WITNESS-Unity PoC with bloc digital

In June 2018 Lanner and bloc digital created a proof-of-concept that demonstrates how the Unity game engine could be used for rendering 3D model in WITNESS.

Project artefacts:

- Project docs: \\develop\Public\Products\WITNESS\Implementation Detail\WitnessToUnityPoC
- Specification for the PoC is located here: WITNESS-Unity Proof of Concept 01.pdf
- Source code: https://dev.azure.com/TwinOps/Witness/_git/WitnessCloud-Poc

The outcome from the PoC is as follows:

Item	Comment
2.1. Create and Position objects in a 3D world	**Done.** 3D geometry instances, linked to 2D WITNESS elements can be positioned sensibly in the Unity scene. Some of the dae/fbx shapes are not centred at 0, 0, 0 so appear offset. This is something that the existing 3D engine does when it loads the shapes.
2.2. Place a Part at a Machine	**Done.**
2.3. Create a simple Path and Position a Part on it	**Done.**
2.4. Apply an Extrusion to a simple Path	**Done.** Paths can be created by extruding a small unit geometry.
2.5. Move a Part along a simple Path	**Done.** Parts can be positioned along a path in response to a % update from WITNESS.
2.6. Position a Part on curved Paths	**Done.** Same as 2.5.
2.7. Move a Part that is contained by another moving Carrier	**Done.** Vehicles can carry parts in the Quick3D model
2.8. Text	Done.
2.9. Animations	**See 2.10. For a**
2.10. Walking people	**Done.** Aminations using a walking person.

Item	Comment
2.11. Robot Arm	Agreed to spend less time on this in preference to connecting with WITNESS.
2.12. Performance	Currently trying to understand why the performance is between 40-50% slower that existing Quick3D. Looks like it is related to communications between WITNESS / Unity but further investigation required by Lanner

Key improvements that effected performance with the PoC:

- Avoid synchronization locking between the background tcp listener thread and the foreground rendering thread by using a separate buffer for processing commands. These buffers are swapped at the appropriate time.
- Tweaked the number of commands that get processes during each Update() – clearly the more we do the faster it runs. This has proved to be the largest factor.
- Moved the parsing of xml commands to background tcp listener thread.
- WITNESS 22.5 reduces the number of broadcast events (by removing AVI support) which improves performance.
- Increasing the size of the read buffers on the background thread.

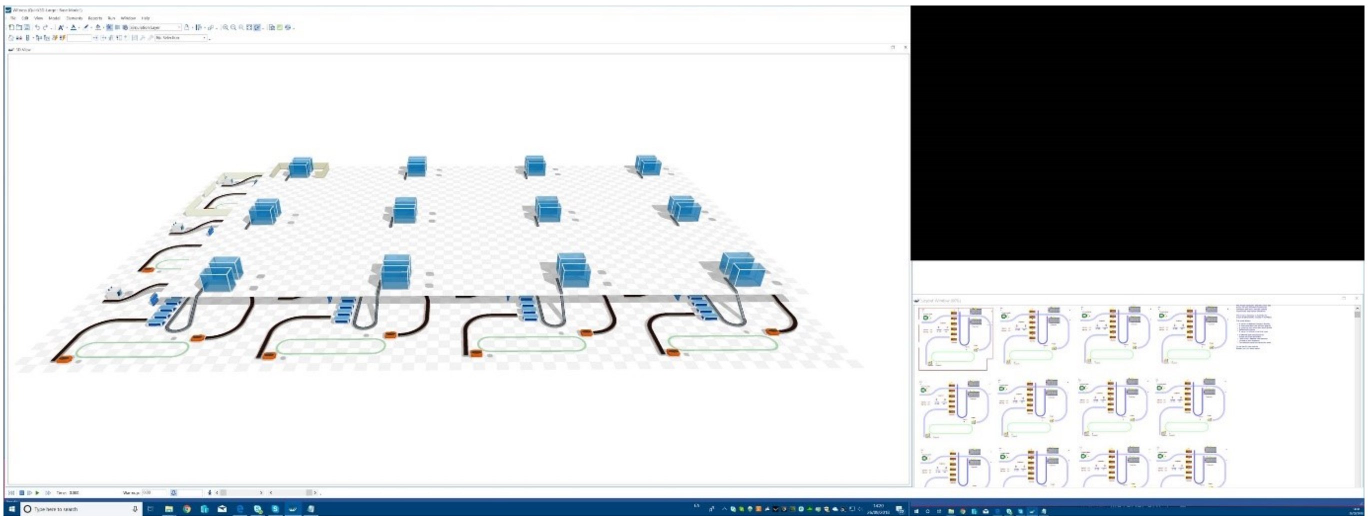
Here are some performance comparisons with Quick 3D

Environments

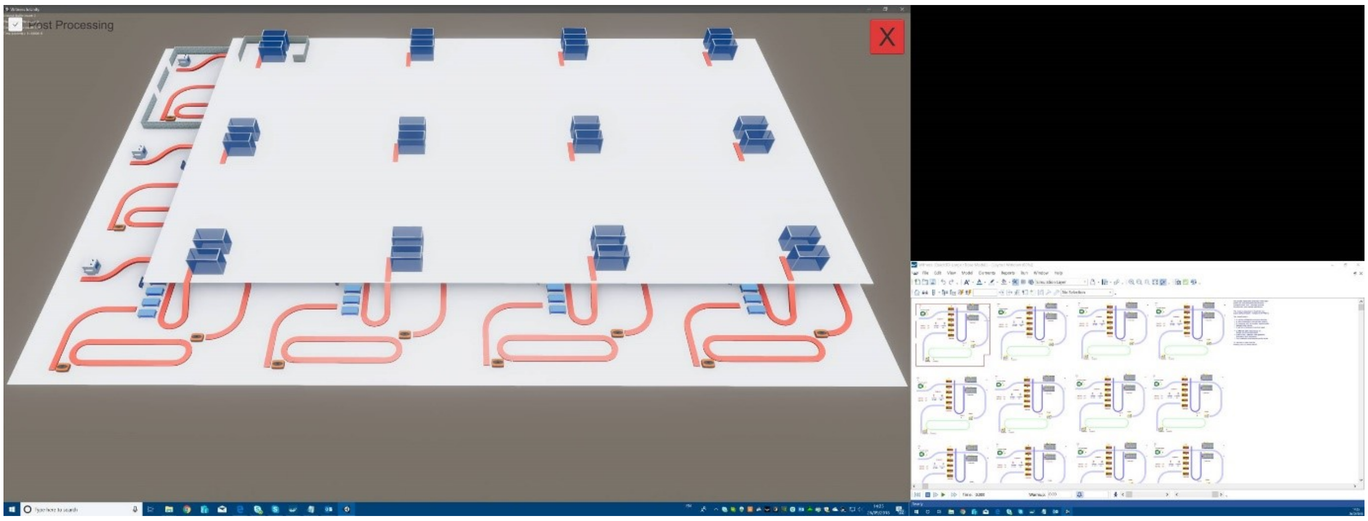
- Dual monitors 4K & HD
- 16GB RAM i7 (7th gen)
- Nvidia GF 1060 (JH) or Nvidia Quadro K4000 (EGA)

Test method

Quick 3D: To get a fair test where overlapping windows did not affect performance Quick 3D was tested using the following screen configuration – 3D window maximized on 4K monitor with the model enlarged to 60% zoom on the HD monitor. For example:



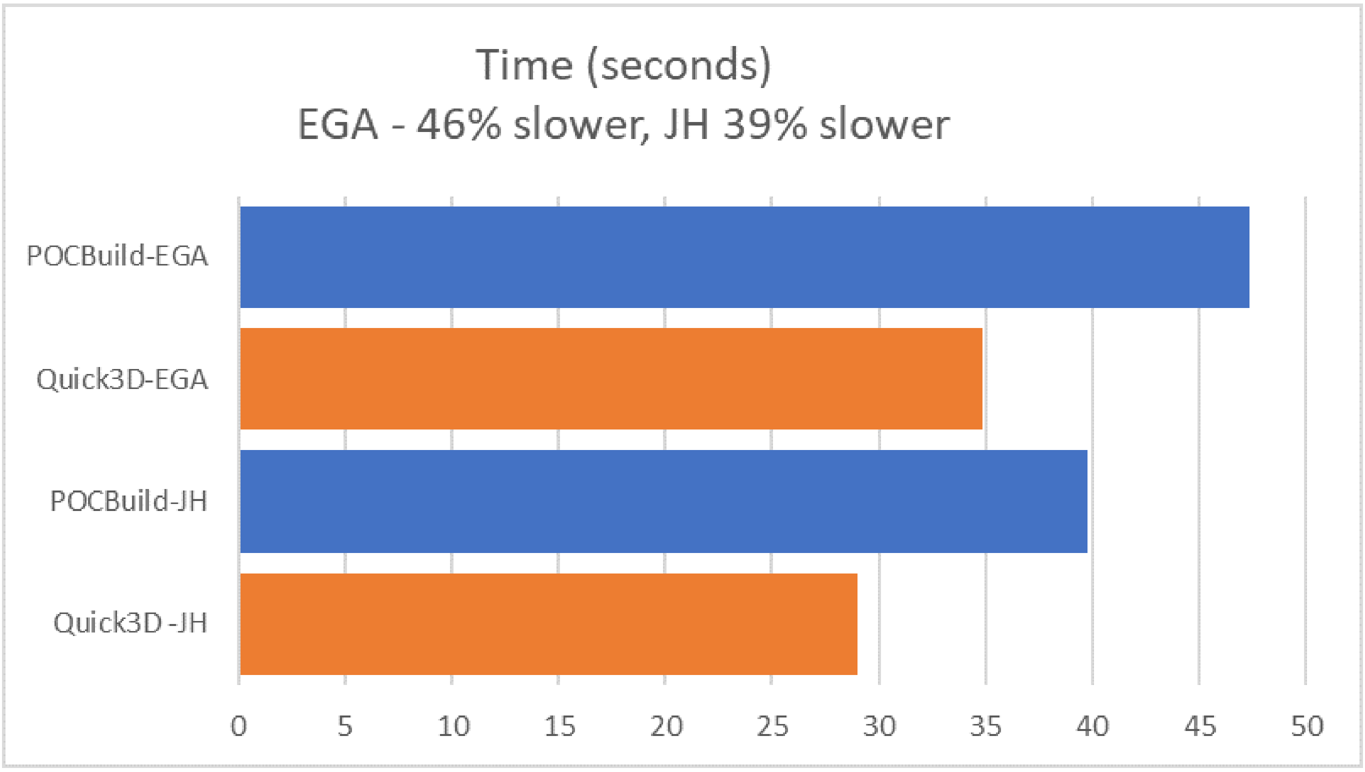
POCBuild: 3D window maximized on 4K monitor with the model enlarged to 60% zoom on the HD monitor. For example:



The model used for the test was Quick3D-Large.mod – and enlarged version of the Quick3D.mod

Results:

Test	Description	S1	S2	S3	S4	S5	S6	S7	S8	S9	Avg	% Slower
Quick3D -JH	3D on 4K, 2D on HD (60% zoom)	30	29	28	28	29	28	27	32	30	29	
POCBuild-JH	3D on 4K, 2D on HD (60% zoom)	41	41	43	43	37	39	39	38	37	40	39
Quick3D-EGA	3D on 4K, 2D on HD (60% zoom)	35	38	36	35	35	33	34	33	35	35	
POCBuild-EGA	3D on 4K, 2D on HD (60% zoom)	49	49	50	47	49	46	47	44	45	47	46



Other Observations

- One other factor is that WITNESS 2D graphics performs poorly on a higher resolution displays e.g. 4K.
- The tcp listener implementation was also re-written using different approaches (native C++ socket, .Net Scoket) but did not change performance.
- Running Unity without connecting to WITNESS but reading the events from a file, takes 20 seconds.
- Running Quick3D with no 2D window takes 9 seconds.
- Running Unity with no 2D window takes 18 seconds.