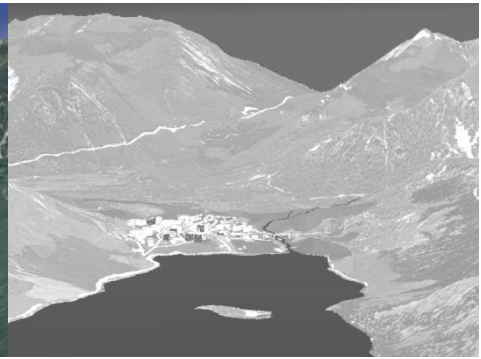
EMIT

Extensible Multi-spectral Image Generation Toolset



EMIT

An infrastructure for real-time infrared image generation

Paul Obermeier

MBDA MISSILE SYSTEMS

MBDA – Company Structure

AIRBUS 37.5 %

BAE SYSTEMS 37.5 %

LEONARDO 25 %

MBDA MISSILE SYSTEMS

- MBDA FRANCE — 100 %
  - Matra Électronique — 100 %
  - GDI SIMULATION — 100 %
- MBDA UK — 100 %
  - MBDA USA — 100 %
- MBDA ITALY — 100 %
- MBDA GERMANY — 100 %
  - Bayern-Chemie — 100 %
  - TDW — 100 %
- MBDA SPAIN — 100 %

MBDA MISSILE SYSTEMS

MBDA – Product Portfolio

Ground Based Air Defence

Air Dominance

Battlefield Engagement

Maritime Superiority

Components

20 years ago I gave my first Tcl presentation at the 3rd EuroTcl in Munich

2002: New image formats for the Img extension

Img Inside & Out

Paul Obermeier

obermeier@poSoft.de
paul.obermeier@lfk.eads.net

Third European Tcl/Tk User Meeting.
Munich, June 2002

2008: First version of EMIT

EMIT - **E**xtensible **M**ultispectral **I**mage Generation **T**oolset

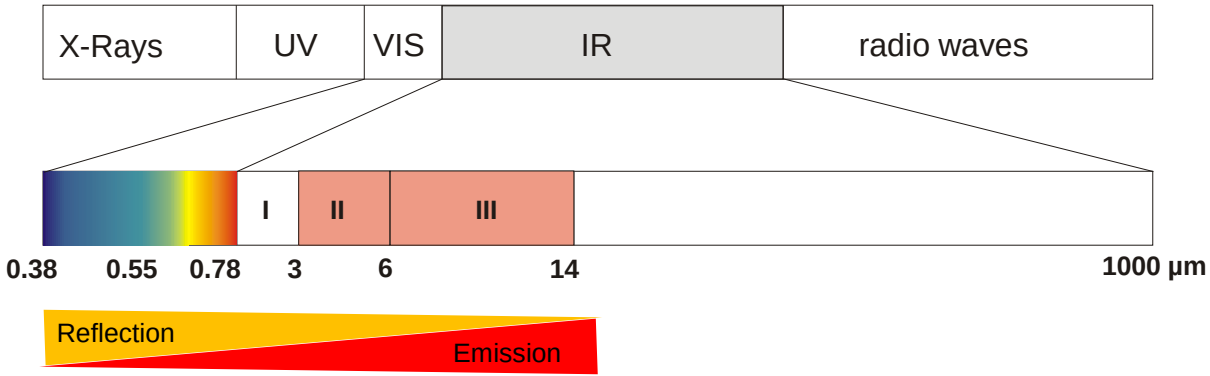
Using Tcl/Tk for simulation and visualization

Dipl.-Inf. Paul Obermeier

Software Architect for Simulation and 3D Computer Graphics
MBDA Germany

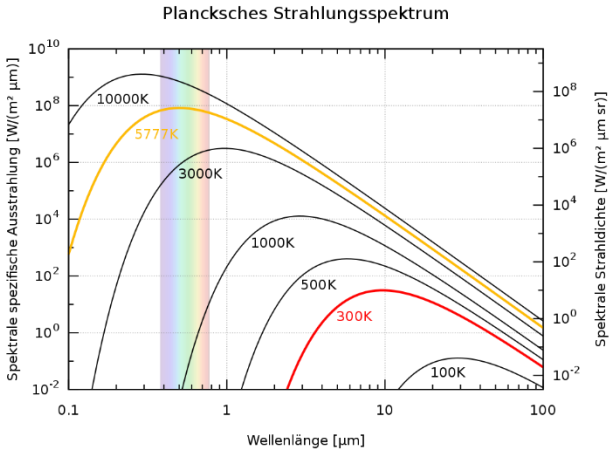
European Tcl/Tk User Meeting 2008, Strassbourg

Infrared: Spectral waveband adjacent to the visible waveband



The emission of IR radiation of solids is described by Planck's law

$$R(\lambda, T) = \frac{2hc_0^2}{\lambda^5\left(e^{hc_0/\lambda kT} - 1\right)} d\lambda$$



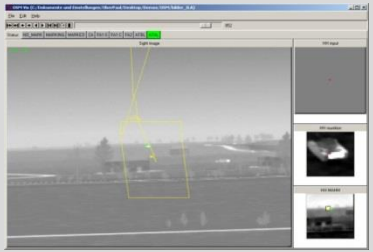
Reference: EMIT – An infrastructure for realtime infrared image generation

Visibility in the infrared spectrum is different from what we are used seeing with our eyes

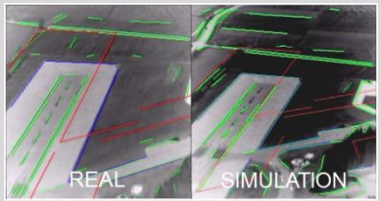
The Extensible Multispectral Image Generation Toolset (EMIT) is a modular software library developed at MBDA Germany for the generation of physics-based infrared images in realtime.

It is able to render infrared images in full 32-bit floating point precision using state-of-the-art computer graphics cards and advanced shader programs.

The core modules of the EMIT rendering engine are written in C++ and GLSL, but EMIT also makes heavy use of Tcl/Tk (including several extension packages) for development, maintenance and usage purposes.



Software-in-the-Loop



Hardware-in-the-Loop

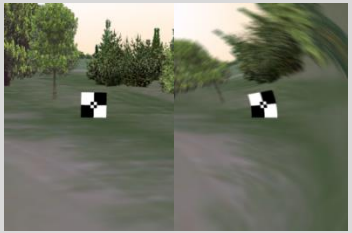
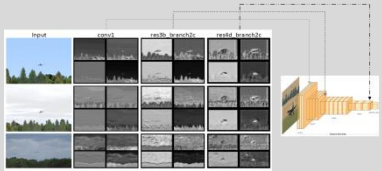


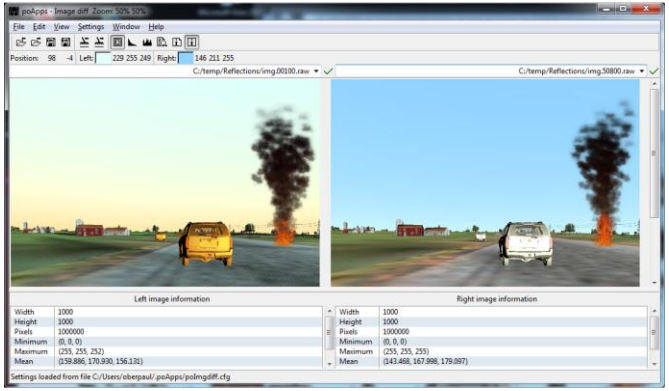
Image Processing

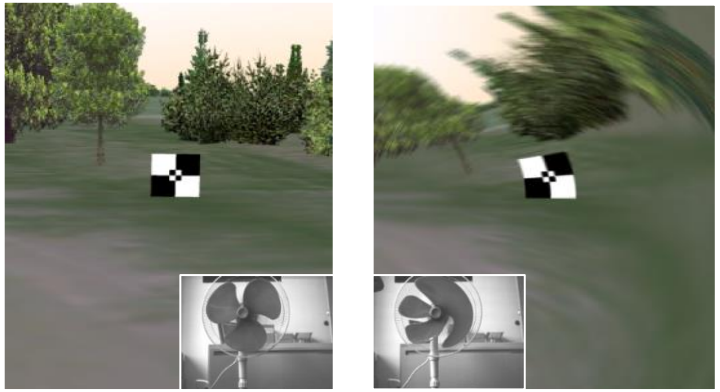


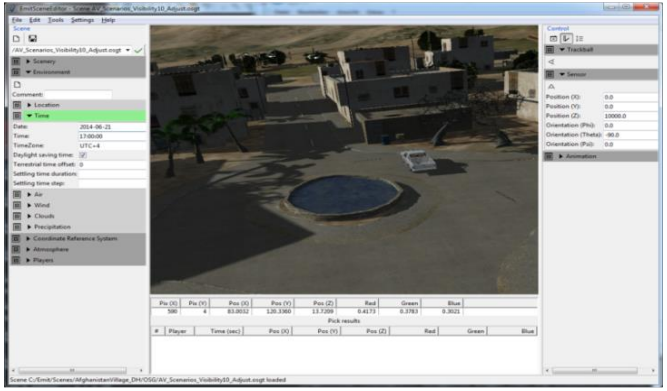
Machine Learning

EMIT Use Case - Software-in-the-Loop Simulation

- Verification simulation.
- Geotypic databases for development of image processing.
- Thread Enabled
  - Thread for image generation
  - Thread for terrain height queries at high rates
- Generation of seeker images with EMIT
  - Reflections and soft shadows
  - Rolling-shutter investigations

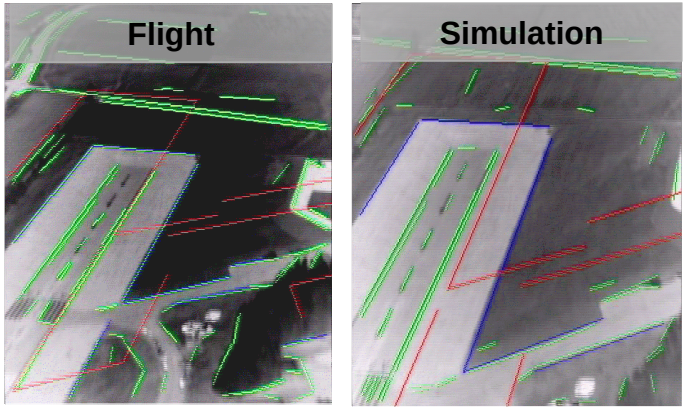


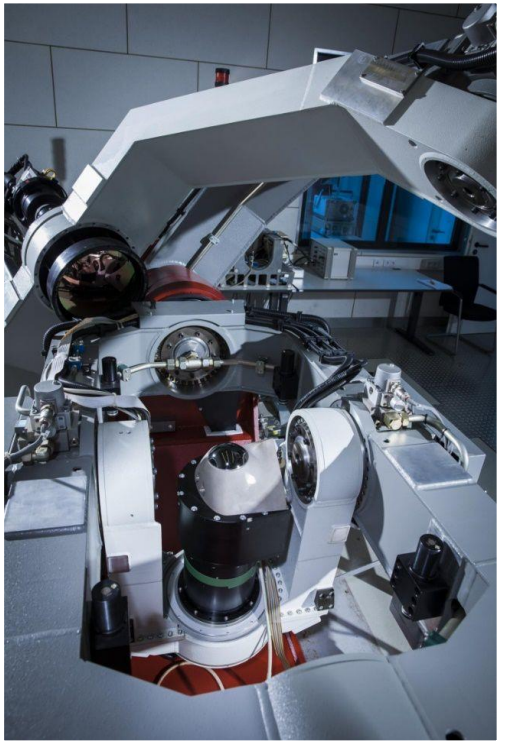




EMIT Use Case - Hardware-in-the-Loop Simulation

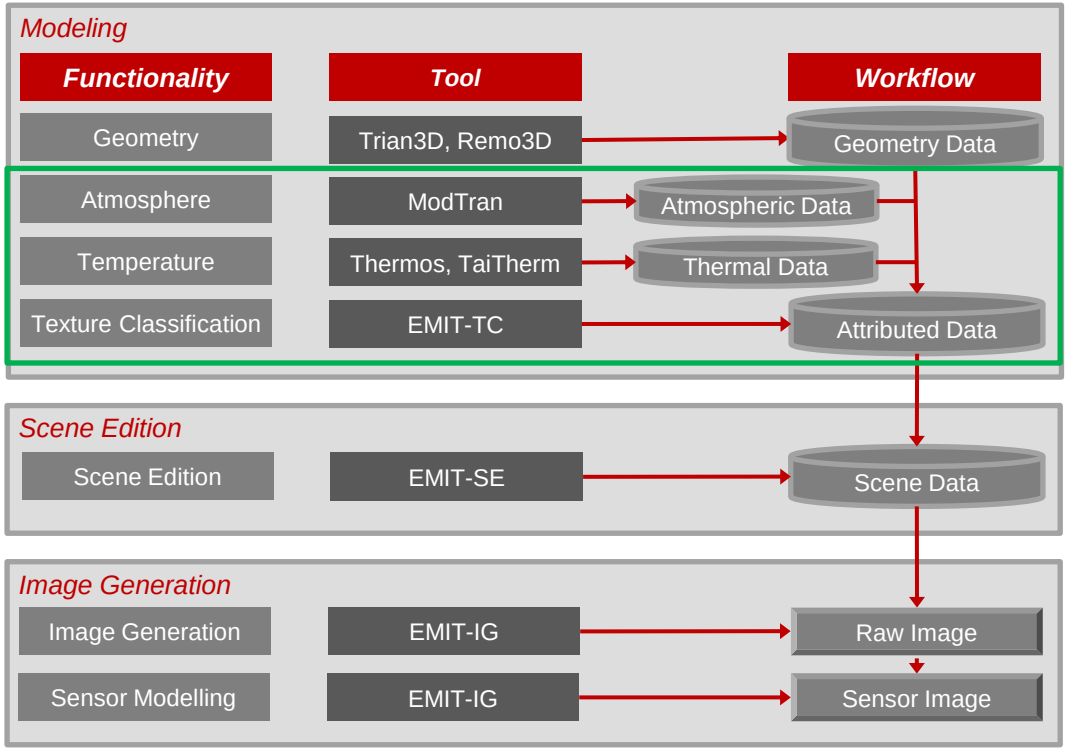
- Accredidated verification simulation
- Geospecific databases for flight campaigns
- 5-axis motion table
- SBIR Mirage Infrared Projector
- EMIT Infrared Image Generation

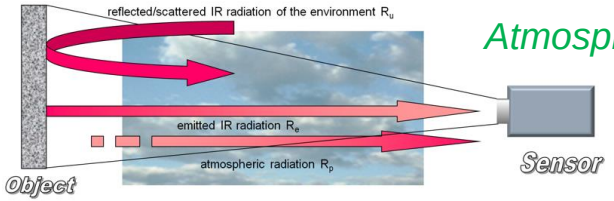




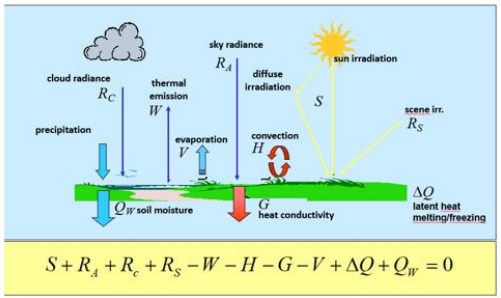
EMIT Overview – Workflow

Workflow for IR image generation needs some additional modeling steps





Atmosphere

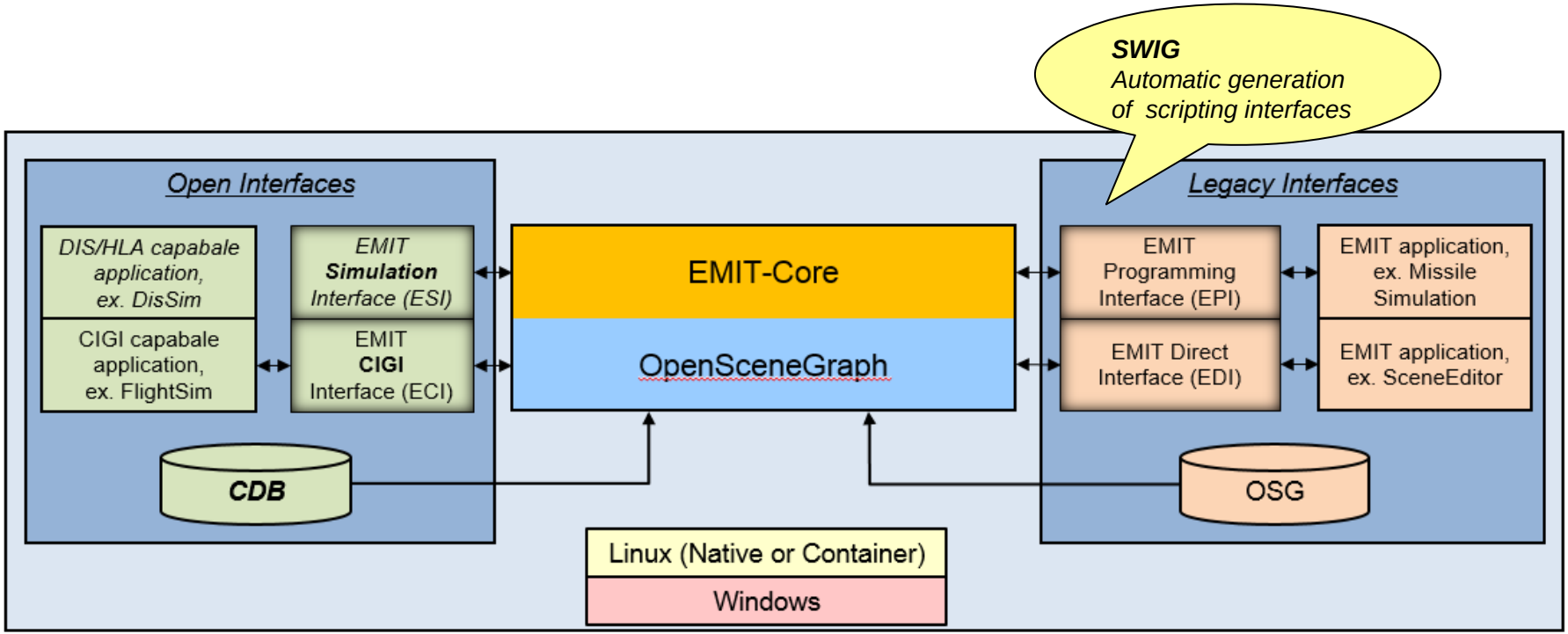


Heat Balance

Material Classification

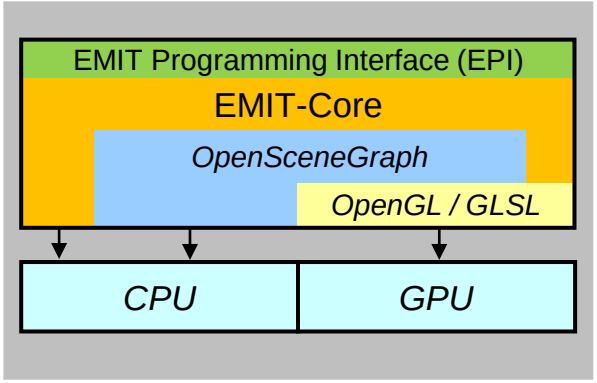
MBDA MISSILE SYSTEMS

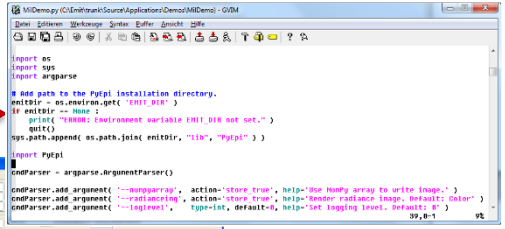
EMIT uses OpenSceneGraph and adds functionality for infrared specific calculations

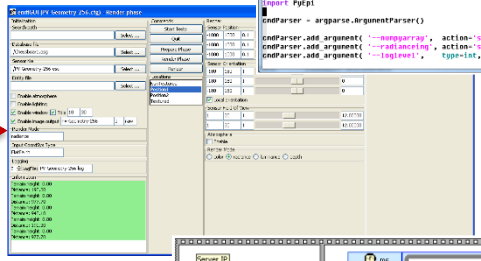


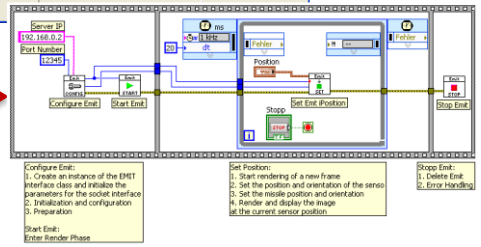
EPI offers a network-transparent interface in C, C++, Python and Tcl for application programming and connection to other simulation tools such as Matlab / Simulink, Labview or TensorFlow.

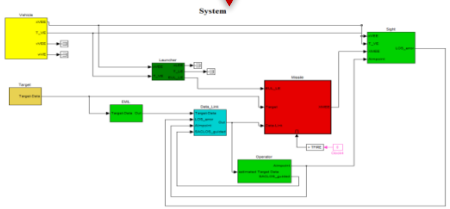
EPI-Python	Machine Learning
EPI-Tcl	Tests, Prototypes
EPI-C	Fortran, LabView
EPI-C++	SIL, HIL, SimuLink











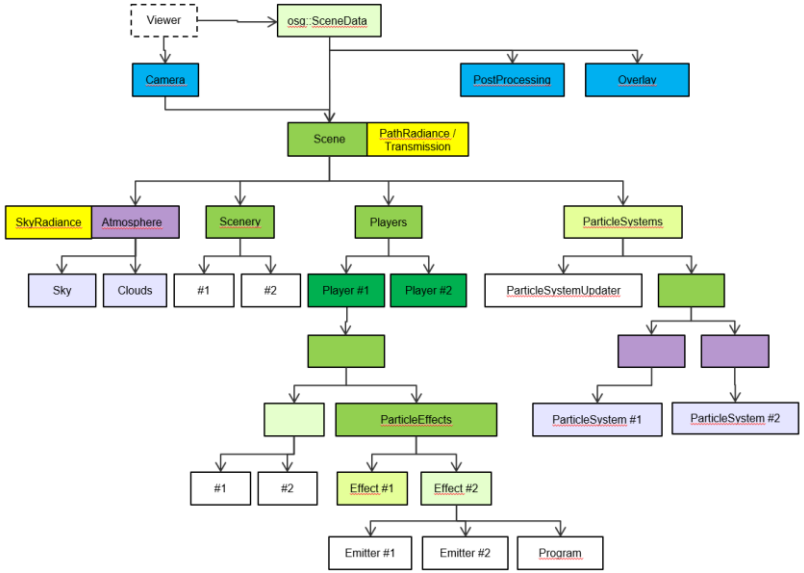
EDI is the interface used for script-based generation of scenes, realizing the graphical user interface EMIT-SceneEditor and tight integration into other applications.
Available in C++ and Tcl.

BatchPseudoIR.tcl (C:\Emit\Scenes\PV-EMIT-SE) - GVIM

Datei Editieren Werkzeuge Syntax Puffer Ansicht Hilfe

```
package require tclemit
package require tcledi

# Enable verbose messages.
Edi VerboseMode true

# Define the name of the scene to generate and the wavelength it is used for.
Edi NewScene "TestScenePseudoIRBatch.osgt" 3 5

Edi SetSkyTemperature -38 10

# Add a scenery (terrain) file.
Edi AddScenery "terrain/master.flt"

# Set date and time.
Edi SetDateAndTime "2016-04-20" "12:00:00" "-10:00"

# Set the temperature range of the scenery in Celsius for Pseudo-IR.
Edi SetTemperatureRange 10 20

# Add player with EntityID 1 using explicit waypoint definitions.
set ply1ID 1
Edi AddStaticPlayer "Players/VBL.osgt" $ply1ID
Edi AddPlayerWayPoint $ply1ID  0.000000  1482.770000 -533.728000 137.000000
Edi AddPlayerWayPoint $ply1ID  2.000000  1510.550000 -520.156000 137.000000
Edi AddPlayerWayPoint $ply1ID  4.000000  1515.080000 -530.172000 137.000000

# Set the temperature range of the player in Celsius for Pseudo-IR.
Edi SetPlayerTemperature $ply1ID 20 30

# Add player with EntityID 2 using a waypoint import file.
set ply2ID 2
Edi AddStaticPlayer "Players/VBL.osgt" $ply2ID
Edi AddPlayerWayPointFile   $ply2ID "WayPoints/Runway-t01-t14.wp"
Edi SetPlayerAnimationTypes $ply2ID "CubicSpline"

# Set the temperature range of the player in Celsius for Pseudo-IR.
Edi SetPlayerTemperature $ply2ID 30 50

# Write out scene file.
Edi WriteScene
```

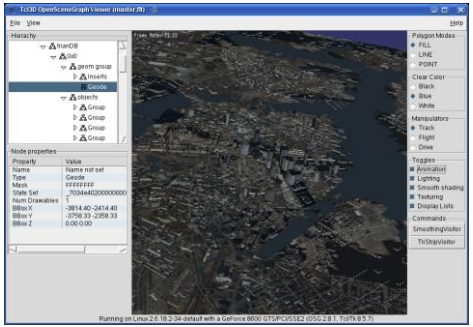
30,0-1 Alles

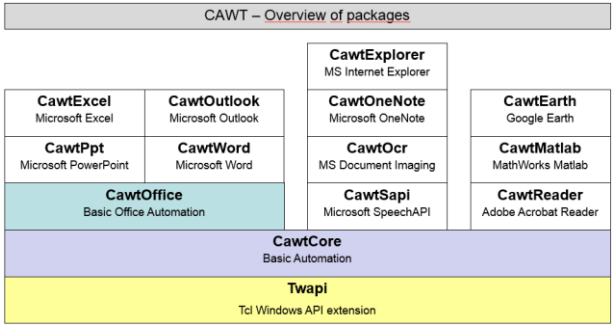


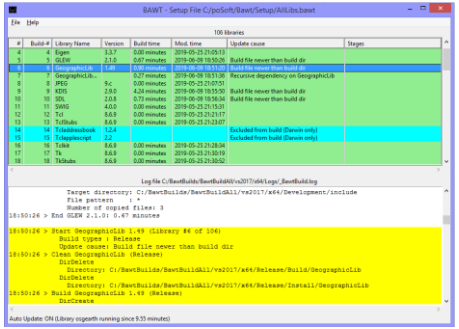
Tcl extensions to make life at work easier

Img	Additional formats parsers (SUN, SGI, RAW)
Tcl3D	Tcl wrapper for OpenGL and OpenSceneGraph
CAWT	COM Automation With Tcl
BAWT	Build Automation With Tcl

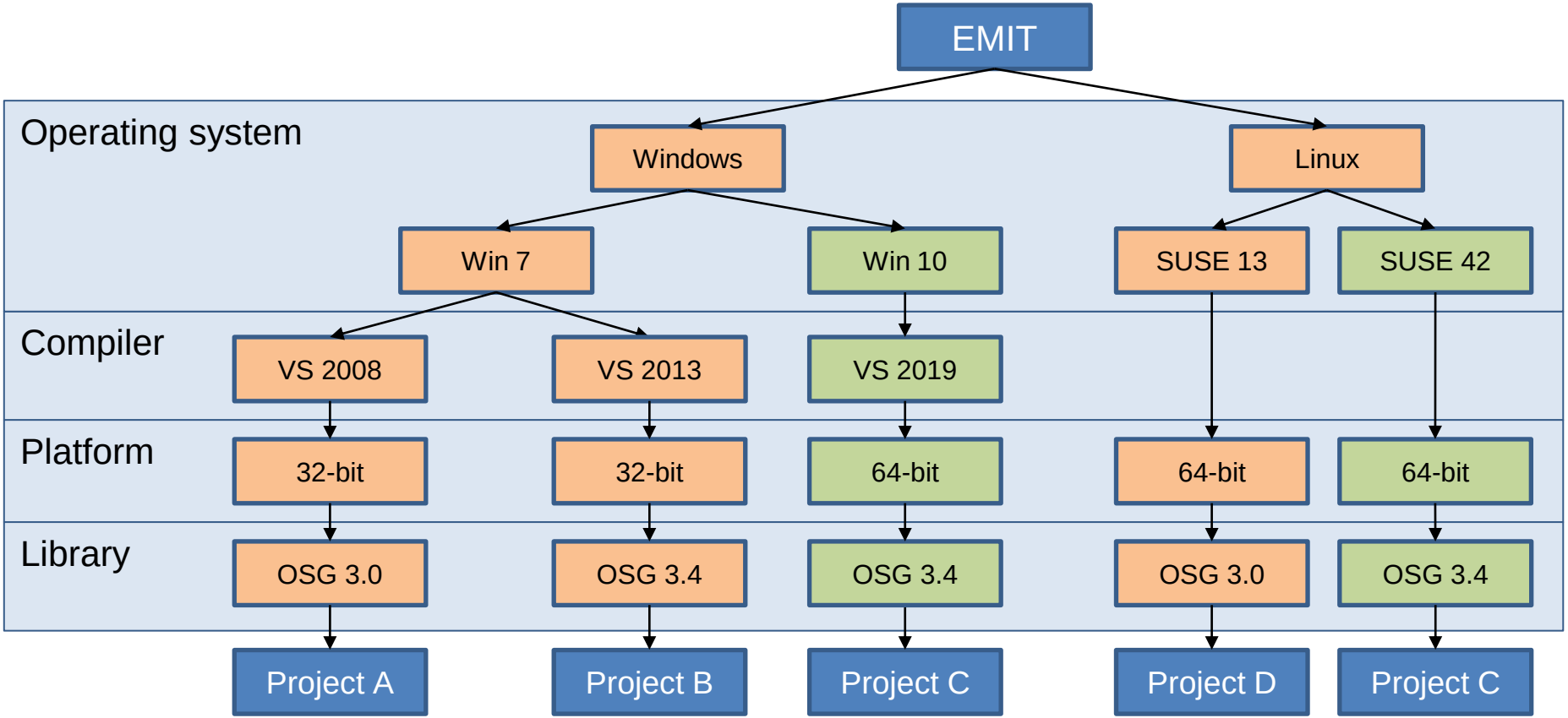
Format	Description
BMP	Windows Bitmap Format
DTED	Digital Terrain Elevation Data
GIF	Graphics Interchange Format
ICO	Windows Icon Format
JPEG	Joint Picture Experts Group Format
PCX	Paintbrush Format
PIXMAP	Pixmap Image Type
PNG	Portable Network Graphics
PPM	Portable Pixmap Format
PS	Postscript and PDF
RAW	Raw Binary Data
SGI	Silicon Graphics Format
SUN	Sun Raster Format
TGA	Targa Format
TIFF	Tagged Interchange File Format
WINDOW	Tk window as photo image
XBM	X Bitmap Format
XPM	X Pixmap Format







BAWT based daily builds



EMIT and Tcl/Tk – BAWT based daily builds

EMIT depends on a large number of third party libraries.
All libraries are compiled with BAWT under Windows and Linux.

Tools:

- ✓ CMake
- ✓ SWIG
- ✓ Doxygen
- ✓ InnoSetup

Tcl/Tk environment:

- ✓ Tcl/Tk (+ appr. 20 Tcl Packages)

Base Libraries:

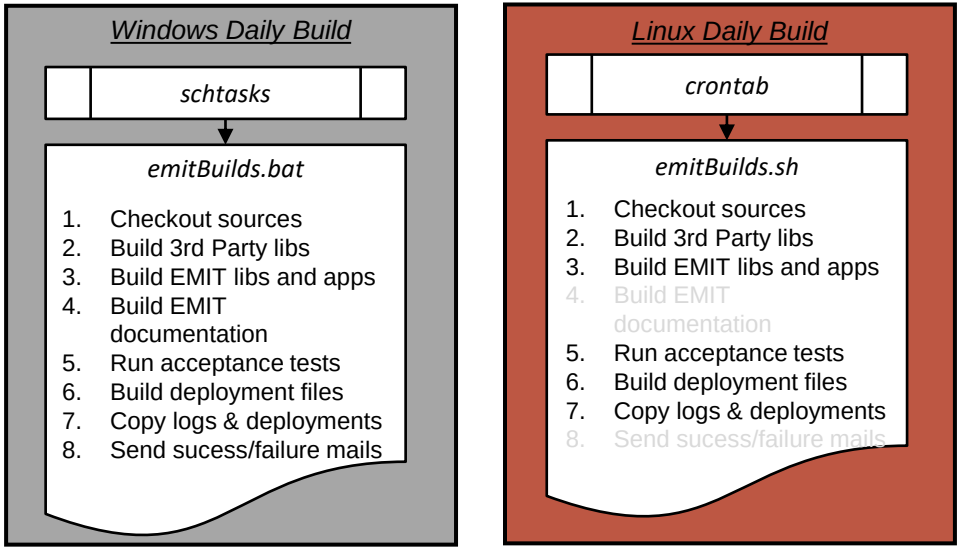
- ✓ zlib
- ✓ giflib
- ✓ libjpeg
- ✓ libpng
- ✓ libtiff
- ✓ Freetype
- ✓ libressl
- ✓ Curl
- ✓ Boost
- ✓ Eigen
- ✓ Fftw
- ✓ Xerces

Simulation/Graphics Libraries:

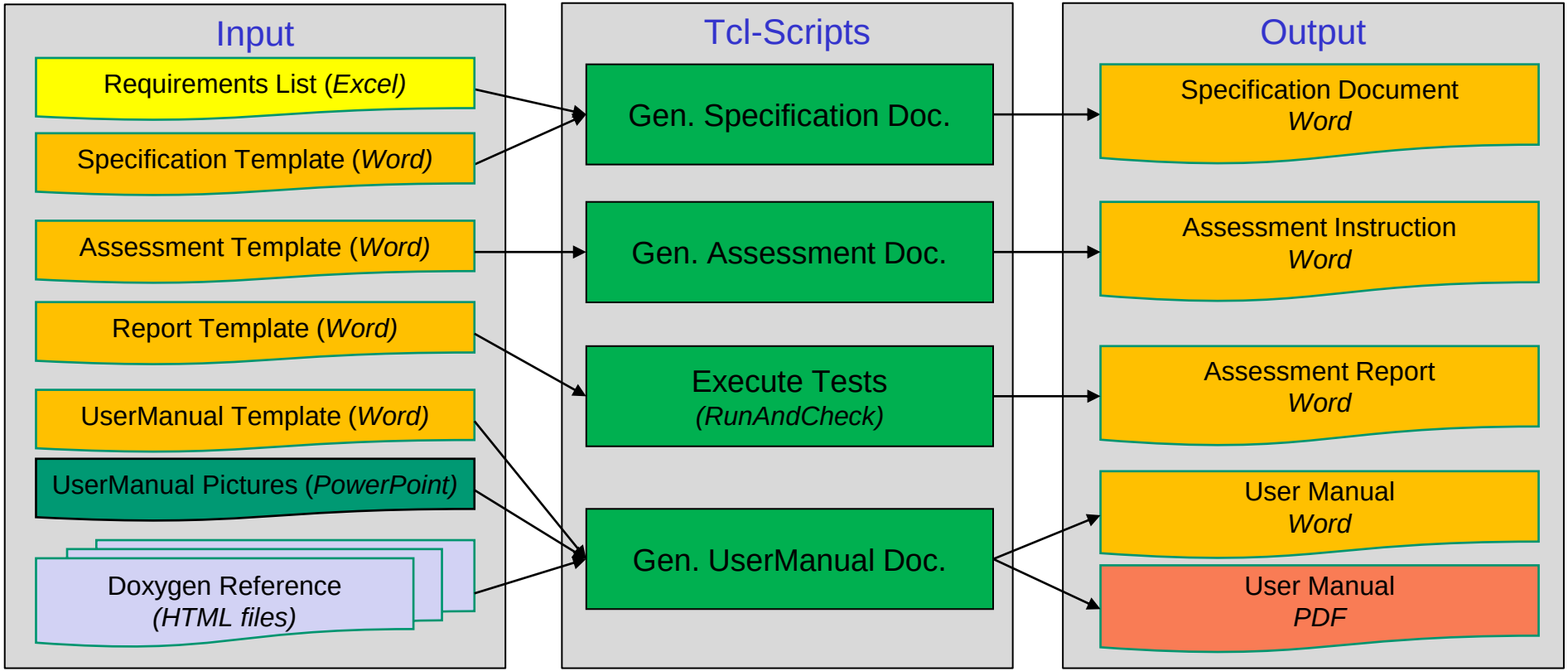
- ✓ GeographicLib
- ✓ KDIS
- ✓ Freeglut
- ✓ Ftgl
- ✓ Glew
- ✓ OpenSceneGraph
- ✓ WxWidgets

Continous Development and Integration

- An automated infrastructure for creating, testing and delivering the various EMIT product versions, including documentation and test protocols.
- Daily Build computer for Windows and Linux (Build, Test, Deploy).
- No additional license costs due to combination of OpenSource (SVN, Tcl) and Microsoft Office (Word, Excel, OneNote) products.



EMIT and Tcl/Tk – CAWT based generation of documentation



Test Input

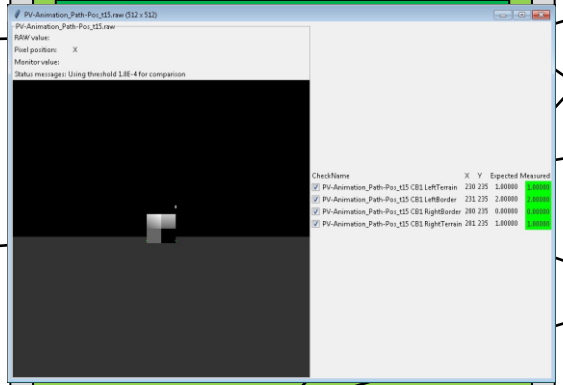
```
SpecifySceneFile     "Chessboard-Animation.osgt"
SpecifySensorFile    "PV-Animation/PV-Animation_Path.esc"
SpecifyWindow        1 1 10 30
SpecifyImageOutput   1 "PV-Animation_Path" 0 raw
SpecifyCoordSysType FlatEarth
SpecifyLocalOri      1
SpecifyLogging       2 "PV-Animation_Path.log"
set camPosY [expr -(1000.0 - 44.370)]
SpecifyLocation      Pos_t00    0.0 $camPosY 20.0   0.0 0.0 0.0    0.0
SpecifyLocation      Pos_t05    0.0 44.370   20.0   0.0 0.0 0.0    5.0
```

```
#-------------------------------------------------------------------------
# New location
set locationName "Pos_t05"

# Check correct image size.
CheckImgSize $testName $locationName $testWidth $testHeight

# Check border of CB1. Object width must be 50 pixels.
CheckPoint "eq" $testName $locationName "CB1 LeftTerrain"   230 235    1.0
CheckPoint "eq" $testName $locationName "CB1 LeftBorder"    231 235    2.0
CheckPoint "eq" $testName $locationName "CB1 RightBorder"   280 235    0.0
CheckPoint "eq" $testName $locationName "CB1 RightTerrain"  281 235    1.0
```

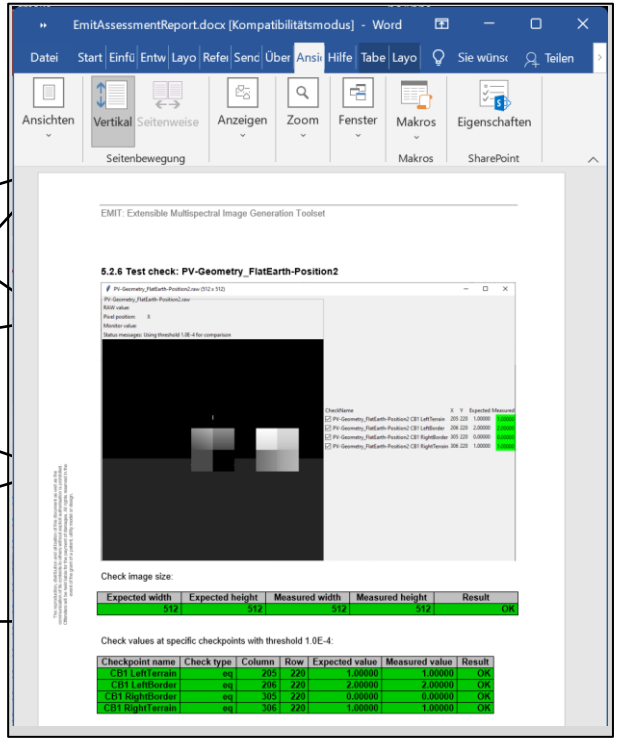
Report Template
Word

Test Scripts

RunAndCheck



Generate Report

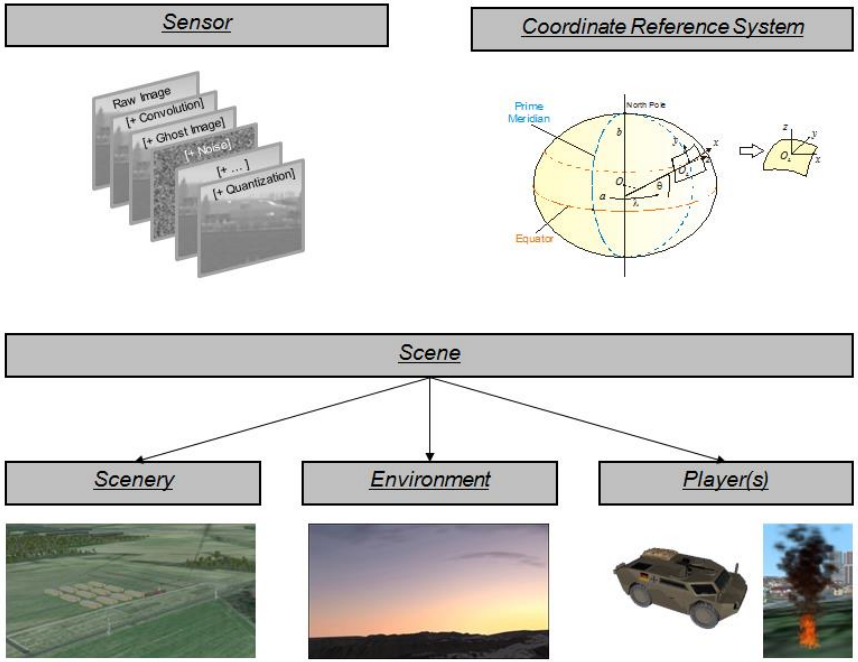


The EMIT Scene Editor builds up an EMIT renderable scene by

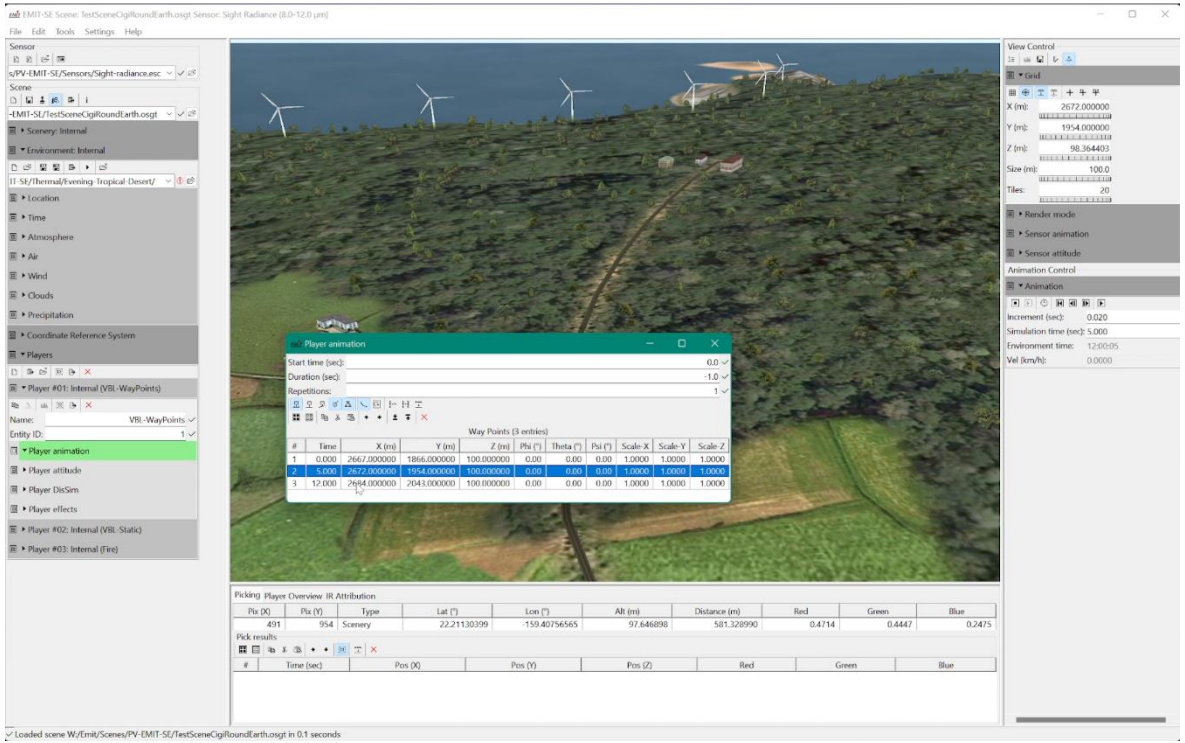
- combining components created in the modelling step (Players, Terrain)
- animating dynamic players
- defining the environment
- configuring virtual sensors

The EMIT SceneEditor is written in Tcl by using the wrapped EMIT functionality supplied by the EDI interface.

Additionally a separate Viewer class allows the incorporation of the OpenSceneGraph based EMIT 3D window into a Tk widget.



Overview of the Graphical User Interface

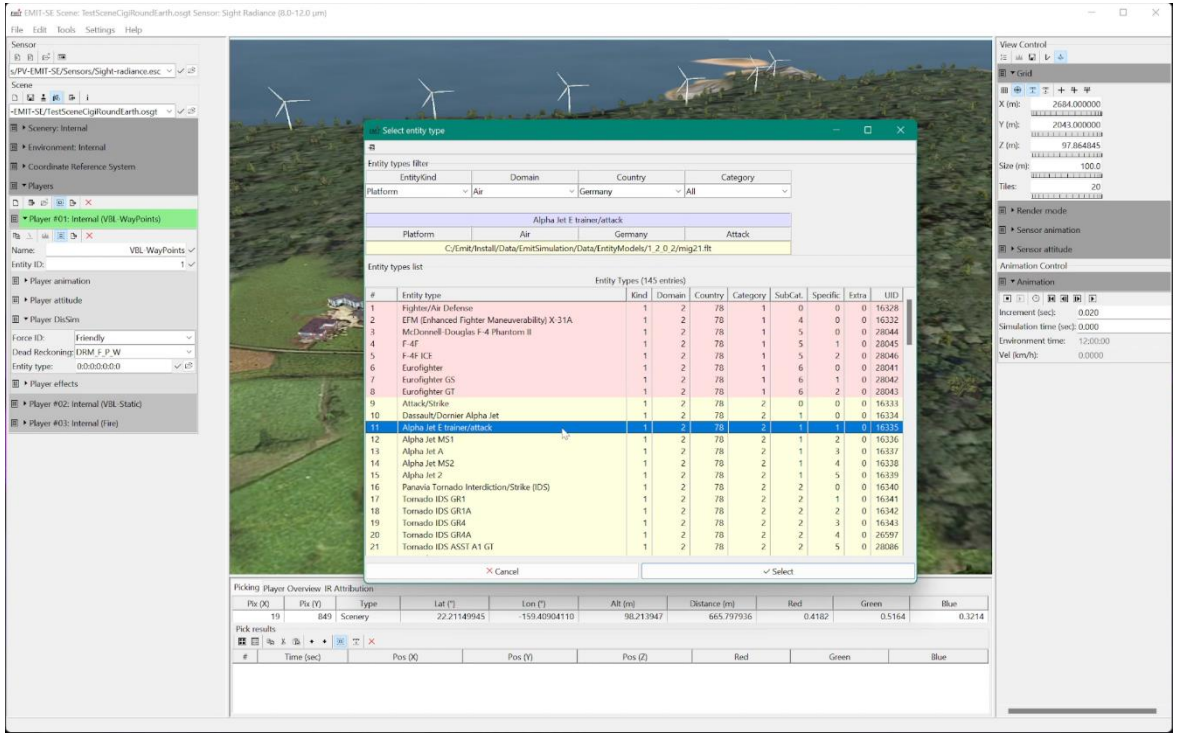


Tcl/Tk specialities:

- Togl widget
- Tk events -> OSG

- Rollups
- Dials
- Tablelists

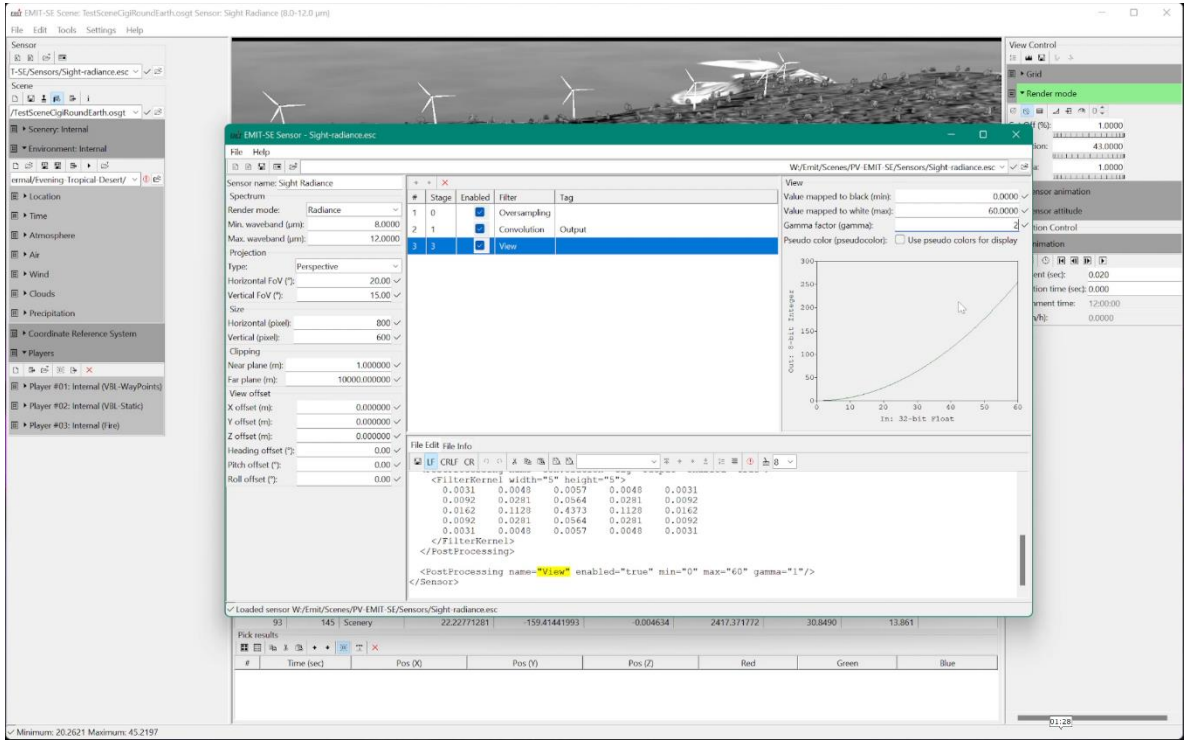
Player Management



Tcl/Tk specialities:

- C++ Code-Generation for EntityTypes with Tcl script

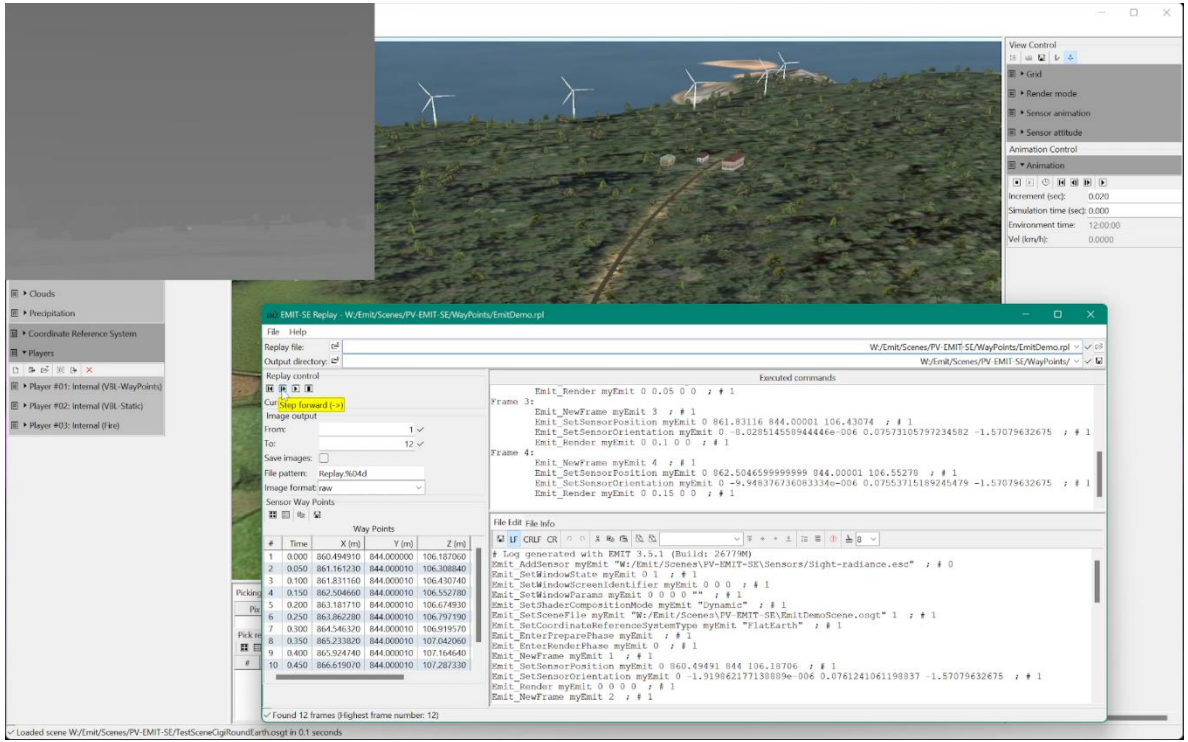
Render Modes & Sensor Management



Tcl/Tk specialities:

- ukaz
- tdom

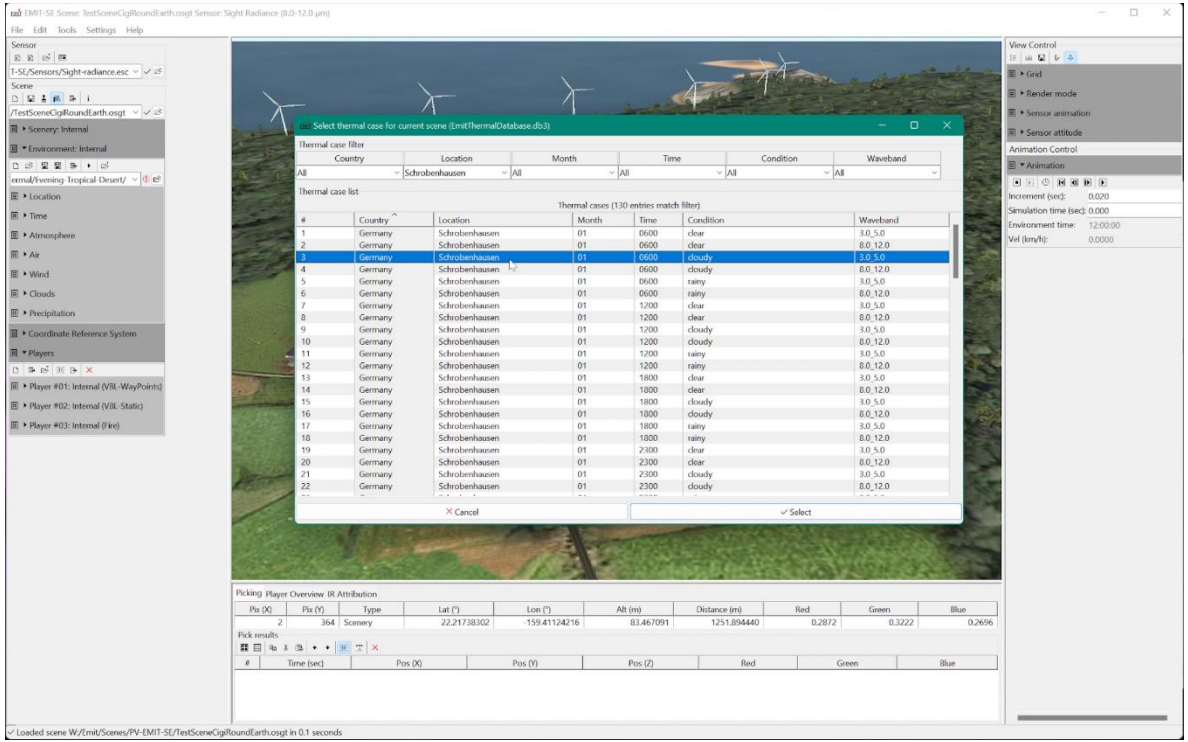
Replay Functionality



Tcl/Tk specialities:

- Render log files are written as Tcl scripts and can directly be interpreted.

Atmospheric and Thermal Calculations



Tcl/Tk specialities:

- Ukaz
- Tablelists
- Sqlite binding

Main features:

- Realtime Image Generation
- Realtime sensor effects
- Realtime special effects (ex. fire, smoke)
- Flexible, network transparent interface
- E/O and IR mode
- Linux and Windows

Tcl usage:

- Tcl/Tk based graphical user interface
- C++ interfaces wrapped for Tcl with SWIG
- CD/CI workflow based on Tcl