Inside Biosphere3D
The free digital globe from a developer prospect

Steffen Ernst

Short Description

- Biosphere3D targets interactive landscape rendering based on a virtual globe
- Input data: DEM, satellite and aerial images, 3D models (Collada), 3D plant models (Lenné3D Flora3D), Shapefiles
- Scale independent: range from First-person view to Globe view
- Modular: replaceable software components
- Scalable: multi-core CPUs and GPUs
- Portable: minimized OS and 3D-API dependencies
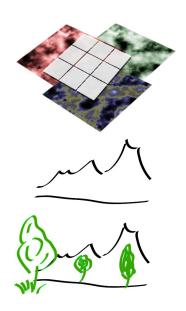- Interactive: no pre-calculations

Software Architecture: Overview

- Base: generic low-level libraries
  - Math, Geometry, Image, Thread, Crypto, OpenGL, …
  - All tested by automatic tests
- API: plant libraries
  - DRM, Plant, Render
- Plant Distributor ("oik"): plant instance management
  - Sketch of plant distribution methods
- Player: 3D scene processing and rendering
- GUI: Biosphere3D application
  - translates between user and computer
- Sandbox: experimental code
  - Can link to stable code, but not vice-versa

Software Architecture: Player

- Map:
  data management, images/instances
- Terrain:
  clipmap based LOD
- Plant:
  plant distribution framework
- Scene:
  contains Terrain, Plants, atmosphere
- View:
  camera framework for Scene views
- Job:
  multithreaded Map processing
- Renderer:
  OpenGL output of a scene view
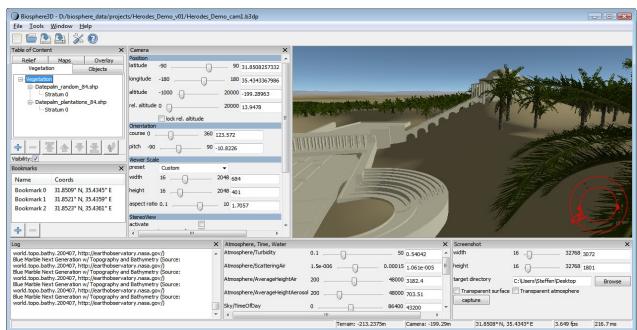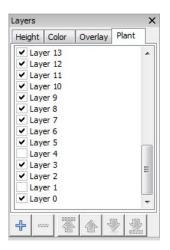
Source: de.wikipedia_KMJ

Current State: GUI

- FPS-, sphere- and compass-navigation
- Add/remove layers
- Save/load views
- Save/load scenes
- Set scene, global and camera parameters
- Set global parameters
- Render stills GPU-accelerated and in virtual resolutions to hard disk

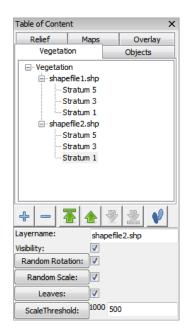# Extension 1: show vegetation layers in tree view

Preconditions:

- Layers are stored in Player → Terrain
- Strata are numbered incrementally
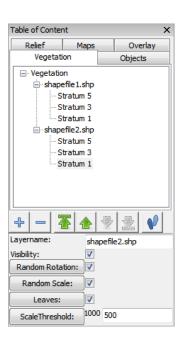- relation to its shapefile isn't obvious

Goal:

- show the relation between a Shapefile and its
    embedded Strata
- allow adjustment of attributes like name and
    visibility

Extension 1: show vegetation layers in tree view

Implementation:
- Derive new widget from TreeView-wxWidget
  - → replaces old ListView
  - → stored in GUI
- Add new panel, to show properties of selected (sub-)layer
  - → changes influence all sublayers
- New attributes have to be added to Player → Terrain
- Add possibility to store default values to registry

Extension 2: adding sphere navigation

Preconditions:
- Game-like FPS navigation implemented in Player → View
- Used in GUI → PlayerController

Goal:
- Sphere navigation following Google Earth
- Add Space Mouse Support

Source: www.3dconnexion.de

Extension 2: adding a sphere navigation

Implementation:
- Add new camera to Player → View, which allows
      drag, tilt, rotation, zoom
- Implement here also the SDK for the Space Mouse
- SDK needed to compile → activated by putting an #define in:
      "project\player\current\src\UserDefs.txt": -DUSE_3D_DEVICE
- GUI → PlayerController passes user input to the cameras and
      switches between them
- Input comes from:
    - wxWidgets Events (directly implemented in PlayerController)
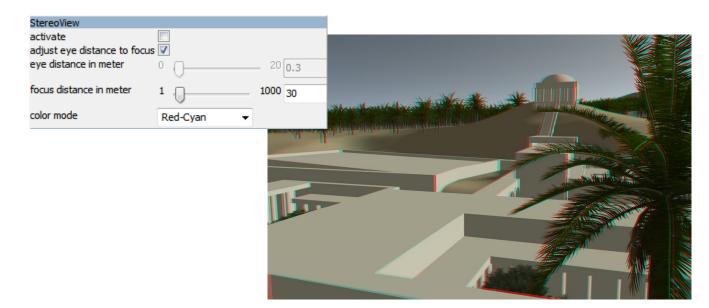   - Space Mouse implementation (Player → View)

Extension 3: adding stereo rendering

Preconditions:
- GUI calls Renderer under Player → Renderer_OpenGL
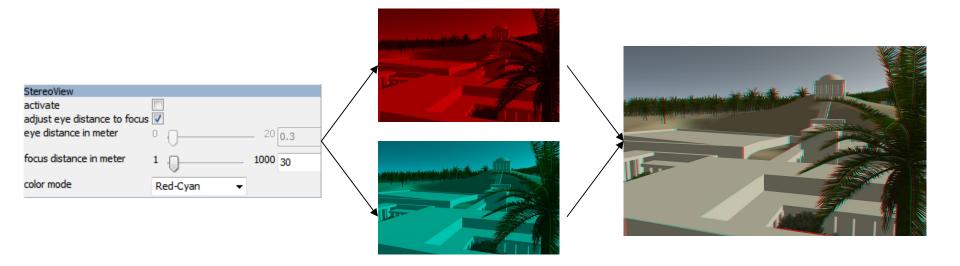- Panel with all the camera parameters

Goal:
- Stereo rendering (anaglyph)
- Adjust Parameters through GUI

Extension 3: adding stereo rendering

Implementation:
- Sliders in camera panel controls eye- and focus distance
- Pass settings to Renderer as new arguments
- Add new camera to Player → View, which "looks through two eyes"
- Combine renderings from the two eyes to the final image by using color masks from OpenGL

Summary

Plain and simple: There's no USP. But there is:
- Clean software architecture
- Modern programming style
- Minimal code duplication (continuously refactored)
- Generic implementations of algorithms (e.g.1-nD images)
- Minimal/orthogonal interfaces
- Easy to add user visible unique features when required

More infos at:
- www.biosphere3d.org
- sourceforge.net → project Biosphere3D