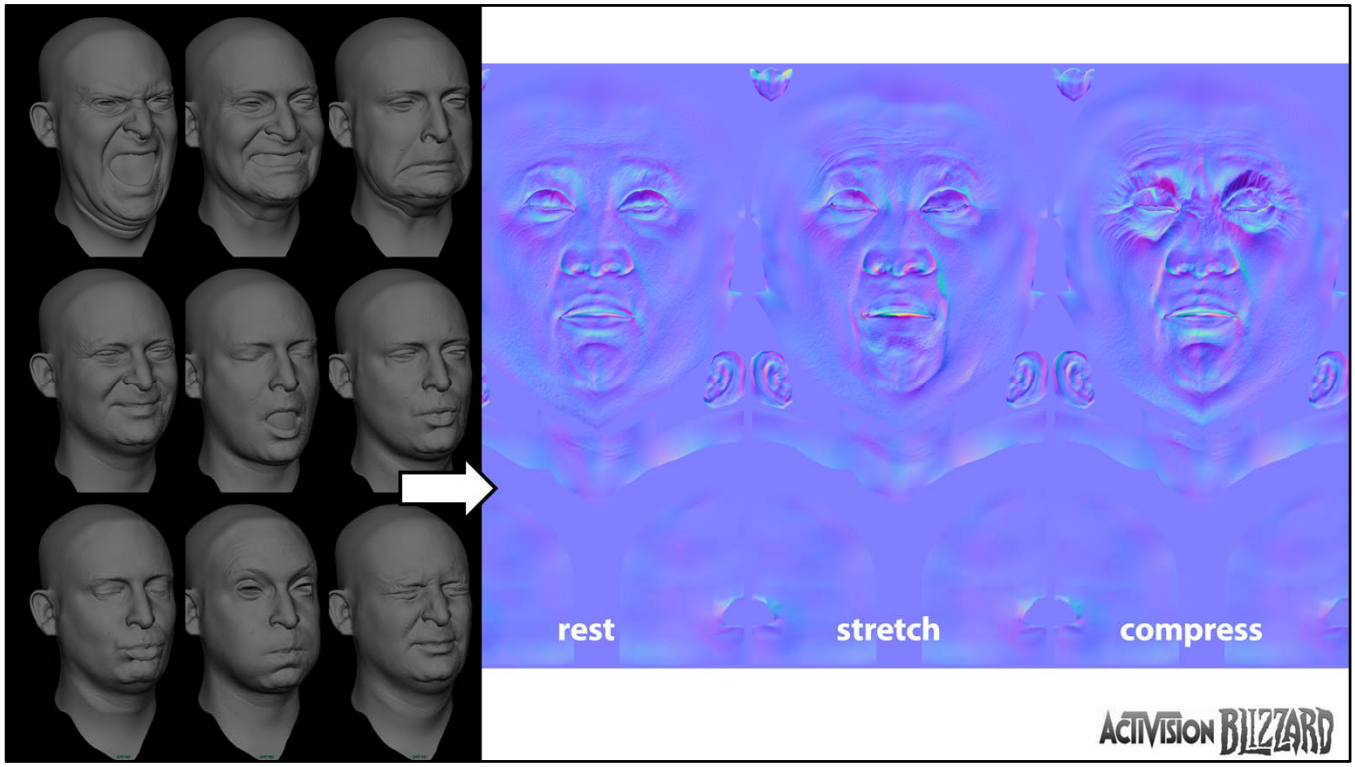
We are currently computed in CPU by looking at the average length of neighbor edges to a vertex. A better form of stress could be computed by looking at the standard deviation of a set of points.

While some of these stress measures could yield better stress, we chose to use the standard 1d Strain based stress, we need more comparative data to analyze the differences in end result.



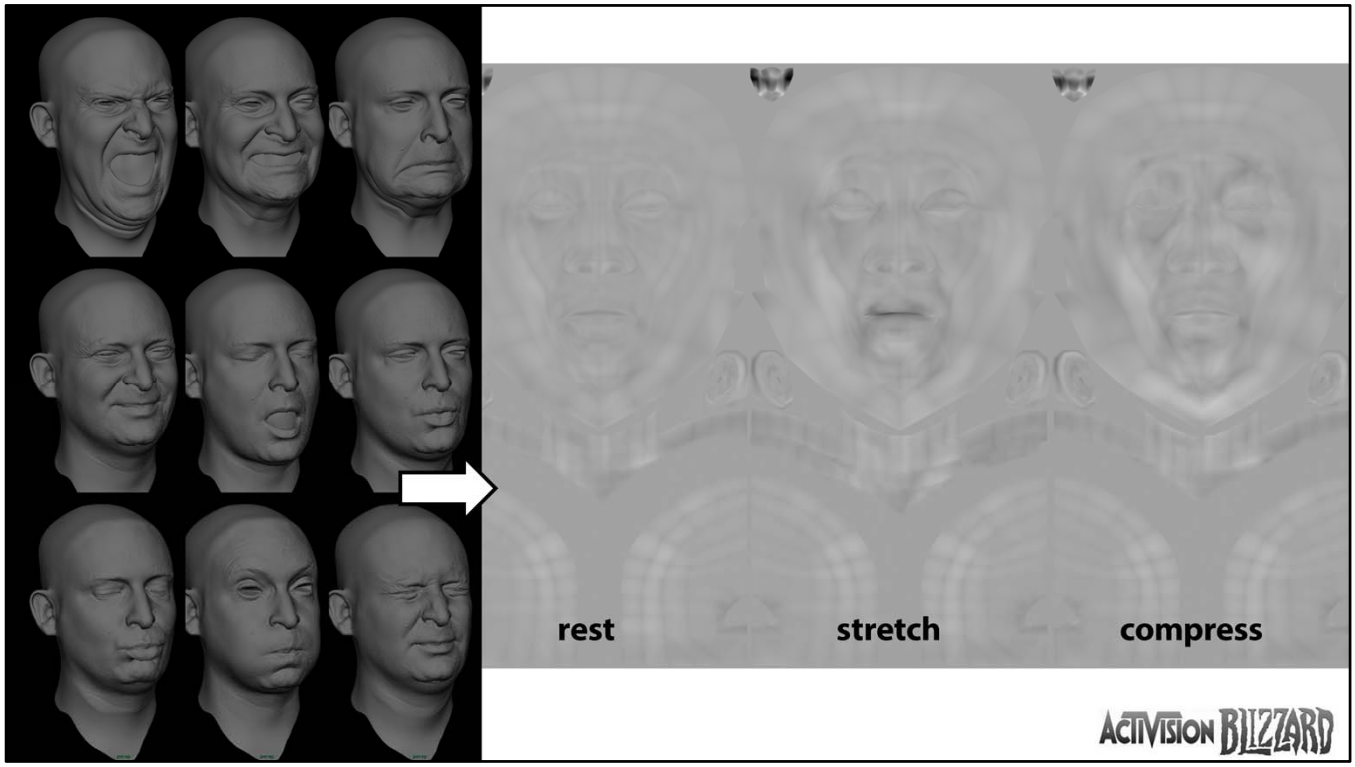
The process to derive our composite maps utilizes whatever form of stress we will later use to blend the maps in the games.
For every vertex we find the stress.
For every bone, we find the frame for which the stress is maximum for the collection of weighted points to that bone.
Once we have this frame, we compute the normal map between the high and the low resolution meshes.
Next, using the area of the bone, we composite this normal map against all other normal maps.
The smooth falloff nature of the bone weight maps results in a very smooth composite normal map.

We could do the composite per vertex but that would result in a very uneven normal map.

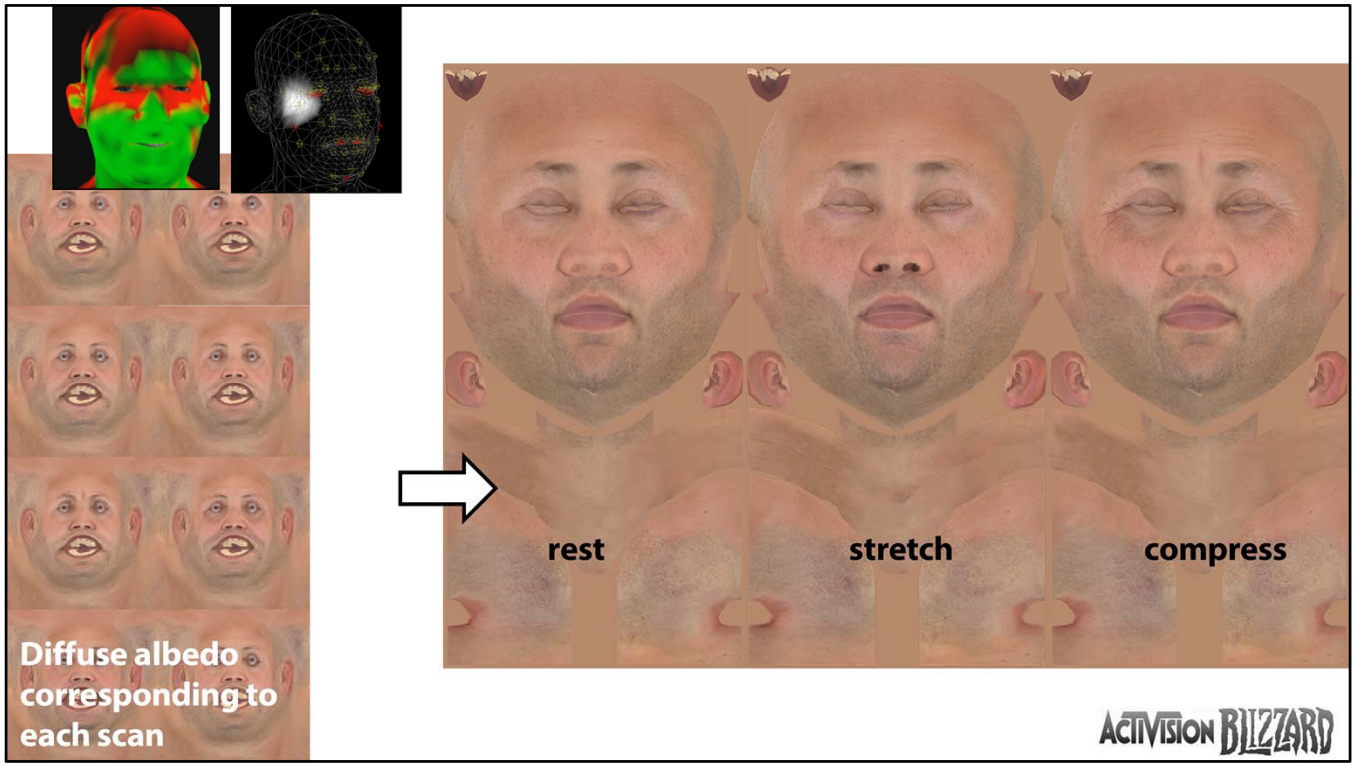


From our 30 high res meshes, We compute 3 normal maps, one at rest one at compression and one at stretch.

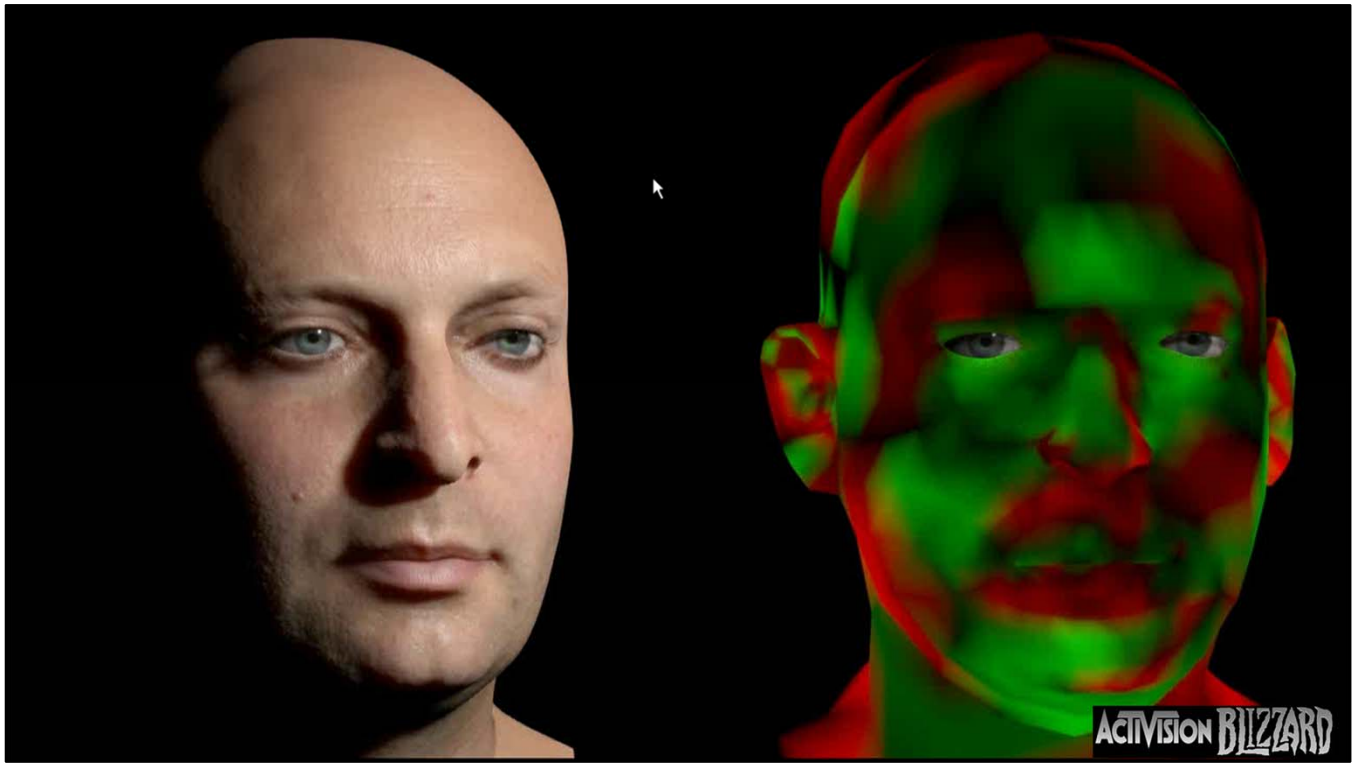
These maps will then be blended together in runtime.



We computed the same for displacement maps.



If we have diffuse maps for the same meshes, we can also composite them with this method.



The result of blending normals, displacement and albedo maps. Driving the mesh with 70 bones, 3000 point mesh.

Thanks for your attention!

Special thanks to (in alphabetical order):

Angelo Pesce, Carl Schnurr, Chris Codding, Chris Ellis, Colin Barré-Brisebois, David Bullat, Dimitar Lazarov, Lee Perry, Jennifer Velazquez, María López, Sébastien Lagarde, Stephen Hill, Steve McAuley , Xian-Chun Wu.

For more information:

YouTube Channel: ActivisionRnD

Blog: www.iryoku.com

ACTIVISION BLIZZARD