Intro to OASIS Baseline Scanpath Code

America Makes (ALSAM) project 3024

Updated August 12, 2020 to differentiate OASIS from ALSAM code

Contact: David Toledano, GE Research ([toledano@ge.com](mailto:toledano@ge.com))

Contents

[1. Overview 2](#_Toc48119005)

[Scope 2](#_Toc48119006)

[End-to-end process from STL files to physical parts 2](#_Toc48119007)

[Current status 3](#_Toc48119008)

[2. Quick start: download the precompiled code and run some examples 3](#_Toc48119009)

[3. Requirements 4](#_Toc48119010)

[4. Software locations 4](#_Toc48119011)

[OASIS repository on GitHub 4](#_Toc48119012)

[Slic3r 4](#_Toc48119013)

[5. Target folder structures 5](#_Toc48119014)

[Precompiled (executable) code 5](#_Toc48119015)

[Source code 5](#_Toc48119016)

[6. Additional documentation 6](#_Toc48119017)

[The multi-laser XML schema 6](#_Toc48119018)

[Setting up a configuration file 6](#_Toc48119019)

[Specifications and limitations 6](#_Toc48119020)

# 1. Overview

## Scope

The code described here converts CAD files in STL format into scanpath files in XML format. The XML scan files are intended for use by a LabVIEW-based machine controller to build parts using laser-melted metallic powder. Both source and compiled code are included

The desired parameters for each build are described by an Excel configuration file. Several sample configuration files (along with their required STL files) are included, which can used to evaluate the compiled code and XML scan format

## End-to-end process from STL files to physical parts

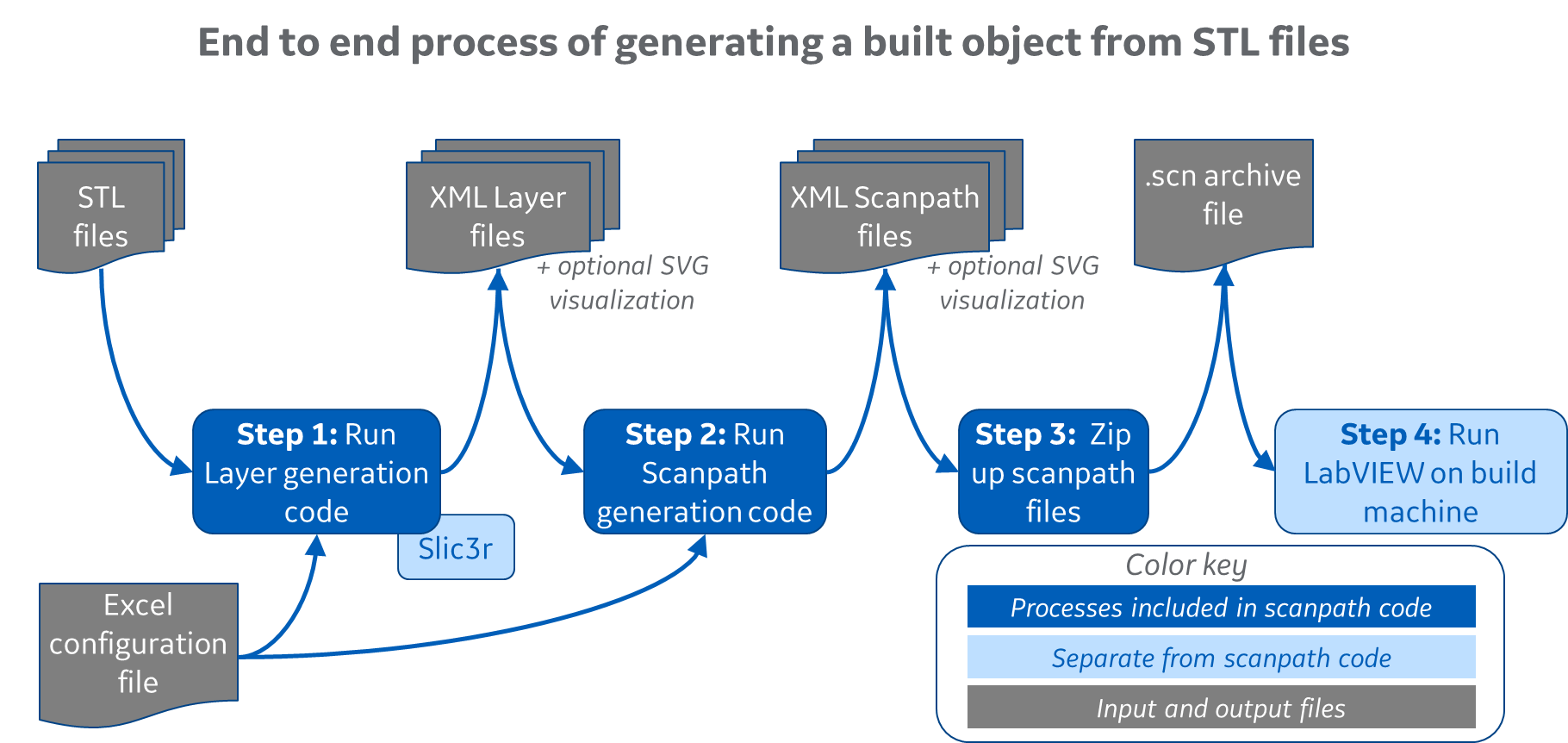
Creating a physical part proceeds in several steps as indicated in the list below and diagram that follows:

Step 1: Generate layer files (in XML layer format) based on STL files. One XML file is created per layer

Step 2: Generate scanpath files from the layer files. One scanpath file is created per layer

Step 3: Zip up the scanpath files and change the extension of the resulting archive to “.scn” to form a scan package for LabVIEW

Step 4: Run the ALSAM LabVIEW code to control a machine based on the given scanpaths



Steps 1-3 are executed by the current code, while step 4 is conducted by the forthcoming LabVIEW code

The layer files created in step 1 are not needed after step 2 is completed; the files can be deleted or retained to generate an alternate series of scanpath files. The code allows you to execute any combination of steps 1-3 with the understanding that you must have (or be creating) a set of layer files before scan files can be generated

In addition to XML outputs, you may optionally create SVG (structured vector graphics) files which can be used to quickly visualize layers or scanpaths in any web browser

## Current status

As of this writing the scanpath code on GitHub is functionally complete with respect to the XML schema dated March 3, 2020: all XML schema features may be generated via the current code and configuration parameters. The original PBFAM machine controller code is not compatible with the new multi-laser schema files

The code also utilizes a further refinement to the original Excel configuration file (version 3) which differs from the previous (version 2) file format. The code will flag an error if an old-format config file is used

# 2. Quick start: download the precompiled code and run some examples

1. Download the compiled binaries from <https://github.com/AmericaMakes/OASIS-Challenge-Baseline-Code-and-Models> via the green “Clone or Download” button. Unzip. The new folder can be named whatever you’d like (doesn’t have to match the GitHub folder name)
2. Download Slic3r version 1.3.0 from <https://dl.slic3r.org/win/Slic3r-1.3.0.64bit.zip>
3. Unzip the Slic3r package into the slic3r\_130 folder that appears in the Multilaser\_Precompiled-binaries\Executables folder created in the previous step. The slic3r executables and four subfolders should now appear in OASIS baseline precompiled binaries\Executables\slic3r\_130
4. Several examples are provided in the Project Examples folder, along with a blank template for your use. The outputs of scanpath generation are included for the examples so that you can view XML and SVG results and verify that your execution returns the same results
   1. Folder “A original examples – updated format” contains the original two examples in updated format:
      1. *OASIS Ex1 - many parts.xls* generates a build files populated with multiple 1mm x 1mm cubes, and demonstrates most features of the Excel configuration file and XML schema
      2. *OASIS Ex2 - coin.xls* generates build files describing a single 40mm-wide coin on top of circular supports. This build contains more detailed (interesting) scan lines but is still fairly compact in size and generation runtime
   2. Folder “B additional examples” contains two new examples:
      1. *OASIS Ex3 - larger parts.xls* generates a few parts of somewhat larger dimension (30-50mm wide, up to 60mm high) to evaluate several hundred layers at a time
      2. *OASIS Ex4 - singlestripes plus parts.xls* demonstrates the new single-stripe tab, which may be used to conduct bead-on-plate experiments for parameter development, in combination with an STL file containing markings and identifiers
5. To generate any of the examples, execute **createScanpaths.exe** in the Executables folder. You can also create a shortcut to this file and place it on your desktop or other convenient location
   1. Double-clicking on createScanpaths.exe will open a file-select window
      1. Navigate up and over to the Project Examples folder and select any of the Excel configuration files. Click cancel to quit without generating anything
      2. The select window will only permit you to select Excel files ending in .xls
   2. The code will quickly scan the config file for errors, such as missing or conflicting identifiers and values out of range
      1. If an error is found, the code will identify the error, write the information to an error-report file and then quit (after giving you a chance to read the text and press a key). The error report will be created in the same folder as the config file you selected
      2. If no errors are found, the code next determines whether any layer and/or scan outputs exist in the project output folder listed in the config file. This is used to determine which output-generation options (layer-only, layer-plus-scan, scan-only) are available
   3. You are then presented with a list of output-generation options. Input your selection (in upper or lower case) and press Enter. The options may include
      1. L = generate layers files only
      2. S = generate scan files only (only available if the project output folder already contains layer outputs)
      3. B = generate both layer and scan files (layers first, then scans, of course)
      4. C = cancel without overwriting or deleting anything
   4. If your choice involves re-generating or overwriting something which currently exists in the project output folder, you’ll be asked to confirm. Y or y (and Enter) will proceed, all else will quit/cancel
   5. The process will then create (or erase and recreate) the project output folder and/or layer and scan subfolders, and then proceed with file generation
   6. If your choice involves layer generation, the process proceeds in two sub-steps
      1. Slicing all the indicated STL files into layers, contained in one or more new SVG files
      2. Generating XML and SVG layer files (layer by layer) and place them into subfolders of the layer folder
   7. If your choice involves scan generation, the process iterates layer-by-layer until done and then (optionally) creates a zip file with the XML scan output
6. Before or after generating the examples, you may view the (pre-)generated results by navigating to any of the example folders, then into a project output folder, then into a LayerFiles or ScanFiles folder
   * XMLdir subfolder contains the scan files, which can be viewed in an XML editor such as Notepad++
   * SVGdir subfolder contains visualization files, which are not required for the actual build. Double-clicking on an SVG file will likely give you the option to view the file in a web browser (Chrome does this well). The SVG files display a fixed 2000 pixel-wide view of the populated build surface, which may omit small details and hatching that is present in the XML build files
7. The sample Excel files can be used as a guide to develop and generate your own build using the existing or alternate STL files. Both text and binary STL files are suitable and may be mixed in the same build

# 3. Requirements

The scanpath code was compiled to run on a 64-bit Windows PC. Executing the layer-generation code also requires the open-source Slic3r package (version 1.3.0). Creating a zip archive from the scanpath files requires a zip manager or 7zip

The C++ source code was compiled using Microsoft Visual Studio 2015 and the GitHub repository includes all solution files needed by Visual Studio. Visual Studio is not required to execute the compiled code. The code could be recompiled in an alternate development environment with a small amount of work, but does require Windows processes such as windows.h and msxml6.h. See the basline scanpath source code documentation for more info

# 4. Software locations

The majority of the scanpath codebase is custom open-source C++ available on the OASIS repository

Execution requires an additional open-source package, Slic3r, which is used to slice STL files into SVG layers. Slic3r may be downloaded separately or included in the America Makes repository; however, this package does not fall under the America Makes project and is licensed separately by Slic3r.org

## OASIS repository on GitHub

Source code, documentation, pre-compiled binaries and examples: <https://github.com/AmericaMakes/OASIS-Challenge-Baseline-Code-and-Models>

## Slic3r

The layer-generation code requires the open-source Slic3r package, available from [www.slic3r.org](http://www.slic3r.org)

Use the 64-bit package. Slic3r version 1.3.0 (the latest as of this writing) was used for development. Version 1.3.0 has a slightly different API (interface protocol) compared with version 1.2.9, was used for the original single-laser scanpath code. As a result, the Slic3r versions utilized by these two versions of America Makes code are not interchangeable

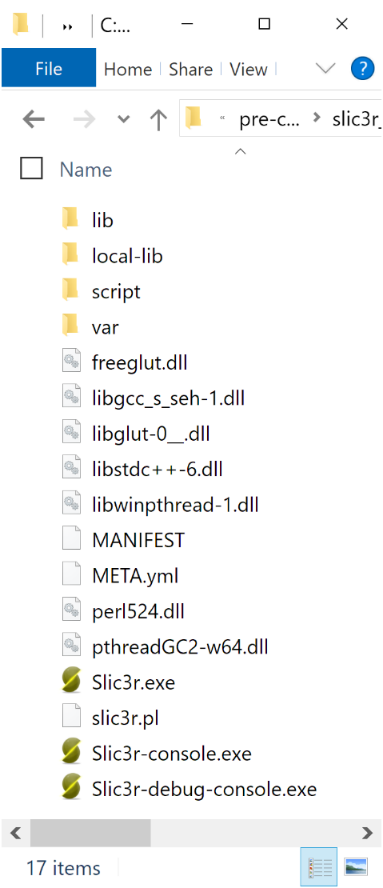
# 5. Target folder structures

## Precompiled (executable) code

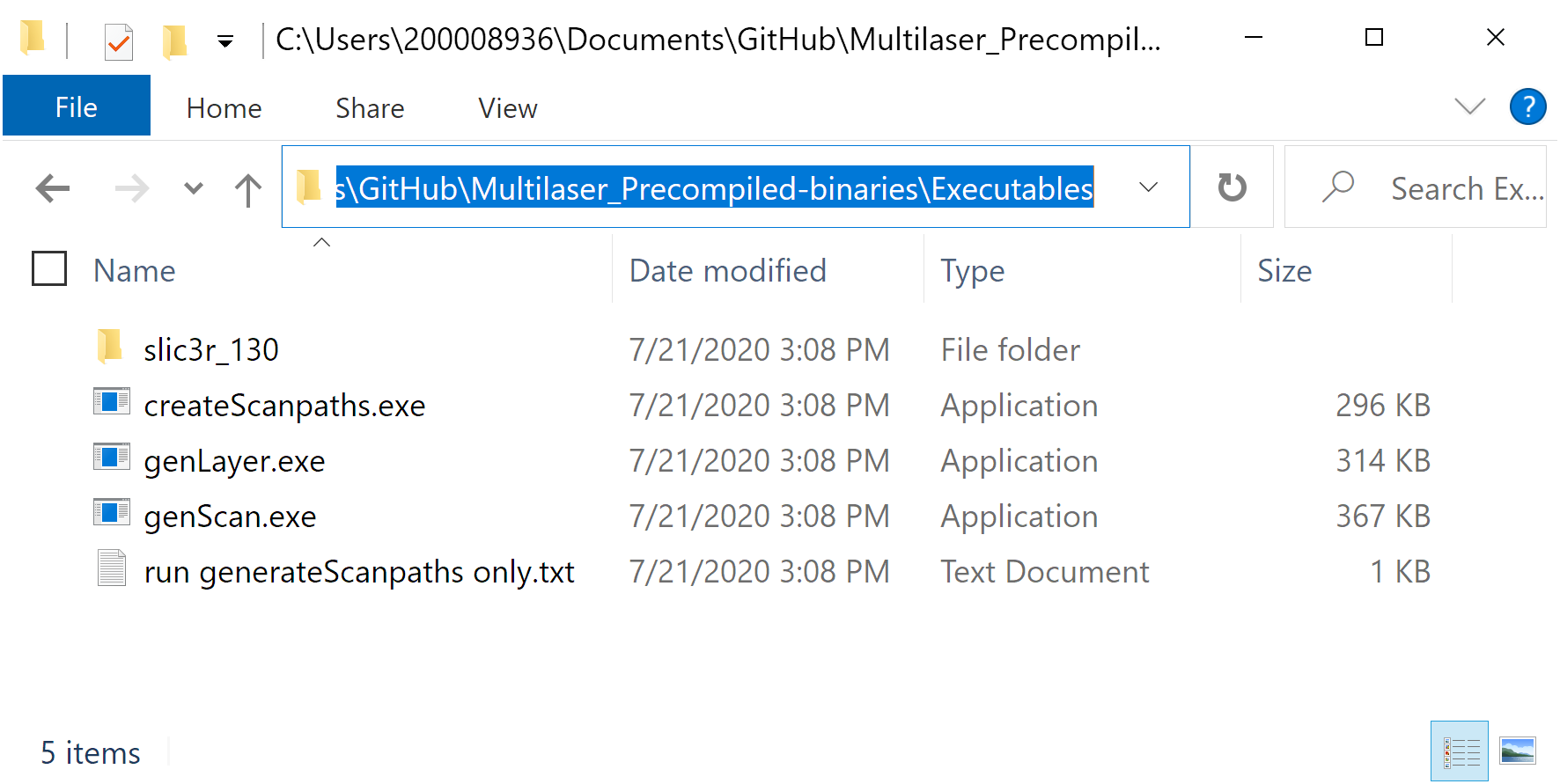
After downloading and unpacking the precompiled code as described in the QuickStart section above, the runtime code and all required elements should be contained in a single Executables folder. The code uses relative paths, so this runtime folder can be placed in any location on your machine. Slic3r 1.3.0 should appear in a subfolder named *slic3r\_130* as shown

Within the Precompiled-binaries folder, the following elements appear:

* Executables: contains the .exe files and a blank folder for slic3r download
* Project Examples: contains four sample projects. Your own projects can be in any folder; they don’t need to be contained within this folder

Each new project should be assigned a separate output folder (as named in tab 2 of the config file) which will contain subfolders for layer and scan files

Target structure of the Executable folder and slic3r\_130 subfolder



## Source code

Detailed information on the structure and operation of the source code are provided in a separate Word file; here’s a quick rundown:

C++ source code for layer and scan generation appears in a repository with one main folder (OASIS baseline source code) that contains a single Visual Studio solution, OASIS\_baseline.sln. Opening this file in Visual Studio will display several projects with their associated header (.h or .hpp) and code (.cpp) files. The projects are:

* createScanpaths – provides a basic user interface to select configuration files and execute layer and scan generation
* genLayer – code to generate layers from STL files. Called by createScanpaths (not executable in isolation)
* genScan – code to generate scanpaths from XML layer files (output of genLayer, above). Also called by createScanpaths and not executable in isolation

# 6. Additional documentation

## The multi-laser XML schema

The current (and assumed final) version of the XML scanpath schema is documented in **ALSAM3024 multiLaser XML schema 2020323.docx** which is available in the documentation folder within the scanpath source code folder

## Setting up a configuration file

See the User Manual, and use the provided examples as guides

## Specifications and limitations

The scanpath generation code is capable of generating all features of the new multi-laser XML schema. Here are a few additional nuances and figures:

* Dimensions
  + *Millimeter*-denominated values include the desired layer thickness via tab 2 of the configuration file, X/Y/Z positioning of each part via tab 6, scan velocities listed on tab 3 and contour/hatch spacings and offsets on tab 5
  + *Micron*-denominated items include spot size on tab 4
* SVG coordinates
  + SVG visualization files created by the code display the build area, centered on areas containing parts
  + Positive X proceeds to the right and positive Y proceeds downward
* Build plate dimensions and coordinates
  + The scanpath code is not aware of the dimensions of the target build plate, so there is no check that all parts fall on the plate
  + The code is also not aware of the plate location of coordinate 0,0 for the target scanning system (whether at upper left or in middle of plate, for instance), so it is up to the user to determine this and offset each part appropriately via tab 6 of the configuration file
  + Some trial and error may be required to determine the generated X,Y coordinates of a particular STL file when placed at scanpath coordinate 0,0. This will be resolved in further refinement of the scanpath code and documentation
* Layer thickness
  + Layer thicknesses from 0.001 mm (1 microns) to at least 5 mm can be generated by the current release of the Slic3r package
  + The minimum layer thickness that has been extensively tested is 0.01 mm (10 microns). Even smaller values are likely feasible, but 0.01 mm should be sufficient for all current build systems
* Maximum number of layers
  + This is not known, but builds containing several thousand layers have been generated with no issues
  + To reduce generation speed for very large builds, disable or reduce the frequency of SVG-file generation via tab 2 of the configuration file