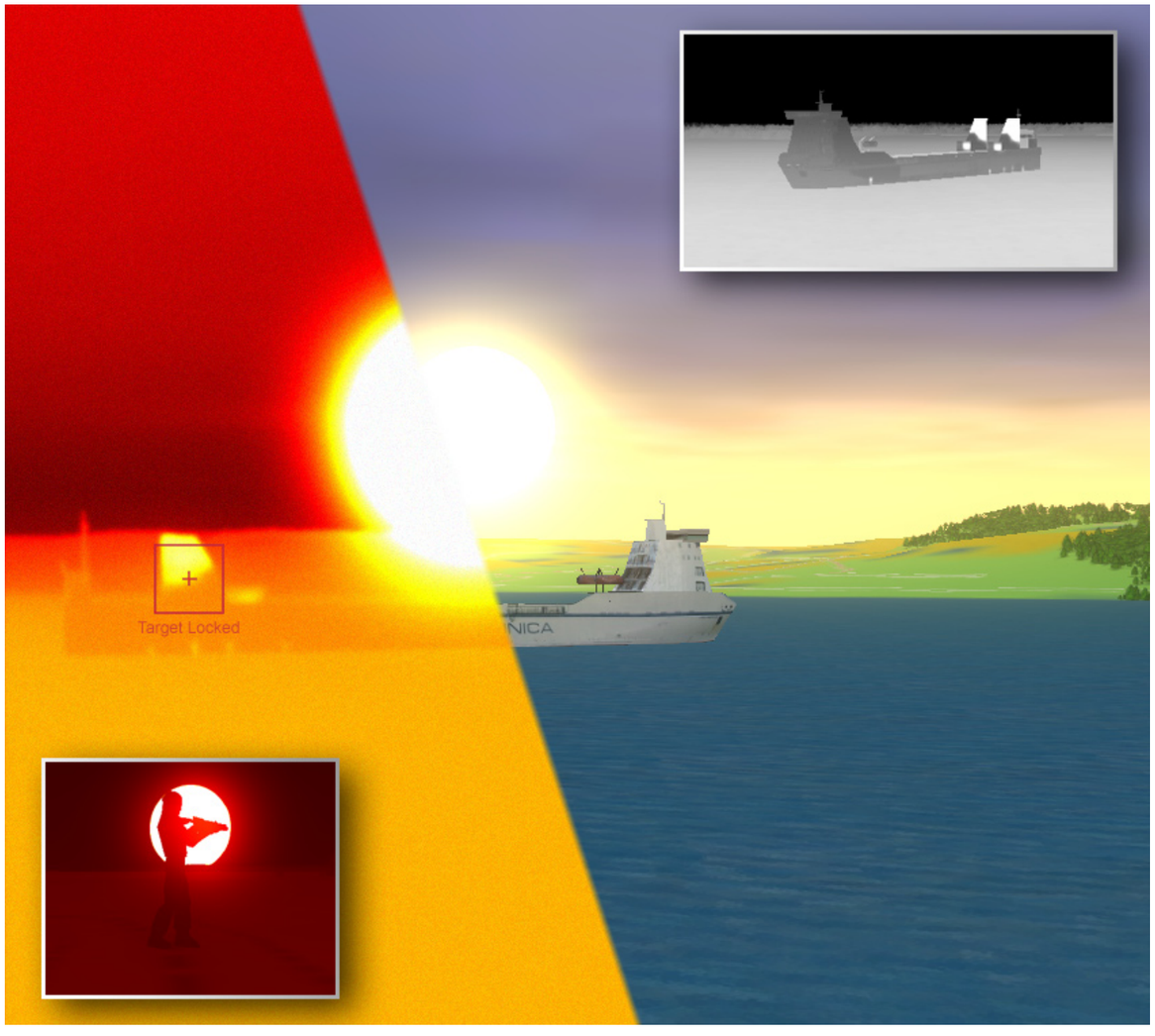
High Dynamic Range Imaging i EWSim

Användarhandledning och teknisk dokumentation

HENRIC KÄRNHALL, KJELL ANDERSSON, LINUS HILDING,
RAGNAR HAMMARQVIST, LARS TYDÉN



FOI-R--2773--SE Metodrapport
ISSN 1650-1942 Juni 2009

Informationssystem

FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1000 anställda varav ungefär 800 är forskare. Detta gör organisationen till Sveriges största forskningsinstitut. FOI ger kunderna tillgång till ledande expertis inom ett stort antal tillämpningsområden såsom säkerhetspolitiska studier och analyser inom försvar och säkerhet, bedömning av olika typer av hot, system för ledning och hantering av kriser, skydd mot och hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets Forskningsinstitut
Informationssystem
Box 1165
581 11 Linköping

Tel: 013-37 80 00
Fax: 013-37 81 00

www.foi.se

Henric Kärnhall, Kjell Andersson, Linus Hilding,
Ragnar Hammarqvist, Lars Tydén

High Dynamic Range Imaging i EWSim

Användarhandledning och teknisk dokumentation

Titel	High Dynamic Range Imaging i EWSim
Title	High Dynamic Range Imaging in EWSim

Rapportnr/Report no	FOI-R--2773--SE
Rapporttyp Report Type	Metodrapport Methodology report
Sidor/Pages	33 p
Månad/Month	Juni/June
Utgivningsår/Year	2009
ISSN	ISSN 1650-1942
Kund/Customer	FM
Forskningsområde Programme area	6. Telekrig och vilseledning Electronic Warfare
Delområde Subcategory	61 Telekrigföring med EM-vapen och skydd 61 Electronic Warfare including Electromagnetic Weapons and Protection
Projektnr/Project no	E7015
Godkänd av/Approved by	Mikael Sjöman

FOI, Totalförsvarets Forskningsinstitut
Avdelningen för Informationssystem
Box 1165
581 11 Linköping

FOI, Swedish Defence Research Agency
Information Systems
Box 1165
SE-581 11 Linköping

Sammanfattning

Inom FoT-projektet dynamisk duellsimulering telekrig vidareutvecklas ramverket EWSim evolutionärt. EWSim ger möjlighet att värdera telekrig i multispektrala scenarier. Detta ramverk har kompletterats med hantering av HDR-grafik (High Dynamic Range Imaging), spektrumhantering och posteffekter. HDR-tekniker gör det möjligt att arbeta med ett bildformat som har högre precision och större dynamik än en normal datorskärm kan visa, dvs mer än 8 bitar per färgkanal. Med den nya metoden så kan de bildanalysalgoritmer som finns i EWSim och används i de elektrooptiska målsökarmodellerna och siktena arbeta direkt på HDR-bilddata för att på ett bättre sätt efterlikna verkligheten. Med HDR så ökar upplösningen i temperatur från 256 nivåer till ett flyttal med upp till 48 bitar.

Inom telekrig finns det flera aspekter som är kopplade till färgdynamiken som är svåra att simulera med vanlig 8-bitarsgrafik i multispektrala scenarier. I dessa scenarier kan samtidigt finnas ett flertal elektrooptiskasensorer och medel för att störa dessa såsom facklor, rök och lasrar, som alla arbetar inom olika våglängdsområden och med mycket stora skillnader i intensiteter. När verkan av dessa motmedel ska värderas där de simuleras i samma scenario så krävs att bildrenderingen arbetar med full dynamik och att sensorerna i slutändan bestämmer vilket intensitetsspann de vill använda. Exempelvis så kan exponeringen för att se ett fordon i terräng med god kontrast sättas till 10–30°C. Motorutblåset är dock 300°C vilket gör att kontrasten med vanlig grafik blir mycket dålig om hela spannet 10–300°C används, men med HDR kan elektrooptiska system, såsom kameror och målsökare, använda valfri exponering och ändå få korrekta kontraster.

HDR-grafiken möjliggör dessutom att en eller flera posteffekter kan användas. Dessa effekter appliceras efter det att hela scenen är renderad för att efterlikna olika typer av optiska fenomen eller oönskade artefakter som uppkommer i elektrooptiska system. Effekterna hanteras som moduler i en sammankopplad dynamisk kedja där man kan slå av och på eller byta ordning på dem. Den modulbaserad design gör det även enkelt att implementera och lägga till nya effekter. För tillfället är följande posteffekter tillgängliga, oskärpa, brus för att försämra signal till brusförhållandet och glödeffekten även känt som optisk punktförstoring, blooming eller glow. Effekten får ljusintensiva områden att blöda över på mörkare delar.

I dagens datorers grafikkort finns funktionalitet som kan genererar HDR grafik. Detta innebär att det med hög noggrannhet och med en snabb beräkningshastighet går att simulera de effekter som uppkommer i ett elektrooptiskt siktessytem eller målsökare. Tack vare den höga beräkningshastigheten så kan bilderna användas vid simulering med människan i loopen eller med hårdvara som till exempel målsökare i loopen. Med användning av HDR-funktionaliteten i grafikkort möjliggörs dessutom att bildalstrande målsökare kan användas i hårdvara i loopen simuleringar till en förhållandevis låg kostnad jämfört med andra befintliga tekniker. Sammantaget ger HDR tekniken nya och förbättrade möjligheter att på ett realistiskt sätt simulera telekrigets påverkan på EO-sensorer.

Nyckelord: datorgrafik, högdynamik, bildgenerering, bildanalys, elektrooptik, EWSim

Summary

In the FOI project duel simulation in electronic warefare there has been work performed to enhance the simulation of sensors, sights and cameras that normally use higher dynamic color range than traditional computer graphics and computer screens can show. This report describes the framework for high dynamic range imaging that has been implemented in the duel simulator EWSim.

Keywords:

computer graphics, high dynamic range, imaging, image analysis, electro optics, EWSim

Innehållsförteckning

1 Inledning

EWSim, Electronic Warfare Simulation interface model, är ett ramverk för distribuerade telekrigssimuleringar till vilket modeller för telekrigsscenarion utvecklas och utvärderas. Ramverket består av tre delar:

- Scenarioplanering och konfiguration (NetScene) där olika plattformar, t ex stridsvagnar och helikoptrar, konfigureras med olika komponenter såsom t ex sensorer och vapen.
- Dynamisk duell. Ett scenario där ingående delar interagerar i en simulerad och visualiserad miljö. Den simulerade tiden kan gå i realtid, snabbare än realtid eller långsammare än realtid.
- Utvärdering där dueller kan återuppspelas och loggade data analyseras.

Visualiseringen som används vid den dynamiska duellsimuleringen i EWSim har utökats med en modell för visualisering av högre färgdynamik än vad som traditionellt varit möjligt med datorgrafik och datorskärmar. Det här dokumentet beskriver ramverket för högdynamisk rendering, HDR-rendering[HDR], som har införts i EWSim.

2 Läsanvisning

Det här dokumentet riktar sig dels till programmerare som ska underhålla eller vidareutveckla HDR-ramverket, men också till scenarioskapare som bara vill använda HDR-grafik i sina simuleringar. Utvecklare bör läsa hela dokumentet men användare av EWSim och NetScene klarar sig med att läsa kapitel 3, 9 och 10. Kapitel 3 beskriver bakgrunden till vad HDR är och vad det används till. Kapitel 9 beskriver hur man inför HDR-grafik i ett scenario med NetScene och kapitel 10 beskriver hur man använder HDR-grafiken i EWSim. För den som ska använda HDR-grafik i EWSim kan det även vara bra att läsa kapitel 5 som går in på detaljer om exponeringsfunktioner.

3 Bakgrund

EWSim har kompletterats med ett ramverk för hantering av HDR-grafik (High Dynamic Range Imaging), spektrumhantering och posteffekter. HDR-tekniker gör det möjligt att arbeta med ett bildformat som har högre precision och större dynamik än en normal datorskärm kan visa, dvs mer än 8 bitar per färgkanal. Den färgrymd som visas på en skärm när man visualiserar HDR-bilder kontrolleras av en exponeringsfunktion. Denna exponeringsfunktion transformerar en delmängd av den totala HDR-färgrymden in till 8-bitars heltal. De värden som är för stora klipps till att visa det ljusaste möjliga värdet, dvs helt vitt, och värden som hamnar utanför den undre gränsen blir helt svarta. Vilken mängd som visas styrs av en exponeringsfunktion som kan ställas manuellt eller automatiskt.

Bildanalysalgoritmer som t ex används av målsökande missiler kan arbeta direkt på HDR-bilddata. Med HDR så ökar upplösningen i temperatur från 256 nivåer till ett flyttal med 48 bitar. Detta ger möjlighet att få med temperatur- och färgkontraster som återfinns i verkligheten med varma motorutblås, sol, facklor och laserstörning som exempel. Exempelvis så kan exponeringen för att se ett fordon i terräng med god kontrast sättas till 10–30°C. Motorutblåset är dock 300°C vilket gör att kontrasten med vanlig grafik blir mycket dålig om hela spannet 10–300°C används, men med HDR kan elektrooptiska system, såsom kameror och målsökare, använda valfri exponering och ändå få korrekta kontraster.

Utvecklingen inom datorgrafik och grafikprocessorer har gjort att det idag finns grafikkort med hårdvarustöd för rendering med hög dynamik. EWSim bygger på OpenSceneGraph [OSG], vilket är ett öppet ramverk för visualisering. OpenSceneGraph bygger i sin tur på OpenGL [OPENGL], vilket är en industristandard för grafik. HDR-ramverket som är utvecklat för EWSim använder sig av OpenSceneGraph och OpenGL för att utnyttja hårdvaruaccelereringen av HDR-rendering som finns i moderna grafikkort.

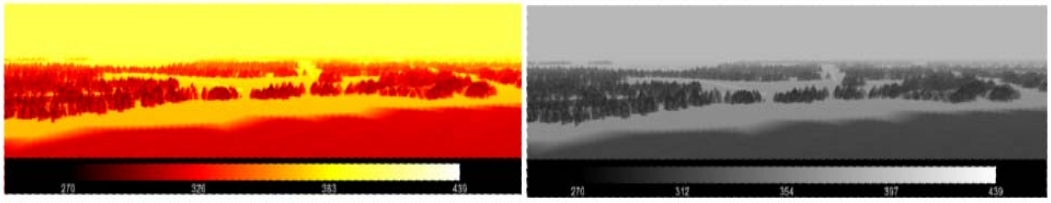
3.1 HDR ur ett telekrigsperspektiv

Inom telekrig finns det flera aspekter som är kopplade till färgdynamiken som är svåra att simulera med vanlig 8-bitarsgrafik. Man använder sig av olika sensorer, kameror och målsökare, samt medel för att störa dessa såsom facklor, rök och lasrar, som alla arbetar inom olika våglängdsområden och intensiteter. För att kunna simulera dessa i samma scenario så krävs att visualiseringen arbetar med full dynamik och att sensorerna i slutändan bestämmer vilket intensitetsspann de vill titta på.

Införandet av HDR-rendering i EWSim har gjort att de sensormodeller som simuleras nu fungerar tillsammans. En fackla som fälls från ett flygplan strålar med mycket högre intensitet än vad flygplanet själv gör. Samtidigt så kan målsökningen av en missil ändå urskilja detaljer i mindre strålande föremål såsom båtar och fordon.

3.2 Spektrumhantering

I samband med övergången till HDR-grafik infördes möjligheten att konfigurera ett eller flera alternativa spektrum där varje spektrum har sin unika uppsättning av texturer och inställningar för t ex dimma, sol och exponering. För tillfället finns utöver det visuella spektrumet även Shortwave Infrared och Longwave Infrared som valbara spektrum. I de infraröda spektrumen har texturernas ljusintensitet kopplats direkt till motsvarande temperaturvärde mätt i Kelvin. För att få en mer intuitiv känsla av vad t ex ett infrarött spektrum visar kan man aktivera olika färgmappningar med tillhörande legend som visar kopplingen mellan färg och faktiska temperaturvärden, se Figur 1.



Figur 1 Exempel på färgmappningar med tillhörande legend. Från vänster: Iron, Gråskala

3.3 Posteffekter

När HDR-grafik är aktiverad finns också möjligheten att aktivera en eller flera posteffekter. Dessa effekter appliceras efter det att hela scenen är renderad och används för att efterlikna olika typer av optiska fenomen eller oönskade artefakter som uppkommer i elektrooptiska system. Detta innebär att det med hög noggrannhet och med en snabb beräkningshastighet går att simulera de effekter som uppkommer i ett elektrooptiskt siktessytem eller målsökare. Tack vare den höga beräkningshastigheten så kan bilderna till och med användas vid simulering med människan i loopen. Effekterna hanteras som moduler i en sammankopplad dynamisk kedja där man kan slå av och på eller byta ordning på dem. Tack vare modulbaserad design är det också enkelt att implementera och lägga till nya effekter. För tillfället är följande posteffekter tillgängliga (se Figur 2):

- **Oskärpa** – Effekten applicerar ett gaussiskt filter för att jämna ut detaljer och kan användas för att t ex simulera oskärpa i en lins.
- **Brus** – Effekten adderar brus för att försämra signal till brusförhållandet och kan t ex användas för att simulera en brusig kamera.
- **Glöd** – Även känt som optisk punktförstoring, blooming eller glow. Effekten får ljusintensiva områden att blöda över på mörkare delar. Den fysikaliska förklaringen är att inga linser är helt perfekta och dessa felaktigheter blir synbara vid starkt ljus och höga kontraster. En annan orsak är också mättnadseffekter i sensorer. Effekten används för att återskapa ett mer realistiskt resultat när man tittar på ljusintensiva föremål så som en brinnande fackla, en varm raket eller solen på himlen.

Figur 2 Exempel på posteffekter. Från vänster: oskärpa, brus, glöd.

3.4 Högdynamiktexturer

För att få ut så mycket som möjligt av HDR-grafiken har de flesta texturer i EWSim bytts ut mot HDR-texturer. Det har också medfört att alla terrängtexturer har genererats om. De bildformat som nu används är Radiance HDR [RADIANCE] och OpenEXR [OPENEXR], vilka båda stödjer högre dynamik i sina färgvärden.

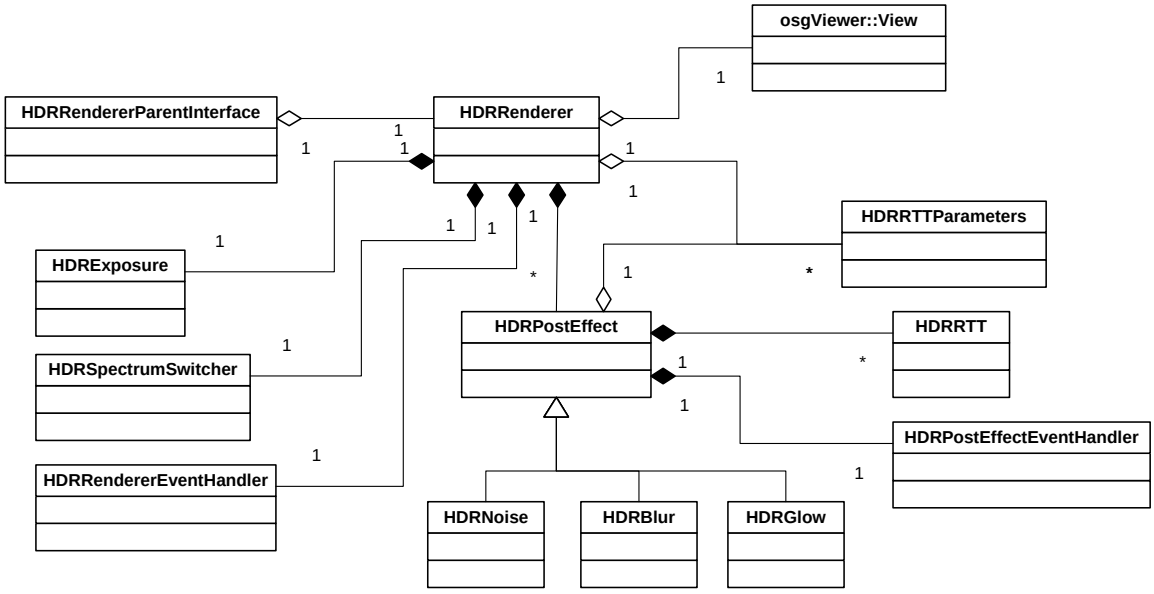
Radiance HDR finns stöd för i OpenSceneGraph men har något sämre precision än EXR, varför OpenEXR integrerats i EWSim.

4 HDR ramverket i EWSim

HDR-ramverket fungerar som en plugin till den befintliga visualiseringen och möjliggör att vyer som tidigare använde normal 8-bitars rendering enkelt kan göras om till att rendera i HDR-format.

4.1 Översikt

Ramverket för HDR består av en topklass, HDRRenderer, vilken ansvarar för HDR-renderingen av en vy. För att kunna rendera i högre dynamik så krävs att renderingen görs till en bild in minnet, en så kallad off-screen textur, vilken inte direkt visas på skärmen. Före denna textur ritas ut på skärmen i lägre dynamik så kan ett antal efterbearbetningssteg, eller så kallade post-effekter, appliceras på den renderade bilden. Dessa effekter arbetar på bilden som har högre dynamik och kan t ex användas för att skapa effekter såsom brus (noise), suddighet (blur) och blödningseffekter (glow). Efterbearbetningseffekterna använder sig också av bilder i minnet för att rendera sina mellanresultat. Abstraktionen av dessa off-screen texturer görs av en klass som i ramverket heter HDRRTT, vilket är en förkortning av High-Dynamic-Range Render-To-Texture. Efterbearbetningseffekter implementeras i form av klasser som ärver från en basklass HDRPostEffect.



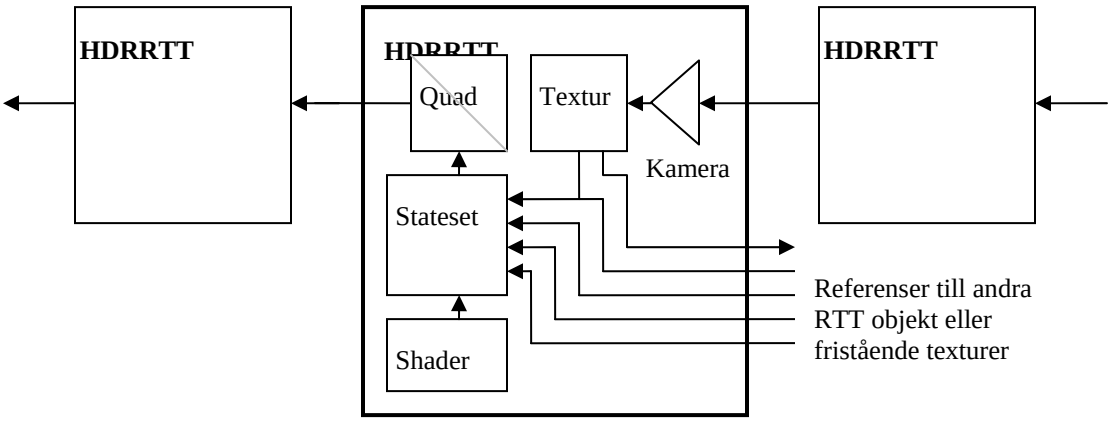
Figur 3 Klassdiagram över HDR ramverket

Följande tre kapitel 4.2, 4.3 och 4.4 beskriver HDR ramverket på klass-nivå i ett botten-upp perspektiv. Först förklaras grundobjektet HDRRTT, därefter HDRPostEffet som bygger sina effekter med hjälp av HDRRTT-objekt och sist beskrivs HDRRenderer som förutom hanterar själva renderingen även fungerar som en container för HDRPostEffect-objekt. I kapitel 4.3 om post-effekter finns också tre underkapitel som beskriver respektive post-effekt.

4.2 HDRRTT

Den minsta byggstenen i HDR-ramverket är klassen HDRRTT som implementerar ett RTT-objekt. RTT står för Render To Texture och har som uppgift att rendera en godtycklig scengraf till en off-screen textur. RTT-objektet har också en quad (två polygoner som formar en kvadrat) med ett stateset som pekar ut en shader och en till fyra texturer, där en av texturerna kan vara RTT-objektets egen

textur. Quaden i sig är en scengraf vilket betyder att ett annat RTT-objekt kan använda ett RTT-objekt som input-scengraf och därmed forma en av ihopkopplade RTT-objekt. Kapitel 4.3, som behandlar post-effekter visar exempel på hur RTT-objekten kan kopplas samman för att bilda önskade post-effekter. Figur 4 visar en kedja av tre RTT-objekt där det mellersta RTT-objektet beskrivs i detalj. Observera att den interna texturen kan användas både som textur till sin egen quad eller till ett annat RTT-objekt.



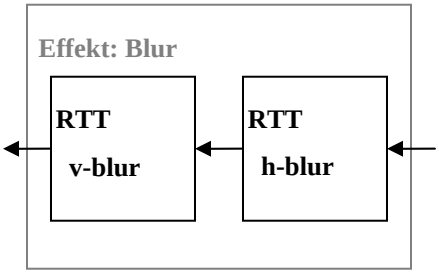
Figur 4 Kedja av tre RTT-objekt där det mellersta RTT-objektet beskrivs i detalj

4.3 HDRPostEffect

HDRPostEffect är en basklass som implementerar all gemensam hantering av en post-effekt. Klassen är abstrakt eftersom vissa funktioner är rent virtuella. Dessa funktioner är listade i kapitel 7.3 som beskriver hur man implementerar en ny effekt. En effekt består av en eller flera sammanlänkade RTT-objekt. Följande tre kapitel 4.3.1, 4.3.2 och 4.3.3 beskriver tre olika effekter och hur de byggs upp med RTT objekt.

4.3.1 HDRBlur

Effekten applicerar ett gaussiskt filter för att jämna ut detaljer och kan användas för att t.ex. simulera oskärpa i en lins. Filtret har en kärna med 13x13 element men beräkningen har delats upp i två steg för att öka prestanda. Först appliceras en horisontell kärna och sedan en vertikal kärna. Mängden oskärpa kan styras med en parameter (parametern Blur). Parametern är en skalfaktor som skalar kärnans utbredning. Om faktorn sätts till 0.0 appliceras hela kärnan på en enda pixel och resultatet blir då samma som originalbilden. Om faktorn sätts till 1.0 sker ingen skalning och kärnan mappar direkt mot 13x13 pixlar. Om faktorn är större än 1.0 omfattar kärnan ett större område men blir då också glesare. Figur 6 visar hur Blur-effekten påverkar originalbilden.

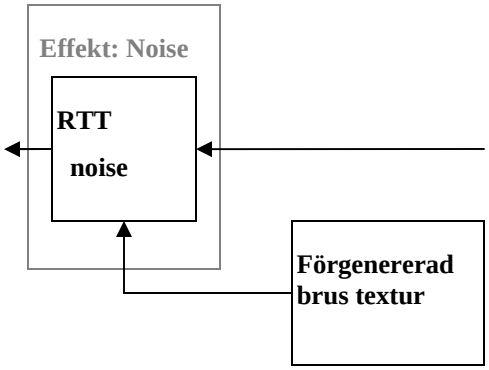


Figur 5 Effekten blur består av två RTT-objekt, v-blur och h-blur

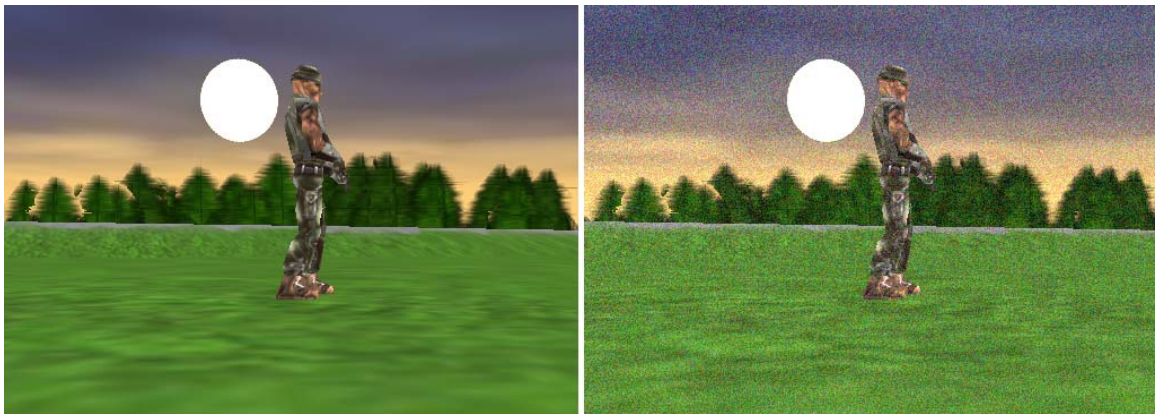
Figur 6 Före och efter-bild vid applicerande av blur-effekt.

4.3.2 HDRNoise

Noise-effekten används för att simulera brus i sensorer. Bruset är slumpmässigt i varje bilduppdatering. Brusets styrka sätts med parametrarna index och range som motsvarar en offset i intensitet samt en skalning av bruset som vid skalning 1.0 motsvarar brusvärden mellan 0.0 och 1.0. Effekten består av endast en RTT och en förgenererad brustextur. Brustexturen slumpas fram när effekten initieras. Den pixel-shader som är kopplad till RTT objektet använder sedan brustexturen dels som textur och dels som distorsionstextur. Brustexturen adderas till originalbilden, men först efter att den har förvrängts, där brustexturen i sig används som förvrängningskälla, i inverkan av ett slumpfrö (seed). Detta får ett väldigt slumpmässigt beteende. Det sker också en skalning och offsetförskjutning så att bruset anpassas till parametrarna index och range. Slumpfröet ändras varje frame. Figur 8 visar hur Noise-effekten påverkar originalbilden.



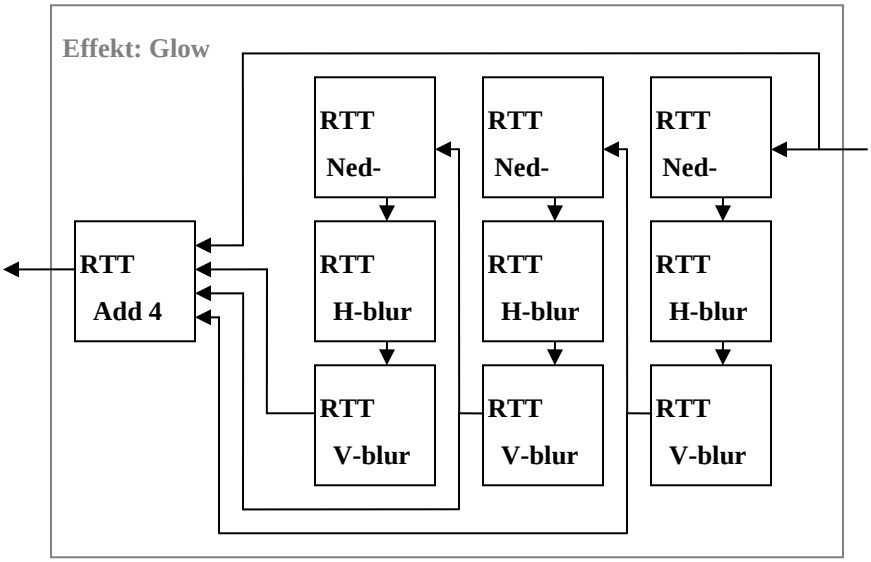
Figur 7 Noise effekten består av endast ett RTT objekt och en förgenererad noise textur



Figur 8 Före och efter-bild vid applicerande av noise-effekt.

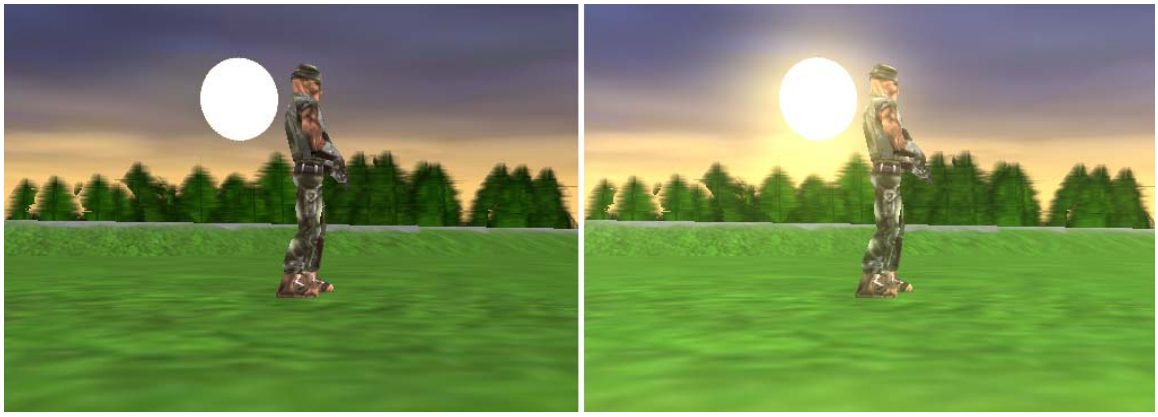
4.3.3 HDRGlow

Glow används för att simulera blödningseffekter runt starka ljuskällor p.g.a. atmosfärisk spridning, linsorenheter och mättnadsspridningar i sensorer.



Figur 9 Glow effekten består av tio RTT object och har referenser bakåt i kedjan. Lägg märke till återkopplingen till originalbilden.

Glow-effekten är lite mer komplex än Blur och Noise, den består av tio RTT-objekt och har referenser till RTT-objekt bakåt i kedjan. Glow-effekten konstrueras genom att omväxlande skala ner och göra bilden mer oskarp. Sedan avslutar man med att addera ihop de tre nedskalade och oskarpa bilderna med originalbilden. När de fyra bilderna adderas ihop har man möjlighet att ange en viktfaktor som anger hur mycket respektive delresultat ska påverka originalbilden (parameter ScalePass1..3). Man kan också välja hur mycket oskärpa varje Blur-steg ska tillföra (parameter BlurPass1..3). Figur 10 visar hur Glow-effekten påverkar originalbilden.



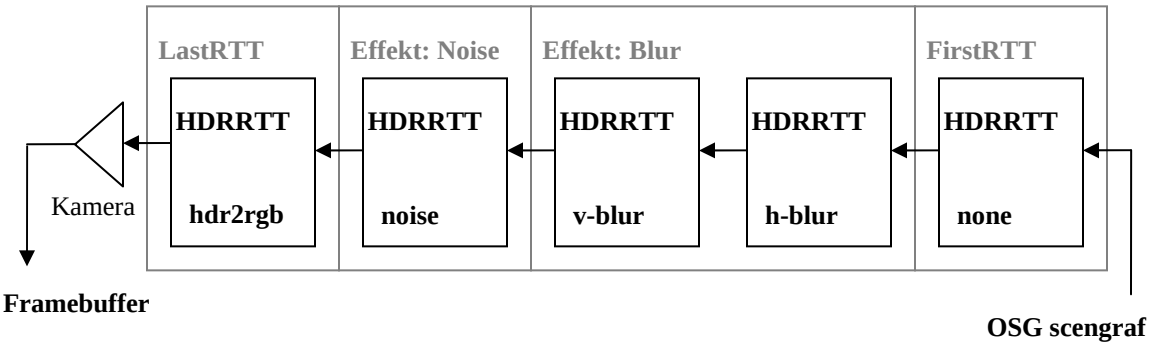
Figur 10 Före och efter-bild vid applicerande av glow-effekt. Solen har här fått en lysande aura runt sig som sprids till omkringliggande pixlar av bilden p.g.a. dess starka intensitet.

4.4 HDRRenderer

Modulen HDRRenderer använder existerande osgViewer::View-objekt och transformerar dessa till HDR-renderande vyer. Detta gör den genom att byta ut vyns topp-scen-nod och kamera mot en ny scengraf-nod som utför HDR-rendering av vyns föregående scen-nod. Hur man gör om en vanlig view till en HDR-renderande vy beskrivs i kapitel 7.

En HDR renderare har alltid minst två RTT objekt. FirstRTT och LastRTT. FirstRTT kopplas till scengrafen när man gör om en vanlig vy till att bli HDR-vy. FirstRTT har ingen shader men LastRTT har en shader som hanterar exponeringen och gör om HDR-grafiken till användbara RGB värden för att visas på en datorskärm. De RTT-objekt som finns mellan FirstRTT och LastRTT kan ändras dynamiskt beroende på vilka effekter som är aktiva och vilken renderingsprioritet de har. Se Figur 11.

HDRRenderer hanterar också beräkning av autoexponering samt rendering av legender till färgöversättningen. Mer om det i kapitel 5.



Figur 11 Exempel på en kedja av RTT objekt för en HDRRenderer där en Noise och en Blur-effekt har aktiverats

4.5 Parameterhantering

Parameterhanteringen i HDR-ramverket är konstruerad för att få en transparent koppling mellan attribut i NetScene, effektparametrar i EWSim och de uniforms som används av effektens pixel shaders. Uppdaterar man ett attribut i NetScene så uppdateras alltså motsvarande parameter i effekten och motsvarande uniform i shadern. En parameter behöver inte användas hela vägen från attribut till uniform. Den kan t ex användas för att endast skicka information från post-effekten till shadern eller för att skicka information från NetScene till post-effekten. Parameterhanteringen är generiskt skriven och man behöver inte lägga till någon kod för att hantera nya parametrar. Man registrerar bara parametern när post-effekten skapas, samt lägger till parametern i scenariomodellen om man vill komma åt den i NetScene. Det finns också en koppling till input-systemet som gör att man kan ändra värden på parametrarna ifrån drop-down menyn för ett scenarioobjekt. HDR-ramverket har två event-hanterare, där en är specialskriven för HDR-renderaren och den andra är generisk för alla post-effekter.

En parameter tillhör alltid en parametergrupp (HDRRTTParameters), där en parametergrupp kan innehålla en eller flera parametrar. Grupperna används för att gruppera alla parametrar som är kopplade till en viss RTT (eller shader). Om man av någon anledning skulle vilja traversera alla parametrar som tillhör en HDRRenderer kan man göra det på följande sätt:

För en HDRRenderer

  För varje HDRPostEffekt

    För varje RTT i effekten (dvs parametergrupper)

      För varje parameter i parametergruppen

Det finns åtkomstfunktioner för att sätta och hämta parametrar på alla nivåer.

```
HDRRenderer::setParameter  (effectName, groupName, paramName, value);
HDRPostEffect::setParameter(            groupName, paramName, value);
HDRRTTParameters::set      (                       paramName, value);
```

5 Exponering

För att kunna presentera en HDR-bild på skärmen så måste bilden transformeras för att minska dynamiken till 8-bitar per färgkanal, vilket är vad en skärm klarar att visa. När det sista RTT-objektet (LastRTT) renderas används en pixel-shader som implementerar en exponeringsfunktion vilken transformerar flyttalsvärdena i HDR-bilden till 8-bitars färgvärden för att visas på skärmen.

5.1 Exponeringsfunktioner

Försök har gjorts för att utveckla lämpliga exponeringsfunktioner som visualiserar den dynamik som telekrigsscenarier ger i form av IR-spektra med höga och låga intensiteter. Det finns för tillfället två linjära exponeringsfunktioner implementerade som täcker större delen av våra behov. Det har även gjorts försök med att implementera olika typer av olinjära funktioner men dessa har inte gett något bra resultat.

LINEAR Linjär exponering sker genom att definiera ett min- och ett max- värde i HDR-rymden. Värden mellan dessa gränser interpoleras sedan linjärt mellan värdena 0 och 255. Om ett värde är utanför gränserna så klipps det till att stanna på gränsvärdet. Den här algoritmen lämpar sig när färgvärdena är ganska jämnt fördelade i bilden.

LINEAR2 Den andra varianten av linjär exponering sker genom att först beräkna medelintensiteten i bilden. Ett värde linjärinterpoleras sedan mellan (min) och (min+(mean-min)*2). Den här algoritmen lämpar sig när bilden innehåller ”hot spots”, dvs små områden i bilden som är väldigt ljusa. Dessa punkter blir då överexponerade men den generella ljusnivån i bilden behålls. Beräkningen av min, max och medelvärdet kan ske genom automatisk analys av HDR-bilden eller låsas av användaren. I låst läge kan användaren själv, via användargränssnittet i EWSim, ändra på värdena för att ändra på exponeringen.

5.2 Standardexponering

Alla spektrum har hårdkodade standardexponeringsinställningar (min, mean, max). Standardinställningar för exponering till respektive spektrum är lagrade i klassen HDRExposure och sätts i konstrueraren för OSGScene-klassen i EWSim.

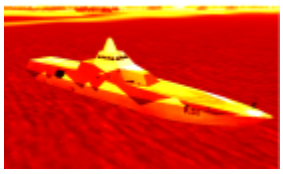
```
HDRSpectrumExposure::getInstance()->setupSpectrumExposure(
      "IRSW", 200.0, 400.0, 300.0,
      HDRExposure::LUMINANCE,
      HDRExposure::LINEAR,
      HDRExposure::GRAYSCALE,
      HDRExposure::AUTO);
```

5.3 Färgöversättning

Förutom exponeringsfunktionen så finns även möjligheten att förändra med vilka färger som HDR-bilden presenteras på skärmen. Tre algoritmer finns implementerade; NoColorMap, GrayScale och Iron.

NoColorMap förändrar inte färgerna från HDR-bilden annat än att exponeringen förändras.

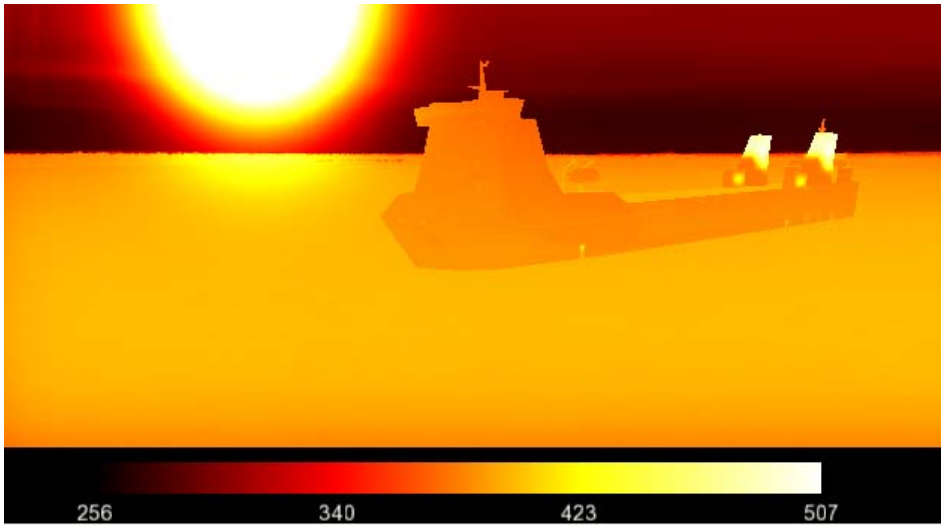
GrayScale använder sig av gråskalesummering av RGB-värdena i HDR-bilden för att skapa en gråskalebild som representerar ljusintensiteten så som den upplevs av det mänskliga ögat.



Iron ritar ut ljusintensiteten som en skala mellan svart, röd, gul till vit, där svart är låg intensitet och vit är hög intensitet.

6 Multipla färgspektra

En stor möjlighet med rendering i HDR-format är att kunna rendera bilder i andra spektra än det för människan visuella spektrumet. EWSim har sedan tidigare stöd för att på ett halvt fungerande sätt rendera i IR-spektrum. Detta har gjorts genom att använda ett alternativt textur-set för objekt. Problemet har varit att man i IR-spektrat ofta jobbar med väldigt varierande strålningsintensitet vilket enbart har fuskats till med tidigare 8-bitars färgdjup på texturer. Nu när färgdjupsproblematiken är löst så öppnar sig möjligheten till att på ett enkelt sätt växla till flera olika färgspektrum. EWSim kan nu hantera godtyckligt många alternativa textur-set i stället för bara ett IR-set.



Figur 12 IR-texturer med högre dynamik som visualiseras med Iron-färgsättning.

6.1 Spektrum med alternativa texturer

Huvudfunktionen för växling av alternativa texturuppsättningar på scenen sker av klassen EWSpectrumTextureHandler. För att lägga till nya alternativa spektra som skall kunna användas så används metoden addAlternativeTextureSet(). Denna metod anropas i konstruktorn till OSGScene vid uppstart. Namnet på de spektra som skall kunna hanteras anges som argument till metoden.

Idag finns det hantering av ett ”IRSW” och ett ”IRLW” spektra vilket i OSGScene:s konstruktor har lagts till genom följande anrop:

```
m_spectrumTextureHandler.addAlternativeTextureSet("IRSW");
m_spectrumTextureHandler.addAlternativeTextureSet("IRLW");
```

Dessa anrop säger att objekt som finns i scengrafen skall kunna byta till texturer som har ett IRSW eller IRLW-prefix. Detta innebär att om t ex en växling till spektrat som är namngivet ”IRSW” sker så kommer alla objekt i OSGs scengraf att ladda in nya texturer. För att detta skall fungera så krävs det att texturerna för ett objekt som finns i data-katalogen också finns i en uppsättning som har ”IRSW_” som filnamnsprefix. Förutom de spektra som anges med addAlternativeTextureSet() så finns alltid

eller ”NORMAL” spektra, vilket använder de originaltexturer som angivits för objekt. Byte till ett alternativt spektra sker genom ett anrop till switchToAlternativeSpectrum() i EWSpectrumTextureHandler, vilken tar namnet på spektrat som argument.

6.2 HDRSpectrumSwitcher

I och med införandet av HDR-ramverket så sker hela hanteringen av växling mellan olika spektrum i klassen HDRSpectrumSwitcher. Varje HDR-renderare har en HDRSpectrumSwitcher och man måste ha en HDR-renderare på sin vy för att kunna växla spektrum. HDRSpectrumSwitcher ser till så att EWSpectrumTextureHandler byter ut texturerna, men också så att rätt dimma, sol, exponering och andra inställningar används för rätt spektrum.

6.3 Terränggenerering för alternativa spektra

För att kunna simulera värmekameror i både lång- och kortvågiga IR-spektra (IRLW och IRSW), så har man vid terränggenereringen låtit skapa två extra uppsättningar marktexturer vid sidan av de normala. Texturerna är namngivna med prefix som matchar det spektrumnamn som används vid spektrumbyte. I dagsläget har man använt sig av ett separat program för terränggenereringen, vilket tar ett begränsat antal grundtexturer som input, t.ex. gräs, fält, vatten, etc. För att skapa nya spektra har således motsvarande bilder skapats för varje nytt spektrum. Inputbilderna till detta program har endast haft 8 bitar per färgkanal. För IRLW och IRSW har det varit gråskalebilder (0-255), som programmatiskt skalats om till önskade temperaturintervall. De slutliga texturerna är i HDR-formatet OpenEXR, vilket kan läsas av EWSim.

Utöver terrängen, har även övriga objekt som renderas i EWSim fått lämpliga texturer för varje nytt spektrum. Någonting som underlättar detta arbete, är att det i Adobe Photoshop finns stöd för att direkt redigera och spara bilder i formatet OpenEXR

7 Att använda och underhålla HDR ramverket

I det här kapitlet beskrivs hur man gör om en vanlig OSG-vy till att rendera HDR-grafik. Det förklaras också hur man lägger till nya post-effekter samt hur man kommer åt HDR-bilddata för bildanalys.

7.1 Uppsättning av vy för HDR-rendering

Att transformera en befintlig OSG-vy till att rendera HDR-grafik är enkelt och kräver inte så mycket extra kod. För det första så måste klassen som ansvarar för vyn ärva av klassen HDRRendererParentInterface. Det gör att klassen som ansvarar för vyn får tillgång till funktioner för att kolla om det finns en HDR-renderare, hur stor den är och möjlighet att hämta en pekare till denna.

```
class osgSomeSensor : public osgSensor, public HDRRendererParentInterface
-------------------------------------------------------------------------
HDRRenderer*   m_pHDRRenderer;
float                    m_maxViewSize;
-------------------------------------------------------------------------
m_pHDRRenderer = getHDRRenderer();
m_maxViewSize =  getHDRRendererMaxViewSize();

if(m_pHDRRenderer)
{
   m_pHDRRenderer->setupHDRRendering(pView, m_maxViewSize);
}
```

Argumentet pView är den vy som ska göras om till HDR-vy och argumentet m_maxViewSize anger vad vyn maximalt kan använda för bredd och höjd. Detta värde måste vara en tvåpotens, d.v.s. 64, 128, 256, 512 etc. Minne kommer att allokeras som om hela vyn användes så det är bra att hålla detta värde så lågt som möjligt.

Användaren måste vara aktsam och inte förutsätta att en vys kamera definierar den kamera som visar scenen efter att HDR har aktiverats. Vyns kamera efter aktivering av HDR kommer att vara en specialkamera som är till för att transformera HDR-bilden till en bild som kan visas av en datorskärm.

7.2 Ny post-effekt

För att skapa en ny post-effekt och integrera den i HDR ramverket så är det bästa tipset att helt enkelt titta på en redan befintlig post-effekt och kopiera all kod som hör till den. Ett enkelt sätt att hitta all kod som hör till en effekt är att t ex söka på ”HDRBlur” i källkoden till EWSim. Nedan följer en lista på filer som måste ändras och läggas till när man skapar en ny effekt. HDRBlur är använt som exempel. Till att börja med räcker det med att göra steg 1-5. Man kan då lägga till sin nya effekt i ett

scenario och testköra lokalt i EWSim. Vill man däremot kunna köra simuleringen distribuerat över HLA måste man även göra steg 6-15.

Ny post-effekt i HDR ramverket

För att lägga till en post-effekt i HDR ramverket behöver man endast ändra och lägga till filerna 1-4. Hur man implementerar de nya filerna beskrivs i kapitel 7.3.

1. \EWMSim\HDRBlur.h Ny
2. \EWMSim\HDRBlur.cpp Ny
3. \EWMabLib\EntityTypesDB.h Ändra
4. \EWMabLib\EntityTypesDB.cpp Ändra

Uppdatera scenario modellen

För att man ska kunna lägga till post-effekten i ett scenario måste scenariomodellen uppdateras. Därefter kommer man åt effekten i NetScene.

5. \netscene_mount3\DtScenarioModel.xml Ändra

Skapa HLA objekt för post-effekten

För att skapa ett HLA objekt för den nya effekten måste man modifiera och lägga till följande filer.

6. \EWFOMExtKit\include\se\foi\hla\ewfom\hlaobject\HDRBlur.h Ny
7. \EWFOMExtKit\include\se\foi\hla\ewfom\EWFOMClasses.h Ändra
8. \EWFOMExtKit\include\se\foi\hla\ewfom\EWFOMEnums.h Ändra
9. \EWFOMExtKit\src\EWFOMClasses.cpp Ändra
10. \EWFOMExtKit\src\EWFOMEnums.cpp Ändra
11. \EWFOMExtKit\src\HDRBlur.cpp Ny
12. \EWLKit\src\EWLFedApp.cpp Ändra
13. \EWMSim\OSGScene.cpp Ändra

Utöver detta krävs också att man uppdaterar FOM-filen samt jar-filer för NetScene.

14. MOSART-reference-FOM.xml Ändra
15. jar-filer till NetScene Ändra

7.3 Implementera den nya post-effekten

Nedan följer en lista på funktioner som måste implementeras för att en effekt ska kompilera och fungera. Funktionerna 4-6 används bara då man kör med HLA så de kan lämnas tomma om man testar en effekt lokalt utan HLA. Funktionerna 1-6 är rent virtuella och måste därför implementeras, men updateCallback kan utelämnas. updateCallback är en callback som anropas varje frame innan effekten renderas. Det kan vara användbart om man har någon slags dynamisk uppdatering av effekten, Noise-effekten byter t ex seed varje frame. Man kan också behöva lägga till nya shaders om effekten kräver det.

Skapa effekt
1. createParameters(...)
2. createEffect(...)

Parameterhantering
3. parseAttributeList(...)
4. setupFromHLA(...)
5. updateFromHLA(...)
6. updateToHLA(...)

Uppdatering
7. updateCallback(...)

Effekten definieras av funktionerna createParameters och createEffect. De beskrivs här mer i detalj.

createParameters(...)
Parameterhanteringen är konstruerad för att få en koppling mellan attribut i NetScene, effektparametrar i EWSim och de uniforms som används av effektens shaders. Uppdaterar man ett attribut i NetScene så uppdateras alltså motsvarande parameter i effekten och motsvarande uniform i shadern.

I createParameters skapar man grupper av parametrar, se klassen HDRRTTParameters. En grupp kan bestå av en eller flera parametrar, där varje parameter har ett namn som måste vara exakt samma som namnet som attributet i scenariomodellen samt i de shaders som använder motsvarande uniform. Om man vill kunna styra parametern från input-systemet i EWSim så anropar man setIOControll och specificerar, min- maxvärden, förklarande text och vilka snabbtangenter som ska användas mm.

createEffect()
Den här funktionen har till uppgift att skapa de RTT-objekt som bygger upp effekten samt att koppla ihop dessa på önskat sätt. En effekt kan bestå av endast ett RTT-objekt eller flera RTT-objekt. RTT-objekten kan kopplas i en rak kedja eller ha mer komplicerade former med referenser till tidigare RTT-objekt. Man skapar och lägger till ett RTT-objekt med funktionen addNextRTTObject.

```
addNextRTTObject("original",1.0,m_pBlurParam,"shader_hblur","", . . .);
addNextRTTObject("hblur",   1.0,m_pBlurParam,"shader_vblur","", . . .);
```

Första argumentet är RTT-objektets referensnamn, andra argumentet anger skalningen på texturstorleken i förhållande till max-storleken på effekten. Max-storleken är alltid samma som för föräldern, dvs HDR-renderaren. Alla effekter börjar och slutar med max-storlek, dvs resultattexturen är lika stor som originaltexturen. Tredje argumentet är en pekare till en parametergrupp. Dess parametrar kopplas som Uniforms för shadern. Fjärde argumentet är namnet på shadern, utan shader saknar RTT-objektet i princip funktion. Argument fem till åtta är referenser till andra RTT-objekt eller fristående texturer. Alla referenser är default NULL och om inget anges som första referens kommer den automatiskt peka på RTT objektets egen textur. De fyra referenserna kan ses som ”indata” till RTT-objektet. Referenserna, som i slutändan blir referenser till texturer, sätts som texturenhet 0-3 för shadern och kan användas för bearbetning. Shadern används för att rendera en ny textur och resultatet landar i nästa RTT objekt. Det förklarar förskjutningen i referensnamnen, andra RTT objektet heter ”hblur” trots att det första har shadern ”shader_hblur”. Resultatet från ”shader_vblur” resulterar i

”original” i nästa effekt, eller i LastRTT hos HDR renderaren. Studera figurerna i kapitel 3.3 och titta på koden för befintliga effekter för att förstå sammanhanget.

När man utvecklar och testar nya post-effekter så finns användbara debug-funktioner inbyggda i ramverket. De är beskrivna i kapitel 8.

7.4 Bildanalys av HDR-data

Rendering i HDR-format öppnar upp möjligheten att implementera bildalgoritmer som använder sig av den höga dynamiken för bildanalys. Den flyttalsbaserade HDR-renderade bilden kan hämtas ut för efteranalys genom att använda setPostDrawCallbackAfterPostEffects() eller setPostDrawCallbackBeforePostEffects()-metoderna på ett HDRRenderer-objekt. Metoderna tar ett callback-objekt som argument, vilket anropas varje gång HDR-bilden uppdaterats, och som då får tillgång till den renderade bilden. Skillnaden mellan de två metoderna är att den första ger den renderade bilden så som den ser ut efter att alla posteffekter har applicerats, medan den andra metoden ger bilden så som den ser ut före posteffekterna lagts på.

Callbacken sätts med följande kod:

```
pMyPostDrawCallback = new MyDrawCallback();
pHDRRenderer->setPostDrawCallbackAfterPostEffects(pMyPostDrawCallback);
```

Nedan är ett exempel på implementation av ett en post draw callback som konverterar och sparar HDR-bilden i ett 8-bitars råformat till disk varje frame:

```
class MyDrawCallback : public HDRRenderer::DrawCallback
{
public:
   virtual void operator () (const osg::Image *pImage) const
   {
      FILE *captureFile = fopen("Capture.raw", "wb");

      for (int row = pImage->t() - 1; row >= 0; row--)
      {
         float* data = (float*)pImage->data(0, row);
         for (int column=0; column < pImage->s(); column++)
         {
            unsigned char byteData[3];
            byteData[0] = (*data) < 1.0 ? (*data * 255) : 255;
            data++;
```

```
            byteData[1] = (*data) < 1.0 ? (*data * 255) : 255;
            data++;
            byteData[2] = (*data) < 1.0 ? (*data * 255) : 255;
            data++;
            data++;

            fwrite(byteData, 1, 3, captureFile);
         }
      }
      fclose(captureFile);
   }
};
```

8 Debug-funktioner

Två debug-funktioner har implementerats för att underlätta utvecklingsarbetet av nya post-effekter. Debug-funktionerna fungerar bara på huvudfönstret, så placera effekten som ska testas på en renderare till huvudfönstret. Man finner funktionerna i EWSim under View-menyn och alternativet Debug GUI.

8.1 Capture HDR pipeline

Klickar man på Capture HDR pipeline så tas en skärmdump av varje RTT-textur i HDR pipelinen för huvudfönstret och sparas som png-filer i katalogen src\EWMSim. Pipelinen börjar alltid med FirstRTT, sedan kommer alla aktiverade post-effekter ordnade efter renderingsprioritet och sist kommer LastRTT. Man kan då se resultatet av alla shaders samt storleken på alla interna RTT-texturer. Det är en bra funktion för att se var i pipelinen det har gått fel om man inte får ut ett förväntat resultat.

8.2 Reload Shaders

När man finjusterar nya shaders är det behändigt att inte behöva starta om EWSim varje gång man uppdaterat en siffra eller en kodrad. Därför finns funktionen Reload Shaders som laddar om alla shaders i HDR-pipelinen för huvudvyn. Det är rekommenderat att använda en utvecklingsmiljö för shaders parallellt med EWSim. Man kan då få syntax-kontroll och se resultatet av enskilda shaders i utvecklingsmiljön, spara filen, sedan ladda om alla shaders i EWSim med Reload Shaders och se slutresultatet av hela pipelinen.

9 HDR i NetScene

Detta kapitel beskriver hur man anpassar ett scenario i NetScene för att köras med HDR-grafik. Det beskrivs hur man lägger till HDR renderare på sikten och huvudfönstret och hur man kopplar post-effekter till dessa. Det beskrivs också hur man anpassar sol och dimma för flera spektrum. De spektrum som finns är: NORMAL, IRSW och IRLW.

Väljer man att använda HDR-renderare får man tillgång till följande funktionalitet:

- HDR-grafik
- Möjlighet att använda post-effekter
- Manuell och automatisk exponering
- Färgöversättning, gråskala eller iron
- Spektrumhantering, växla mellan olika spektrum.

De spektrum som finns är: NORMAL, IRSW och IRLW.

9.1 Lägg till HDR renderare

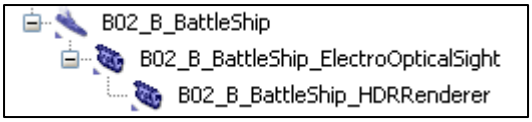
För att huvudfönstret eller ett sikte ska renderas med HDR-grafik måste man lägga till HDR-renderare. Detta måste göras för respektive sikte som man vill ska rendera HDR-grafik. Objektet för en HDR-renderare i NetScene heter HDRRenderer och finns med i drop-down listan för objekt som kan skapas. HDR-renderaren hanteras olika beroende på om den är avsedd för huvudfönstret eller ett sikte. Följande sektioner går därför igenom hur man anpassar respektive vy och därefter följer en genomgång av parametrar.

9.1.1 HDR rendering för sikten

De sikten som för tillfället är HDR anpassade och därmed kan hantera HDR-rendering är:

- ElectroOpticalSight
- MissileLauncher
- RemoteMissileLauncher

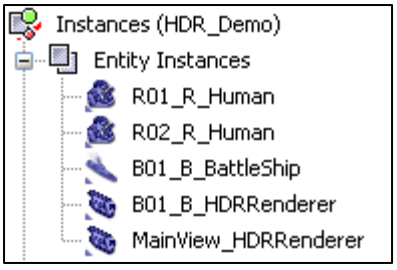
HDR renderaren ska placeras under respektive sikte så att siktet blir förälder till HDR-renderaren. Se exemplet för ett elektrooptiskt sikte på ett Battleship i Figur 13. HDR-renderare för sikten ska vara remote.



Figur 13 HDR renderare på elektrooptiskt sikte

9.1.2 HDR rendering för huvudfönstret

Om man vill att huvudfönstret ska rendera HDR-grafik så ska man lägga till en HDR-renderare i roten på scenariot, d v s direkt under Entity Instances. Det finns två sätt att namnge HDR-renderaren för huvudfönstret beroende på om man vill att den ska vara specifik för en viss federat eller om den ska gälla allmänt för alla federater. Den allmänna HDR-renderaren ska namnges med prefixet ”MainView”



Figur 14 HDR renderare på huvudfönstret

och de federatspecifika med respektive federatnamn som prefix, t ex ”R01” eller ”B01”. Federatspecifika renderare har högre prioritet än den allmänna renderaren. Kör man lokalt, d v s utan HLA, används alltid den allmänna renderaren eftersom man då inte kör som någon specifik federat. HDR-renderare för huvudfönstret ska inte vara remote.

Exemplet i Figur 14 visar ett scenario där federaterna R01 och R02 kommer att använda den allmänna renderaren MainView_HDRRenderer och B01 kommer att använda den federatspecifika renderaren B01_B_HDRRenderer. Man kan tänka sig följande fyra fall när man väljer HDR-renderare för huvudfönstret:

1. Inga federater ska ha HDR-grafik
   Lägg inte till några renderare

2. Alla federater ska ha HDR-grafik med samma inställningar
   Skapa en allmän renderare (MainView_HDRRenderer)

3. Alla federater ska ha HDR-grafik, några med speciella inställningar
   Skapa en allmän renderare och federatspecifika renderare för de federater som ska ha speciella inställnignar

4. Några federater ska ha HDR-grafik, resten ska inte ha HDR-grafik
   Skapa federatspecifika renderare för de federater som ska ha HDR-grafik

Anledningen till att man kan vilja ha olika inställningar för renderaren på huvudfönstret kan t ex vara för att man har olika snabba datorer eller att man vill ha olika startspektrum eller posteffektkuppsättningar.

9.1.3 Parametrar

I NetScene kan man ställa in tre parametrar för HDR-renderaren som påverkar dess egenskaper. Det är dock valet av posteffekter och dess inställningar som man kopplar till renderaren som utgör den största skillnaden mellan olika renderare. Mer om detta i nästa sektion. Följande parametrar finns för HDR-renderaren:

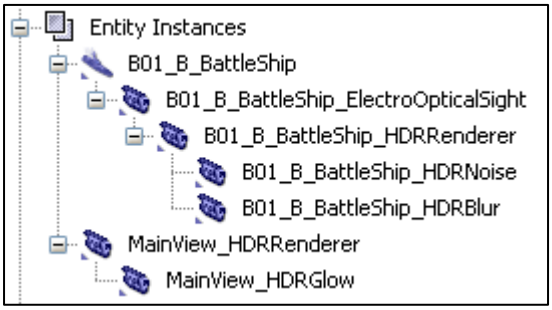
1. **MaxViewSize** – Ställer in den maximala storleken för fönstret eller siktet som använder renderaren. Storleken påverkar prestanda och minnesåtgång så håll den så liten som möjligt. Storleken måste vara en tvåpotens. För huvudfönstret är 1024 en bra storlek.
2. **SpectrumList** – En textsträng med en lista på de spektrum som renderaren ska ha tillgång till.
3. **StartupSpectrum** – Det spektrum som ska vara aktivt när fönstret visas första gången.

MaxViewSize	1024
SpectrumList	NORMAL IRSW IRLW ...
StartupSpectrum	NORMAL ...

Figur 15 Parametrar för HDRRenderer i NetScene

9.2 Lägg till post-effekter

I det här kapitlet beskrivs hur man lägger till post-effekter till HDR-renderare samt hur man konfigurerar dem. Hur post-effekterna fungerar och vad de är till för går att läsa i kapitel 4.3. För tillfället finns tre post-effekter implementerade (Blur, Glow och Noise) och de heter i NetScene HDRBlur, HDRGlow och HDRNoise och hittas i drop-down listan där man skapar objekt.



Figur 16 Post-effekter på elektrooptiskt sikte och huvudfönstret

För att lägga till en post-effekt till en HDR-vy placerar man post-effekten under HDR-renderaren som hör till den aktuella vyn så att HDR-renderaren blir förälder till post-effekten. Se exemplet i Figur 16 där huvudfönstret och ett elektrooptiskt sikte på ett Battleship har post-effekter.

9.2.1 Parametrar

I NetScene kan man ställa in parametrar för post-effekterna som påverkar deras effekt på originalbilden. Antalet parametrar och vad de betyder skiljer sig mellan effekterna men samtliga effekter har två generella parametrar:

1. **RenderPrio** - Om en renderare har mer än en post-effekt avgör RenderPrio vilken ordning de ska verka. Ett högre nummer ger högre prioritet och kommer därför att verka före övriga effekter.
2. **SpectrumList** – Textsträng med en lista på de spektrum som effekten ska vara påslagen för som standard. Denna lista ska vara en delmängd av den listan som beskriver spektrumen för HDR-renderaren. Alla effekter kan tvingas av eller på från input-systemet för renderarens alla spektrum, men om man växlar spektrum kommer bara de effekter som är default påslagna för det nya spektrumet aktiveras.

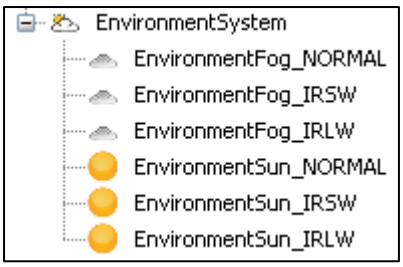
Effektspecifika parametrar beskrivs i kapitel 4.3 för respektive effekt.

9.2.2 Något om HLA

Samtliga HDR-objekt, effekter och renderare, är implementerade som HLA-objekt och distribuerar därmed sina parametrar över HLA. Eftersom den allmänna HDR-renderaren inte är remote och ägs av NetScene så kan man ändra alla parametrar på effekter som tillhör den allmänna renderaren från NetScene och få dem uppdaterade i EWSim över HLA. Detta är väldigt smidigt när man ska testa parametersättningar för effekter innan man väljer att spara dem i scenariot. Övriga HDR-objekt kan man inte uppdatera från NetScene men man kan däremot se de aktuella parameterinställningarna i NetScene även om man ändrar dem i EWSim.

9.3 Sol och dimma för flera spektrum

Sol och dimma fungerar ungefär som tidigare. Sol- och dimmaobjekten ska läggas i containern EnvironmentSystem och om man vill ha sol och/eller dimma för flera spektrum så ska man skapa ett objekt per spektrum. Namngivningen har ingen påverkan men kan vara bra för att hålla isär objekten. Vilket spektrum som respektive dimma och sol ska kopplas till specifieras i det nya attributet



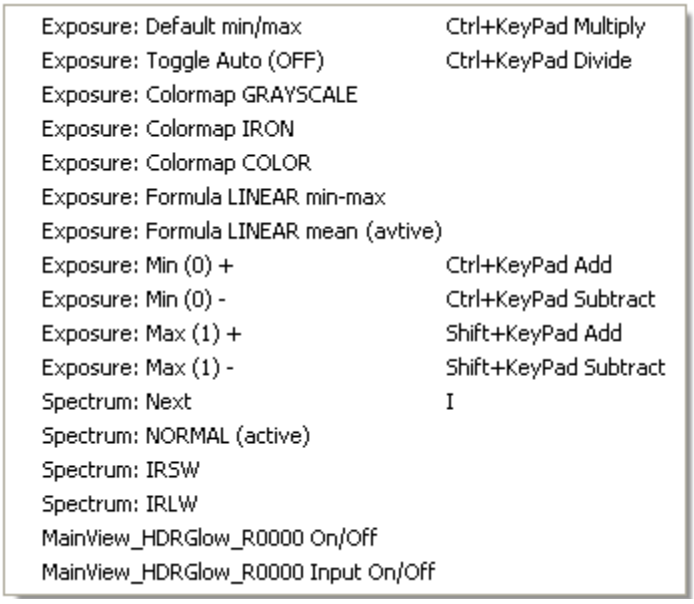
Figur 17 Exempel på sol och dimma för flera spektrum

”Spectrum”. Det är en sträng som anger namnet på spektrumet där dimman eller solen ska aktiveras. Solen visualiseras numera på himmlen och dess position beräknas genom att man i NetScene anger tid på dagen, datum på året och vilken latitud man befinner sig på. Ny funktionallitet är också att färg för sol och dimma kan distrubueras över HLA.

10 Input-systemet och objektlistan i EWSim

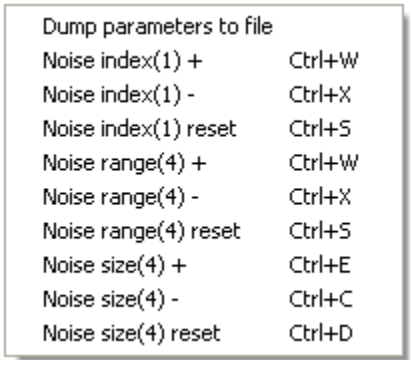
Här beskrivs de drop-down menyer som man kommer åt från objektlistan i EWSim när man högerklickar på en HDR-renderare eller en post-effekt.

För renderaren kommer man åt inställningar för exponering, byte av spektrum samt möjlighet att slå på eller av de post-effekter som är kopplade till den aktuella renderaren. Se Figur 18 för ett exempel på en renderare med en glow effekt. Menyn är till viss del dynamisk och visar aktuella inställningar så som aktiv färgöversättning (colormap), aktiv exponerings-formel (LINEAR mean), aktivt spektrum, om autoexponering är påslagen samt den aktuella exponeringen (Min och Max). Om man har post-effekter kopplade till renderaren så kan man aktivera och avaktivera dessa (On/Off) samt aktivera och avaktivera deras underliggande input möjligheter (Input On/Off). Om man aktiverar input för en effekt kommer dess inställningar läggas till i renderarens drop-down meny.



Figur 18 Exempel objektlistans drop-down meny för HDR renderaren

Man kan även komma till post-effektens inställningar genom att direkt högerklicka på effekten och ta fram dess drop-down meny, förutsatt att input för effekten är aktiverat. Anledningen till att man kan aktivera och avaktivera input för effekter är för att det inte ska uppstå krockar för snabbtangenterna samt för att minska storleken på drop-down menyn. Se Figur 19 för ett exempel på drop-down meny för en noise-effekt. Det översta alternativet ”Dump parameters to file” sparar ner alla aktuella parameterinställningar för effekten i en textfil. Det kan vara en bra funktion när man testar ut nya inställningar för en effekt. Det smidigaste sätter för att testa parametrar är dock att koppla effekten till en allmän renderare och styra parametrarna över HLA som beskrevs i kapitel 9.2.



Figur 19 Exempel på drop-down meny för en Noise effekt

11 Referenser

[HDR] 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 Press, 189–198.

[OSG] http://www.openscenegraph.org, 2009-02-16.

[OPENGL] http://www.opengl.org, 2009-02-16.

[RADIANCE] http://en.wikipedia.org/wiki/Radiance_(software)#HDR_image_format, 2009-02-16.

[OPENEXR] http://www.openexr.com, 2009-02-16.