Project acronym: POPART Project number: 644874 Work package: Integration Deliverable number and name: D4.1: API Definition	Title: Define component APIs Work Package: WP4 Version: 1 Date: 31/03/2015
	Author: Fabien Castan
Type: [X] Report [ ] Demonstrator, pilot, prototype [ ] Website, patent filings, videos, etc. [ ] Other	Co-Author(s): Gunleik Groven To: Albert Gauthier, Project Officer
Status: [ ] Draft [ ] To be reviewed [ ] Proposal [X] Final / Released to EC	Confidentiality: [X] PU – Public [ ] CO – Confidential [ ] CL - Classified
Revision: Final	
Contents: Description: POPART develops components of the visualization system that allow for “drop-in” replacements of algorithms to trade computing speed for accuracy and quality. The APIs of these components are fixed early in the development phase of POPART to ensure that development can proceed although the internals of components are still under investigation. These APIs become part of POPART’s open source release to ensure forward compatibility.	

Introduction

The ambition of this work package is to design the workflow and dataflow of the POPART Previz solution. It includes an early description of the structure and applications of the technical components of the POPART system. This will define future data structures and how APIs are connected. Obviously, it is not possible to define rigidly how all interfaces between components should be implemented and this is outside the scope of this deliverable. The purpose here is to give a concrete global view of all the components of the project and how they will interact. That will ensure the partners are aligned in their development efforts.

On-set workflow

The shooting step is a critical step in the creation of a movie. The daily budget is huge, the planning is very tight and defined in details but it remains unpredictable. Under these conditions, it’s really challenging to create a software for Previz. It is not a standard software with one software on one computer with one user. Here we have a lot of hardware, many users, many software that need to communicate together and the whole result of that could be a bottleneck for the whole team on-set (about 50 people).
So one key element is to clearly define the different users of the solution, the different kind of interactions they will have with the software and between them.

The relevant roles for Previz on a film set are:

- **Director** - The director will need to visualize the different takes of the final shot at any moment.
    - Need Live final rendering & deferred final rendering
- **Camera Operator** - The camera operator will always need to receive the live camera composited with the virtual world in the viewfinder. If the camera tracking fails, we need to detect it and send the original live stream to limit the disruptions.
    - Need Live final rendering
- **CG supervisor** - The CG supervisor is responsible for the previz servicing. He will discuss with the Director and will coordinate the Previz Artists work with the shooting needs.
    - Doesn’t use any software directly but interacts with all people involved.
- **Main Previz Artist** - The main Previz Artist will manipulate the main software to provide live feedback and deferred rendering if needed. He will relaunch the camera RIG calibration if we need to modify it from one shot to another.
    - Update the POPART database with new pictures, check the result, remove problematic points and manage the database versions (could be a separate person).
    - Connect all the hardware (the main and witness cameras to the main computer).

    - Check if we have the focal lens information in the stream or add and calibrate C-Motion.
    - Change the orientation of the cameras RIG, from one shot to another, to ensure the camera tracking and then relaunch the camera RIG calibration.
    - Check the camera tracking.
    - Choose the 3D scene and the 3D garbage mask scene.
    - Configure the keying and grading.
    - Create shots and takes. Trigger the records and add comments.
- ***Second Previz Artist*** - This optional second Previz Artist will be required on complex shots to modify the 3D scene on-set. This person needs to work on the current or the next shoot. This work can be time consuming and needs a lot of testing, so this person needs to be independent from the on-set team to avoid becoming a bottleneck. He needs to show his work to the CG supervisor and provide the result to the Main Previz Artist when it is validated.
    - Edit the 3D scene of the current shot.
        - Need a 3D Editing Software with live camera pose feedback.
    - Prepare the scene for the next shot.
        - Need a 3D Editing Software.
    - Need to provide the new 3D scene versions to the Main Previz Artist.
- ***Post-Production Artist*** - This optional person can make some post-production online. This can be useful to provide deferred visualization of really complex VFX (fluids interactions, virtual actors, etc.). Or on the other hand, to provide the final compositing online for really small budget productions (like TV shows).

That defines the different components:

1. **POPART Database Tool**

A software that can receive still images over Wifi to create and update the 3D reconstruction on-set. It can be used during the preparation stage or during the shooting. It can also be used for 3D reconstruction without previz.

2. **HAL Hardware and Software**

This component receives 3 synchronized videos streams and lens information and streams that over the network.
There is no UI on this tool, it is completely controlled by the *POPART cockpit*. It may be useful to be able to control it from a small device like a smart phone to use it outside of the previz context.

3. **POPART Cockpit**

This is the main tool that has access to all the database versions (provided by the *POPART Database Tool*), all the 3D scenes versions (provided by the “3D Editing Software Integration”) and receives the live data from the *HAL*.
It will record all the data, compute the camera tracking and stream the camera pose over the

network. It also has the control on the *POPART Viewer* and on the *POPART Deferred Renderer*.

4. **POPART Viewer**
   - **Realtime**

It receives a 3D scene and the live camera pose from the *POPART Cockpit* and display the live 3D rendering, keying, grading and compositing. It also records the result on disk.
There is no UI on this tool, it is completely controlled by the *POPART cockpit*.

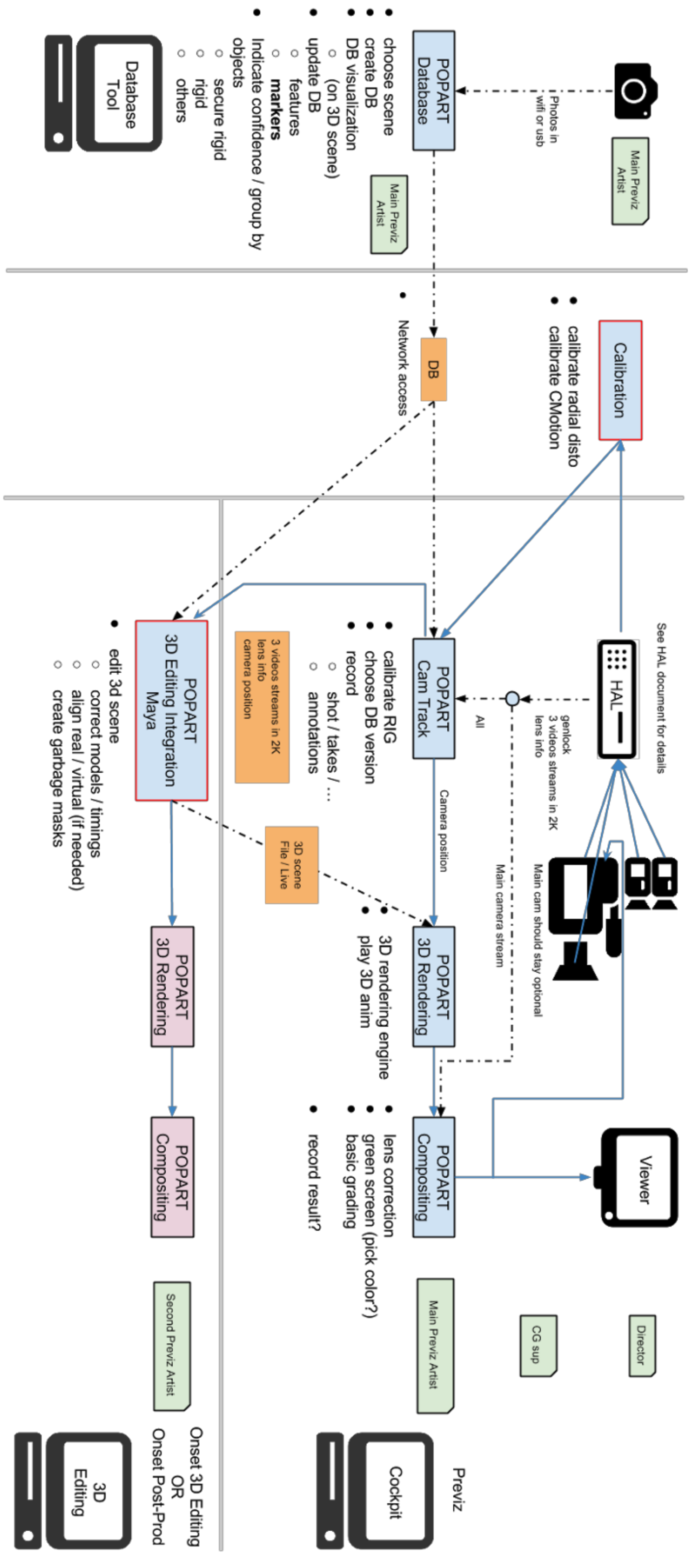
   - **Deferred Renderer**

Batch process that receives the 3D scene and a camera motion from the *POPART Cockpit* and generates a movie with a good quality 3D rendering, keying, grading and compositing.
There is no UI on this tool, it is completely controlled by the *POPART cockpit*.

5. **3D Editing Software Integration**

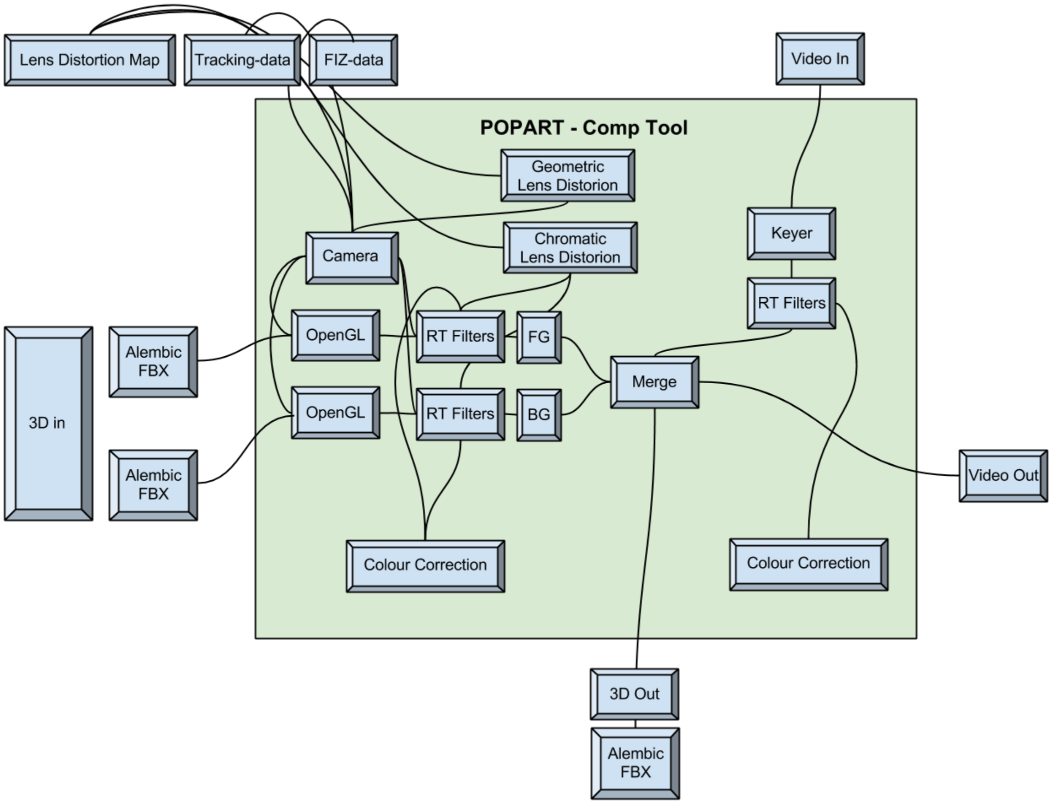
This plugin receives the Camera Pose in Live and will allow the Second Previz Artist to provide new versions of the scene to the *POPART Cockpit*.
This software will also be able to control an additional *Viewer* or *Deferred Renderer*.

Overview of the API structure



On-Set workflow, to describe API components

Workflow examples - background, foreground and video layer



Generic 3d application, working with interchange formats

Off-set workflow

The POPART Previz Solution stores information for all takes of all shots.
Typically that would be a bundle of:

- Recorded focal and radial lens distortion map
- Link to distortion maps per shot basis
- Main-camera proxy
- Witness camera video
- FIZ data (Focus, Iris and Zoom)
- Additional information from the on-set team:
    - Compositing parameters
    - Annotations
    - 3D scene modified on-set
    - Right version of the database
        - Geometrical markers setup (if applicable)
    - Live camera tracking
    - On-set composited shot (shown to the Director)

Synchro script

Few weeks later, the Post-Production team will receive the selected takes from the Video Editing in full resolution. So we need to match all our previz information with these movies (using the naming and timecode information).

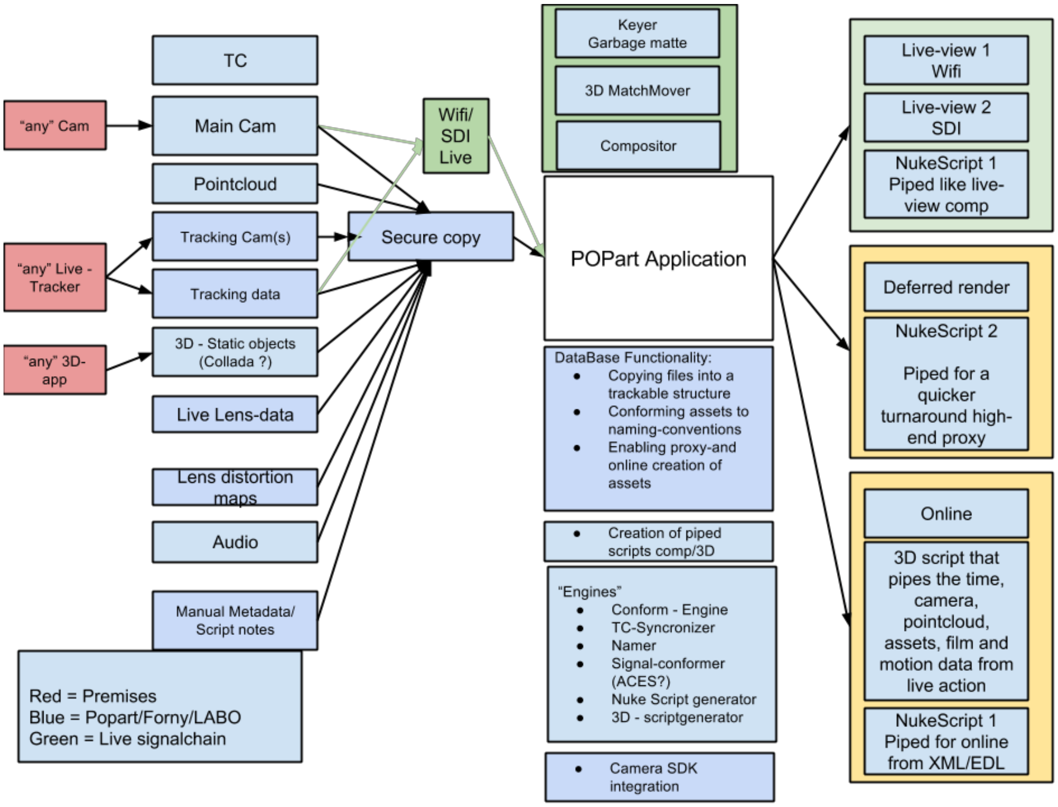
Export to standard Camera Tracking Softwares

Then we should be able to automatically re-compute the camera tracking without real-time constraints. In some challenging cases, this fully automatic camera tracking will not be good enough. So we will also provide exporters to standard Post-Production camera tracking solutions:

- 3DEqualizer
- PFTrack
- SynthEyes

Dataflow

Below follows an overview of the planned inputs and outputs of the workflow as in the POPART project:



Global dataflow

The solution is constructed in such a way that components in the workflow could be replaced with 3rd parties during the project. And at the end customer will be able to adjust the workflow to their needs.

Input sources:
For 3D elements

For the exchange format, we will use Alembic (.abc files). Alembic is an open source exchange format that is becoming the industry standard for exchanging animated computer graphics between content creation software packages. Alembic bakes any rig, constraints, deformation to the vertices and save them as is. It has a mechanism to instantiate copies and minimizes memory spending.
The main alternative is FBX which is a more generic 3D scene format that claims to support all 3D descriptors but its unstable support into major 3D software has resulted into troubles in production. So Alembic is getting more and more preferably used for stability reasons.
We will provide a full integration into Maya (Autodesk) and may provide additional support in the future (like Lightwave from NewTek or Cinema4D from MAXON). We hope to build the architecture in such a way that most 3d applications can be supported down the line.

Lens data:
To get accurate Focus, Iris and Zoom data, we need to read out the data either from an automatical system integrated in lenses (like ARRI LDS) or from a calibrated FIZ unit.

FIZ systems:

- ARRI
- C-motion
- Preston
- RED

Direct data from electronically equipped lenses:

- i-data
- ARRI LDS

Post-Production Camera Tracking softwares
(Creating automatic scripts for conformed calls for tracking data)

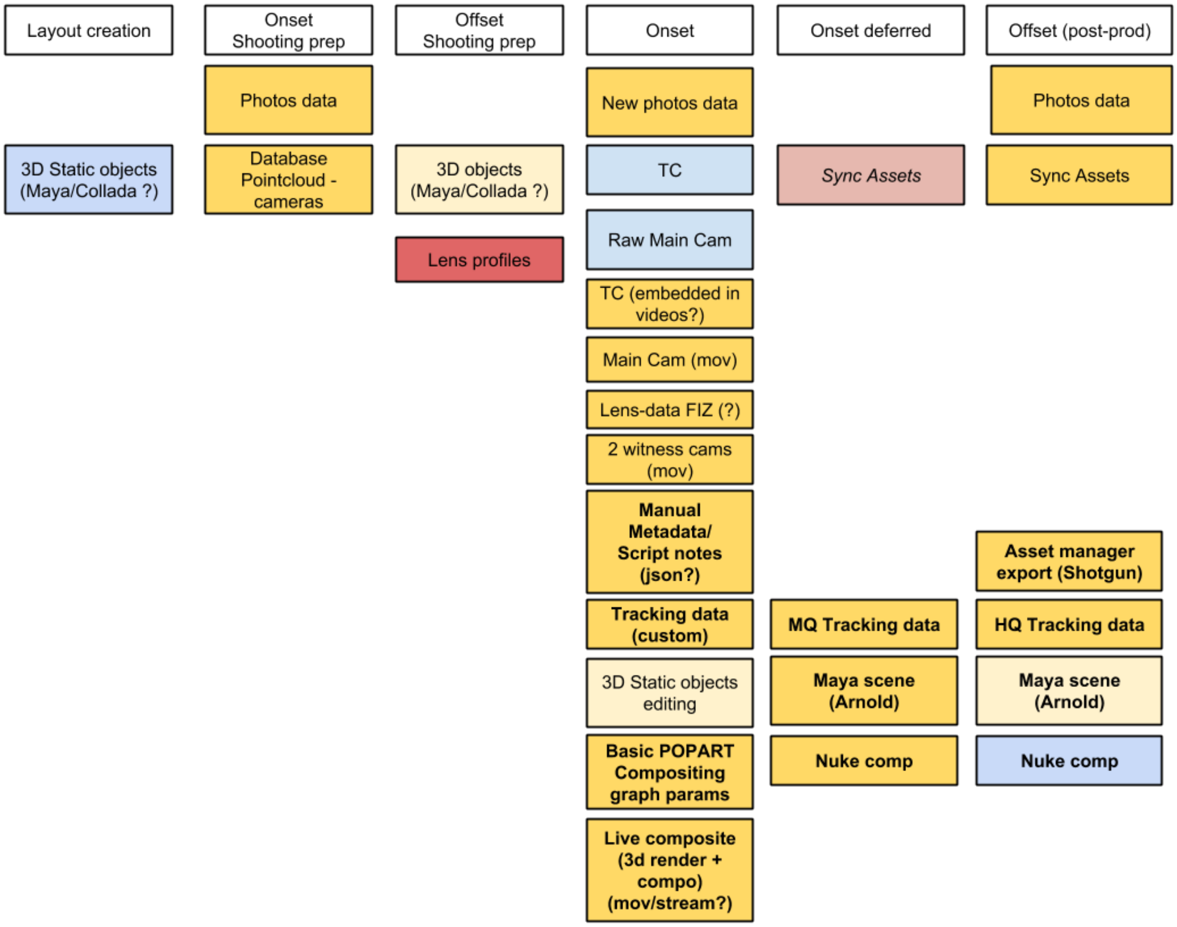
- 3D equalizer
- PFtrack
- Syntheyes

Compositors (for finishing):
Exporting finalized piped comps for Nuke (The Foundry). In the future, other comp software could be considered like After Effects (Adobe) and Fusion (Eyeon / Blackmagic).

Piping live 3D data from POPART-system for Maya (Autodesk). In the future, other 3D software could be considered like LightWave (NewTek).

Exporting 3D project (tracked and with 3D, 2D, tracker and pointcloud assets to Alembic.

Mapping between different production phases and data types:



Layout creation	Onset Shooting prep	Offset Shooting prep	Onset	Onset deferred	Offset (post-prod)
	Photos data		New photos data		Photos data
3D Static objects (Maya/Collada ?)	Database Pointcloud - cameras	3D objects (Maya/Collada ?)	TC	*Sync Assets*	Sync Assets
		Lens profiles	Raw Main Cam		
			TC (embedded in videos?)		
			Main Cam (mov)		
			Lens-data FIZ (?)		
			2 witness cams (mov)		
			Manual Metadata/ Script notes (json?)		**Asset manager export (Shotgun)**
			Tracking data (custom)	**MQ Tracking data**	**HQ Tracking data**
			3D Static objects editing	**Maya scene (Arnold)**	**Maya scene (Arnold)**
			Basic POPART Compositing graph params	**Nuke comp**	**Nuke comp**
			Live composite (3d render + compo) (mov/stream?)		

Production phases and datatypes

File-structures and naming conventions.

1. File structures

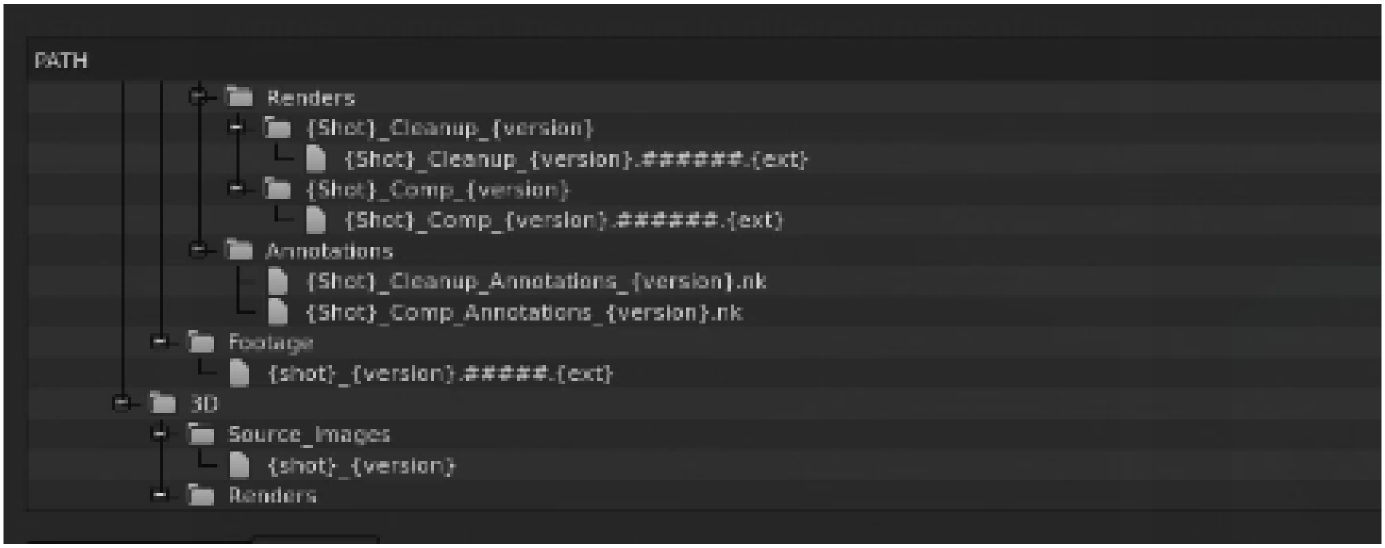
All data onset are organized directly during the shooting. Then script-notes are used for linking shots with episodes and scenes. Our challenges are that we create a lot of additional assets and we also have pre-generated assets that need to be synchronized with live on-set generated assets.
The pre-generated assets are:

- Point cloud
- Marker setup
- 3D asset(s)
- Lens distortion maps

2. Naming conventions

A POPART shooting day structure might look like this with two groups of assets, the “live”, and the corresponding pre-generated assets with the right naming convention. First group is Live, second is copied to the file-bundle

The project should fit the existing team workflow. So the POPART solution will use a template description to generate the data. It is similar to solutions provided by Nukestudio (The Foundry) and Tank (Shotgun Software).



```
PATH
        Renders
            {Shot}_Cleanup_{version}
                {Shot}_Cleanup_{version}.######.{ext}
            {Shot}_Comp_{version}
                {Shot}_Comp_{version}.######.{ext}
        Annotations
            {Shot}_Cleanup_Annotations_{version}.nk
            {Shot}_Comp_Annotations_{version}.nk
    Footage
        {shot}_{version}.#####.{ext}
3D
    Source_Images
        {shot}_{version}
    Renders
```

NukeStudio (The Foundry) file exporter example.

The default template will look like this:

- {day}
    - {seq}
        - {set-name}.####.popartdb
            - *Database should be global to the sequence but contains many versions to deal with changes onset.*
        - *lens distortion maps for each optic*
        - {shot}
            - {take}
                - Acquisition
                    - mainCamera.####.mov
                    - witnessCameraA.####.mov
                    - witnessCameraB.####.mov
                    - *FIZ/lens data*
                - {seq}-{shot}-{take}.popart
                    - *File format to describe the* Popart scene *with the database version used, recording Timecodes, the compositing parameters, etc.*
                - Output
                    - liveCompositing.####.mov
                    - onsetCompositing.####.mov

This default template will try to fit the naming convention used by most of the professional cameras.

File structuring

The files of a production is commonly structured by “all assets that belong to a shooting day”, rather than scene/episode, as a shooting day can cover shots from scenes from the whole span of the production. (Say episode scene 1, is followed by episode 13, scene 8)

Scriptsupervisor notes and info from the clapper creates the meta-relation between roll/shot and episode/scene. Associating shot-bundles with scene for final render/coloring, will usually happen after log’ing and editing of the material, and we need a database-application to maintain and update the meta-dependencies between assets and asset-bundles and scenes.

Usually this is done in post only on the shots selected through the edit, and we need to be ale to export Nuke-scripts as described above post edit. But we also move a lot of traditional post-complexity (normally reserved for single shots) to on-set, and need to have a way to automate as much as possible of that, without logging every single shot into a shotgun/nuke studio workflow.

Assets exported for scene/shot bundles, only happens towards de end of a production-pipeline.

Assets belonging to a day also trickle in asynchronously over and after the day (typically). And we have the extra challenge of having three types of assets:

Global/Pregenerated assets, which will be associated with single shots.
Example: Pointcloud, 3D assets, Lens distortion maps.

Live generated assets, the bundle that is captured when we hit “Record” and has TC
Example: Cameras, live-tracker/matchmover data, live comp, audio, 3D render

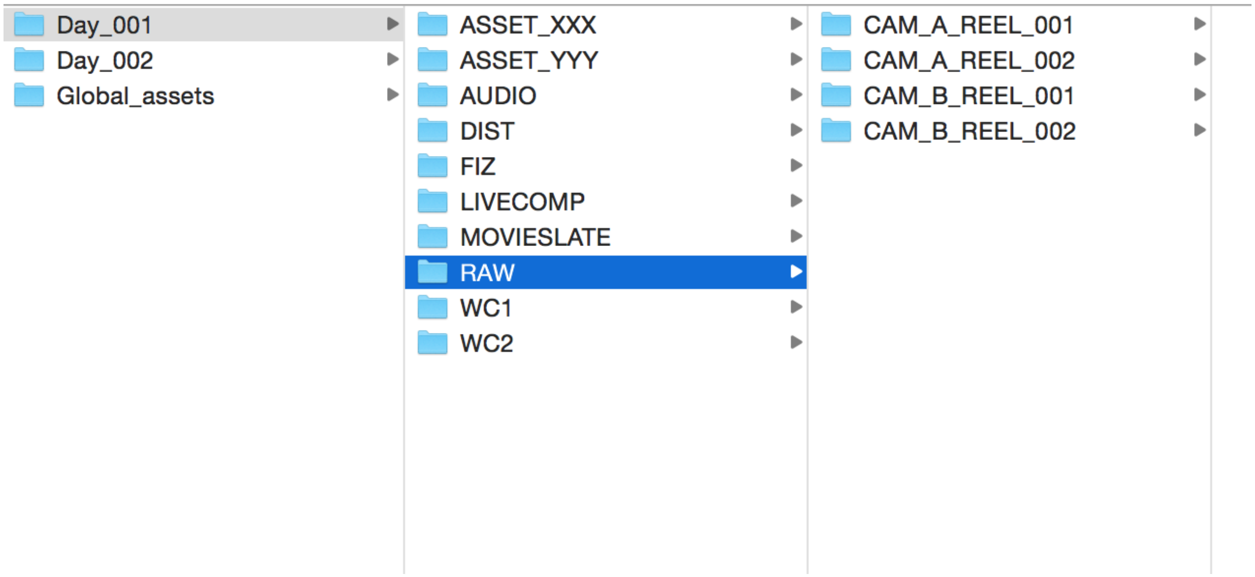
Meta assets:
Typically script-supervisor notes (which amongst other things associate shots with episode/scenes), and tech notes, like unique lens-ID (which assoicates specific lens with specific distortion map)

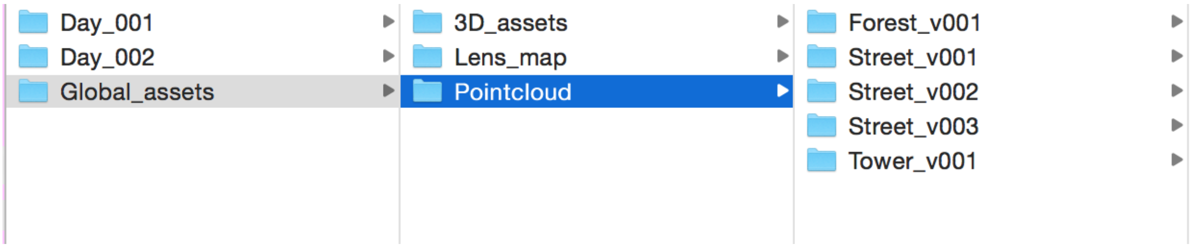
<u>Here is an example of a shooting day file-structure:</u>

Shoot day 1 → X /
Asset type A (Main cam)
Asset type B (Witness cam 1)
Asset type C (Witness cam 2)
Asset type D (FIZ/lens-data)
Asset type E (Audio)
Asset type F (Live composite)
Asset type G (3D files copied from global assets, merged with FIZ-data and RT camera tracking)

Asset type I (Script metadata)

Asset type H (Pointcloud) - copied from global assets
Asset type J (Lens map) - copied from global assets





We have yet to complete a full mapping of all the assets generated when we start recording, however we know that it is least 20 assets.

Naming convention for on-set assets.

We strongly suggest using the main-cam naming convention as a basis for naming of all assets.

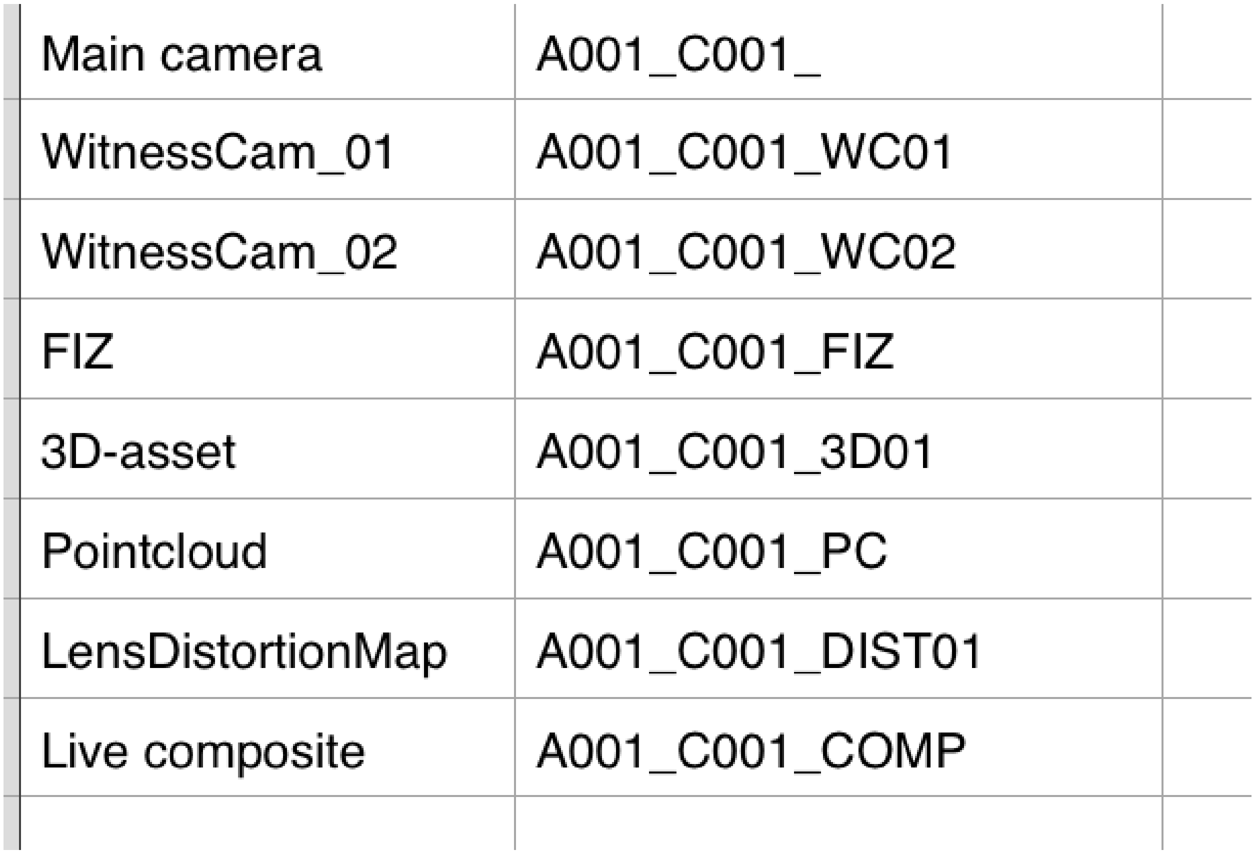
Typically (for most professional cameras) the naming structure looks like this:

A001_C001_XXXXXXXX

The:
A001 - means Camera “A” roll “001”

The:
C001 - Means “Clip 001”.

For our use, it would thus be practical if all the related files follow the same convention:



Main camera	A001_C001_	
WitnessCam_01	A001_C001_WC01	
WitnessCam_02	A001_C001_WC02	
FIZ	A001_C001_FIZ	
3D-asset	A001_C001_3D01	
Pointcloud	A001_C001_PC	
LensDistortionMap	A001_C001_DIST01	
Live composite	A001_C001_COMP	

Naming convention for off-set VFX assets.

To be able to re-ingest a VFX shot into an edit, a naming convention for shot-exports can be handy.

Structure could easily be:

Production_Episode_Scene_Shot

PROD_EP01_0001_0001_XXXXXXXXX

Where the X’s represent post-house specific metadata