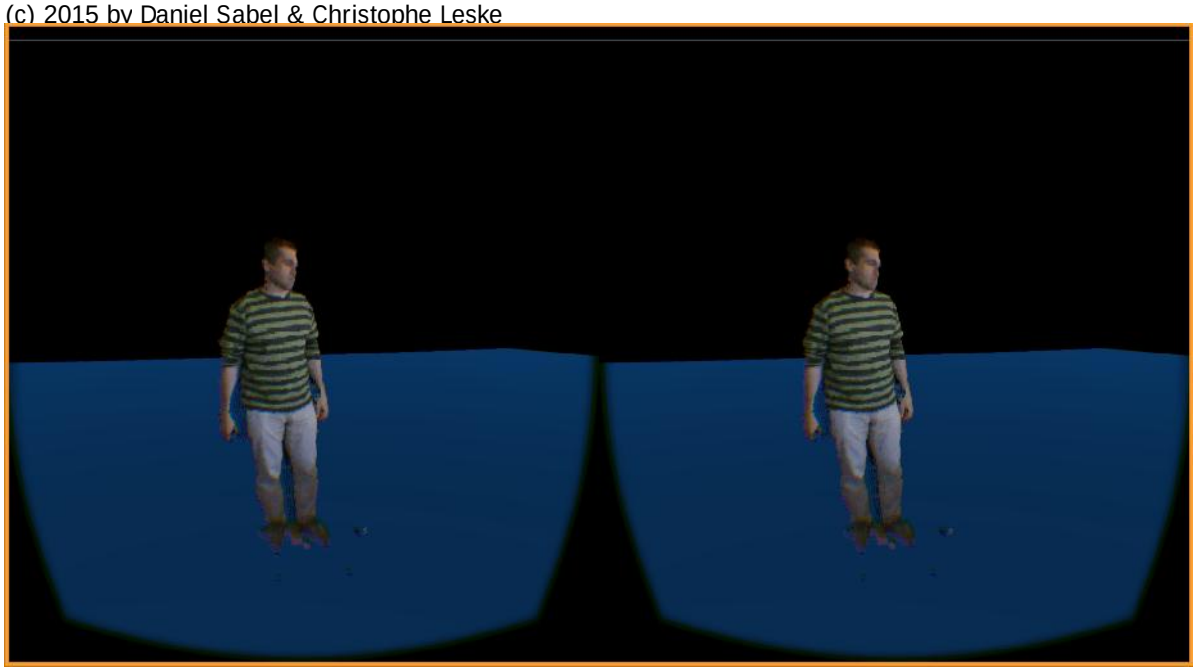
Documentation for VrVideo

Autor: info@multimedial.de

Welcome to VrVideo



Version history:
v1.0 - 6th of oct 2015 - christophe leske (info@multimedial.de)
v1.1 - 1st of dec 2015 -christophe leske (info@multimedial.de)

What is this?

VrVideo is a software application for recording and playing back 3D video captured by a depth camera like the Microsoft Kinect1 or 2. The 3D video is being captured as a normal Matroska .MKV file, using standard video codecs (H265HEVC and VP8). The said file can then be played back in a game engine like Unity in 3D to be seen in Virtual Reality. Since the files are standard Matroska files, one can also preview the content in a normal media player like VLC. So you get the best of both worlds. The filesize is comparable to the filesizes of compressed video files .

Requirements

We currently only support Windows 8 or higher as a recording and playback platform. For recording footage with a Kinect2, you'll need WIndows 8 in 64bit or higher, whereas playback can happen on systems with Windows 7, provided the graphics card of the target system supports DirectX11. We might support mobile platforms like the GearVR in future, but this is currently not supported. We are also working on streaming, but this is not yet publicly available. We plan to make this available as an additional plugin module, but due to the complex nature of the subject, this hasn't been finally decided yet. We also plan to release an editor in order to allow the editing of recorded footage.

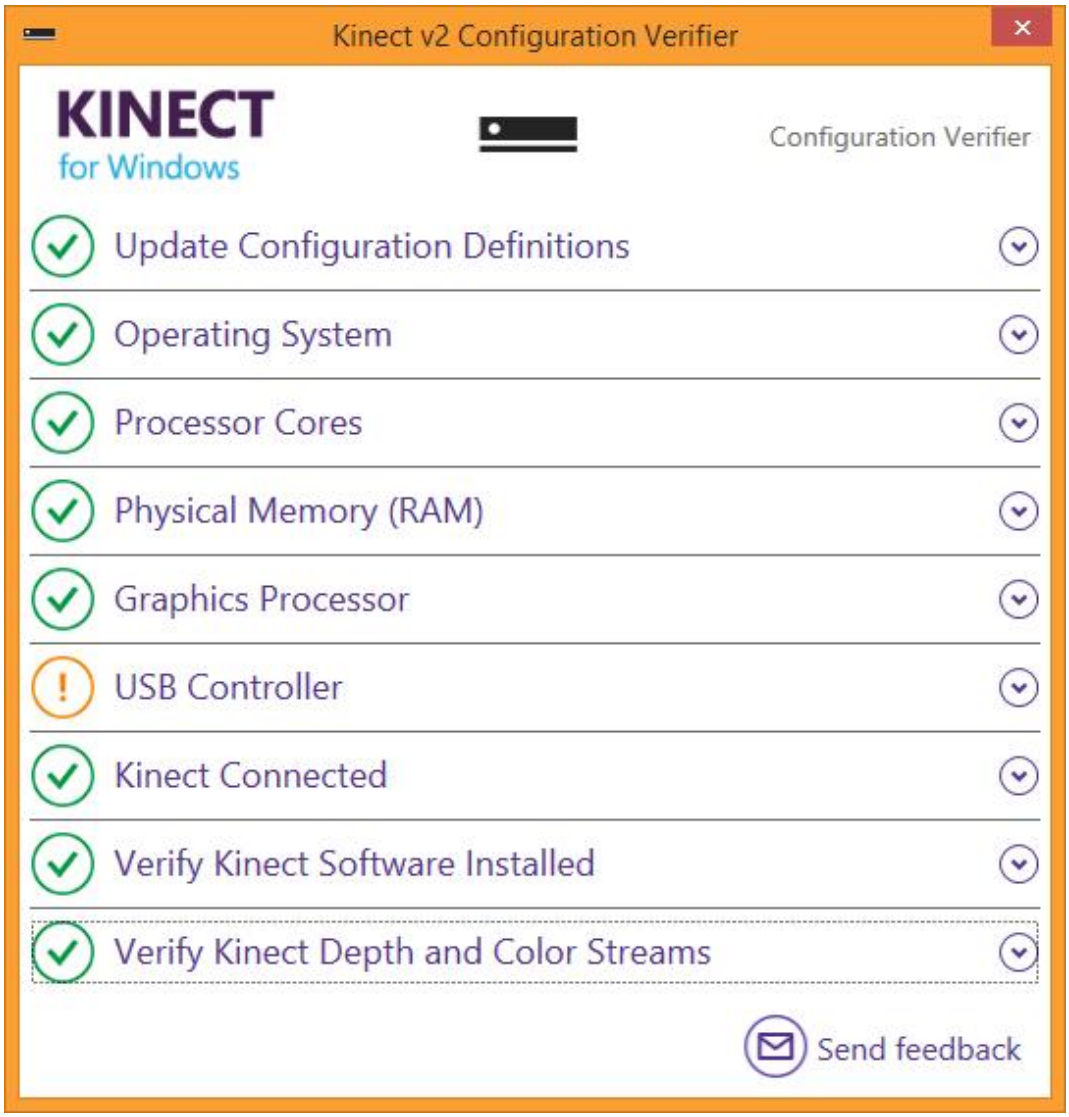
Workflow

The workflow of the solution consists of three steps:

1. Recording your footage
2. Setting up Unity for playback
3. Increasing the visual quality (optional)

1. Recording your footage

First, you need to install one of the supported 3D depth cameras. At the time of writing, these are either the Kinect1 or 2. For the rest of this documentation, we will assume that you got a Microsoft Kinect2 aka Kinect4Windows. Make sure that you got the latest driver installed and that the camera is working. You can easily check this by using the Kinect configuration Verifier, which is part of the KinectV2 SDK which in return can be downloaded from Microsoft:



Once you are certain that your camera is working, you can launch the recording application

(VrRecorder, see screenshot below). The app is divided into 3 parts: the upper preview windows, the optional settings for recording, as well as the start/stop button at the bottom.

Registering the app (TBA)

(to be added) The recording application is available upon demand from info@multimedial.de as a free trial version and will expire 1st of January 2016.

Recording footage

Make sure that you have a valid path specified for your recording. The default is "C:\temp\test.mkv", but you can of course point the application to any directory you wish.
There are several options you can choose from in the middle part of the application:

- make sure that the source device is selected correctly (we chose the Kinect 2)

- decide how you would like to shoot the footage by checking or unchecking the "Body separation" checkbox. This will make the application record additional data that will allow you to switch the type of playback at runtime. Unchecked means that you will record the full 3D frame without any additional body data in it, even if there are humans in the image. If checked however, the body silhouette as recognized by the Kinect will be saved with the stream, and you can chose at runtime to either see the whole picture (bodies + background), just the bodies as in a greenscreen application, or the cut-out background footage. See the Unity chapter for more information about this. Note that having the checkbox checked slightly increases the filesize and slows down the encoding process due to the additional information being saved in the stream

- the encoder parameters allow you to tweak the codec settings:

    - the "Lossless" option provides the best quality by losslessly encoding the depth data - if checked, the bitrate is not taken into account
    - encoding bitrate (which defines the resulting file size) for encoding the depth data
    - Interframe NR/Intraframe NR: noise reduction values for interframe/intraframe encoding (acceptable values are from 0 to 2000)
    - Intraframe smoothing smoothes transitions between frames
    - CU lossless
    - Distribute mode Analysis
    - Asymmetrical Motion prediction

By clicking on the "Record" button, you start the recording. The preview window will change and show the video stream of the camera as a rough preview.

2. Setting up Unity for playback

As of time of this writing, the only playback platform supported is Unity3D for Windows. The reason for this is that the software uses DirectX11 commands for playback.

Setting up Unity for playback is simple:

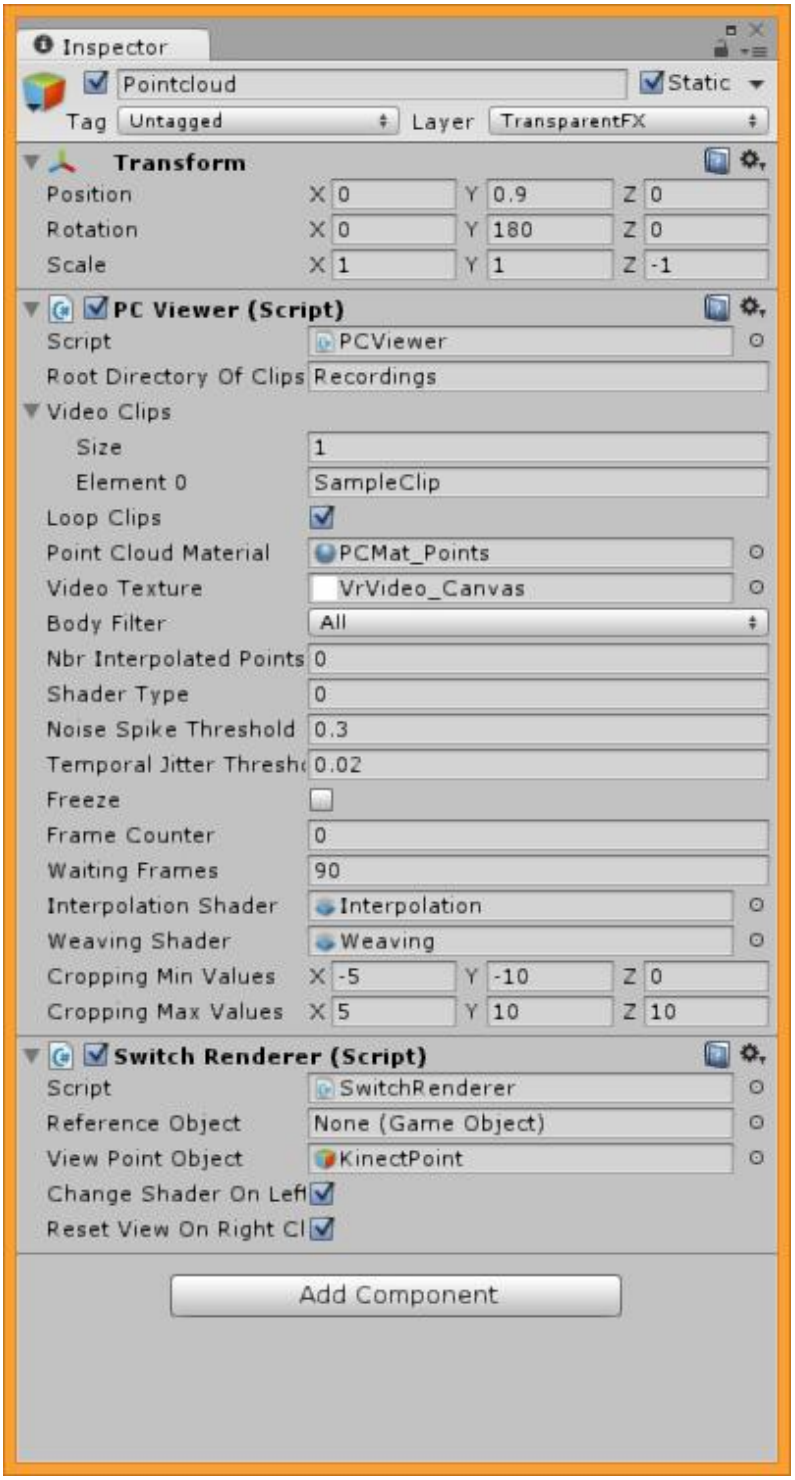
- set up an empty GameObject
- add the playbackscript to it
- customize the playback script for visual quality

There are currently two different rendering modes for the content in Unity - either

1. pointcloud rendering (individual dots form the 3d object) or
2. mesh rendering

Additional rendering modes like lines may be added in future if there is enough interest.

The Unity playback script has some customizable options:



Root Directory of Clips	Full or relative path to the recorded mkv file to be played back, amount of files to be specified in VideoClips, including their filename
Point Cloud Material	Should be the material provided with the Unity asset, pointing to the according shader.
Video Texture	Placeholder Texture Object for playing back the recorded video (white image)

Nbr of Interpolated Points	Optional number of points to be inserted into the pointcloud - reasonable values are 0 to 4
ShaderType	Shaderstyle to use. 0 = mesh rendering, 1 = pointcloud rendering
NoiseSpikeThreshold	An value describing the maximum length of a polygon to be considered for filtering. Lower values yield to flatter and more concise surfaces but may also reduce details in the scene, higher values will produce spiky artefacts around the meshes. Default is 1.
Temporal Jitter Threshold	A value describing the allowed distance discrepancy (jitter) for any pixel between two adjacent frames. If the discrepancy is lower than the specified value, the pixel won't be changed from one frame to the next. Lower values (default is 0.02) work best, ensuring a good compromise between a steady image and changing geometry. Higher values will freeze the geometry, whereas the video texture will continue to play.
Freeze	Allows to freeze the rendering of the stream - does not stop the stream itself, meaning that in the background, the clip continues to play.
InterpolationShader	ComputeShader used for doing the point interpolation
Weaving Shader	ComputeShader used for weaving together the vertices to triangles, and to weave the uv coordinates
Cropping Min Values	Defines the minimum cropping volume from which to draw the scene.
Cropping Max Values	Defines the maximum cropping volume from which to draw the scene.

Increasing the visual quality

The current consumer depth cams suffer from low depth image resolution. The Kinect2 for instance records depth (theoretically) at 512x424 pixels. This number is further reduced by the color mapping process when recording a 3d image from the device, as pixels of the HD video need to be mapped to their depth pixel counterpart in the image from the depth sensor. The two cameras in the Kinect are therefore aligned to have sweat spot about 1.5-6m from the device. This intrinsic convergence of the cameras however decreases the effective resolution of the depth image, as many depth pixel cannot be mapped to a corresponding color pixel in the video stream, and vice-versa. As a result, these pixels either don't get valid depth data, and/or no color data. Often enough, the depth image is therefore further reduced to rather 450x380 pixels or even less. Additionally, the signal generated from the sensor is also quite noisy, meaning that pixels are jittering from image to image.

We offer three different solutions to counter these effects and thereby hopefully increase the visual quality of your recordings:

1. Interpolation (pertains to point cloud rendering only)
2. a depth filter ("Noise Spike Threshold")
3. and a temporal jitter filter ("Temporal Jitter Threshold")

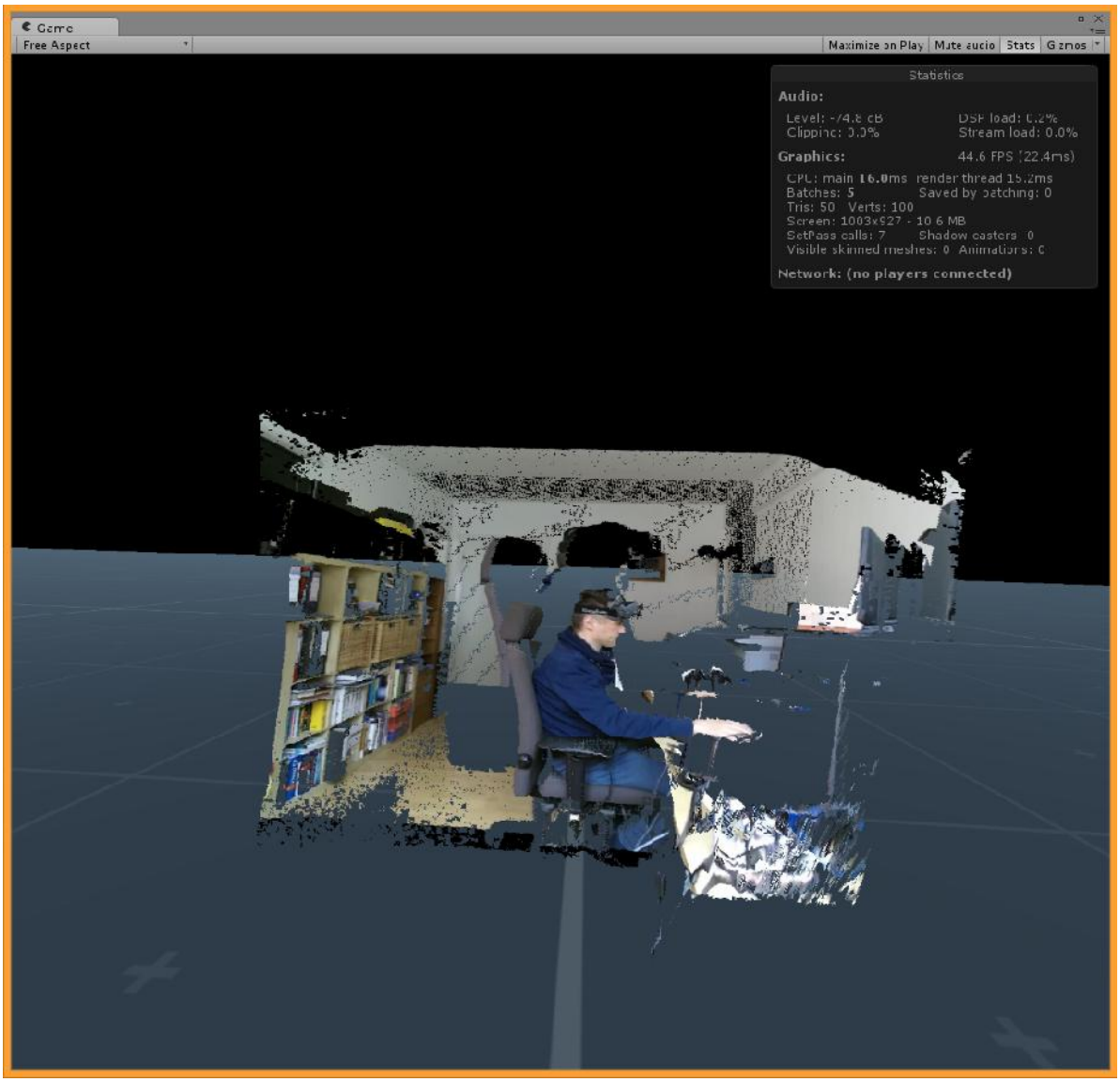
As an additional option, we also offer the recuperation of otherwise lost information in the color stream, namely all those color pixels coming from the HD video stream of the camera that couldn't be mapped to an according depth pixel and which would therefore be lost.

Rendering modes

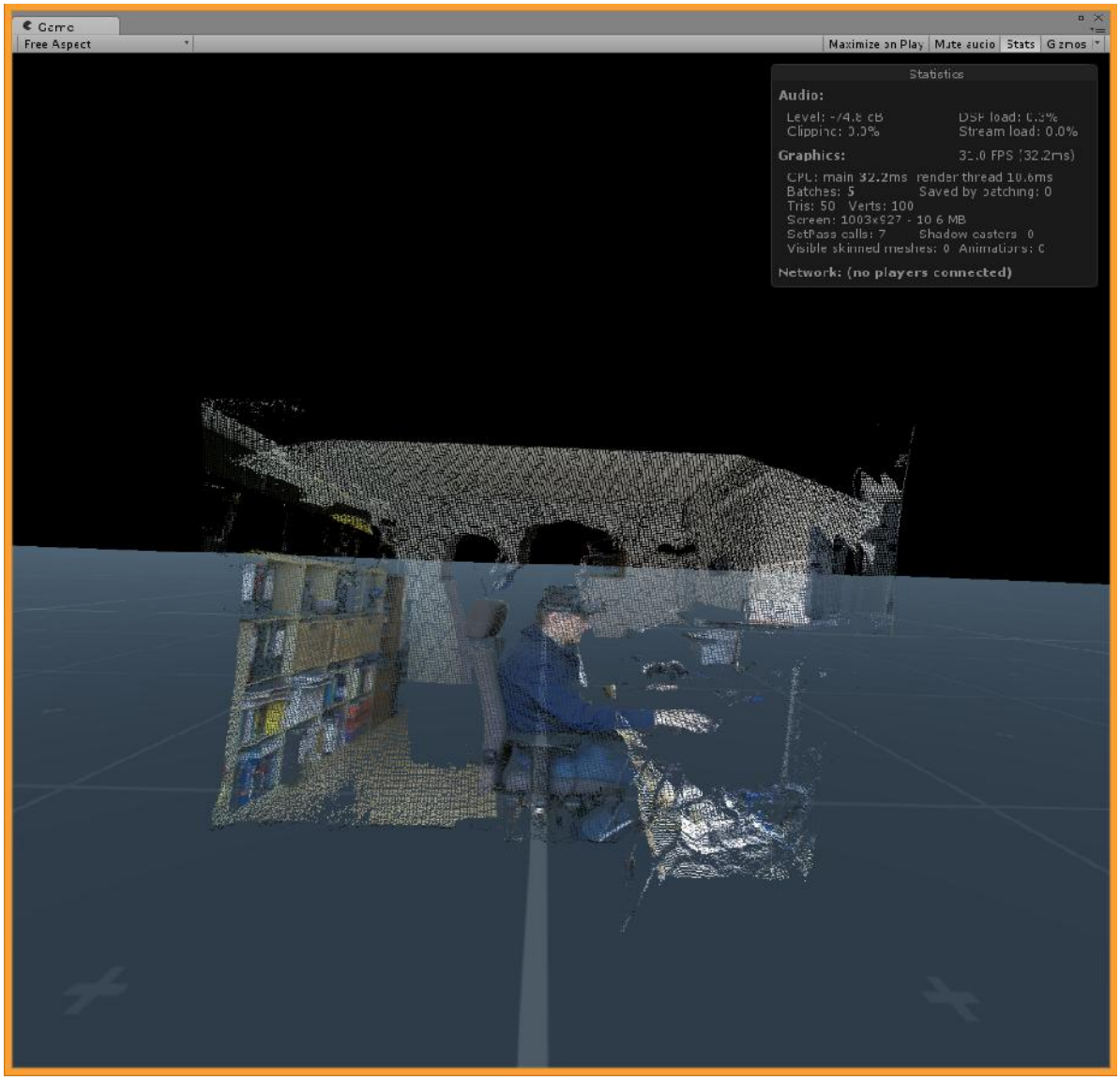
We are currently offering two different rendering modes:

Mode 0: mesh rendering
Mode 1: point cloud rendering

When rendering in mesh mode, the points are being reordered in the stream and get connected to a closed mesh:



When rendering in point cloud mode, the recorded data is rendered as individual points in space:



Interpolation (currently broken due to changes in the interpolation algorithm)

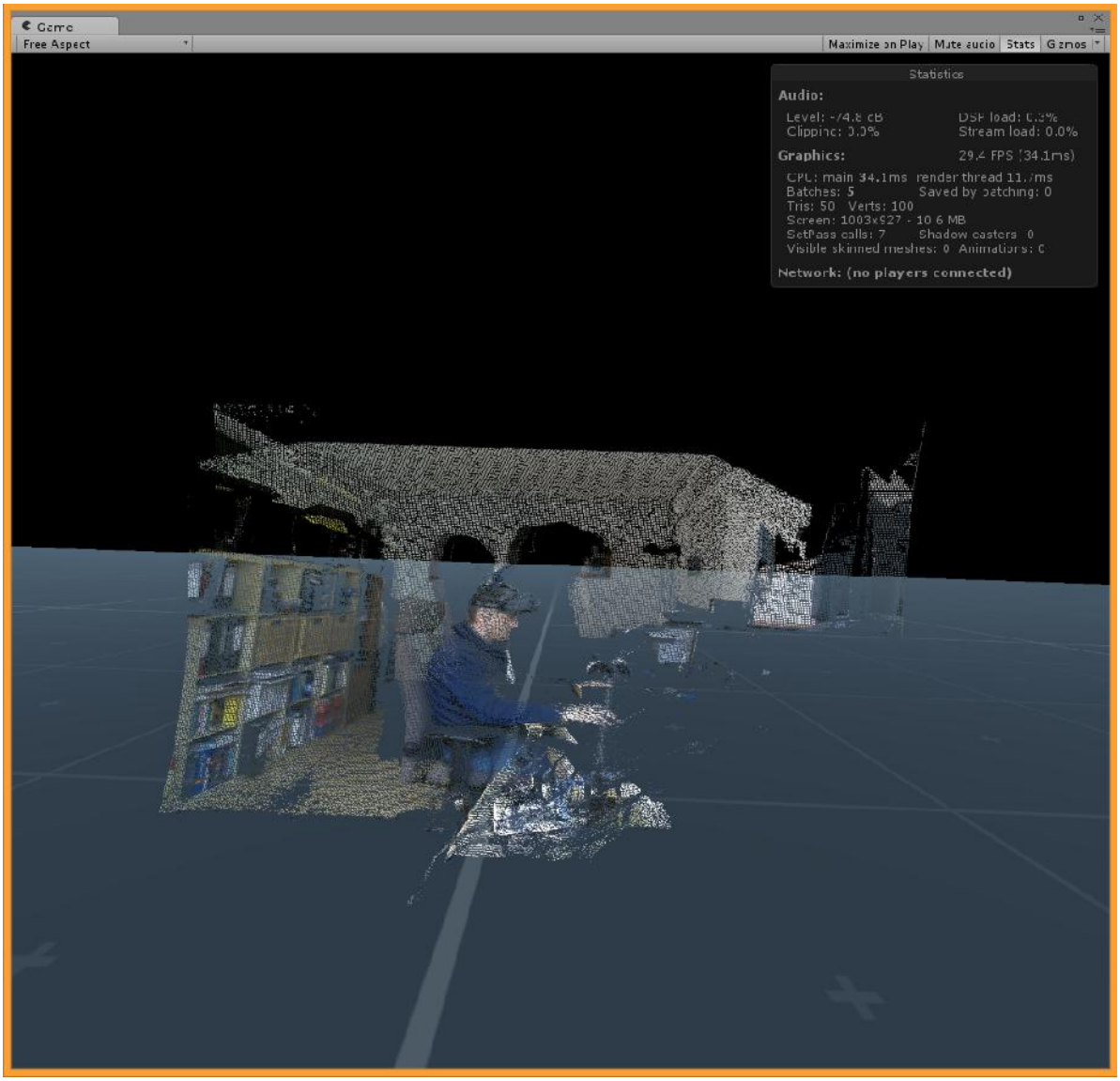
The following pertains only to the point rendering mode:

Interpolation is the process of adding points into the image by computing the difference between existing points that were recorded. This methods enhances the visual quality, but depending on the amount of points you add to the stream, the rendering speed may decrease, due to the additional computational load. The interpolation is done on the GPU, using compute shaders.

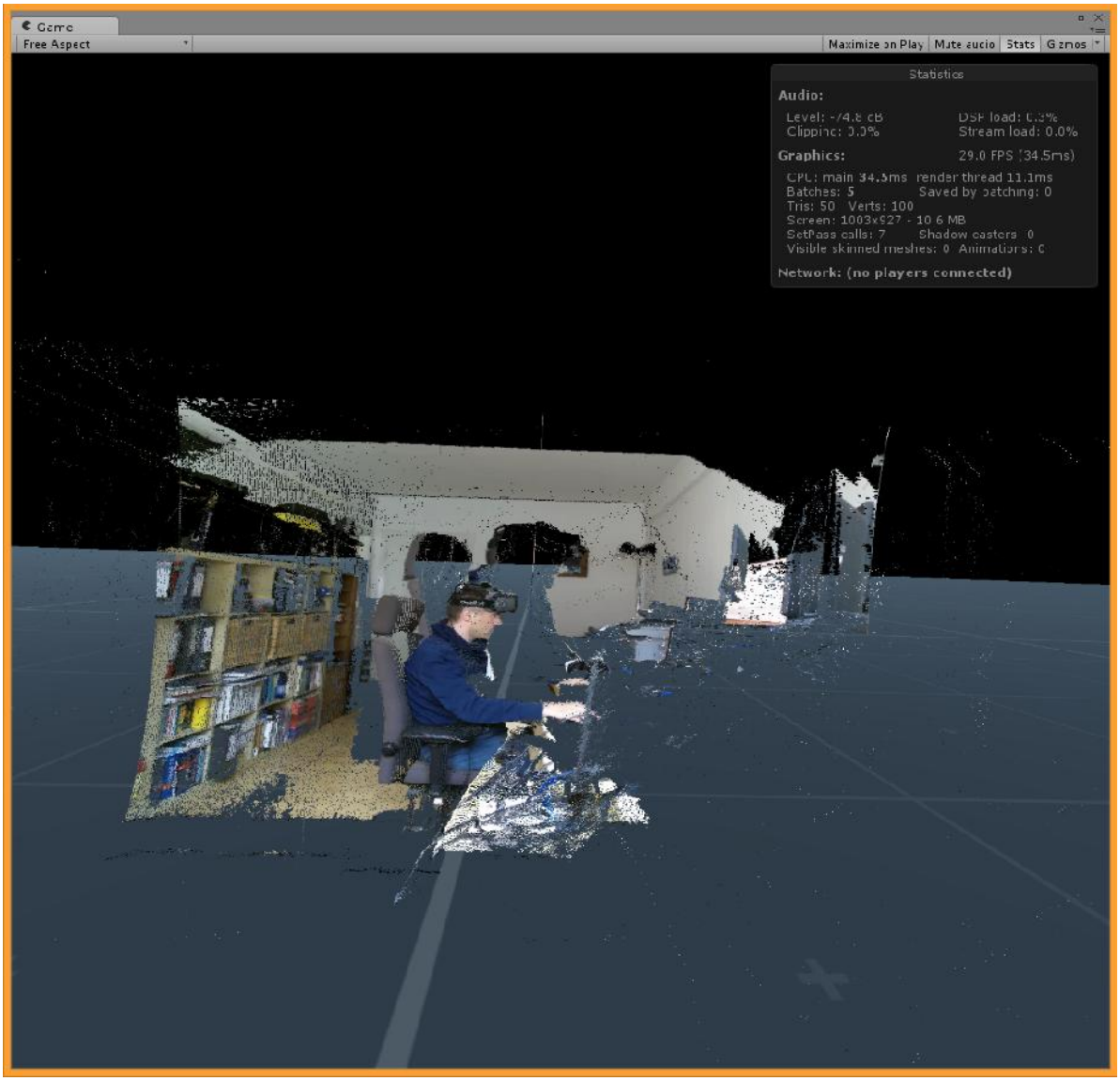
Please note:
every increment of the "NbrOfInterpolatedPoints" is effectively DOUBLING the amount of points being rendered! If you add one interpolated point, you double the theoretical resolution from 512x424 to 1024 by 848. The more points you add, the better the visual quality gets, but the rendering speed will most probably decrease, depending on the power of your machine.

Values that make sense range from 0 to 4 eventually.

Zero interpolated points (recorded stream in raw physical resolution):



One interpolated point added (note that noise in the image also gets enhanced):



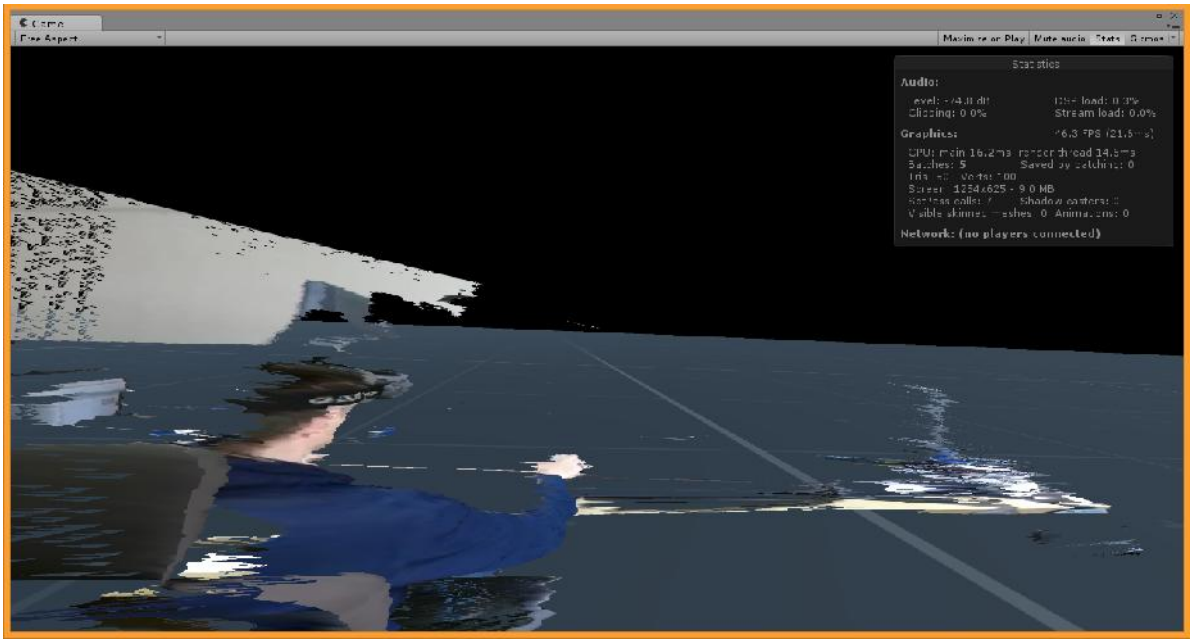
Filtering

Two additional issues occuring when playing back recorded footage are spikey polygons (partially due to the noisy signal of the sensor, as well to the lossy compression of the video codec). On top of that, there is also the inherent temporal jitter coming from the kinect - the points jitter in space due to fluctuations while recording. We offer two different filter values to remedy the situation.

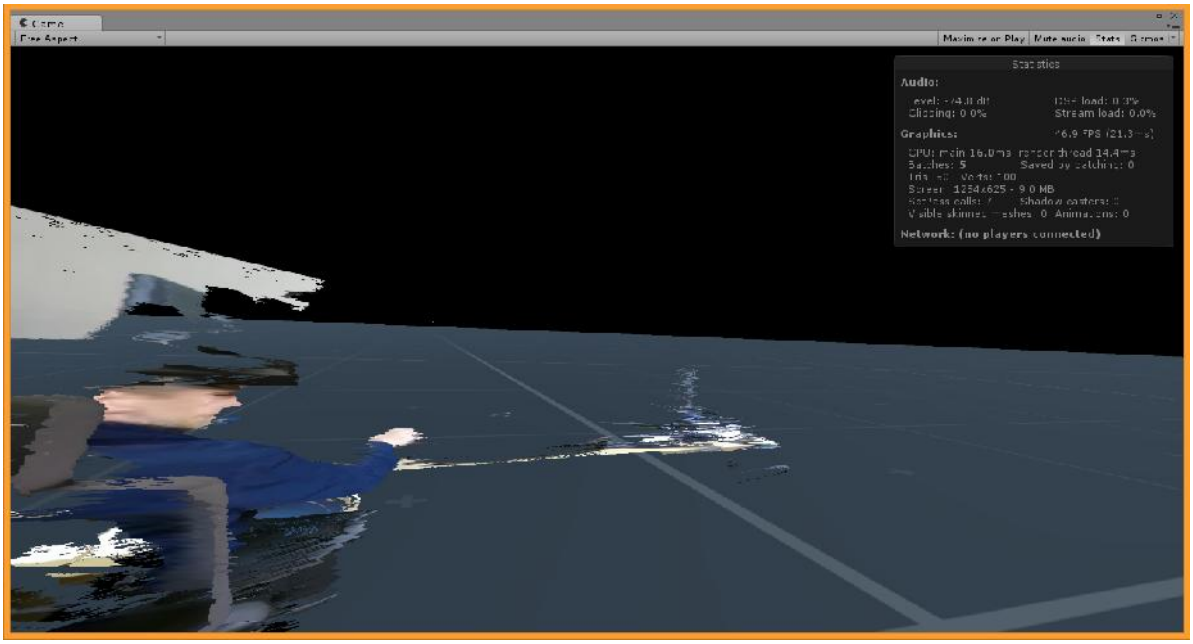
1. NoiseSpikeThreshold

Spikey triangle artefacts can be reduced by lowering the value for NoiseSpikeThreshold (default is 1, good values are between 0.2 to 1).
If you set this too low, some triangles of the mesh won't be formed. If you set this too high, you will get spikey artefacts in the scene.

Value of 1:



Value of 0.2:



2. Temporal Jitter Threshold (point cloud rendering only)

The Kinect produces inherently noisy depth images, meaning that between individual frames, the depth of the individual pixels slightly jitters. This produces noisy footage. We therefore introduced a temporal interpolation filter which we called the temporal jitter threshold. It's value is a double edged sword: the threshold describes the amount of difference (=jitter) allowed for a given point between two given frames. If the difference between two frames for a given pixel is lower than that value, it won't get redrawn - it stays in the image, thus producing a more static stream. However, if you choose this value too low, you'll end up with the raw jittery footage coming from the Kinect. If you choose this value too high, you'll end up with a scene that will hardly budge - the pixels won't move, unless there is really some hefty movement going on in the scene. Since this won't affect the video texture, you might end up with static geometry onto which the video stream is layered as (somewhat detached) texture. This filter

is therefore a compromise between dampening the temporal noise, and preserving the action in the recorded scene. Good values seem to be between 0.02 and 1.

Tips and Tricks

Setting up your own Unity scenes

If you want to set up your own 3D scenes in which your recordings are ought to play, you'll need the following components as the bare minimum:

- an empty gameobject which will hold your pointcloud/mesh recording
- the PCViewer C# playbackscript, configured for the different options (especially the material and the file to be played back)
- the pointcloud material and its shader, assigned in the PCViewer C# script
- Interpolation and Weaving Compute Shader in the Assets of the project
- the DLLs needed, especially VrVIdeo.dll in the Plugins-Folder of your project

Shiney objects do not work well

Shiney objects like cars or glass don't reflect the IR light of the Kinect. They are best to be avoided.

Sun light limits the range of the kinect

Since the Kinect emits its own IR light, recording in sun light can diminish the range and FOV of the camera. The Kinect operates best in terms of depth in dim or even dark environments, otherwise be prepared for shorter ranges.

Framerates of the Kinect may drop depending on the lighting conditions

The Kinect can change its framerate if lighting conditions aren't favorable. Most of the time it records at 30fps, it can drop down to 15fps however.

Consider using a specialized microphone instead of the built-in ones from the kinect

The Kinect has 4 built-in microphones, which mix down to a Mono signal. A specialized microphone might produce better results than the built-in ones, especially for interview-like setups.

Mobile 3D camera rig

You can easily build yourself a portable 3D camera rig by following the USB3.0 mod hack provided by Julian Tatsch. The hack shows how to hook up the camera to the USB3.0 port directly and have it powered externally. As an alternative, you can also use a mobile Powerbank such as the Xtorm AL390 which offers a normal 240V power outlet. For those of you living in the UK or US, you will have to find a similar powerBank with a fitting power outlet for the Kinect power adapter. The author of this documentation uses such a mobile camera rig along with an Xtorm AL390 and can record about 40minutes of footage with it. A laptop or (even better) a windows based tablet allows for easy set up and recording.