Liberty3D UberCam
For NewTek’s LightWave3D
Version 2.0

July 23rd, 2015

Thank you for purchasing UberCam 2.0!

(Win32Bit, Win64Bit, MacUB Intel 32 Bit / Intel 64 Bit)

Copyright © 2010-2011 Liberty3d.com (Declaration Pictures Inc.)
Version 2.0 Code and Engineering: Chilton Webb
Design and Concept: Kelly Lee Myers
Special Thanks: NewTek, Oliver Hotz at Origami Digital, The Liberty3D.com Citizens and the supporters of Liberty3d.com including our Beta Testers, Customers and Friends around the world.

COPYRIGHT
© 2010-2015 Liberty3D (A Division Of Declaration Pictures Inc.). All Rights Reserved.
Except as otherwise permitted by the authors, this publication, or parts thereof, may not be reproduced in any form, by any means, by any method, for any purpose. Disclaimer THIS PUBLICATION AND THE INFORMATION CONTAINED HEREIN IS MADE AVAILABLE BY THE AUTHORS “AS IS.” LIBERTY3D DISCLAIMS ANY WARRANTIES, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

Liberty3D No-Nonsense License Statement
(We acknowledge inspiration to this in part to the words and the spirit of the Borland No-Nonsense License Statement)
This software is protected by both United States and Canadian copyright law, as well as international copyright treaty provisions. It is software that is licensed only to be used in conjunction with Newtek LightWave3D software (which is described at the URL http://www.lightwave3d.com).
Therefore, you must treat this software just like a part of a legally licensed installation of LightWave. You may copy it onto a computer to be used and you may make archival copies of the software for the sole purpose of backing-up our software and protecting your investment from loss. As such, since the software is license locked to a hardware or software identification device, commonly known as a dongle, any use of the software on a computer workstation in a non-network rendering mode without a legally obtained hardware identification device and the appropriate keys to unlock the software is prohibited.

Further Explanation of Copyright Law Provisions and the Scope of This License Statement
You may not download or transmit the software electronically (either by direct connection or telecommunication transmission) from one computer to another, except as may be specifically allowed in using the software on a computer
network. You may transfer all of your rights to use the software to another person, provided that you transfer to that
person (or destroy) all of the software, as well as the hardware identification device and documentation provided in this
package, together with all copies, tangible or intangible, including copies in RAM or installed on a disk, as well as all back-
up copies.

Remember, once you transfer the software, it may only be used at the single location to which it is transferred and, of
course, only in accordance with copyright law and international treaty provisions. Except as stated in this paragraph, you
may not otherwise transfer, rent, lease, sub-license, time-share, or lend the software, diskettes, or documentation.
Your use of the software is limited to acts that are essential steps in the use of the software on your computer or computer
network as described in the documentation. You may not otherwise modify, alter, adapt, merge, de-compile or reverse-
engineer the software, and you may not remove or obscure copyright or trademark notices within this product.
There is no rights to redistribution of this software that is included in this License. All licenses, software downloads, and
software keys must be obtained via the Liberty 3D website (URL http://www.liberty3d.com ).
By obtaining a software key and using this software, the user signals agreement to this license agreement.
If the hardware device is damaged or lost, our support policy again matches that of NewTek. If NewTek support will verify
that they agreed to reissue a hardware identification device to a legally obtained licensed version of LightWave3D due to
damage or loss, then we will reissue a new key upon request within a reasonable amount of time. The exact content of
this agreement in obscure legal jargon may be supplied upon request, and the laws of Alberta, Canada will apply if there
is any conflict in the interpretation of the law between our place and your place.
All trademarks are the property of their respective owners.

Chapter 1 Liberty3D UberCam v2.0

Liberty3D UberCam is a plug-in for NewTek’s LightWave3D that enhances the
LightWave platform with additional camera capabilities while expanding into the realm of
Virtual Reality. Three years of additional research and development, testing and
customer feedback since the release of Version 1.9 have been brought forward in
Version 2.0.

In version 2.0 we have introduced additional specific tools and cameras to the artist who
wants to explore Virtual Reality content creation using LightWave3D. Virtual Reality over
the last 4 years has exploded and Liberty3d.com has been working quietly to bring
forward tools to LightWave artists that will give them a competitive edge in this new
medium. Many of the tools in UberCam 2.0 are unique in the industry thus far and as a
result are not available for any other package out there in a simple to use and affordable
tool set.

Our development efforts in the areas of Virtual Reality for UberCam continue and we will
be providing additional tools and functionality to the package in the very near future. This
includes support for additional Virtual Reality Headsets beyond the Oculus Rift.
With UberCam 2.0 you get all of the original cameras from previous versions of Ubercam
with many bug fixes, performance enhancements and the added capabilities we have
introduced that opens up new possibilities.

About Liberty3D UberCam

Liberty3D UberCam requires at least NewTek’s LightWave v9.3.1 in order to run most of
the camera features contained in the software.

As of version 2.0 there are at least 16 camera types included in the UberCam package.
In addition to the controls included in the new cameras, the standard position and
orientation of the existing cameras will affect the Linear and Non-Linear UberCam types as well. Although the cameras will function as described in LightWave 9.3.1 and above, we recommend upgrading to LightWave 11.6.3 at least, and above in order to take advantage of additional capabilities, LightWave 2015 for full stability during use. We have tested UberCam 2.0 using LightWave 3D versions 11.6.3, 2015, 2015.2 and 2015.3. In the future, support for LightWave 9.x may need to be removed in order to advance this product.

In addition to the Camera Plug-ins, UberCam 2.0 now ships with a set of additional and very special plug-ins giving LightWave3D artists unique VR production capabilities.

First, there is the UberRift Viewer Window Plug-in simply called "L3DRiftViewer64.p" or "L3DRiftViewer32.p" for PC installations. This plug-in provides for a floating window clone of your primary layout viewport and is intended for use with the Oculus Rift VR Head Mounted Display. This cloned viewport applies the correct barrel distortion to your camera view for use with the Oculus Rift VR Head Mounted Display and eliminates the need for you to export your scene to a game engine such as Unity3D or Unreal Developer Kit in order to see your work in VR with the Oculus Rift. It supports both of the currently available Oculus Rift Developer Kits; 1.1 and 2.0.

Second, there is an Image Filter Plug-in called "L3DStereoF64.p" (64bit version, 32bit and OSX versions are named appropriately) that applies the correct barrel distortion to a Liberty3D Stereo Cam rendered image. This plug-in will apply the distortion in F9, F10 and LWSN renders.

Lastly there is the L3DVRHeadset64.p (x64 Windows) or L3DVRHeadset32.p (x86 Windows) or VRHeadset.plugin (Mac). This plug-in provides for Oculus Rift rotational head tracking (positional tracking will be made available at a later time) capability via LightWave's Virtual Studio Tools. It consists of two main components; a VR Headset Manager which is found under the Virtual Studio Tools Device Manager Panel and an Item Info Device node that connects a Virtual Studio Tools ItemMotion:Camera Trait to the VR Headset Manager. These combine to give you the real-time rotational head tracking of the Oculus Rift. It is intended to be applied to the rotation of a camera, but can be added to any item in a 3D scene such as a Null, Light or Object providing you with an additional real-time input device option with Virtual Studio Tools.

**PLEASE NOTE:** Our UberRift viewer requires at least LightWave3D 11.6.3 running on a Windows PC in 32bit or 64bit at this time. The UberRift viewer is currently not supported on Mac OS systems. It is important to note that Oculus VR has recently dropped support for Mac platforms. While we are working to get a MacOS version of the viewer working for a future update, there is no guarantee that the Oculus Rift will still support the Mac.

In addition to this, to use the Oculus Rift rotational head tracking features provided in UberCam via LightWave3D’s Virtual Studio Tools which became available in version 11.0 you will need to at least be running 11.0 for LightWave3D on the PC or Mac. We have tested the UberRift viewer and L3DVRHeadset plug-ins with both the Oculus Rift DK1.1 and DK2 developer kits extensively. Either headset can be used with these plug-ins as expected.

In order to use both the UberRift viewer and the Oculus Rift rotational head tracking
functions together you will need to be running at least 11.6.3 on a Windows PC in order to provide a full Virtual Reality experience in Layout as intended.

**Installation**

Liberty3D UberCam is installed just like any other plug-in for LightWave. To install UberCam, simply place the appropriate version of the plug-ins that come with this package into a folder named "Liberty3D" or something similar in your LightWave3D plug-ins folder. Then, run layout and add the plug-ins from this directory location. This is done by selecting add-plugin in the Utilities menu bar selection in the menu bar system, or by pressing add plug-in in the edit plugins dialog box. In order to correctly recognize all the camera types we recommend un-installing the old plug-in first, then re-adding the new plugin.

This will recount the plug-in types included in the package. We do not recommend attempting to run older versions of the UberCam plug-ins along side the latest versions. Doing so may be a bad time.

The additional plugins found in the package should be added in the exact same way. Remember to close Layout down afterwards in order to write the plug-ins to your configs on your system. We recommend doing this immediately once you have all the plug-ins loaded so that you are good to go for the next step.

**Key Activation**

When you purchased UberCam from [www.liberty3d.com](http://www.liberty3d.com) you should have been emailed back a Key number after you entered your Product Lock ID into the store section field on check out. If you didn't do this you need to contact us and provide your Product Lock ID (aka hardware lock ID) number to us as well as your receipt. This will help us to get the key back to you faster. You will then be emailed a key.

We pride ourselves on our fast customer service and will have your key back to you very shortly but it is not an automated process as of this release, so please be patient as we get your key to you.

Once received, this key is then entered into a dialog box that pops up when you first try to access the property panel for any of the UberCam Cameras listed inside of the camera properties panel. Simply paste the number or enter it in manually and hit “OK”. You will get a message stating the key is accepted or not. The box, depending on which OS type of LightWave you are running will either go away on its own or not. If it doesn't, just close it by using the “X” button on the license key dialog box. Keys issued from versions prior to UberCam 2.0 will not work. A new key must be issued.

Once you have entered your key and it has been accepted by the software, you should now be able to access the properties of the camera, or switch cameras to any of the others available in the drop down list and make use of them.
**Please note:** The key can only be entered by accessing the UberCam Camera properties. It cannot be entered through accessing the viewer plug-in, the VR headset manager plug-in or the image filter plug-in.

We highly recommend that you set the permissions of your LightWave3D installation or wherever you store your plug-ins to be writeable by the local user and run Layout in Administrator mode when activating your copy of UberCam 2.0. This will ensure the plug-in has permissions to write the key file to the directory where the plug-ins are stored so you won't have to go through this process again each time you start up layout and want to use UberCam 2.0.

If you have any problems with your key, please contact us immediately via email. Email: kat@liberty3d.com

---

**What has changed in UberCam?**

Lots! UberCam over the years has evolved significantly, culminating with this UberCam 2.0. For the sake of space this list is not entirely exhaustive but covers the basics.

**Changed from v1.0 to v1.5**

- Liberty3D UberCam v1.5 was changed to be compatible with LightWave v9.3.1., 9.5, 9.6 and now LightWave3D 10.0 Previous versions of UberCam required LightWave v9.6.
- The Liberty3D SuperPano Camera and the Liberty3D Overscan Camera was added in UberCam v1.5.
- Some camera names were adjusted for clarity.
- This document was added.
- The Liberty3D Performance Camera was included as a separate optional camera plug-in package in order to judge the performance hit from various camera choices.

**Changed from v1.5 to v1.9**

- The Radial Shift Camera, and the DPI Camera were added as new camera types.
- The Liberty3D performance measurement camera was included also in order for you to check your performance changes on your own scenes and object files.
- The one point poly and two point poly problem with multi-up and stereo camera combinations has been fixed by using a workaround for a LightWave 3D rendering bug.
- There was an error in calculating the zoom factor in a rendered sequence if the zoom factor was enveloped. This has been fixed.
- The PPC build type was removed from the Mac UB build in order to reduce the size of the plugin.
- This document was updated, although we were too lazy to update the screen captures for the dialog boxes.
Changed from v1.9 to v2.0

- The Liberty3D Immersive Camera was added.
- The L3D UberRift Viewer was added.
- The L3D VR Headset Manager was added.
- The L3D VR Headset Item Info/Device Node was added.
- The L3D Oculus Rift Barrel Distortion Image Filter Plug-in called "L3DVR-OculusRift" was added.
- The L3D Surface Baking Camera was added. This is a highly experimental camera type at this time and is not documented in this manual.
- Several bug fixes and performance enhancements were made to the original cameras found in UberCam 1.9 and updated for compatibility with changes in LightWave3D 11.x and up especially when used in conjunction with LightWave's VPR.
- The L3D Key Protection system for UberCam has been upgraded in order to protect both our users and our development team from piracy while ensuring future compatibility with LightWave3D 2015 and up.
- Performance Camera has temporarily been removed for this release.
- DPI Camera has been temporarily removed for this release. It is now somewhat redundant since the appearance of Matt Gorner's DPI calculator in the LW 11.x cycle. We may put it back in later.
- This document was updated.

Why does the camera view not show what I expect?

The OpenGL view ports in LightWave Layout are hardware accelerated assuming that a linear transform can be applied, like that used in an Orthographic camera or Perspective camera. In these, a straight line in the scene will always end up as a straight line in the view port image. This can be done only for the Linear camera types in UberCam. For all other cameras we recommend using a visualization plug-in such as F-Prime from Worley Labs or LightWave 3D's VPR. Fully functional support for how images should appear in a F9 or F10 render compared to those of VPR may be limited at this time and may not perform as expected. This may change in future releases of LightWave 3D. We feel it would be a worthwhile effort by The LightWave Group to make this possible for all camera types when viewed in OpenGL.

Chapter 2 Linear Cameras Vs. Non-Linear Cameras

There are 3 linear cameras in the Liberty3D UberCam V2.0 package.
What is a Linear Camera?
A linear camera is a camera type that can be calculated using a 3D linear transform matrix. In this camera type, a straight line in the scene will always result in a straight line in the rendered image.

Liberty3D Perspective Camera
This camera operates identically to the perspective camera that is the default camera in LightWave 3D 9.3.1 - LightWave3D 2015 but has been optimized for speed.

Liberty3D No DoF Camera
This camera operates identically to the perspective camera that is the default camera in LightWave 9.3.1 - LightWave3D 2015, but assumes that there is no Depth of Field being used. By knowing this, the calculations that the Depth of Field rely on can be skipped as they are not used, adding further speed to the final render.
**Liberty3D No DoF No MB Camera**

This camera operates identically to the perspective camera that is the default camera in LightWave3D 9.3.1 - LightWave3D 2015 but assumes that there is no Depth of Field or Motion Blur being used. By knowing this, the calculations that the Depth of Field rely on as well as the Motion Blur calculations can be skipped as they are not used, adding further speed to the final render.

![The Liberty3D Perspective No DoF No MB Camera Option Panel](image)

**What is a Non-Linear camera?**

A linear camera is a camera type that CANNOT ALWAYS be calculated using a 3d linear transform matrix. In this camera type, a straight line in the scene will NOT ALWAYS result in a straight line in the rendered image. There are 8 non-linear cameras in the Liberty3D UberCam v2.0 package. These are the Stereo Camera, the Skybox Camera, the FishEye Camera, the Cylinder Camera, the Panoramic Camera, the SuperPano Camera, the Spherical Camera, and the Immersive Camera.

**Liberty3D Stereo Camera**

To make use of this camera in your scene you will need to change the width of the camera resolution to twice that of the resolution you wish to have a single Eye frame size at. For example - If you want to output in HDTV format which is 1920x1080, you will need to render your scene at 3840x1080. This will give you a horizontally wide image twice that of 1920 pixels (assuming square pixels). This will give you 1920 pixels per eye when recombined to produce the stereoscopic image.
Our Camera Renders both eyes at the same time which is, as far as we know a capability unique to LightWave 3D and this plug-in. The advantages to this are obvious. You no longer need to render a scene twice, once per eye in order to get the images you are looking for. This is why our camera renders the images side by side horizontally.

This is very advantageous when working with modern compositing packages like BlackMagic Designs Fusion 7.0 or higher (Please note that the free version of Fusion 7.x and up does not have a set stereoscopic tools, only the paid for Fusion Studio edition does) as it has a stereo image combing function that specifically can make use of horizontally stacked images and combine them to produce the stereoscopic image result at what is more or less a push of a button.

There are other advantages such as scene file management complexity being reduced and render times over the network are reduced on multiple fronts as well. In UberCam 2.0 this camera has been additionally performance tuned and as a result in most scenes it will very closely perform to that of traditional LightWave Stereoscopic rendering operations. We aim to actually go faster than rendering using the built-in stereoscopic rendering function in LightWave. Actual performance improvements may vary from scene to scene in comparison to the native stereoscopic camera in LightWave. The biggest noticeable advantage of the UberCam Stereoscopic Camera over that of the native LightWave stereoscopic camera is that operations such as "Moving objects" or "Radiosity Render Calculations" or any pre-rendering calculations that take place. By using our Liberty3D Stereo Camera these calculations that take place at render time are done once instead of each time per eye.
A note on frame display in Layout itself: If you need to accurately preview the framing of your scene through the camera view, you can use the Alternative Aspect Display function do so. You can enable this through the Display Options Panel (press d). As a good starting point, use the width of your resolution for your Stereoscopic target frame resolution. An example of this would be if your target stereoscopic frame resolution (the final combined stereo image) is to be 1920x1080, enter in a value of 1.92. This will loosely match the actual framing but not exactly. It's a good starting point, but even the default of 1.85 works right out of the box. Great stuff!

As you can see in the image below we have an example of the UberCam Stereo Camera set up in a scene where a Null Object, called "ConvergencePoint" has been parented to the camera. This Null Object is then selected as the convergence point in the camera options panel. The Convergence Point in stereo is represented by the blue rectangle drawn in the OpenGL view.

Proper convergence point management and eye separation settings, animated over time are key to making great looking stereoscopic imagery. We highly recommend making use of these functions and understanding how they work together to produce the stereoscopic effect.
An Example of the Liberty3D UberCam 2.0 Stereo Camera with LightWave's VPR enabled is below.

There are additional features and functionality improvements that have been made in UberCam 2.0 to match improvements made to the native Stereo camera system in LightWave3D 10.1 and above. The Liberty3d 2.0 Stereo Camera and VPR are for the most part completely compatible except in one odd case and that is if you enable the UberRift viewer window you will see a stereo image pair per eye on each side. The UberRift viewer is specifically intended for use with OpenGL (Point, Wireframe, Shaded, Texture Shaded, etc.) due to interactive performance reasons. If you want to use your Oculus Rift as an inexpensive stereoscopic viewing device, we recommend using these modes exclusively.

For production renders via F9 or F10 (and LWSN), where the OculusVR Barrel Distortion image filter plug-in that ships with UberCam 2.0 is applied will produce renders as expected when combined with our Stereo Camera. Should you wish to use the UberRift viewer with VPR, working in stereoscopic mode with the Oculus Rift, we recommend using our L3D Perspective No MotionBlur, No DoF camera with the stereoscopic mode turned on in the stereo tab under the camera options panel. This will give you best performance results.

**IMPORTANT:** A crash *WILL* happen if one renders either F9 or F10/LWSN using a native LightWave or UberCam Camera is stereoscopic mode combined with the L3DVR-OculusRift Image Filter applied in the processing tab in Layout. Do not use the L3DVR-OculusRift Image Filter in conjunction with any other camera but our own Stereo Camera. Always make sure in the camera options tab, that "Stereoscopic" is turned off. LightWave when told to render in stereoscopic through this tab option renders one eye and then the other independently. It is during the switch from one eye frame render completing to the start of render of the other eye that the crash will occur. **You have been warned!**
**Liberty3D SkyBox Camera**

This camera provides a simple way to create cross shaped skybox files, which can then be cut up into individual files as a post-rendering step. Saves time from having to set up and rotate 6 cameras every time a skybox render is required. The forward camera orientation is always rendered in the square that is the intersection square of the cross. In order to get a near perfect SkyBox Render which can then be split up into separate images for use in a game engine package such as Unity3D or Unreal3D we recommend using a 4:3 resolution.

For example, 1000x750 for each SkyBox side to be 250x250 pixels each. There are two modes supporting the two standard skybox shapes.

- Liberty3D Skybox Camera - Unfolded to Cross Shape
- Liberty3D Skybox Camera - Unfolded to Alternate Shape

You can switch shapes by using the dropdown box next to the text "Skybox Type" as shown in the image below.

---

**Liberty3D FishEye Camera**

This camera presents spherical lens distortion similar to that found in actual lenses. If the Distortion power is set to 1 and the Distortion Scale is set to 1, then the resulting image is identical to the default LightWave Perspective Camera. A distortion Power greater than 1 pincushions the field of view frustum inward. A distortion power less than 1 barrels the field of view frustum outward. That means parallel lines within the scene
show up bending in the opposite way in the final render. As the power value increase results in a smaller field of view, a second control called distortion scale is provided to scale back up the field of view frustum without changing the focal length. A check box control is provided to automatically calculate the appropriate distortion scale in order to perfectly match the selected Focal Length and Zoom Factor.

Finally, as there is no way to accurately represent non-linear cameras like this in the camera view when working with OpenGL, we alternatively draw the camera view rectangles in order to allow you to match camera shots within the camera view. FishEye Camera in action with Fprime 3.51 or VPR (in non-draft mode) being used to view the results.

A quick note: While we have worked hard on elimination of crash bugs dealing with VPR and our cameras working together at the same time, we recommend that you save your scene before making use of the mini-sliders in the options panel for the FishEye Camera. In the past, VPR has been very touchy when used in this scenario.
**Liberty3D Cylinder Camera**
This camera presents a panoramic view, but with the vertical axis having orthogonal perfectly horizontal direction of views from all points along a vertical axis. The controls are the length of the imaging axis, the maximum Field of View Angle, and the minimum field of view angle.

**Liberty3D Panoramic Camera**
This camera presents a perfectly hemispherical panoramic view presented from spinning around the vertical axis. The controls are the horizontal Field of View Angle, the minimum vertical field of view angle, and the maximum vertical field of view angle. The horizontal field of view assumes that half is on either side of the direction the camera is pointing.
As with many of these camera types, they cannot be accurately represented in OpenGL through the camera view port. However they can be viewed correctly working with Fprime 3.51 or LightWave’s native VPR in non-draft mode.

**Liberty3D SuperPano Camera**

This camera presents an imperfect hemispherical panoramic view. In addition to the controls on the panoramic camera there are additional controls to provide anisotropic scaling and shifting of the final image. These controls were added in order to help match existing panoramic camera footage, where the program that stitched the still pictures together did so in a non-linear fashion.

![The Liberty3D SuperPano Camera Options Panel](image)

**Liberty3D Immersive Camera**

A new camera in our collection for UberCam 2.0 is the Immersive Camera and it’s one of the most exciting features of our camera package update because of how it can be used or rather what you can do with it to make money. We have made it so as to be straightforward as possible but don’t let that simplicity fool you. This camera is specifically intended for use in the production of 360 degree spherical videos for use on Youtube.com.
LightWave artists can now produce their animations in a format that works with Google’s Cardboard VR and Android based smart phones directly using this camera type.

Chilton Webb has produced a quick video here that you can watch that shows this functionality. [https://www.youtube.com/watch?v=uDc679zKW50&feature=youtu.be](https://www.youtube.com/watch?v=uDc679zKW50&feature=youtu.be)

For a really kick ass example of what you can do with Immersive Camera check out this link. [https://www.youtube.com/watch?v=LdTm7Vpape0](https://www.youtube.com/watch?v=LdTm7Vpape0)

We've also included a still shot of this to highlight the little 4 corner circle in the upper left hand side of the video window on youtube.com This circle means you can navigate your view around using your mouse and see the action looking in any direction while the video is playing or even paused.

Click the image to watch the video. Make sure to check out your surroundings by left clicking your mouse on the video window and dragging it around while it plays.

A super simple interface but this camera can make you money!

The Immersive Camera in UberCam 2.0 on its own allows you to produce these spherical renders. That much is clear, but there is a trick to getting a resulting MP4 (we
suggest you use Fusion 7.x or QuickTime Pro to make your MP4 video from an image sequence render) video file onto youtube.com so that it is understood to be a 360 degree spherical video for use with cardboard and your android phone or be able to look around using your mouse via a web browser.

Google has produced a MetaData file injector that is super easy to use that makes this work. You can download the tool for Mac or PC here and find out more about how it works.

https://support.google.com/youtube/answer/6178631?hl=en

The Google 360 Degree Video MetaData Injection tool shown above.

Google and Youtube.com are both pushing 360 Degree VR videos on youtube heavily. At Comicon 2015 several companies, studios and productions handed out over 1.1 Million (or something like that) Google Cardboard units. That's a very big audience that
needs some stuff to watch and now you can make it happen. Also, Google recently announced that they will start injecting commercials produced in 360 Degree spherical VR video format. This means you can get in on the business of producing advertisements for clients who want 360 degree spherical videos for Youtube.com using LightWave3D and the UberCam 2.0 package.

**VERY IMPORTANT NOTE:**
It is important to note that you should use resolutions that are 2:1 (twice as wide as it is tall) when rendering out Immersive Camera spherical renders. This is critical if you want to produce absolutely distortion free material for use on Youtube.com in the spherical video player.

Suggested resolutions are 2048x1024 or 4098x2048. Youtube.com will process material even in these high-resolution formats. Of course the higher you go, the more rendering time is involved, but if you can, aim for 4K by 2K. This will make your videos stand out and look amazing. Google is watching!

One other recommendation is that when you produce your content, higher frame speeds like 30fps rather than 24fps will be better in terms of fluid playback and cleaner compression once YouTube's encoders step on it. Always use the best source material you can produce to feed their encoding technology and you will get the best results out on the other end. If you upload heavily or even modestly compressed MP4 format video you will regret the final results for sure, especially in this format.

**Liberty3D Spherical Camera**
Almost identical to the Immersive Camera is our Spherical Camera. This camera has found uses though in the world of hemisphere dome film production. The kinds of movies you would see at amusement parks or at planetariums. We have a few customers around the world that make use of this camera specifically for those purposes.

The only input function on the Spherical camera is the Angular Field of View input value. This function has a default value of 180 degrees which is pretty much standard for dome film projection set ups. In addition this function can be enveloped over time.
Liberty3D Radial Shift Camera

This camera presents controls to match actual camera bodies and lenses, such as used in panoramic stitching software like hugin or panotools, or in Denis Pontonnier’s Radial Shift camera. It uses a 4th order polynomial coefficient values to take into account various lens barrel or pincushion effects, including the difficult to match mustache effect.

This can be used so that rendered output files for compositing can be made to match existing footage at render time, rather than going through a distortion matching step during composition. Although the steps involved in calculating the radial distortion values are fairly complicated, there are databases of existing lenses and camera bodies that have already been determined and published on the Internet. If you have a certain camera and lens combination that you need information about, please refer to either the panotools website, the hugin tool website, or contact us directly here at Liberty3D.com. Chances are that the information regarding the lens and camera combination can be found fairly quickly, or be calculated from a captured still image.

The Radial Shift Camera options panel.
Chapter 4 Multi-up Cameras

There are 4 Multi-up cameras in the Liberty3D UberCam v2.0 package.

What is a Multi-Cam camera?

These cameras allow for multiple existing LightWave cameras in a scene to be rendered at the same time. In addition to this there is an OverScan Camera that allows a single existing LightWave camera to be rendered with additional control modifications like OverScan and jitter. Also, there is a DPI camera that it is really useful for people who need to go to print and works wonderfully. Other uses for Multi-Cam Camera set ups are pretty numerous. Working with "Holographic" reflective pyramids or boxes where reflections of renders are produced from a tablet or smart phone creating a "holographic" effect for product displays are a snap with a Multi-Cam Camera. Here is an example: http://thefutureofthings.com/5151-holho-make-your-tablet-a-video-hologram-projector/

There are many uses, so many that we can't name them all but here are some more ideas. Multi-Cam cameras are useful not only in outputting an existing stereoscopic camera rig that you have built in side-by-side stacked stereo files but also as a pre-visualization tool, where the scene can be adjusted and viewed from multiple camera locations. The 12Up camera can also be used to create custom skybox camera rigs.

Only one instance of Multi-Cam is allowed in a scene, and it is not permitted to select the Multi-Cam as one of the sub cameras. There is only one instance of an Overscan Camera allowed in a scene, which can then select a multi-up camera.

Liberty3D Dual Camera

This camera allows for two existing cameras to be rendered simultaneously side by side on the same render. The main use for this is for custom stereo camera rigs, although this is not the only use for such a camera.
Dual Camera set ups can be used with our L3DVR-OculusRift Image Filter to produce Oculus Rift compatible still images or animations. You can also use our UberRift viewer should you wish to combine real-time viewing of your camera view with the Oculus Rift.

**Liberty3D 4Up Camera**
This camera allows for up to four existing cameras to be rendered simultaneously in a 2 by 2 grid on the same render.

As mentioned before, one of the great uses of a 4 up camera is to produce "Holographic" pyramid style renderings for use with tablets and smart phones. You may have to flip your cameras upside down to get it to work right. Also, since our 4Up Camera renders each camera view in a quadrant, you will need to use Fusion or another compositing application to reposition these areas into a checker board arrangement with each image in a cross shape.
Here is an example of a flow set up in Fusion 7.x demonstrating what this looks like.

A 4 Up Camera Pyramid Example render is shown on the left, the flow in Fusion 7.x produces the arrangement of the Viper render as required to be reflected correctly into a HoloPyramid.

The image above shows the results of this workflow using a primitively built holo-pyramid display made out of old flat screen TV material and a Microsoft Surface tablet as the source video device.
If you choose to build one of these things yourself here are some links for you to check out.

http://www.instructables.com/id/Reflective-Prism/

https://www.youtube.com/watch?v=UFhhIOaGW7U

https://www.google.com/search?q=making+your+own+hologram+pyramid&espv=2&biw=1057&bih=682&tbm=isch&tbo=u&source=univ&sa=X&ved=0CDAQsARqFQoTCMGr_YWt8sYCFQq2iAodcAUG4Q&dpr=1

There is almost no limit to how large you can make one of these displays. You can even use a video projector or a large flat screen TV as your video source. This is another great opportunity to make money using LightWave3D and UberCam 2.0.

We have included the example Fusion flow showing the Viper set up in this package for you to work with as a start.
Liberty3D 12Up Camera
This camera allows for up to 12 existing cameras to be rendered simultaneously in a 4 by 3 grid on the same render.

The 12 up camera can be used to build highly specialized stereoscopic spherical rigs similar to those used in the real world to shoot 360 degree video. There are other uses for a camera of this type that we haven’t even thought of and we encourage users to experiment with it. We would love to hear from you and see what you come up with.

Liberty3D OverScan Camera
This camera allows for an existing camera to be rendered with an overscan percentage and/or horizontal and vertical jitter. If the overscan is used in addition to modifying the render resolution of the overscan camera output then the render step can recover the original rendered output in a compositing step from the overscan camera output, but would have additional pixels available if something just out of frame turns out to be needed during composition. In addition the shaky camera type of shot can be applied to existing cameras without complex camera paths by jittering using envelopes.
NOTE: If horizontal shredding occurs in the final renders, try using a single thread for the render. This is a known issue. Alternatively however this can be corrected for by baking out key frames on the envelopes and the camera will multi-thread without shredding the image.
Chapter 5 Oculus Rift Functionality and Use with UberCam 2.0

A big part of UberCam 2.0 is the added support for the Oculus Rift Virtual Reality Headset. Ubercam 2.0 provides for real-time rotational head tracking a special viewer known as UberRift and a image filter that let's you produce properly distorted renders for viewing in the Oculus Rift headset. This section details the use and functionality of these tools.

Before we get started though we need to cover off some things concerning functionality and performance.

Oculus Rift DK1.1 and DK2.0 headsets were used in the development and testing of UberCam 2.0 and while the head tracking functionality for rotation is supported from either unit, at this time the positional head tracking (depth, side to side, up and down motions) possible using the DK2.0 head set is not supported. We left this functionality out for this release but it is something we will be adding in once the Oculus Rift SDK evolves a couple of steps further as Oculus prepares for the release of the Consumer version of the Oculus rift in Q1 or Q2 of 2016. You won't have to wait that long however to get positional head track in Ubercam 2.0. We just want to see a couple more updates done to the SDK before including it.

In addition, while every effort has been made to optimize the performance of the UberRift viewer, the overhead involved in producing the floating view in UberRift combined with the real-time data monitoring of the Oculus Rift and LightWave Layout's OpenGL/Geometry engine performance means that UberCam 2.0 is not intended to be used as a platform for VR experiences as you would have in a game engine such as unreal or unity, but instead as a content creation tool.

Performance of the tools when used in combination with each other highly depends on your system and the limitations of LightWave's performance itself. For example, the OpenGL viewports in Layout are single threaded. It is for this reason that you will experience sluggish response while the UberRift viewer is in use even in lightly populated scenes on the average system. The barrel distortion mathematics used to distort the stereoscopic view is also fairly heavy in terms of processing time. For now in the interest of maximum compatibility the way that these tools are used in combination can put a heavy load on your system and real-time feed back may be impeded. We are working on a faster solution that reduces some of the overhead to a bare minimum and this will be provided as an update to UberCam 2.0.

Needless to say, we encourage users of UberCam 2.0 to stay up to date by making sure they check our website at www.liberty3d.com and staying subscribed to our mailing list. When we make updates you will be notified through these channels.

With these limitations understood, we will now show you how to set up UberCam 2.0 using Virtual Studio Tools so you can get the real-time head tracking rotation of the
Oculus Rift to drive the rotation of a UberCam 2.0 stereo camera and see the scene via the UberRift viewer through your Oculus Rift.

**Step 1: The Virtual Studio Tools Device Manager and Setting up a camera**

In order to get the rotational head tracking working, ensure that the plug-ins are installed and you have fired up a fresh instance of LightWave Layout. Your Oculus Rift should be connected, turned on and calibrated before you begin. For both DK 1.1 and DK 2.0 users make sure the Rift configuration utility is set up so that you are in extended display mode. This means that the Oculus Rift is acting as an attached display to your video card. For further information on this, please consult the Oculus Rift Configuration Utility Help and the documentation that comes with the Oculus Rift DKs.

Now then... In layout we need to start up the Virtual Studio Tools "Device Manager Window" before proceeding any further. This is found under "Virtual Studio Tools" dropdown menu commonly located just below the Image Editor button on the left hand of Layout’s interface.

Yup. It's there, just click "Virtual Studio" and you will see a Device Manager button. You want that. When you select that button you will get a window much like this that comes up below.
Enable the L3D Oculus Rift check box as shown.

Once you have this window up, you need to check both of the “Enable” boxes for the VRHeadSet Manager. This essentially “activates” the Oculus Rift and Virtual Studio Tools is now listening for the data coming off the unit. You can minimize this window for now but we will come back to it later.

Next, select your camera that you want to make work with the real-time rotational head tracking being driven by the Oculus Rift in Layout. From here we want to add a motion modifier.
Hit "m" to access the motion modifiers panel. An example of this is shown below.

![Motion Options for Camera](image)

An example of the Virtual Studio Trait Motion Modifier added to the Motion options of a camera.

The motion modifier you want to add is called "Virtual Studio Trait". Once you add this, you should see the Trait much like it is shown above. From here double click on the area where it says "Trait (Takes:1 Active:1 dur: 0.00sec.)".

This will expand the window displaying a button called "Studio" and a check box called "Relative". Click the Studio button to continue to the next step.
Step 2: Setting up nodes for the ItemMotion:Camera Trait.

We know that this area of LightWave can be daunting but it's really not that bad. Once you click studio you will be presented by a window called "Studio". Yeah... Which looks like what we have below.

This is the Virtual Studio Tools Studio Window and while it doesn't look very interesting, it's where a lot of magic happens. From here, we want to double click the "Edit" text under where it says "Nodes". You can actually double click anywhere along that line to continue on. Once you have done that you will be presented with a very dry looking window entitled "Node Editor - Trait:ItemMotion:Camera".

The Node Editor. Don't worry, it's not evil.
Next, you want to add a "Device" node. You can do this by typing "Device" into the search box and then double clicking it to load it into the editor window or by finding it under the dropdown list "Item Info" and then double clicking the word "Device". This will add it to the into the nodal editor window area. It should like this once you do that.

![Device node added to the editor window.](image)

On its own this isn't going to do anything for you and as you can see it doesn't have any "connections" on its right hand side that you can feed to the "Trait" node just yet. First we need to get this device node to be "something". To do that, double click in the gray area inside of the Device Node itself and you will find a new window that pops up called "Trait:ItemMotion:Camera:Device...". This is a Device Manager component. From here we need to make some selections using the drop down arrows for "Manager Name" and "Device Name". Don't worry, this is all standard Virtual Studios Tools stuff that has been around since the release of LightWave 11.0. We didn't do this to make your life complicated. It's just how it needs to be done.

From here we need to get this Device Node to do some work for us. In the dropdown menu for Manager Name, select "VRHeadset Manager" as shown below.

![Select VRHeadset Manager](image)
Next we need to do this for "Device Name".

When you are done both slots will be occupied much like the example below and you can now make your connection in the node editor taking the Rotation output from the Device and feeding that into the rotation input of the Studio Trait.

Everything is good.

Notice in the example above that the text displayed in the grey area in the "Device (1)" node is green. This is a good thing as it means everything is set up here correctly and you can move on.

**Step 3: Making it move.**
From here Virtual Studio Tools can be used as normally would be done with any other input device but in our case the Oculus Rift headset is driving the rotation of your camera. If your moving your Rift around and you don’t get any movement, make sure the "Active and Live" buttons turned on in the Studio Panel.
Make sure those buttons are on if you want your headset to rotate the camera when you are moving it.

If you want to turn the Oculus Rift Virtual Studio functionality off for a moment, just disable the "Live" button. Recording takes with your Rift works in exactly the same way as any other input device that works with Virtual Studio Tools. Please refer to the LightWave 11.0 documentation regarding further information regarding Virtual Studio Tools and how the different panels work, recording and playing back takes, etc.

**Performance Tuning and Centering:**
There is one more small feature to this area that we want you to be aware of and that is the VRHeadset Settings Function. This is found in the Device Manager window for Virtual Studio tools.

In the image above, next to the "Enable" check boxes where it says "open..." on the bottom line saying "L3D Oculus Rift" you can click that and get a little box. In this box are
two options that are very handy. Polling Interval (ms) and a default value of 100 and "Recenter Headset" button.

The Polling Interval deals with how often we ask the Oculus Rift Driver for an update on what it's rotational position values are. The default is 100ms but this is actually quite high. Internally in the Rift, all of its gyros, magnetometers are updating very fast and the Rift is sampling these updates at over 1000 sampling cycles per second. Because we can't synchronize LightWave's Virtual Studio Tools, the Layout OpenGL drawing rate and the Oculus Rift hardware and driver all together, we came up with a way to help users dial things in, although loosely. This value can be reduced to smaller numbers like 30ms or even less for increased responsiveness, but it can also be increased to higher numbers so as to reduce the interruption that we are making to the process which can impact other things such as LightWave's interface menus drawing themselves, OpenGL updates, etc.

We recommend that users play around with this number moving up or down in 10ms steps when they dial in their set ups.

As for the "Recenter Headset Button" this is fairly self explanatory. This button recenters the camera relative to the direction that the headset is pointed. We should point out that you may want to do this from the get go when setting up your scene. Experiment with this function and you will get the hang of how it works in seconds. Also, while this VRHeadset Settings options box is open, Layout stops doing anything until you hit continue. So no, Layout didn't lock up on you and there is no need to swing your Rift all about trying to get it to respond. You have to hit continue in order to get things going again and interactive with Layout's interface.

Once you have your Oculus Rift driving the camera motion, we want to see something through it.

Let's move onto the next step.

**Step 4: Configuring your camera**

The appropriate camera to use in real-time Oculus Rift set ups for UberCam 2.0 is our own Liberty3D No DoF No MB camera. Alternatively you can use the native perspective camera. However, if you do have the system power available to you to run VPR at the same time, we recommend our No DoF No MB camera because it turns off the checking for motion blur and depth of field. This check is done per pixel in LightWave’s native perspective camera (as well as other cameras even if not supported) and this will slow VPR down considerably; something we are already trying to avoid in general.

Again, we don't recommend people try and use their Oculus Rift with the UberRift viewing window and expect to be able to look around in a fluid motion. This is incredibly tasking on even the most powerful systems. It was our intention from the start to make this all work using OpenGL views like bounding box, vertices, wire frame, etc. Still we allow for VPR to be used but please understand what our design parameters were in the first place for this update to UberCam.

With this considered, here in the example below we have our Liberty3D No DoF No MB camera in a scene pointed to an object in layout. Take note of the camera resolution
settings of 960x1080. This is the native resolution of the Oculus Rift DK2 VR headset per eye. You can also see that the Stereoscopic Rendering option in the Stereo tab of the camera is enabled. This will by default produce an anaglyph representation of the scene through the camera but don’t worry, it stops there. More on that in just a second.

For DK 1.1 set ups, the recommended resolution, matching that of the DK1.1 display is 640x800.

Now you are probably wondering why not full HD at 1920x1080 for use with the DK2 or 1280x800 for the DK1.1 unit? Well in short this is because the width is representative of each eye in stereoscopic inside the displays for each developer kit. Multiply the width of each eye by two and you get the full resolution of the display for each unit. Simple.

While we are on the subject of stereoscopic, we should point out that Liberty3d.com was instrumental in advancing LightWave’s stereoscopic capabilities during the 11.6.3 cycle. It is because of us that you have more than just anaglyph stereoscopic rendering available to you in Layout. Today in 11.6.3 and above you now get additional options found in the stereo tab under the camera panel including convergence point (taken from our Stereoscopic camera) the stereo tracked eye option and different stereoscopic OpenGL modes. Under the Stereo OpenGL dropdown box where it currently says Anaglyph you can set this to be “3D Glasses”. If you choose this option, that ugly anaglyph OpenGL representation goes away, but under the hood, Layout is still making the OpenGL camera view port data available in stereo to quadbuffered stereoscopic enabled video cards such as AMD’s FirePro line or nVidia’s Quadro line of products. IF you don’t have one of these types of cards, don’t panic. We take advantage of this through our UberRift viewer so you can see the stereoscopic effect in that window which can then be sent to the Oculus Rift.
To do this, we need to access the UberRift viewer window. You can find it by clicking on the little set of 3 lines which happens to be VPR's options panel access button.

Access the VPR options window by clicking that little button.

From here you will be presented with the VPR options panel window. No we are not going to mess with VPR here, instead we are going to add a our UberRift Viewer to the "Add Display" drop down menu.

Click "Add Display" and add "UberRift".
Once you have added the plug-in to this list, double click it so that the "Toggle Window" and "Full Screen" buttons appear as shown below.

Here is a screen grab that show the UberRift viewer open and the options available to you in the VPR options panel window.

The UberRift viewer will appear and you can now drag it off to the side display where you have your Oculus Rift connected. You don't have to get it all the way onto the screen, just more than half. From here you can hit "Full Screen" on the VPR options window and it will maximize itself to fit the display, in this case the Oculus Rift.

If we could see both displays at once, they would look something like this in the image below, with Layout on the left and the Oculus Rift display being feed the layout camera viewport, in stereoscopic no less with the proper barrel distortion applied to each of the eye views, on the right. Notice how both views lack that ugly anaglyph filter but the Oculus Rift display view is in stereo? Cool huh? You betcha it's cool!

Yeah. It's cool.

At anytime you can use either the toggle window (for when you want to move around your scene quickly repositioning the camera) or the full screen button so you can move the display around. To disable the viewer entirely you can either remove the plug-in from
the drop down list in the VPR options panel window or uncheck it where it says "On" next to the name "UberRift: View 0".

A couple of small notes before we move on. First, yes - it is possible to have more than one UberRift Viewer active at one time. While this would be incredibly tasking on a system, we do allow for it. Even though not many people have more than one Oculus Rift available to them but we didn't want to limit people's options and this was done to make certain things in the future possible. Be advised that only one Rift can be used to drive the selected camera at a time. Second, if you save your scene in with the UberRift viewer loaded in the VPR options panel and come back to it later, you will get a harmless error that pops up on scene load. In order to get the viewer to work again you will have to re-add the plug-in to the VPR options panel like before. This error message isn't very descriptive and simply says "Instance creation failed. %1" with an ok button next to it.

![Loading Scene Error](image)

A somewhat harmless error when loading a scene that has the UberRift Viewer active when save and then reloaded. Just hit ok and your scene will load.

Normally just hit ok and your scene will load just fine.

**HOWEVER,** we recommend that if you ever want to render this scene over the network using LWSN that you remove the UberRift Viewer before you save the scene and send it to your render farm. Otherwise it will stop the rendering process on this error, and it will sit there waiting for someone to hit ok on the button which can't be seen by anyone - especially your render controller software. **You have been warned.**
Chapter 6: The L3D Oculus Rift Barrel Distortion Image Filter

The L3D Oculus Rift Barrel Distortion Image Filter Plug-in is provided as another simple tool for those who want to make "lean back" stereoscopic content for viewing with the Oculus Rift or to check how something will look in all its fully rendered glory using LightWave3D's rendering engine.

The L3D Oculus Rift Barrel Distortion Image Filter Plug-in can be accessed through the image processing tab or by hitting Cntrl+F8 (or Command+F8 on the Mac) under the "Add Image Filter" dropdown menu box. In the list of plugins in that area the L3D Oculus Rift Barrel Distortion Image Filter Plug-in is called "L3DVR-OculusRift". Simply select it. It doesn't have any options that you can change so don't worry about that part.
As we mentioned earlier in this document, the L3D Oculus Rift Barrel Distortion Filter is ONLY to be used with our Liberty3D Stereoscopic camera. Use with other cameras and the native stereoscopic functions in LightWave will produce a crash. If you enable "stereoscopic" under the L3D Stereoscopic camera tab this will also produce a crash. So don't use the native stereoscopic options here.

To set up the Liberty3D Stereoscopic camera, ready for producing a proper image result we will use a normal resolution such as 1920x1080 in the case of the DK2. In general we recommend this resolution anyway even if you are looking at it on a DK1.1 because the display with down res the image to fit.

The L3D Oculus Rift Barrel Distortion Filter works on normal width images in this way because if you think about it 1920x1080 is the value of the right and left eye (960x1080 each) combined side by side.

While it isn't required, you can make use of the convergence point function in our Stereoscopic Camera options panel. Once you have things set up you can simply hit render and once the render is complete you can save the image out or render a full animation and then play it back on the Rift through various commercial and free players. You can also maximize your image viewer from layout on the rift display.

Here is an example of the settings normally used and the render result.

F9, F10 and LWSN renders using the image filter will produce images as you see above.
Chapter 7 Troubleshooting
The following section covers common questions about Liberty3D UberCam for
LightWave. The section also covers a list of known issues. If you have any questions,
suggestions or comments about Liberty3D UberCam for LightWave, e-mail
kat@liberty3d.com

Common Questions
The following table contains answers to some common questions about
Liberty3D UberCam for LightWave.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are volume purchases available?</td>
<td>Yes, contact <a href="mailto:kat@liberty3d.com">kat@liberty3d.com</a> for more information.</td>
</tr>
</tbody>
</table>

Are any other cameras being planned?

All suggestions are welcome, and will be evaluated based on viability and usefulness.
We have several new cameras that we want to build and are working on them based on
our own production needs and priorities but we are always willing to hear your thoughts
and comments. Please send them to us!

Known Issues

Lens Flares do not work in multi-up cameras correctly. Why?
Lens flares and other effects which are the result of a post-render compositing step will
not work correctly in a multi-up camera, such as the Stereo Camera. This is because
they assume that a linear camera transform is being used in order to determine the
location of the lens flare light effect. The current workaround is to use comparable
volumetric light effects in order to obtain the desired visual effect. A more efficient
solution is being examined for Liberty3D UberCam in the future.

Cameras parented to Other objects or items may not perform correctly. Why?
Some beta testers have reported to us that when certain cameras are parented to other
objects or items in the scene, that the camera doesn’t perform exactly as expected. One
known instance of this is the stereoscopic convergence plane in the StereoCam
Camera. If this camera is parented to an object while the convergence plane is being
previewed, it will generate the plane at the world axis point instead of in front of the
camera at the appropriate convergence distance set in the properties panel for the
StereoCam Camera. This issue and others may take place when parenting UberCam
cameras to other items in the scene. We are looking into what causes this and will
update the package immediately once a solution is found.
Overscan camera with MB and jitter shows horizontal tearing on multi-core machines or with Multi-Threading Enabled in the render globals panel. How do I fix this?

Some beta testers have reported to us that when the overscan camera is used to render a motion blurred scene rendered with a multi-core system using multiple threads, there are horizontal tearing lines in the final image. This can happen when certain motion modifiers or textured motion tools are applied to the jitter envelope itself in the graph editor. Some of these tools such as Noisy Channel are not able to work with the Jitter function without first baking the key frames on the envelope that Noisy Channel creates. Baking the key frames of the envelope will eliminate any issues with shredding of the image during render while using multi-threaded rendering. We recommend you do this anyway if you are going to render your shot over a network as some machines may be different processor types and translate the randomness different that others, giving you undesirable results - however in our testing on mixed machine networks at Declaration Pictures in Vancouver and in use on several productions by our users and beta testers around the world this problem has rarely popped up.

A final note:
Thank you again for purchasing Liberty3D UberCam 2.0 and supporting our development efforts! We appreciate your support and welcome your ideas on new tools and products. Suggest them by visiting our site at www.liberty3d.com and signing up to our forums or send us an email! Make sure to check out our other great tools like QuadPanels and Weighter2 for LightWave Modeler. More tools and video tutorials are always being released and we look forward to seeing your images made with LightWave3D and our products. You can submit them to the gallery area in our forums at Liberty3d.com

Liberty3D UberCam
For NewTek’s LightWave3D
Version 2.0