Solar3D User Guide

Contents

1. Background

Solar3D is a software application designed to interactively calculate solar irradiation and sky view factor at points on 3D surfaces. It is essentially a 3D extension of the GRASS GIS r.sun solar radiation model. According to the GRASS GIS documentation [1]:

"r.sun - Solar irradiance and irradiation model.
Computes direct (beam), diffuse and reflected solar irradiation raster maps for given day, latitude, surface and atmospheric conditions. Solar parameters (e.g. sunrise, sunset times, declination, extraterrestrial irradiance, daylight length) are saved in the map history file. Alternatively, a local time can be specified to compute solar incidence angle and/or irradiance raster maps. The shadowing effect of the topography is optionally incorporated."

To calculate the solar irradiation over a certain time interval at a point on a 3D surface, Solar3D first derives the slope and aspect value from the surface normal, and then use a cube map to determine if the point is shaded at each time step for the entire duration. Instead of using the brute-force ray-casting which relies on ray-triangle intersection and therefore is computationally intensive, Solar3D generates a cube map-based panoramic view of the 3D scene at the point with sky and non-sky pixels encoded in different color values, and then determines if the point is shaded at each time step by looking up the intersected pixel in the corresponding cube map face (Figure 1). In this way, Solar3D can rapidly calculate daily to annual solar irradiation at arbitrary points in sufficiently small time-steps (smaller than one hour). However, a limitation with Solar3D is that it is designed specifically to rapidly calculate solar irradiation at discrete points and is not equipped with adequate performance to calculate for large areas where points are densely and uniformly distributed, and the main reason is that generating millions of cube maps is not computationally affordable.

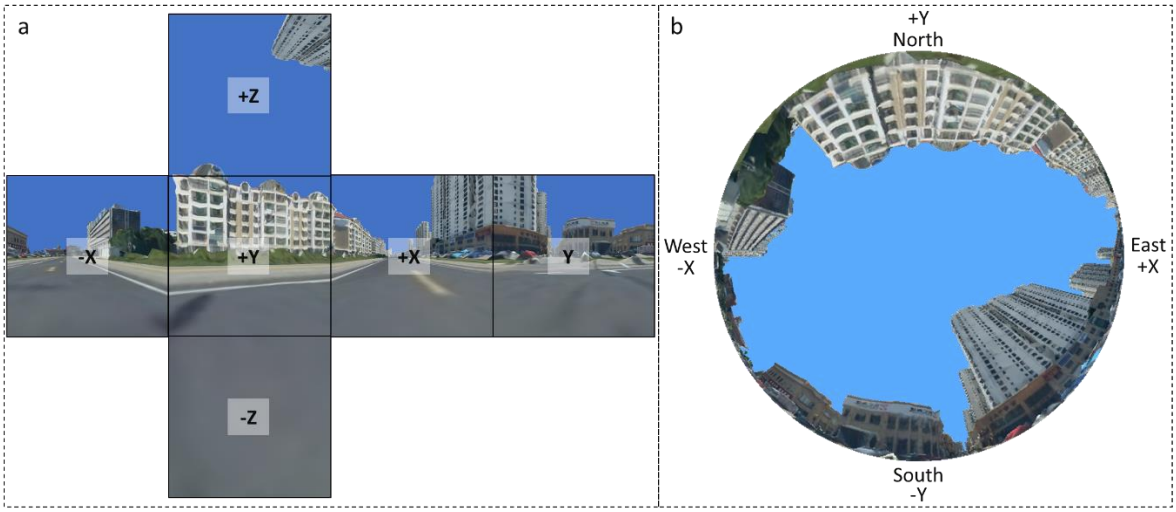

Figure 1. A cube map (a) and hemispherical view at the same ground location.

2. Software Architecture and Business Logic

The core framework is constructed by integrating the r.sun solar radiation model into a 3D graphics engine, OpenSceneGraph [2], an OpenGL-based 3D graphics toolkit widely used in visualization and simulation. OpenSceneGraph is essentially an OpenGL state manager with extended support for scene graph and data management. The reasons for choosing OpenSceneGraph are multifold: firstly, OpenSceneGraph provides user-friendly, object-oriented access to OpenGL interfaces; secondly, OpenSceneGraph provides built-in support for interactive rendering and loading of a wide variety of common 3D model formats including osg, ive, 3ds, dae, obj, x, fbx and flt; thirdly, OpenSceneGraph supports smooth loading and rendering of massive oblique airborne photogrammetry-based 3D city models (OAP3Ds or integrated meshes), which are already being widely used in urban and energy planning. Once

exported from image-based 3D reconstruction tools such as Esri Drone2Map and Skyline PhotoMesh into OpenSceneGraph's Paged LOD format, OAP3Ds can be rapidly loaded into OpenSceneGraph for view-dependent data streaming and rendering. The r.sun model in Solar3D also relies on OpenSceneGraph for supplying the key parameters needed for irradiation calculation: (1) location identified at a 3D surface; (2) slope and aspect angles of the surface. (3) time-resolved shadow masks evaluated from a cube map rendered at the identified position.

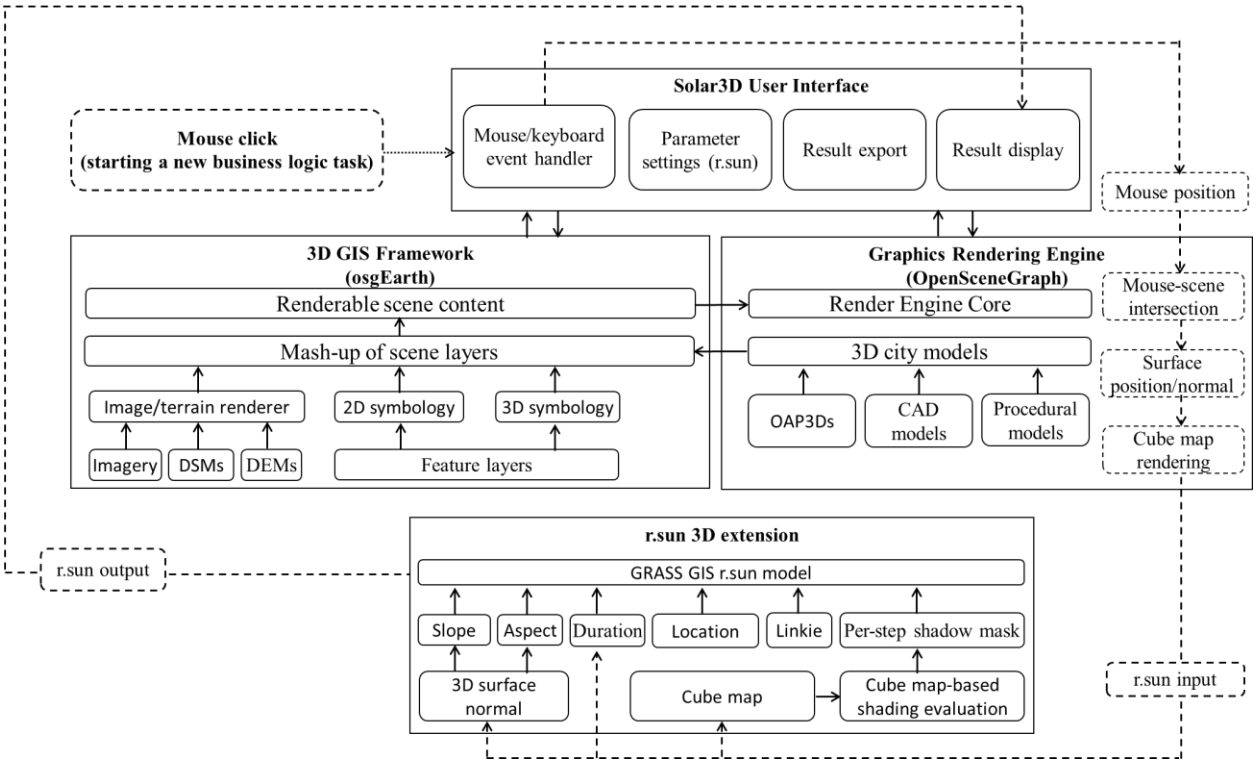

Figure 2. Solar3D architecture (solid boxes and arrows) and business logic (dashed boxes and arrows)

The business logic of the core framework works in a loop triggered by user requests (Figure 2): (1) a user request is started by mouse-clicking at an exposed surface in a 3D scene rendered in an OpenSceneGraph view overlaid with the Solar3D user interface (UI) elements; (2) the 3D position, slope and aspect angle are derived from the clicked surface; (3) a cube map is rendered at the 3D position as described above; (4) all required model input [1], including the geographic location (latitude, longitude, elevation), Linkie turbidity factor, duration (start day and end day), temporal resolution (in decimal hours), slope, aspect and shadow masks for each time step, is gathered, compiled and fed to r.sun for calculating irradiation. The shadow masks are obtained by sampling the cube map with the solar altitude and azimuth angle for each time step; (5) the r.sun model is run with the supplied input to generate the global, beam, diffuse and reflective irradiation values for the given location; (6) the r.sun-generated irradiation results (in units of kWh/m2) are returned to the Solar3D UI for immediate display.

To better facilitate urban and energy planning, the core framework is further extended by integrating into a 3D GIS framework, osgEarth [3], an OpenSceneGraph-based 3D geospatial library used to author and render planetary- to local-scale 3D GIS scenes with support for most common GIS content formats, including DSMs, DSMs, local imagery, web map services, web feature service and Esri Shapefile. With the 3D GIS extension, Solar3D can serve more specialized and advanced user needs, including: (1) hosting multiple 3D city models distributed over a large geographic region; (2) overlaying 3D city models on top

of custom basemaps to provide an enriched geographic context in support of energy analysis and decision making; (3) incorporating the topography surrounding a 3D city model into shading evaluation; (4) interactively calculating solar irradiation with only DSMs.

The code was written in C++ and complied in Visual Studio 2019 on Windows 10. The three main dependent libraries used, OpenSceneGraph, osgEarth and Qt5, were all pulled from vcpkg [4], a C++ package manager for Windows, Linux, and MacOS, and therefore Solar3D can potentially be complied on Linux and MacOS with additional work to set up the build environment.

Note: If Solar3D is not able to start due to missing runtime dlls, try to install the Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019 from https://aka.ms/vs/16/release/vc_redist.x64.exe.

3. Basic Usage and Workflow

The Solar3D UI consists of three components respectively responsible for parameter settings, result display and status update. The parameter settings panels are located in the top left with UI elements for setting the Linkie factor, start day, end day, time step, latitude and base elevation overrides (used in case of a non-geoferenced scene). The result display UI consists of two panels located in the left side right below the parameter settings panel used for immediate display of feedback from the latest request and a pop-up panel used to display results at the cursor point. The status UI elements include a compass at the top right and a status bar displaying cursor and camera coordinates toward the bottom.

A first step in the workflow of Solar3D is scene preparation. The users are expected to prepare their own scenes with a least one 3D model and optionally some basemaps. An easy usage is to start Solar3D with the path of a single 3D model exported form CAD or an OAP3D exported from a photo-based 3D reconstruction software (Figure 3), but in this way, the scene will not be geoferenced and thus the users need to specify the latitude and base elevation override.

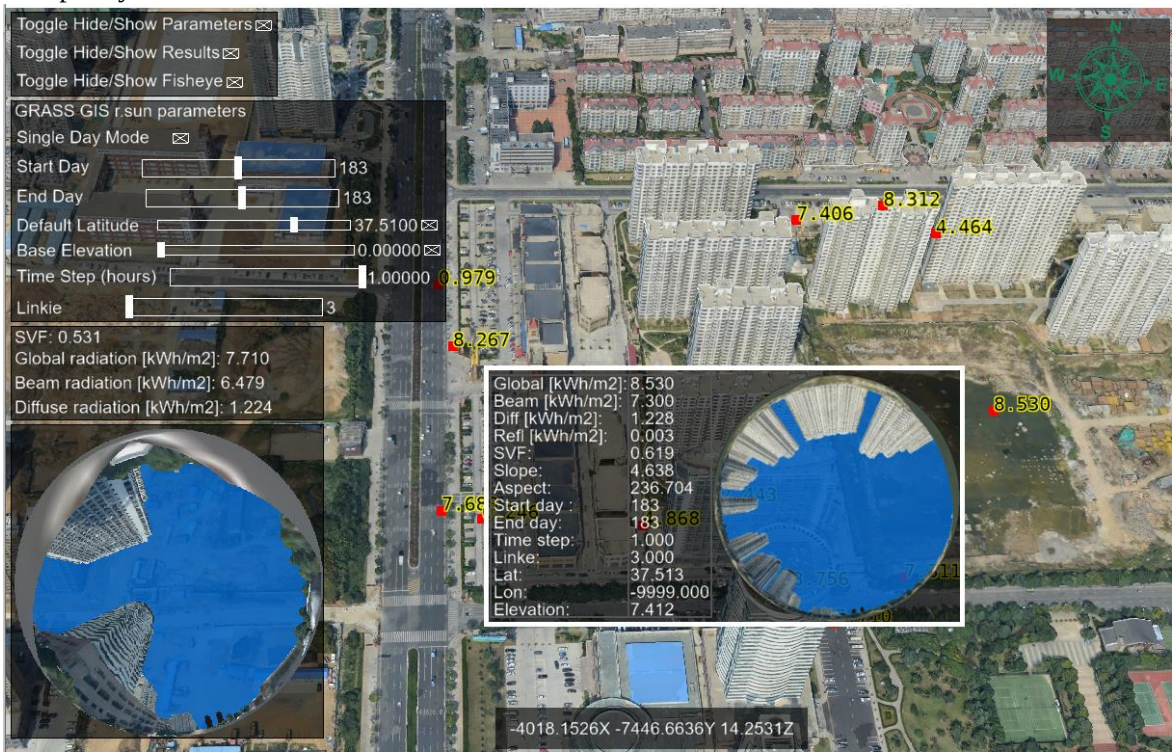

Figure 3. Calculating solar irradiation with an OAP3D in Solar3D

To integrate 3D models into a geoferenced scene with basemaps (Figure 4), the users will need to follow the instructions [3] and examples ("Solar3D/bin/tests/") provided by osgEarth. An inconvenience is that osgEarth does not offer a scene editor with a graphic UI for scene authoring, and instead, the users will need to manually add and configure scene layers in a text editor based on one of the example scene files (*.earth). An advantage with osgEarth is it can be used to author advanced scenes with 3D models overlaid on DEMs and DSMs distributed all over the Earth. Additionally, osgEarth also provides the ability to extrude building footprints from polygon features into 3D models for use in Solar3D.

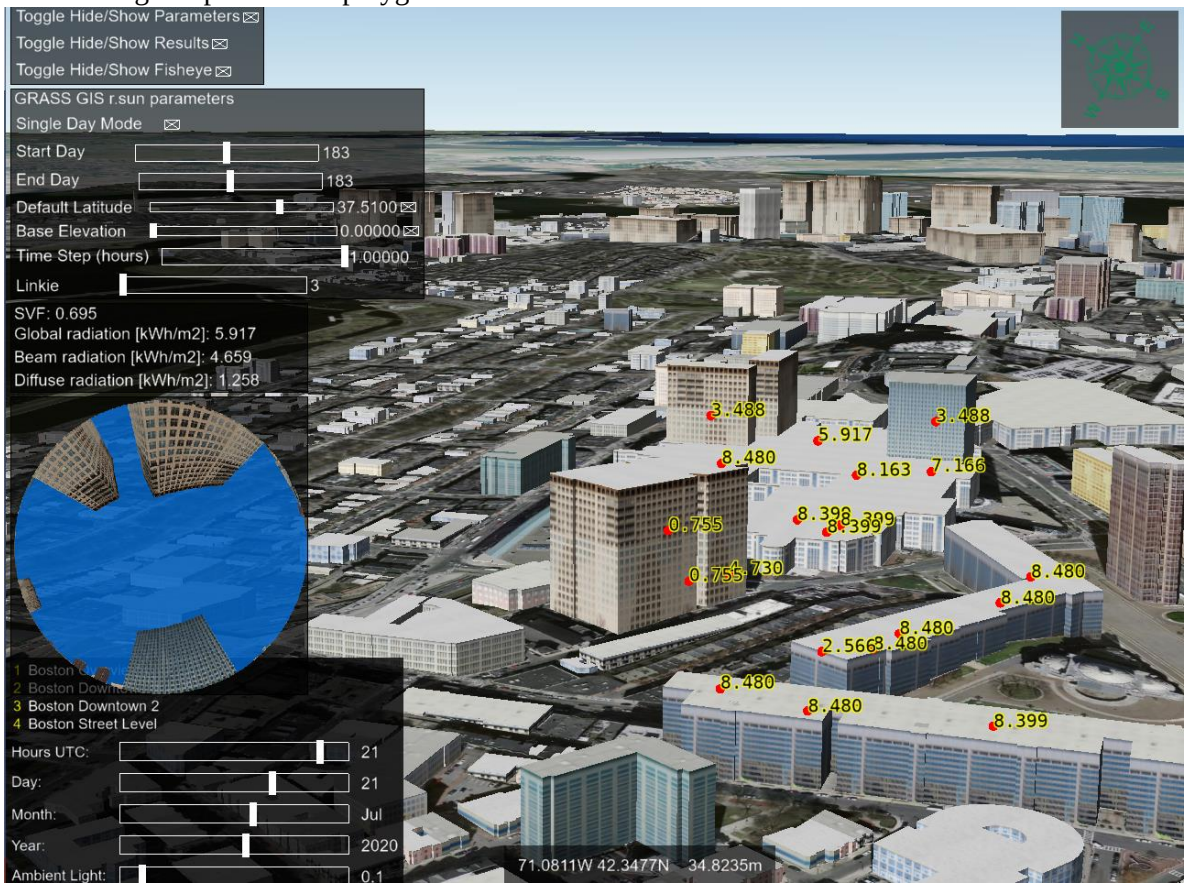

Figure 4. Calculating solar irradiation with a building footprints-extruded 3D city model (Boston) overlaid on a basemap in Solar3D

In a typical use scenario, after starting Solar3D with a single 3D model or an osgEarth scene, the user zooms to an area of interest with buildings on which PV arrays are planned to be deployed, and then irradiation estimates are obtained by interactively clicking at rooftops and facades to identify suitable surface areas for PV deployment. Solar3D processes calculation request upon mouse click (with Ctrl down) once at a time and typically finishes a request within a couple of seconds and displays the results immediately. A marker with text label will be displayed at the location at which a calculation request has been finished. The irradiation results obtained during a session can be exported in a batch to a comma-delimited text file for analysis. When exporting an irradiation record, the associated r.sun parameters and 3D coordinates will be packed in a single row.

4. Keyboard/Mouse Operations

Operation	Keyboard/Mouse
Add a point (start a calculation request)	***Ctrl + left click*** at a surface in the scene, and the calculation will normally be finished instantly with feedback displayed in the left panel. All irradiation values are in units of kWh/m2. A point marker with a label will also be added at the clicked position.
Undo Add	***Ctrl + Z***
Redo Add	***Ctrl + Y***
Show pop-up at a point	***Ctrl + Mouse Hover*** over a point marker
Toggle display text labels	***Ctrl + T*** to hide/show the point text labels
Hide/Show parameter settings/feedback panels	***Toggle*** the checkboxes in the top left panel
Export	***Ctrl + E*** All points will be exported in a .csv named with the current time stamp to the path "Solar3D/bin".
Toggle Scene Lighting	***Toggle Key L***

5. Scene Preparation

CAD models can be created in CAD software such as Autodesk 3ds Max, SketchUp and Blender and exported in a format that OpenSceneGraph supports [5]. OAP3Ds can be created using photo-based 3D reconstruction tools such as Esri Drone2Map and Skyline PhotoMesh and exported into OpenSceneGraph's Paged LOD format. Building footprints-extruded 3D models need to be created in an osgEarth scene by apply 3D symbology to a feature layer.

To create an osgEarth scene, find a template under "Solar3D/bin/tests/" that best suits your needs and then make a copy and modify in a text editor. Table 1 is a selected list of examples demonstrating how to start Solar3D with different types of 3D models (OAP3D, CAD, building footprint extrusions) with or without osgEarth scenes (global or local).

Table 1. Examples of Solar3D command line usage

Example	Command line (format: Solar3D [model/scene path])
Solar3D/bin/Example_OAP3D.bat (Start with an OAP3D model)	Solar3D ./data/models/OAP3D.oap3d
Solar3D/bin/Example_CAD.bat (Start with an CAD model)	Solar3D ./data/models/CAD.osgb
Solar3D/bin/Example_Boston_Global.bat (Start with a global osgEarth scene) This example shows how to extrude build footprints from a shapefile	Solar3D ./tests/boston.earth
Solar3D/bin/Example_Boston_Projected.bat (Start with a local (projected) osgEarth scene) This example shows how to extrude build footprints from a shapefile	Solar3D ./tests/boston_projected.earth
Solar3D/bin/Example_Solar3D_Global.bat (Start with a global osgEarth scene) This example shows how to position CAD and OAP3D models in a global scene.	Solar3D ./tests/solar3d_global.earth

Solar3D/bin/ Example_Solar3D_Projected.bat (Start with a global osgEarth scene) This example shows how to position CAD and OAP3D models in a local scene.	Solar3D ./tests/solar3d_projected.earth

5.1. OAP3D data structure and loading

OAP3Ds can be created using photo-based 3D reconstruction tools such as Esri Drone2Map and Skyline PhotoMesh and exported into OpenSceneGraph's Paged LOD format. An OAP3D is spatially divided into a number of nearly square tiles stored in separate folders. Each OAP3D tile folder contains a progressive mesh present in OpenSceneGraph's Paged LOD format (.osgb) consisting a number of .osgb files representing different levels of detail (Figure 5), and to explore such a progressive mesh in an OpenSceneGrap view, one only needs to load the main .osgb file with the same name as the tile folder (Tile_-002_-016.osgb in Figure 6) which will stream in the higher-level .osgb files in a view dependent manner as needed.

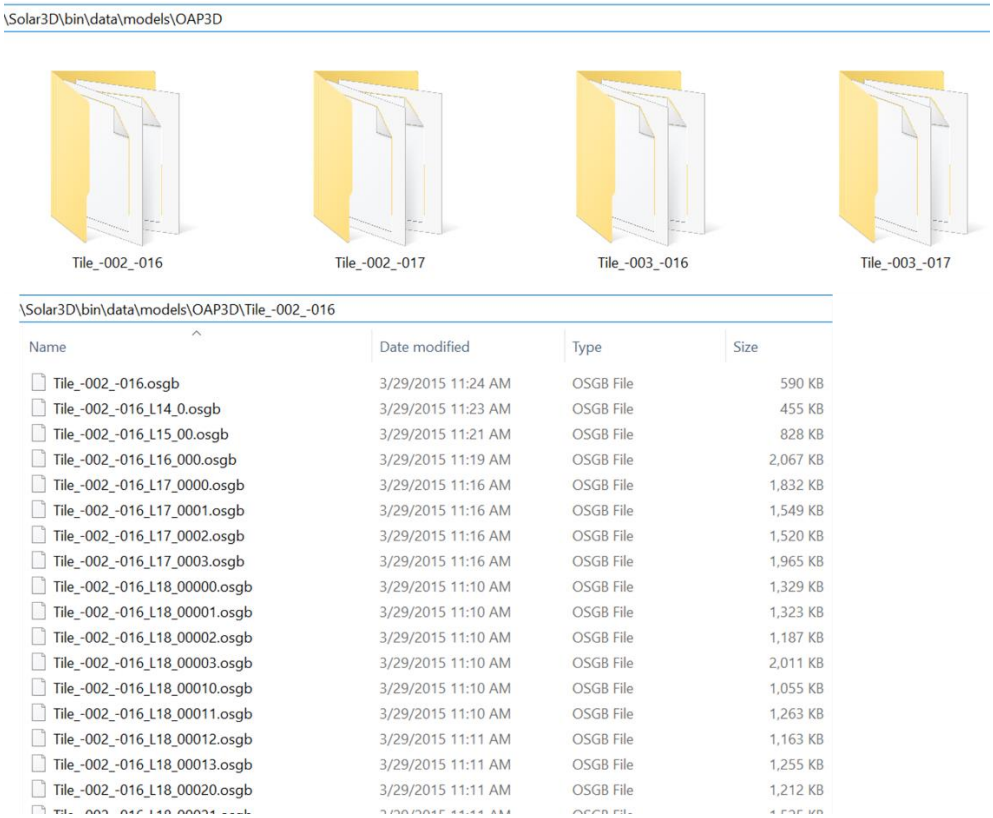

Figure 5. Folder and file organization of the sample OAP3D dataset

An OpenSceneGraph plugin was developed specifically for Solar3D to load an OAP3D dataset consisting of multiple tile folders. In order to specifically request that the OAP3D plugin be used, when providing the path of an OAP3D dataset, the plugin extension ".oap3d" must be appended to the root path of the dataset. For example, the sample OAP3D dataset is located at "bin/data/models/OAP3D", so the path to provide in a Solar3D command line or an osgEarth scene file (.earth) should be "bin/data/models/OAP3D.oap3d".

5.2. Create a map

An osgEarth scene (.earth) starts with a map node. A map can be created as either a global or local scene by specifying the type attribute (type="geocentric" for a gobal scene or type="projected" for a local scene) in the map node. If the map is created as a local scene, the projected coordinate system must be provided as a child node ("options") as shown in Table 3.

In Table 2, a global (geocentric) scene is created with an image layer, an OAP3D, a CAD model, a text label and several view points which can be clicked to zoom to. In Table 3, the same scene content is created in a local (projected) scene.

Table 2. An osgEarth global scene ("Solar3D/bin/tests/solar3d_global.earth")

```
<?xml version="1.0" encoding="UTF-8"?>
<map name="Global OAP3D Demo" type="geocentric" version="2">
  <image name="arcgis-world-imagery" driver="arcgis">
    <url>http://services.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer/</url>
    <nodata_image>http://services.arcgisonline.com/ArcGIS/rest/services/World_Topo_Map/MapServer/tile/100/0/0.jpeg</nodata_image>
    <cache_policy usage="no_cache" />
  </image>
  <!-- <image name="readymap_imagery" driver="tms">
     <url>http://readymap.org/readymap/tiles/1.0.0/22/</url>
   </image>-->
   <!--<elevation name="readymap_elevation" driver="tms">
       <url>http://readymap.org/readymap/tiles/1.0.0/116/</url>
    </elevation>-->
   <annotations>
     <label text="OAP3D">
       <position x="-71.0589" y="42.3601" alt="0" srs="wgs84" />
       <style type="text/css">text-align:     center_center;
      text-size:      20;
      text-declutter: true;
      text-halo:      #777;
      text-bbox-fill: #00FF0033;
      text-bbox-margin: 3;
      text-bbox-border: #FFFFFFFF;
      text-bbox-border-width: 1;</style>
     </label>
     <model name="OAP3D">
       <position x="-71.0589" y="42.3601" alt="0" srs="wgs84" />
       <local_offset x="3024.08521" y="9106.22070" z="0" />
       <scale x="1.0" y="1.0" z="1.0" />
       <style>model: "../data/models/OAP3D.oap3d";
      model-heading:0.0;
      model-pitch: 0.0;
      model-roll: 0.0;</style>
     </model>
     <model name="CAD">
       <position x="-71.0589" y="42.3601" alt="0" srs="wgs84" />
       <local_offset x="0" y="1000" z="10" />
       <scale x="1.0" y="1.0" z="1.0" />
       <style>model: "../data/models/CAD/CAD.osgb";
      model-heading:0.0;
      model-pitch: 0.0;
      model-roll: 0.0;</style>
     </model>
   </annotations>
   <viewpoints time="1.0">
     <viewpoint name="OAP3D" heading="24.261" height="0" lat="42.3601" long="-71.0589" pitch="-21.6" range="2450" />
     <viewpoint name="CAD" heading="24.261" height="0" lat="42.3701" long="-71.0589" pitch="-21.6" range="2450" />
   </viewpoints>
</map>
```

Table 3. An osgEarth local (projected) scene ("Solar3D/bin/tests/solar3d_global.earth")

```
<?xml version="1.0" encoding="UTF-8"?>
<map name="Projected OAP3D Demo" type="projected">
  <options>
```

```
    <profile srs="+proj=utm +zone=19 +ellps=GRS80 +units=m +no_defs" xmin="291343" xmax="367343" ymin="4653876"
ymax="4722276" num_tiles_wide_at_lod_0="1" num_tiles_high_at_lod_0="1" />
  </options>
 <image name="arcgis-world-imagery" driver="arcgis">
   <url>http://services.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer/</url>
   <nodata_image>http://services.arcgisonline.com/ArcGIS/rest/services/World_Topo_Map/MapServer/tile/100/0/0.jpeg</nodata_image>
   <cache_policy usage="no_cache" />
 </image>
 <!-- <image name="readymap_imagery" driver="tms">
    <url>http://readymap.org/readymap/tiles/1.0.0/22/</url>
 </image>-->
 <!--<elevation name="readymap_elevation" driver="tms">
     <url>http://readymap.org/readymap/tiles/1.0.0/116/</url>
 </elevation>-->
 <annotations>
   <label text="OAP3D">
     <position x="329343" y="4688076" z="0" srs="+proj=utm +zone=19 +ellps=GRS80 +units=m +no_defs" />
     <!--<position x="-71.0589" y="42.3601" alt="0" srs="wgs84"/>-->
     <!--<position lon="-71.0589" lat="42.3601" alt="0" srs="wgs84"/>-->
     <style type="text/css">text-align:     center_center;
     text-size:      20;
     text-declutter: true;
     text-halo:      #777;
     text-bbox-fill: #00FF0033;
     text-bbox-margin: 3;
     text-bbox-border: #FFFFFFFF;
     text-bbox-border-width: 1;</style>
   </label>
   <model name="OAP3D">
     <position x="329343" y="4688076" z="0" srs="+proj=utm +zone=19 +ellps=GRS80 +units=m +no_defs" />
     <!--<position x="-71.0589" y="42.3601" alt="0" srs="wgs84"/>-->
     <!--<position lon="-71.0589" lat="42.3601" alt="0" srs="wgs84"/>-->
     <local_offset x="3024.08521" y="9106.22070" z="0" />
     <scale x="1.0" y="1.0" z="1.0" />
     <style>model: "E:/Code/Solar3D/bin/data/models/OAP3D.oap3d";
     model-heading:0.0;
     model-pitch: 0.0;
     model-roll: 0.0;</style>
   </model>
   <model name="CAD">
     <position x="329343" y="4688076" z="0" srs="+proj=utm +zone=19 +ellps=GRS80 +units=m +no_defs" />
     <local_offset x="0" y="1000" z="10" />
     <scale x="1.0" y="1.0" z="1.0" />
     <style>model: "../data/models/CAD/CAD.osgb";
     model-heading:0.0;
     model-pitch: 0.0;
     model-roll: 0.0;</style>
   </model>
 </annotations>
 <viewpoints time="1.0">
   <viewpoint name="OAP3D" heading="24.261" height="0" lat="42.3301" long="-71.0689" pitch="-21.6" range="2450" />
   <viewpoint name="CAD" heading="24.261" height="0" lat="42.34201" long="-71.0689" pitch="-10.6" range="1450" />
   <viewpoint name="OAP3D" heading="24.261" height="0" x="329343" y="4688076" pitch="-21.6" range="2450" srs="+proj=utm
+zone=19 +ellps=GRS80 +units=m +no_defs" />
   <viewpoint name="CAD" heading="24.261" height="0" x="329343" y="4689076" pitch="-21.6" range="2450" srs="+proj=utm
+zone=19 +ellps=GRS80 +units=m +no_defs" />
 </viewpoints>
</map>
```

5.3. Add image layers

Table 5. Add different types of image layers

Image Layer	XML Node
TMS	<image name="readymap_imagery" driver="tms"> <url>http://readymap.org/readymap/tiles/1.0.0/22/</url> </image>
GDAL	<image name="world" driver="gdal"> <url>..\data\world.tif</url>

	</image>
Bing Map	<image name="Bing imagery" driver="bing"> <key>your-api-key-here</key> <imagery_set>Aerial</imagery_set> </image>
Mapbox	<image name="mapbox_satellite" driver="xyz"> <url>http://a.tiles.mapbox.com/v4/mapbox.satellite/{z}/{x}/{y}.jpg?access_token=YOUR_TOKEN_HERE</url> <profile>spherical-mercator</profile> </image>
OpenStreet Map	<image name="osm_mapnik" driver="xyz"> <url>http://[abc].tile.openstreetmap.org/{z}/{x}/{y}.png</url> <profile>spherical-mercator</profile> <cache_policy usage="none"/> <attribution>©OpenStreetMap contributors</attribution> </image>
WMS	<!-- NEXRAD 45 minute RADAR returns --> <image name="nexrad45min" driver="wms"> <url>http://mesonet.agron.iastate.edu/cgi-bin/wms/nexrad/n0r.cgi</url> <format>png</format> <layers>nexrad-n0r</layers> <tile_size>256</tile_size> <srs>EPSG:4326</srs> <transparent>true</transparent> <cache_policy usage="no_cache"/> <altitude>20000</altitude> </image>
ArcGIS	<image name="arcgis-world-imagery" driver="arcgis"> <url>http://services.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer/</url> <nodata_image>http://services.arcgisonline.com/ArcGIS/rest/services/World_Topo_Map/MapServer/tile/100/0/0.jpeg</nodata_image> <cache_policy usage="no_cache"/> </image>

5.4. Add 3D models and labels

Models, labels and other annotations (circle, rectangle, triangle, line, custom polygon, etc.) must be added within an “annotations” node. The coordinates are provided in a “position” node either in WGS84 lat/lon or in projected xyz. Models can also, translated (“local_offset” node) scaled (“scale” node) and rotated (“model-heading”, “model-roll”, “model-pitch” in style node).

Table 6. Add different types of 3D models

```
<annotations>
  <label text="OAP3D">
    <position x="329343" y="4688076" z="0" srs="+proj=utm +zone=19 +ellps=GRS80 +units=m +no_defs"/>
    <!--<position x="-71.0589" y="42.3601" alt="0" srs="wgs84"/>-->
    <!--<position lon="-71.0589" lat="42.3601" alt="0" srs="wgs84"/>-->
    <style type="text/css">
      text-align:     center_center;
      text-size:      20;
      text-declutter: true;
      text-halo:      #777;
      text-bbox-fill: #00FF0033;
      text-bbox-margin: 3;
      text-bbox-border: #FFFFFFFF;
      text-bbox-border-width: 1;
    </style>
  </label>

  <model name="OAP3D">
    <position x="329343" y="4688076" z="0" srs="+proj=utm +zone=19 +ellps=GRS80 +units=m +no_defs"/>
    <!--<position x="-71.0589" y="42.3601" alt="0" srs="wgs84"/>-->
    <!--<position lon="-71.0589" lat="42.3601" alt="0" srs="wgs84"/>-->
    <local_offset x="3024.08521" y="9106.22070" z="0"/>
    <scale x="1.0" y="1.0" z="1.0"/>
    <style>
      model: "E:/Code/Solar3D/bin/data/models/OAP3D.oap3d";
```

```
      model-heading:0.0;
      model-pitch: 0.0;
      model-roll: 0.0;
    </style>
  </model>

  <model name="CAD">
    <position x="329343" y="4688076" z="0" srs="+proj=utm +zone=19 +ellps=GRS80 +units=m +no_defs"/>
    <local_offset x="0" y="1000" z="10"/>
    <scale x="1.0" y="1.0" z="1.0"/>
    <style>
      model: "../data/models/CAD/CAD.osgb";
      model-heading:0.0;
      model-pitch: 0.0;
      model-roll: 0.0;
    </style>
  </model>
</annotations>
```

5.5. Symbolize feature layers

A 3D city model can be created by applying an extrusion symbol to a feature layer of building footprints.

Table 7. Extrude building footprints

```
<model name="buildings" driver="feature_geom">

    <features name="buildings" driver="ogr">
      <url>../data/boston_buildings_utm19.shp</url>
      <build_spatial_index>true</build_spatial_index>
    </features>

    <layout>
      <tile_size_factor>45</tile_size_factor>
      <level name="default" max_range="20000">
        <selector class="buildings"/>
      </level>
    </layout>

    <styles>
      <library name="us_resources">
        <url>../data/resources/textures_us/catalog.xml</url>
      </library>

      <style type="text/css">
        buildings {
          extrusion-height:     3.5 * max([story_ht_], 1);
          extrusion-flatten:    true;
          extrusion-wall-style: building-wall;
          extrusion-roof-style: building-rooftop;
          altitude-clamping:    none;
        }
        building-wall {
          skin-library:    us_resources;
          skin-tags:       building;
          skin-random-seed: 1;
          fill:            #ffffff;
        }
        building-rooftop {
          skin-library:    us_resources;
          skin-tags:       rooftop;
          skin-tiled:      true;
          skin-random-seed: 1;
          fill:            #ffffff;
        }
      </style>


                    <!--Exclude certain buildings from being rendered b/c they will be replaced with geospecific buildings -->
                        <selector class="buildings">
```

```
            <query>
              <expr><![CDATA[ OBJECTID_1 <> 91506 and OBJECTID_1 <> 12921 and OBJECTID_1 <> 11460 and OBJECTID_1 <>
11474 and OBJECTID_1 <> 11471 and OBJECTID_1 <> 11439 and OBJECTID_1 <> 11432 and  OBJECTID_1 <> 91499 and OBJECTID_1
<> 10878 ]]> </expr>
            </query>
          </selector>

      </styles>

      <lighting>true</lighting>
   </model>
```

5.6. Add elevation layers

Table 7. Add different types of elevation layers

Elevation Layer	XML Node
TMS	<elevation name="readymap_elevation" driver="tms"> <url>http://readymap.org/readymap/tiles/1.0.0/116/</url> </elevation>
GDAL	<!--Load a folder full of terrain data as an elevation source--> <elevation name="terrain" driver="gdal"> <!--To load the files in a directory, just point the URL to a directory instead of a file--> <url>..\data\terrain</url> <!-- You can filter out what files you want to load by providing a semi-colon delimited list of files. Uncomment the following, for instance, to load only DTED levels 0, 1 or 2. If you don't provide an extensions list, the GDAL driver will try every file in the directory. --> <!--<extensions>dt0;dt1;dt2</extensions>--> <!--Tell the GDAL driver to just look for tifs--> <extensions>tif</extensions> </elevation>

6. Build From Source

Required build tools: CMake >= 3.15

Required dependencies: The easiest way to obtain these dependencies is to install using vcpkg, a C++ package management tool for Windows, Linux and MacOS. For Windows, a Visual Studio 2019 solution is available with all required dependencies included.

Table 8. Required dependencies and command lines to install from vcpkg

Dependency	vcpkg package
qt5-base >= 5.12	vcpkg install qt5-base:x64-window
osgEarth >= 2.10	vcpkg install osgearth:x64-windows
OpenSceneGraph >= 3.6.4	vcpkg install osg:x64-windows

References

1. GRASS GIS r.sun. Available online: https://grass.osgeo.org/grass78/manuals/r.sun.html
2. OpenSceneGraph. Available online: http://www.openscenegraph.org
3. osgEarth. Available online: http://osgearth.org
4. vcpkg. Available online: https://github.com/microsoft/vcpkg
5. OpenSceneGraph Plugins. http://www.openscenegraph.org/index.php/documentation/user-guides/61-osgplugins

Developer: Jianming Liang, jian9695@gmail.com