Two variants of Texture2HeightMap are available: a plug-in and a HLSLobject. Each has its own merits.

Texture2HeighMap (plug-in)

General

Texture2Heightmap transforms all pixel of a given luminance or RGB texture into vertices of a 3D height map, and stores them persistently in a 3D ObjectData channel. The new populated 3D Object Data can be used as a source for collision detection. Texture restrictions apply. High dynamic range and monochrome textures are supported too. In addition, provisions are made to generate compound terrains of any size, set weights in the luminance computation, and to set mapping techniques.

Children

1. IN Source RGB/RGBA Texture
2. OUT 3D ObjectData
3. IN RGB Channel Weight (vector), optional
4. IN Object Shift (vector), optional

Description

A typical usage would be to generate a terrain with uniformly spaced vertices. The geometry fits into the upper half of a box with a size of 1 under normal conditions. The geometry stored in the 3D ObjectData channel can be reused anywhere. Texture requirements: width and height must be equal and explicitly set, upper limit 2048.

The height is derived from a luminance computation with default weights of 0.29, 0.58, and 0.1 for the R, G, and B channels, or the luminance in monochrome textures. The default RGB weights can be overwritten with the third child to allow special weighting. So you have three independent channels at your fingertips that can be used in any fashion. Negative weights may even result in negative elevations. You can choose to clamp negative elevations.

Object Shift provides a means to shift all vertices. This can be used to produce compound objects with the Command “Add object data“, or to adjust the medium height. The default value is (-0.5, 0, -0.5). See the appendix for further discussion.

The computation process has to be invoked by a CallChannel parent.

A call from a Value parent delvers the current channel state, without invoking a computation. Possible results are

1	success
0	not called yet
-1	general fail
-2	trial expired (trial version only)
<= -10	Texture preparation error

See the reports in the debug window for details in error conditions. Texture preparation errors are:

Code	Reason	Remedy
-10	Texture Format not supported	Start with a JPG for LDR and a *.HDR for HDR textures. See Appendix A1 for details
-11	texture size > 2048 x 2048	Reduce size
-12	none-squared texture found	Prepare texture with equal width and height
-13	-1 encountered	Set Desired Height and With in Texture property panel explicitly

A SetValue parent call to this channel sets individual states:

10 clamp negative elevations
11 allow negative elevations (default)
40 Set PlanarMapping to support Aerial Textures (default)
42 Set PlanarMapping to support False Color Heightmap Textures (experimental, reload required to take effect)

Tips

- Use regular Matrix operations to stretch the Object, and to set the height scale.
- Actual resolution limits may depend on local resources. Exceeding local limits may result in runtime errors. Try starting with an empty 3D ObjectData channel, if you encounter such problems. Or use lower resolutions.
- Height resolution is not limited to 8 bit. Using a *.HDR texture allows to take full advantage of floating point precision. As a starter, use *.hdr files for floating textures, and *.jpg for 8 bit textures.
- Collision detection works fine, usage in physics is under investigation.
- A typical application scenario is to call the channel on demand only. (my testbed: 220 msec for 1 Mpixel). The computation time needed may exceed real-time requirements.
- Weights are normally in the range 0. to 1.0. They may be negative too. And of any magnitude. None-standard values allow creative results, including negative elevations.
- Avoid multiple computations, set OncePerFrame.
- Extreme vertical slopes may be subject to color degrading.
- Consider the GPU bases solution “HLSLobject Texture2HeightMap” as an alternative.
- Floating textures (*.hdr) vertical scale may exceed the half-box boundary, if the pixel values are out of range 0.0 to 1.0.
- Typically you want to use a lower resolution for the height map, and a higher resolution for the actual texturing of the surface. This requires the provision of two distinct textures.
- A change in height map resolution results in different surface normals. This leads to different render results, if other than plain ambient lighting is involved.
- Consider erasing 3D ObjectData content before saving a *.cgr with the Command “Clear object data”.
- Weights cannot be applied to monochrome textures.
- Consider monochrome textures, if storage cost is an issue.
- Alpha channel is not used.
- Use plug-in **CollisionObjectCommand** to control CollisionObject updates.

Legal note:

Commercial use requires an explicit license agreement. Trial availability is restricted. Acknowledgement: This channel is bases on previous work published in the Quest3D forum under the topic “HeightMap Channel”. Thanks to the author “Lukas” for making the sources available.

Contact: quest3d.godbersen.eu

Supported Quest3D Versions

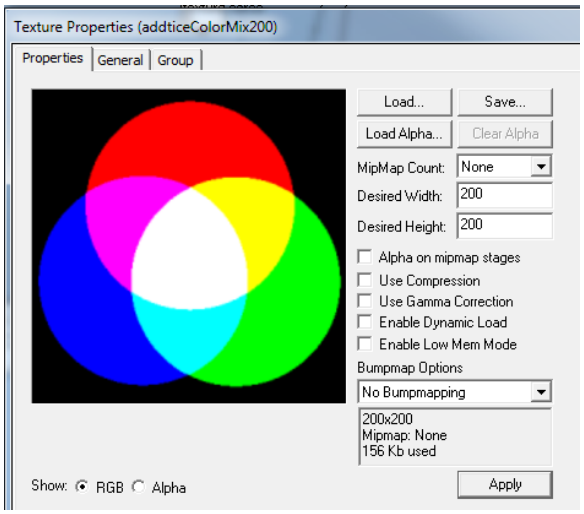
- 4.3.2
- (5.0 under investigation)

Known Problems

1. Remember: Approach quest3D resource limits with care. Make backups of your *.cgr files. Use the compound mechanism to derive at very large 3D Objects.
2. Compound Objects keep their initial UV coordinates. Manually create a new texture UV set and use planar Y mapping

Tutorial:

It is mandatory to pay attention to explicit settings in the Texture Property: MipMapCount None, Desired Width/Height explicit and equal number. And don’t forget to hit “Apply”.



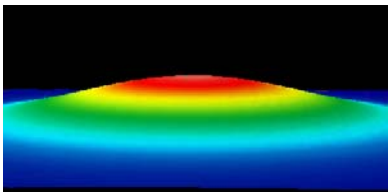
For annotated sample *.cgr explore this shipment. The list includes:

- LandscapeHDRdemo (show difference between LDR and HDR, monochrome and RGB sources.)

- LandscapeGeneratorDemo (Crater Lake Oregon, 3 source resolutions, different surface textures)
- LandscapeProbingDemo (Crater Lake Oregon, external texture, collision detection)
- CompoundTerrainGeneratorDemo (combine single terrains, sample produces 1 Million vertices)
- WeightedExtrusionDemo (select color weight for extrusion)

Using False Color Heightmap Textures

The picture to the right shows the result of False Color mapping to a hill. You achieve this effect by setting the corresponding mode with a SetValue call of 42 (experimental). A new call is required to see the effect.



Appendix A

A1 Supported Texture Formats

Typ	Color channels	#bit	Format	D3DFMT_*	Code	File type
LDR	monochrome	8	unsigned int	L8	50	JPG, PNG, ..
HDR	monochrome	16	unsigned int	L16	81	PNG
HDR	monochrome	32	float	R32F	114	DDS
LDR	RGB	24	unsigned int	R8G8B8	20	JPG
LDR	RGBA	32	unsigned int	X8R8G8B8 A8R8G8B8	22 21	
HDR	RGBA	128	float	A32B32G32R32F	116	HDR

- The file types are just examples.
- The size requirements can be 8-fold with multichannel HDR. Consider monochrome sources as an alternative.
- The three or four channel formats are not restricted to RGB, but can be produced in any other color model (CMY; LUV; ..).
- If you use RenderToTexture (RTT), and use the command “Save render texture in texture”. Set the format accordingly.

A2 Producing Compound Terrains

This feature allows you to add many individually produced objects together and let them act as a single object, e.g. in collision detection. This allows you to produce non-squared terrains, to end up with very large terrains, or to overcome runtime limits of Quest3D. Special attention has to be taken to reach seamless transitions between the tiles:

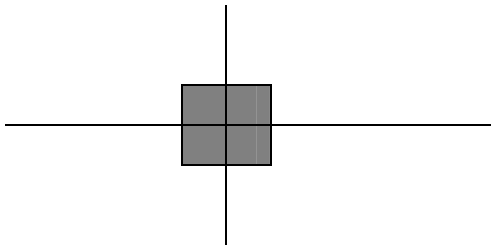
Preparation: You have to ensure, that the borders between tiles align. Cut your source texture in appropriate small square pieces[1], or find a seamless texture.

Note:

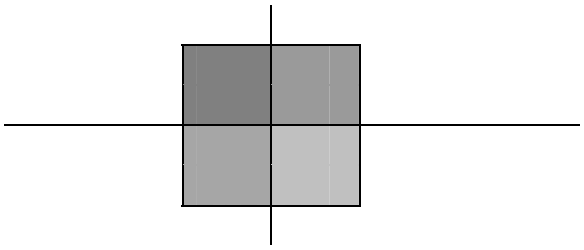
1. In order to approach a seamless visual appearance, consider overlapping, and/or repeat the border pixel elements in the texture cut process.
2. The resulting Object has no uniform texture coordinate space. You have to manually set new mapping: Create new UV Set 1, and Planar Y. This means: Not applicable in published projects!

Seamless texture Manually cut

Production: Produce the first Object and store the 3D ObjectData with the Command “Add Object Data”. Then, load an appropriate texture and generate the second object with a shift of 1 unit (x, or z). This will place the new geometry next to the old one. Repeat these steps as long as desired. The final result is a single Object.



Default (x,z): -0.5, 0.5



Upper left tile: -1, 0
Upper right tile: 0, 0
Lower left tile: -1, -1
Lower right tile: 0, -1

[1] Border pixel must reoccur on neighboring tiles. Example: To get two tiles with a size of 10x10 each, the source must have a size of 19x10. The centre row is reused in both tiles.

Texture2HeightMap (HLSLobject)

General

The HLSLobject variant of Texture2Heightmap extrudes all vertices of a given 2D Object Data child along the normal, according to the luminance of the corresponding texture position. The shader supports one point light. LDR and HDR textures are accepted. As the computation of takes place in the GPU, high throughput rates can be expected. LDR, HDR, monochrome, and RGB textures are supported.

Children

1. Underlying geometry (3D ObjectData)
2. not used
3. Technique:
   0 Point Light and emissive color, (default)
   1 emissive color only,
   2 point light only
4. Height Multiplier (Value), default 1.0
5. Ambient color (Vector), default black
6. Diffuse color (Vector), default white
7. Specular color (Vector), default black
8. Emissive color (Vector), default white

Additional input: Surface channel: 1. RGB texture used as height map, 2. regular surface texture, Light channel: position, point mode.

Description

A typical usage is to compute extrusions from a predefined 3D object. In case of planes, this could be landscapes. Other examples start with a sphere.

Tips

- Extreme vertical slopes may be subject to color degrading.
- No support for collision detection or persistence. Consider the CPU bases plug-in “Texture2HeightMap” as an alternative.
- A change in height map resolution results in different surface normals. This leads to different rendering results, if other than plain ambient lighting is involved.

- Create your geometry in your favourite 3D software and export it to Ques3D, or use one of the samples provided in the *.cgr. The Quest3D Primitive channel is not suitable.
- No texture format restrictions.

Legal note:

Commercial use requires an explicit license agreement.

Contact: quest3d.godbersen.eu

Supported Quest3D Versions

- 4.3.2
- (5.0 under investigation)

Known Problems

Light computation has some weaknesses.

Tutorial:

The source HLSLobject Texture2HeightMap is stored in the file **Texture2HeightMap-HLSLobject.cgr.** Keep a version of this file for reference.

This file provides some 3D ObjectData samples for reuse (planes, spheres, torus, box). Please note, that basically any imported 3D Data Object can be used.

For annotated sample *.cgr explore this shipment. The list includes:

- LiveTerrainGeneratorDemo (allows different source geometries, different source textures including video, different visualizations)
- SetTexturePixelHDRdemo (requires SetTexturePixel plug-in, trial version OK)