LSIE Tool Workflow

This document describes how to author and convert content for use in the SCCN LSIE experiment. If you are authoring for a different study note that particular details about naming of objects and experiment script parameters below are going to be different for your study script. The sections referring to CityEngine are specific to the way in which cities happen to be generated in the LSIE experiment -- you can safely ignore this if you are using hand-created environments in your study. All sections about converting 3d content will apply more or less directly to other SNAP experiments with 3d content.

Complementary documentation on the Panda3d asset pipeline is available at: http://www.panda3d.org/manual/index.php/Main_Page (section Panda3d Tools at the bottom)

Programs used in our workflow:

* 3ds max 2010 (64bit); other versions may work but newer versions may export files with syntax that is not recognized by the Panda3d toolchain (Maya and Softimage|XSI 2010+ may also work since they use the same exporter backend)
* CityEngine 2010 (64bit); only for procedural city generation if desired
* Photoshop CS5 (for texture editing); any other image editing software, e.g. Gimp, will work
* GoldWave (for sound editing); free

Useful Guidelines for Content Editing:

- ideally you have a directory where you keep all your content in its various stages
- keep a directory for the original downloaded assets (.zip archives etc) that you started with
- keep a directory with the source data (in the native format of your content-creation program); ideally this has a sub-directory per asset (including the textures, 3d models, material files etc) this should hold the last version of your assets before you exported them into SNAP
- keep a directory where you convert your content (containing export/interchange formats like .dae, intermediate files like .egg and final outputs like .bam and .dat)
- keep a directory where you store all the final content that goes into the game
- try to be consistent with naming of certain kinds of assets (like the terrain or the city) and version the file names; it is advisable to give the original downloaded asset a new (or second) name that you consistently refer to
- avoid using different texture files with the same name for different assets -- the content script (fix_dae.py) currently strips off any directory parts from the texture files, so textures are all in a large pool as far as the SNAP runtime is concerned.
- try to avoid using models with large numbers of polygons (as they slow down the game); when obtaining assets, try to get game assets.

Content Pipeline -- Creating a new world (if using CityEngine):

- obtain a heightmap, colormap, and (optionally) obstacle map (black/white; white=admissible)
- make sure that these maps contain an area where a city can be generated in their ***center***
- make sure that the heightmap is *very* smooth in this area (smooth out in photoshop, do not store as .jpg (due to compression artifacts) but as .png
- open CityEngine, open the wizard, create a new city, load the appropriate maps, select the city size that you need (note: medium and larger are problematic when the conversion pipeline includes 32-bit apps that can't handle these sizes)
- select and delete portions from the city that look too artificial (usually outskirts)
- select all (Ctrl+A)
- go to Export / "Export models of selected shapes"

Export from CityEngine to 3ds max:

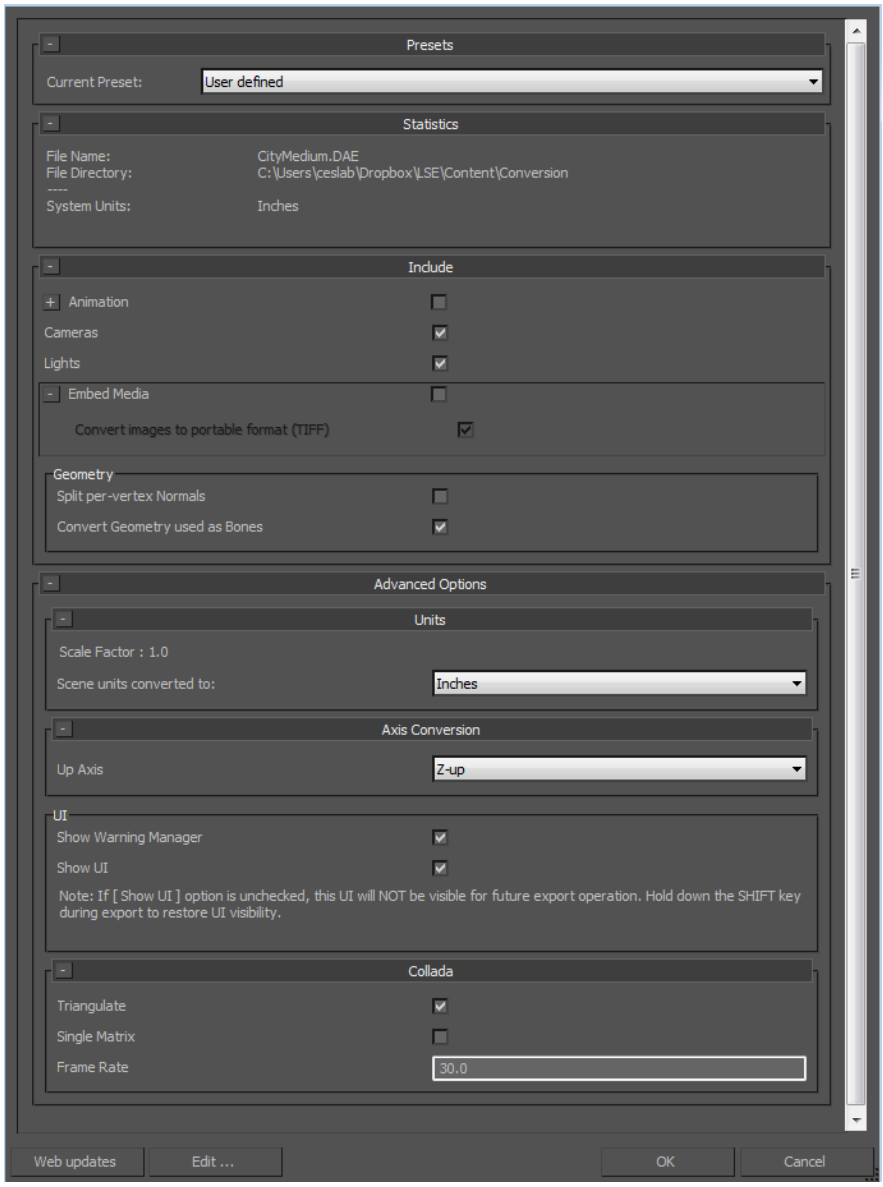
- select as output format FBX
- as preset use Deep Exploration (or along these lines)
- take note of the file name and path
- export
- import into 3ds max
- edit as necessary (for example, add marker objects in various locations with special names)

Notes on 3ds max city content naming:

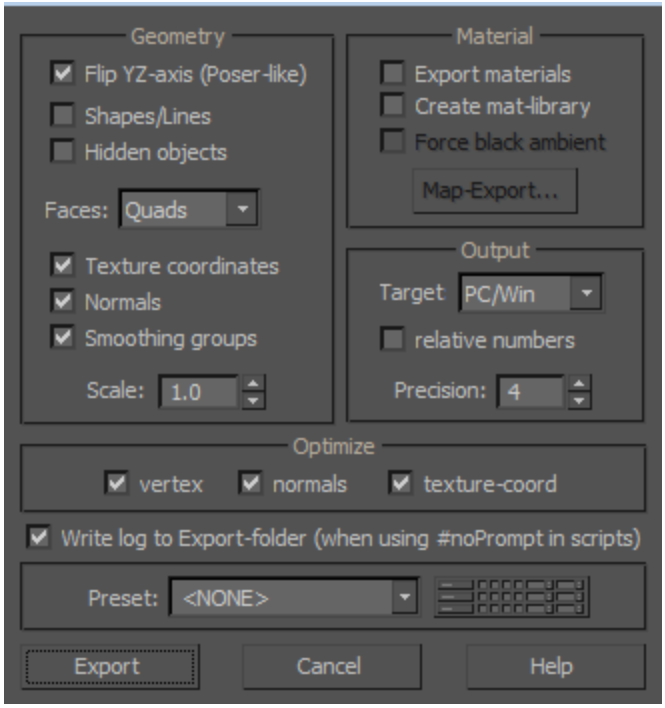
- there can be multiple starting locations in the map; if so:
  - each starting location is a group object with the name ***StartPosXX*** (XX in 1...n)
  - the player objects, truck object and whatever else belongs to the starting location should be grouped under that group (with that name)
  - only one of them will be retained at random, all others will be deleted by the LSE game
- the player objects are named starting with "***PlayerA***" and "***PlayerB***", and each as a subordinate camera object that is named starting with "***Cam***". Make sure that the coordinate system of the cam is oriented such that y is forward, z is up and x is right (this is the Panda3d camera standard); the 3ds max camera will point down
- the truck is called "***Truck***" (this is taken as the "hotspot" location during the secure-the-perimeter mission)
- if the truck model is not reachable from other places of the world (e.g. due to a barrier or the model's own size) one may place a special waypoint as an extra object in the startpos group which indicates the way-point position for closed checkpoint paths going through this startpos; this object is named ***WaypointXX***
- there may be checkpoints with the name ***CheckpointXX***
  - these can have any geometry (e.g. Cylinder), but will not be rendered; only the pivot location is taken as the checkpoint
  - however, the game currently relies on auto-generated checkpoint routes unless the variable (in LSE) named use_manual_checkpoints is set to True

Exporting world environments from 3ds to Panda

- export as Collada (DAE) using the following settings:



- if you have checkpoint geometry or other invisible geometry, temporarily remove it
- if you want to exclude terrain from being used for checkpoint placement and agent navigation, remove it as well.
- export as .OBJ using the following settings:



Conversion Option a): using automated converter script

- open the command line
- cd into the LSE/Tools directory
- run the tool convert_world.bat with the path to the .dae file (but omitting the .dae extension) as argument; note that the script assumes by default that you have a folder C:\DEVEL\Panda3d-1.8.0-64bit that contains a 64-bit build of panda; you may change the path in the batch file or comment the line if you are generating sufficiently small assets -- but note that the conversion process uses easily an order of magnitude more memory than running the final content in-game
- wait until all files, including the .dat file are produced in the same folder as the .dae
- now copy the .bam file, the .dat file and all the textures to a folder that is on Panda's search path (for LSE this would be SNAP/src/studies/LSE/media); if you retain the file, also save a copy in your ForProduction folder for later reference

Conversion Option b): by hand

- open the .dae file with a professional-quality text editor (e.g., Notepad++)
- search and replace any occurrence of "file://c/Users/.../CityEngine/.../models/" (or whatever the path to your textures is) by "" (i.e. by nothing), and save
- start command line and cd into the appropriate directory
- if you are using large files you probably want to use the 64-bit version of Panda3d on the path to do this, run the following line (depending on where you 64-bit build is located; note: you can get it from SCCN, just email christian):
- set PATH=C:\DEVEL\Panda3d-1.8.0-x64-LSL\bin;%PATH%
- run `dae2egg filename.dae -o filename.egg`

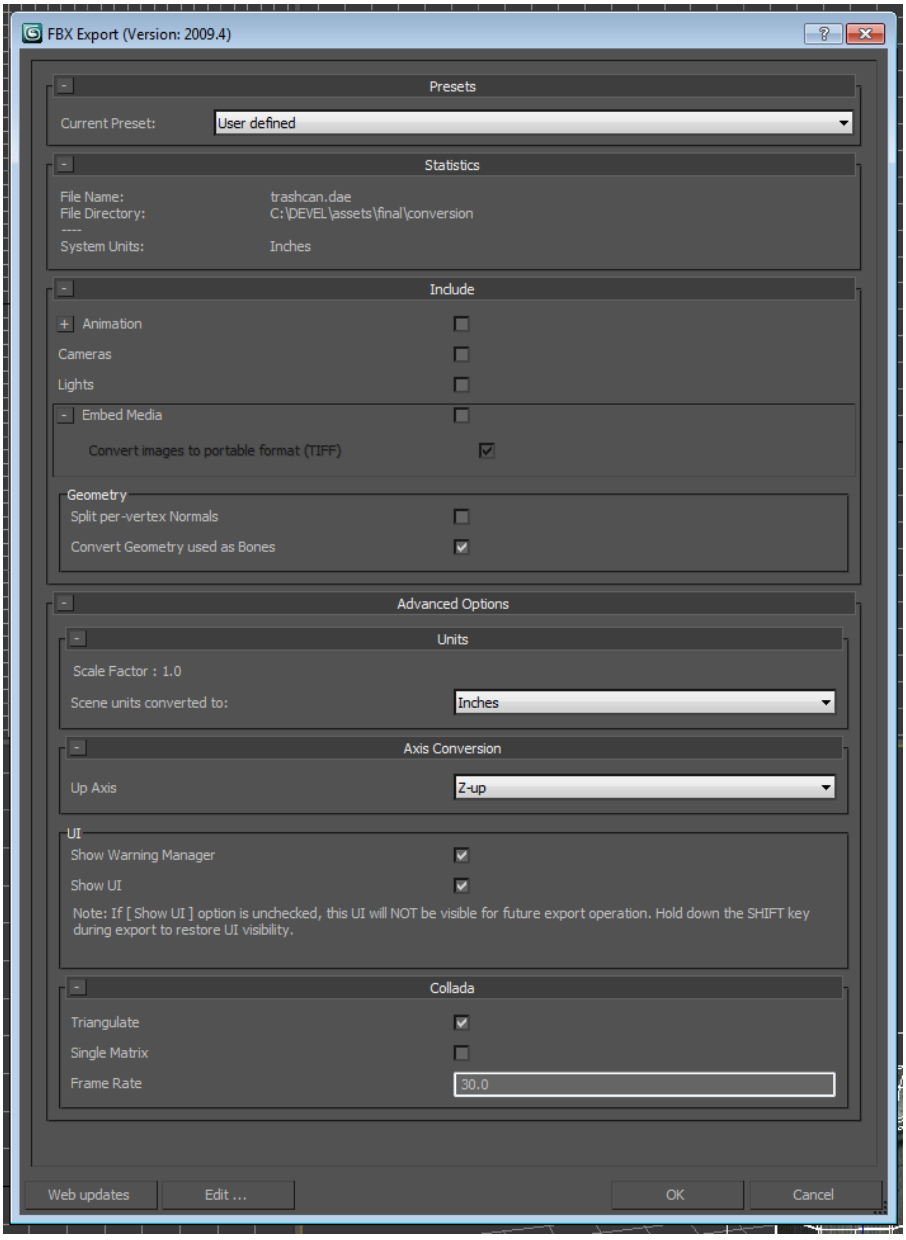
(make sure that you have enough memory available)
- run `egg2bam filename.egg -o filename.bam`
(make sure that you have enough memory)
- run `Recast.exe filename.obj`-- note that it takes a while until the app is done
- now copy the .bam file, the .dat file and all the textures to a folder that is on Panda's search path (for LSE this would be SNAP/src/studies/LSE/media); if you retain the file, also save a copy in your ForProduction folder for later reference

Importing Terrain as Heightmaps:

- it is advisable to import terrain in this way to save memory and graphics bandwidth
- when using heightmaps in Cityengine to constrain city placement, the heightmap and color map needs to be flipped vertically to work with the GeoMipTerrain in Panda
- the heightmaps need to be scaled to 1025x1025 (or 2049x2049...) to work properly
- note that CityEngine does a very poor job at conforming the city to the underlying terrain: usually the city will float some 10s of meters above the terrain and when aligned, the terrain will stick out in lots of places -- ideally the area where the city is generated is essentially absolutely flat

Importing miscellaneous 3d assets (models):

- edit the asset in 3ds max as necessary:
  - remove unnecessary parts (e.g., light sources, cameras)
  - ensure that the object is correctly centered (for physical objects around the center of mass and for street assets such that the origin is on the ground)
  - consider setting the right unit to avoid having to do it in code (but it's good practice to plan for error and, e.g., have the script read the model + scale factor from a text file anyway)
- export as Collada (DAE); we use the following settings in our exporter:



- copy all dependent textures into the same directory
- run the convert_model.bat script

Placing Content in the SNAP project folder:

* the best place for study-specific media files is SNAP/src/studies/LSE/media/
* just copy all textures and .bam and .dat files in this location for them to be found by the experiment script; try to avoid sub-directories for textures unless you adapt fix_dae.py to place textures in certain sub-directories (but note that you then need the same directory tree also in your Conversion folder when running the converter tools)

Editing the LSE Experiment script to refer to new content:

* the most direct way is to edit the SNAP/src/modules/LSE_GameServer.py file and change the respective file names. The relevant lines in the current code are:

```
self.world_types = ['CityMedium2']              # refers to the .bam file
self.terrain_types = ['LSE_desertplains_flat']  # refers to the color/height maps
                                                # (see current file naming scheme!)
self.hostile_filename = 'media/hostile_drone-anim.bam'
self.friendly_filename = 'media/r5_rev.bam'
```

* a fraction of asset files is referred to in text files (for editable lists of objects); these include:
  src/studies/LSE/media/objects_with_labels.txt (the curbside object file names)
  (the file itself is configurable in the LSE_GameServer.py)
* to test assets without changing the game code you can also create or edit a .cfg file that you use to run your experiment; this way you can keep your testing independent of other ongoing development: See existing .cfg files in src/studies/LSE/ for examples. To overide a game parameter, just add lines as below to the .cfg file:

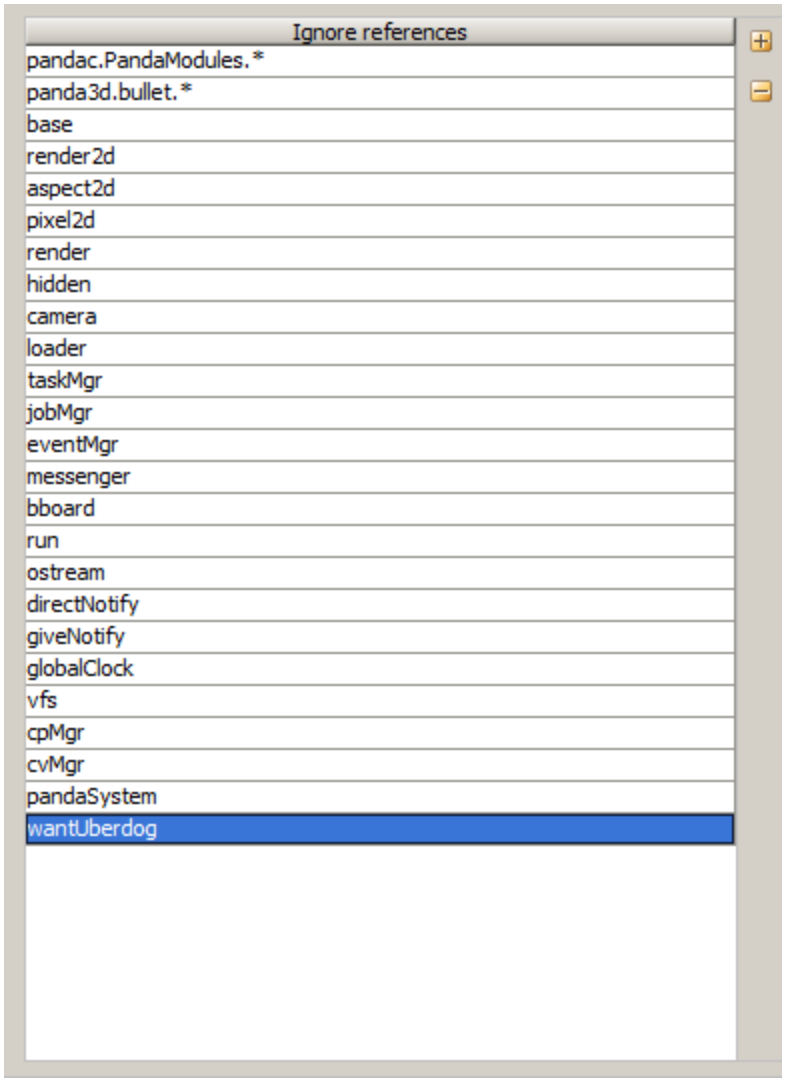
```
world_types = ['MyCustomCity']
friendly_filename = 'media/r5_rev.bam'
```

Testing Content - best practices:

* it is usually fastest to run locally and do a few quick checks
* watch out for incorrect coordinate units (the model is huge or tiny) -- most common unit mismatches arise between: meters, inches, and centimeters
* watch out for missing textures (parts of your model show up as gray)
* watch out for incorrect coordinate axes (your model is rotated or mirrored)
* watch out for incorrect coordinate origin (the model floats above the ground, or is sunken into the ground, or rotates about a point that is not the center of mass)

Setting up the PyCharm IDE for use with SNAP:

* create a new project, set location to the SNAP folder (the one that contains the src folder)
* select that sources from an existing project should be imported (this is a prompt IIRC)
* to stop PyCharm from underlining builtin symbols of Panda3d, go to File / Settings / Inspections / Unresolved references
and under Ignore references (bottom right panel), add the following names:



- note that PyCharm needs a lot of memory to scan large sources files for syntax errors on win32 you will likely not be able to get such error messages