IMAGE 2009 Conference

A Method to Compensate for Display System Contrast Ratio Differences in Distributed Simulation

Michael J. Sieverding
L-3 Communications/Link Simulation & Training
Mesa, AZ

Richard M. Rybacki
MetaVR, Inc.
San Antonio, TX

Amos E. Kent
AFRL/RHAE
Mesa, AZ

ABSTRACT

A considerable challenge facing distributed virtual simulation is to minimize correlation differences between networked simulations so that humans-in-the-loop perceive and respond to the same stimulus--as they would in the real world. There are many causes or domains of correlation differences, including Appearance, Behavior, and Time. This paper addresses one small component within the Appearance domain of correlation.

Considerable work has been done across the Services to develop methods of reusing environmental/spatial datasets that not only reduce the schedule and cost of database generation, but also achieve greater correlation between differing simulations; however, even if all simulations were to share the same database geometry, textures, colors, and rendering engine, how they would look through the simulations' different display systems can vary dramatically. Distributed simulation may include many different virtual entities that are outfitted with considerably different display systems. Each display system type may have widely varying resolutions, luminance, and contrast ratios, resulting in different perceived scenes even if all systems are using the same rendering engine and spatial database.

This paper presents a method of compensating for widely varying display system contrast ratios, which results in more similarly perceived out-the-window scenes across different simulations. This paper presents a novel algorithmic approach of modifying database colors and intensities. The principal variable within the algorithm is the difference in measured display system contrast ratios between two simulator systems. Contrast ratio test methods, tools, and results are also presented to provide objective and repeatable measures. This paper also describes a method used to remap all pixel colors and intensities with the adjustment algorithm during run-time, using plug-in shader techniques.

The method described in this paper offers the potential for application across any simulation network where the environment model is built from common, shared datasets, where different types of display systems with widely varying contrast ratios are employed, and where correlated (or at least more similar) perceptions are required.

Presented at the **IMAGE 2009** Conference St.Louis, Missouri, July 2009

INTRODUCTION

Database Standards and Correlation

The Army, Navy, Air Force, and SOCOM have established database standards programs of widely varying scope, but with a remarkably similar selection of in-process dataset formats. The Navy and the Air Force have both conducted studies to show that dataset investments shared at the in-process format level can yield cost and schedule savings of from 60-95%, assuming somewhat consistent database content requirements. File exchange at the in-process dataset format level (OpenFlight, Shape, and

GeoTIFF) can allow database investments to be shared across Services and programs. This should also improve correlation, since differing programs can start their database development process using more similar value-added source data packages rather than from varying raw source data packages with varying pedigrees. This may not necessarily ensure sufficient correlation in networked simulation, but can serve as an excellent initial step to achieve it. Other methods must be developed to help further improve network correlation.

Network Correlation Background

Network correlation is a hugely complex theme. Correlation differences between networked simulations can occur across the domains of Appearance, Behavior, and Time. Appearance relates to the location, size, color, contrast, material, orientation, etc. of objects within a spatial environment. Behavior relates to how those objects move, spawn, emit, absorb, change state, etc. Time relates to objects' duration, recency, sequence, frequency, latency, and when they start/stop. Behavior causes changes in Appearance over Time. Correlation differences in any domain between networked simulations can be significant enough to limit the validity of network events.

Sufficient correlation between networked simulations can be a key component of network event success. Simply trying to identify what is "sufficient" can be very challenging, since it is typically defined using subjective and sometimes fickle operator opinion. If the network is composed of identical simulations developed by the same vendor, sufficient correlation is comparably easy to achieve. If the network is composed of dissimilar simulations independently developed by different vendors, sufficient correlation can become an impossible task, unless sometimes draconian measures are taken to establish standards that can achieve acceptable correlation, but often at the price of eliminated competition, establishment of sole source acquisition, and stagnated technology. It is desired that methods or processes are developed that can be used to enhance correlation, but remain flexible and adaptive.

This paper focuses on improving network correlation within a small subset of the Appearance domain using a flexible and adaptive method.

CASE STUDY OF TWO SIMULATOR PROGRAMS

The A-10 Full Mission Trainer (FMT) program is funded and managed by the Combat Air Forces (ACC, ANG, and AFRC) to procure, operate, and maintain high fidelity virtual A-10 pilot training simulators (FMTs) across CONUS and at overseas locations. The A-10 FMT program has a very large and extensive, nearly global database, with many higher resolution insets to support training Close Air Support (CAS) tasks and skills. The A-10 FMT program includes an extensive DataBase Generation System (DBGS) that adds and modifies its database as requirements change and technologies evolve. The A-10 FMT out-the-window display system uses eight rear-projected facets or channels to provide a full 360 degree horizontal and nearly 120 degree vertical field of view. The A-10 FMT program participates in the overarching Air Force Distributed Mission Operations (DMO) program, to include network simulation events.

The Joint Terminal Attack Controller (JTAC) simulator program is funded by ACC and managed by AFRL/RHA at Mesa, AZ to prototype, demonstrate, and evaluate simulators to train JTAC operators in CAS tasks and skills, using techniques and equipment peculiar to JTAC operators. As a cost savings method, the JTAC program uses the same database built for the A-10 FMT program (and the same image generator), and includes no provisions for a DBGS. The JTAC program consists of two different types of virtual simulators with two different developmental display systems. One is an internally projected dome with a full 360 degree horizontal and 120 degree vertical field of view, using 14 display channels. The other JTAC display system is an internally projected concatenated dome with an approximate 200 degree horizontal and 120 degree vertical field of view, using 13 display channels. Although developmental, the JTAC program has also participated in DMO network simulation events.

Although both program use the same exact database and the same image generator vendor (MetaVR Virtual Reality Scene Generator (VRSG)) and their display systems use somewhat similar DLP projectors from

different vendors, the scenes viewed through their display systems are considerably different and can compromise or limit the types of scenarios used during CAS training in AF DMO. Differences in display resolution account for some of the correlation differences, but the majority comes from large differences in display system contrast ratios. For example, when at the same location in the same database and with the same viewing conditions (time, day, visibility, etc.), an A-10 FMT scene may appear to have good contrast and strong chroma differences, but the JTAC scene will appear washed out, with little ability to distinguish contrast and chroma between objects or within textures.

During networked operation, the ground-based JTAC may direct the A-10 FMT to a target using plain language feature descriptions based on what the JTAC operator sees in his simulator, but the A-10 FMT pilot sees a different scene and the JTAC description may make no sense to him. Because of large display scene differences, target scenarios must be carefully chosen and scripted to ensure that what the JTAC sees is similar what the A-10 pilot sees. A method to compensate for display system contrast ratio differences and improve apparent scene correlation between A-10 FMT and JTAC systems was desired.

Also, the JTAC operators would not only use unaided eyes to determine targets or objects in the dome-displayed scene, but they would also use binoculars, NVGs, and laser range-finder-designators while within the JTAC dome. These JTAC tools have their own display channels that are not affected by the dome's display system contrast ratio. Objects were often much more discernable through the tools than when using unaided eyes--to a degree that it was felt to be unrealistic. A method to improve scene correlation between dome scene and the scene viewed through JTAC tools was also desired.

ALGORITHM DEVELOPMENT

Key Desired Algorithm Characteristics

First and foremost, an algorithm cannot change a display system's contrast ratio. It is what it is. However, if the algorithm were to artificially adjust colors and intensities before going through the display system, the resultant scene could appear more similar to scenes viewed on other display systems with differing contrast ratios.

Since a low contrast ratio takes away luminance contrast and depletes chroma purity, a method is desired to modify database colors and intensities to increase luminance contrast and chroma purity, where reds appear redder, blues bluer, etc. Also, it is desired that full black should remain full black and full white should remain full white.

Since differences in displayed scenes between the A-10 FMT and JTAC systems are principally caused by differences in display system contrast ratios, it is also desired that the algorithm be sensitive to contrast ratio differences.

Chosen Algorithm

Many different methods could be used to modify database colors and intensities for this purpose. Since equal units of the three color primaries (Red, Green, and Blue) do not result in equal perceived intensities, methods were considered that reflected those differences, as were methods to reflect non-uniform gamma correction for the primaries. A square root luminance adjust function was also considered. Various algorithmic solutions were analyzed for acceptability by applying them to Microsoft PowerPoint color swatches having a wide range of chroma and intensities, such as shown in Figure 1.

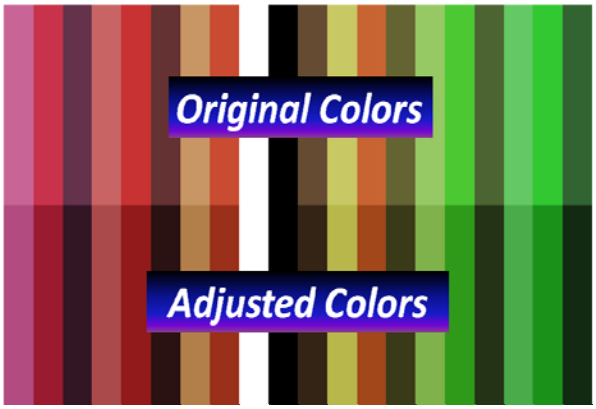

Figure 1

Following subjective analysis of the PowerPoint color swatches, the following method (Table 1) was selected as the most promising.

For each Producer program R,G,B color (range = 000 to 255)
R + G + B = Total Color Units (TCU)
If TCU < 1, then TCU = 1
R, G, B / TCU = Parent Color Ratios (PCR) Rr, Gr, Br
Producer program display contrast ratio = x:1, then Cp = 1/x
User program display contrast ratio = y:1, then Cu = 1/y
$(Cu - Cp)^2$ = a (adjust factor) <This compensates for contrast ratio difference
TCU - ((765 – TCU) • a) = TCU2 <This changes intensity
If TCU2 < 0, then TCU2 = 0
For each PCR Rr, Gr, Br
PCR + ((PCR - .333) • a) = PCR2 <This changes color purity
If PCR2 > 1, then PCR2 = 1
If PCR2 < 0, then PCR2 = 0
For each PCR2 Rr, Gr, Br
PCR2 • TCU2 = Adjusted Ra, Ga, Ba color for the User Program
If Ra, Ga, Ba > 255, then Ra, Ga, Ba = 255

Table 1

Note that the required algorithm inputs are the original RGB from the Producer program (in this case A-10 FMT), and the display system contrast ratios of the Producer (A-10) and User (JTAC) programs. The output is corrected RGB for the User (JTAC) program. Since the algorithm requires display system contrast ratio as an input, the A-10 and JTAC display systems' contrast ratios must first be measured before tests and experiments can be conducted. Neither program had previously captured that data.

CAPTURING CONTRAST RATIOS

What is Contrast Ratio?

Contrast ratio is commonly defined as the ratio of the luminance of the brightest color (white) to that of the darkest color (black) that the display system is capable of producing. The greater the ratio, the greater the dynamic range of the display system luminance, and its ability to mimic the real world. Please note that this definition describes what the display system can attain, not what the projector can produce. Although projectors are frequently described by their vendors as having many thousands-to-one contrast ratios, when the projectors are integrated into a large, multi-channel display system with all projectors/channels operating, it is quite difficult to attain better than a 35:1 contrast ratio with display systems having very large fields of view.

How is Contrast Ratio Measured?

There are numerous ways to measure contrast ratio. The method described here relies heavily upon ANSI Static Contrast Ratio methods, and is based on the very similar FAA AC-120-40B, Appendix 1, Contrast Ratio test methods. The FAA's methods have been applied to testing of hundreds of commercial aviation flight simulator display systems for almost twenty years, and are well accepted and understood by the flight simulation industry. As additional information, the FAA requires all display channels to be operating during contrast ratio measurement and the resulting minimum display system contrast ratio must be at least 5:1.

Contrast Ratio Test Method

To measure contrast ratio, a test sphere 3D model 10 meters in diameter was constructed using 800 emissive polygons in nine degree high rows and nine degree wide columns at the horizon, with the columns becoming narrower toward the zenith and nadir. The polygons were colored in a checkerboard pattern of alternating black and white. Half of the display system was composed of white polygons, the other half black polygons. The A-10 and JTAC computational/ display eyepoints were then positioned at the sphere center. A spot photometer was then used to "shoot" the luminance (in Foot Lamberts) at the center of the two rows of black and white polygons just above and below the horizon line (+9 to -9 degrees elevation), ranging from -108 to +108 degrees azimuth. Sampled polygons (24 white and 24 black) ranged across several display channels for all three tested systems. The average white polygon luminance value was divided by the average black luminance value, yielding a contrast ratio for the display system being tested.

Contrast Ratio Test Results

Using the same test sphere, test equipment, and test methods in all three display systems yielded the following results:

Display System	Contrast Ratio
A-10 FMT	33.65:1
JTAC Concatenated Dome	5.91:1
JTAC 360 Dome	2.05:1

Table 2

The results confirmed the anticipated low contrast ratios in the JTAC domes, especially in the 360 dome. JTAC display system design was selected to satisfy higher priority performance requirements knowing that lower priority performance characteristics would suffer.

Internally projected dome display systems have historically had very low contrast ratios, unless the dome surface is coated or treated with a material having increased specular reflectance, or high gain. The downside of high gain is a very small viewing volume without objectionable intensity falloff. Both JTAC displays require a very large viewing volume, with multiple observers able to roam within the domes using tactical equipment. For this reason the JTAC dome surfaces are nearly Lambertian (diffuse) in their reflectance, having a gain of about 1. In the case of the JTAC dome design, contrast ratio was traded off in favor of a large viewing volume. Also, facility size restrictions prevented use of alternative display technologies for JTAC. Tradeoffs always occur during the selection of any display system. That's just the way it is.

Inputting Contrast Ratios to the Algorithm

The next step was to input the contrast ratio values into the algorithm and adjust color tables and palettes for all polygons and all textures. This step was known to be the most time consuming, since considerable off-line processing would be required.

During informal discussions with MetaVR personnel, they suggested that offline color adjustment may not be necessary, and that the image generator itself could be programmed to make the adjustment during runtime. That seemed very appealing since several algorithm iterations were anticipated before any type of optimal result could be attained.

USING THE GRAPHICS SHADER FUNCTION TO SUPPORT THIS EXPERIMENT

One of the most significant advances in computer graphics in recent years is the development of the programmable vertex and pixel shader. The highly programmable nature of per-vertex and per-pixel operations has opened the door to stunning advances in realism. These advances have been realized in both the commercial gaming sector as well as military visual simulation. While the benefits of shaders for advanced lighting and shading are well understood, the utility of the programmable GPU as a general image processing engine offers many other capabilities as well.

Since both the A-10 FMT and JTAC programs use a MetaVR VRSG image generator, MetaVR developed a plug-in interface that allows the contrast adjust algorithm to be inserted into the scene generation pipeline during run-time as a Dynamic Link Library (DLL) call. The DLL implements a "user-draw" function which VRSG calls after it has rendered the 3D scene. All pixels are redefined using the DLL in accordance with the contrast adjust algorithm previously described in this paper. The processing overhead caused by this additional function is estimated at approximately one millisecond. The MetaVR VRSG implementation is described here to serve as a specific example, but this algorithm and DLL method could be easily adapted to other image generators that offer a similar plug-in interface.

Background and Technical Description of the Shader Implementation

It is often useful to "post process" a rendered scene to add a variety of special effects. For sensors, this could include Gaussian blur (for optical focus), simulated noise, digital zoom, or polarity inversion. MetaVR VRSG also performs a Discrete Cosine Transform (DCT) followed by an Inverse DCT to simulate the compression artifacts of digital video. Atmospheric effects can also be done this way, when used in conjunction with a rendered image representing depth (or slant range). In a post-processing step, saved pixel depths

can be used to reconstruct 3D positions to apply various atmospheric models such as haze and fog. A common technique in games is to remap an input RGB into an output RGB via a cube map, to achieve a different ambiance as you enter different levels of the game.

If a particular effect can be accomplished in a post process, then the set of active effects need not be considered when the pixel is initially rendered. This greatly simplifies the initial rendering step. Furthermore, moving such processing to a post-processing step can be more efficient, as each screen pixel need be visited only once. If performed during the initial rendering pass, pixels that will be later depth-occulted will represent wasted bandwidth. Some effects are only possible when done as a post-processing step. An example of this is an image convolution, such as a Gaussian blur. Since each output pixel is a function of a neighborhood of input pixels, it is not possible to perform such an effect in the initial rendering pass.

Using the GPU for a post-processing effect generally involves the following steps:

1) render the scene to the frame buffer as normal

2) harvest the frame buffer image into a texture map

3) render a quad covering the entire frame buffer, using the texture from 2) as input, outputting a new pixel at each frame buffer location

Step 1 can be performed without any specific knowledge of any forthcoming post-processing effects. Step 2 is necessary to make the rendered image from Step 1 available for access by a pixel shader. In Direct3D, Step 2 is accomplished with the IDirect3DDevice::StretchRect method. This method moves the image rendered as an anti-aliased frame buffer to a video-memory texture that can be read by a pixel shader in the post-processing step.

One of several possible plug-in entry points supported by MetaVR VRSG is the extUserDraw function which is called by VRSG after VRSG has rendered all viewports to the frame buffer. This function provides the opportunity to insert the contrast adjustment algorithm. The details of the algorithm are captured in the source of the pixel shader which will be described later. The C++ code in extUserDraw is straightforward and implements the 3 steps described above.

Another capability of the VRSG plug-in interface is for the ability of a plug-in to extend VRSG's user interface. The Contrast Adjust plug-in adds a tab to VRSG's user interface to allow the user to control the two primary input variables:

· the contrast ratio of the "Producer Program"

· the contrast ratio of the "User Program"

Figure 2 illustrates this user interface slider control. In this case, if the Producer Program (A-10 FMT) display system contrast ratio is 35:1 and the User Program (JTAC) display system contrast ratio is 4.5:1, the sliders can be set accordingly, the algorithm is adjusted, and all display pixels are redefined in accordance with the adjusted algorithm during runtime.

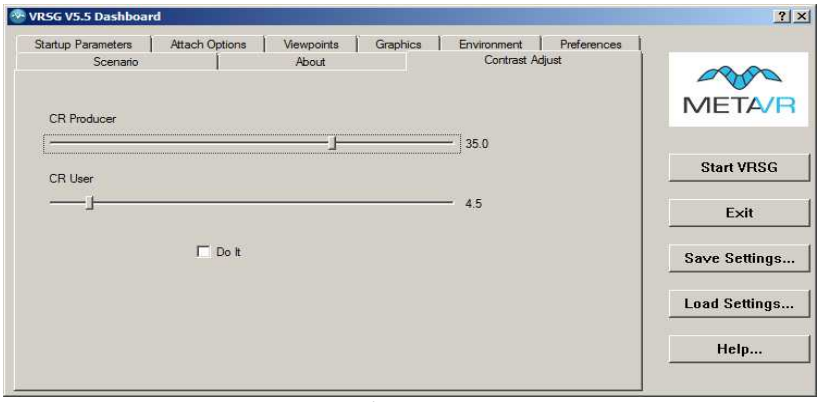

Figure 2

The details of the contrast adjustment implementation are encapsulated in the pixel shader source. The pixel shader takes as input each original RGB value. Each output pixel is a function of the input RGB and 2 scalar constants: CRP - contrast ratio of the "producer" display, and CRU - contrast ratio of the "user" display. Thus the operation of the pixel shader can be described as a function: pixelout = ContrastAdjust(pixelin, CRP, CRU)

The full source of the pixel shader is shown in Table 3.

```
ps.2.0

#define CRP c0.x
#define CRU c0.y
#define ADJ c0.z

; some useful constants
def c1, 1.0, 1.0, 0.333, 3.0
def c2, 0.001, 0, 0, 0        ; min TCU allowed

; declare registers
dcl t0
dcl_2d s0

; sample the input image
texld r0, t0, s0

dp3 r1, r0, c1.x          ; r1 = r+g+b
max r1, r1, c2.x          ; clamp to a minimum value
rcp r1.y, r1.x            ; 1/TCU
mul r0.rgb, r0, r1.y      ; r,g,b / TCU

add r2, c1.w, -r1.x       ; 3.0-TCU
mul r2, r2, ADJ           ; (3.0-TCU)*ADJ
add r2, -r2, r1.x         ; TCU2 = TCU - (3.0-TCU)*ADJ

add r3, r0, -c1.z         ; PCR-0.333
mul r3, r3, ADJ           ; (PCR-0.333)*ADJ
add r3, r3, r0            ; PRC2 = PCR + (PCR-0.333)*ADJ

min r3, r3, c1.x          ; clamp to <= 1
max r3, r3, c2.w          ; clamp to >= 0

mul r3, r3, r2            ; PCR2*TCU2

; Set alpha and output result
mov r3.a, c1.x
mov oC0, r3
```

Table 3

Upon completion of the execution of the pixel shader, the input RGB has been replaced with a contrast-adjusted RGB. It must be noted that this DLL plug-in was implemented only on the JTAC MetaVR image generators, not the A-10, since the appearance of the A-10's displayed scenes was not objectionable and required no correction or adjustment.

ALGORITHM RESULTS

Subjective Assessment

The following two scenes were the first before/after rendering scenes using the algorithm and applying a relatively small contrast ratio difference to the DLL GUI slide bar and not using either of the JTAC

display systems. These scenes were captured on a laptop.

Figure 3 – Laptop Before DLL

Figure 4 – Laptop After DLL

It was felt that there was enough contrast difference between the two images to warrant further algorithm development and testing.

Once contrast ratio measures were collected and the DLL algorithm was installed in JTAC, the before/after differences were even more apparent.

The following two scenes were captured from the JTAC concatenated display system. Figure 5 is with all channels operating without the DLL being applied. Figure 6 is with the forward/center channel modified with the DLL, the surrounding channels remaining unmodified.

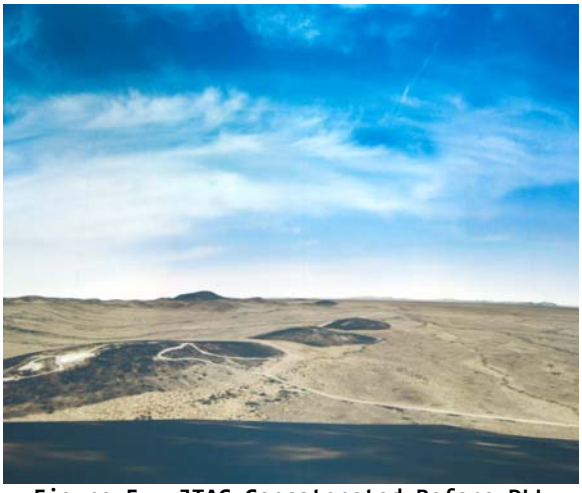

Figure 5 – JTAC Concatenated Before DLL

Figure 6 – JTAC Concatenated After DLL Applied to Center Channel

Objective Measures

Test methods were developed to collect technical measures of the effects of the DLL algorithm. These were used to help objectively validate what observers could subjectively sense between "before" and "after" DLL algorithm application.

Measured Chroma Increase. The spot photometer used to collect data for contrast ratio measurements (Konica Minolta CS-100A with a one degree aperture) also captured X,Y CIE color space. The A-10 and the JTAC 360 dome were initialized to the same location, elevation, and attitude in the same database. Nine easy-to-identify-and-repeat objects in the database scene were selected and X,Y measures were collected from them. Before application of the DLL algorithm, the average distance of

the JTAC X,Y coordinates from the A-10 X,Y coordinates was .053 X,Y units. After application of the DLL algorithm, the average distance from the A-10 color coordinate location decreased to .048 X,Y units or about 10% closer; a small but measurable amount.

Measured Contrast Increase. The 800 polygon test sphere was modified to include a row of twelve colored polygons, each with a different color but of generally mixed chroma. With the JTAC 360 dome initialized to the modified test sphere center, the same photometer was used to capture luminance values from each colored polygon and from a reference full-white polygon, both before and after application of the DLL algorithm. After application of the DLL algorithm, the average luminance ratio of the full-white polygon to the colored polygons increased from 1.44:1 to 1.63:1, a 14% increase in contrast ratio.

Test Limitations

The algorithm presented here results in a loss of brightness and a reduction in color dynamic range. That reduction is theoretical only, since the JTAC's low contrast ratio already places large limits on color dynamic range that the DLL algorithm effectively extends.

Although use of the DLL plug-in provided many advantages during test and validation, it may also have induced artifacts such as the effects of anti-aliasing at the pixel level. It is unknown if test results may differ if anti-aliasing is turned on or off.

The DLL plug-in affects all pixels, including pixels that represent self-illuminating objects such as point lights. Self-illuminating objects should probably be immune from the DLL algorithmic treatment, since any reduction in their brightness is ill-advised for a variety of reasons.

An additional test of the algorithm without using the DLL plug-in, but instead modifying all database color tables and palettes offline would be useful to isolate self-illuminating objects from algorithm effects and identify the effects of DLL artifacts, if any.

Testing and validation of the algorithm occurred only in day scenes. More comprehensive testing is advised.

Algorithm application need not be considered if the differences in display system contrast ratios are relatively small, or if no objectionable characteristics are noted in either stand-alone or in networked operation modes. Don't fix something that isn't broken.

Using measured display system contrast ratios as input to the DLL algorithm should be considered only as a starting point. The algorithmic approach presented here is not based on physics, color science, or what is known of human visual perception. It is based upon desired algorithm characteristics. The final setting must consider subjective assessments of scene improvements. Tweak it until the scene looks best and scene correlation differences appear least objectionable.

The JTAC systems use the A-10 FMT database without change, to save cost and schedule during development and prototyping. The A-10 FMT database colors and intensities have been tailored and tweaked to satisfy the subjective opinions of A-10 pilots for several years. If JTAC had originally built the database and tailored it to suit JTAC operators, it would undoubtedly be subject to criticism (too much contrast, too much chroma) if installed unchanged in the A-10 FMT. A reciprocal method of applying the same algorithm presented here could be used as a starting point to adjust JTAC colors and intensities in the A-10 FMT.

CONCLUSIONS AND RECOMMENDATIONS

Algorithmic options exist to improve scenes in display systems with low contrast ratios and to improve correlation between networked systems having large display system contrast ratio differences.

Use of modern graphics processors' programmable vertex and pixel shader functions can simplify and speed up algorithm development and testing.

It is recommended that additional tests be conducted by modifying all color palettes (not using the DLL plug-in) to ensure optimum system performance with minimal risk of artifacts.

It is recommended that standard tests, methods, and tools to measure display system contrast ratio be developed for networked simulation programs that include devices with a significant range of display system contrast ratios.

AFTERWARD

The three-dimensional test model sphere composed of 800 polygons is available in OpenFlight (in two versions: one black and white checkerboard used to measure display system contrast ratios, the other with several rows of colored squares added to the black and white checkerboard used to measure algorithm effect on contrast). The C++ code used for extUserDraw to set up the DLL plug-in is also available. If interested, please contact any of the authors for copies.

AUTHOR BIOGRAPHIES

Mike Sieverding

Mike is a Senior Engineer with L-3 Communications/Link Simulation and Training. He currently supports the Air Force Research Laboratory/Warfighter Readiness Research Division's (AFRL/RHA) Rehearsal Enabling Simulation Technologies (REST) project in Mesa, Arizona. Mike has over thirty years experience operating, specifying, procuring, testing, managing, and advising DOD simulation R&D and aircrew training programs, and has been a frequent presenter at IMAGE conferences. During his active duty Air Force career he served as an acquisition program manager for several Air Force flight simulation and database standardization programs, served as a C-130 Simulator Certification officer, and flew transport, photo-mapping, and hurricane hunter missions as a C-130 navigator.

Rich Rybacki

Rich is cofounder and chief technology officer at MetaVR, Incorporated. His primary responsibilities include the development and support of VRSG, MetaVR's image generator product. He acquired his passion for computer graphics and distributed simulation while enlisted in the US Air Force developing software for SIMNET programs. Rich received a M.S. in Computer Science from the University of Texas at San Antonio in 1995.

Amos Kent

Amos is a computer engineer supporting the Air Force Research Laboratory/Warfighter Readiness Research Division's (AFRL/RHA) Rehearsal Enabling Simulation Technologies (REST) project in Mesa, Arizona. Amos received a B.S.E. in Computer Systems Engineering from Arizona State University in 2006. Amos was previously enlisted in the U.S. Navy as a submarine nuclear operator for six years, where he was responsible for the operation and maintenance of a nuclear propulsion plant on a Los-Angeles class attack submarine.