The RANCH RenderFarm User Guide
Part II – the RANCH for TERRAGEN
www.ranchcomputing.com

14-06-01 – June 1, 2014

Welcome to the RANCH automated rendering service, the super-powerful - and affordable - supercomputer for all your Terragen projects! This document contains information specific to the use of Terragen on the RANCH. Before reading it, we highly recommend you read Part I first - the General Guide - which includes everything not specifically related to Terragen.

- The RANCH renders projects with the 64-bit Windows versions of Terragen 2.x and 3.x

- The oldest Terragen version supported on the RANCH is 2.4.

- We support animations (one frame per node) and still images (MultiBand rendering).

1) Important information about scene preparation

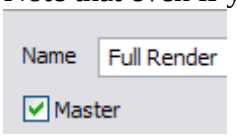
It is **VERY** important that you read this section carefully before submitting a scene. It will help you avoid common mistakes.

- Please verify before sending the scene that you have saved it with all the correct parameters. It is your responsibility to provide the scene with every parameter correctly set. The scene will be rendered exactly as you send it.

- Do not use accentuated / non-alphanumeric characters, spaces, apostrophes and dots in the filenames of your project (scene, textures, objects, etc.). To avoid any possible parsing problems on the RANCH Runner, please use only letters and numbers. You can also use the "_" sign (underscore) to replace spaces.

- For image output, never specify a video format (mov, avi, mpg...) as they are not supported on the RANCH. Always specify individual animation frames rendering with a standard bitmap format, like TIF or EXR.

- If your scene includes several render nodes ("Renderers" panel), only the "Master" node will be rendered, so make sure you have checked the node you want to render. Note that even if you have only one render node, you must check the "Master" square.



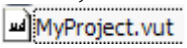
2) Sending your project to the RANCH

As the RANCH is entirely automated, you need to send your scene in a project archive which contains all the files needed for your scene to be rendered correctly. Follow the steps below:

1) When your final .tgd scene is ready, use the '**File / Export Gathered Project...**' function to save your scene, along with all its content (objects, textures), in a new directory. Do not forget to include the GI cache(s) if needed (see sections 3 and 4).

2) You can now **use RANCHpacker** to convert your Terragen project to a ready to render project archive. RANCHpacker will compress the files into a RANCH for Terragen project archive (.vut) that you can directly upload to the RANCH.

Alternatively, if for whatever reason you can't use RANCHpacker:

- compress all the contents of your project directory into a **.zip** or **.7z** (7-zip) archive
- rename the zip archive to a **.vut** file (make sure your OS does not hide the extension from you, the file must have a **.vut** extension, not a **.vut.zip** extension - a common mistake)
- your project is ready to upload! 

3) Projects using Global Illumination (GI)

Global Illumination (GI) produces realistic lighting results, but rendering GI projects on a renderfarm must be done carefully in order to avoid typical problems (flickering in animation, lighting discontinuities in still images).

GI with still images (see 3.1)

When rendered on the RANCH, a still image is subdivided into many bands, each band being rendered by a different node. By default, each node will compute its own GI solution, and the end result will be an image showing clear discontinuities when all the bands are recombined:

To avoid this problem, it is necessary to compute a GI solution first and to save it in a GI cache file that will be included in the project. Each node will then use this GI solution, and the image will not show obvious tiles anymore.

GI with Animations (see 3.2 - 3.3)

When an animation is rendered on the RANCH Runner, each render node is in charge of one or several frames of the sequence. In the case of a GI project, if we do not take necessary measures, each render node will compute its own GI solution and apply it while rendering. In the end, a flickering effect will ruin the animated sequence because each frame will have a different lighting. To eliminate this problem, it is necessary for every node to use the same GI solution.

To do this, individual GI solutions are first computed for all the frames and stored in GI cache files: this is the *prepass* phase. Then, during the rendering phase, the render node will interpolate between several of these GI cache files (corresponding to the rendered frame and to the frames immediately before and after the current frame). It will then use the result of this interpolation as the GI solution for the current frame; this technique produces clean and flicker-free animations.

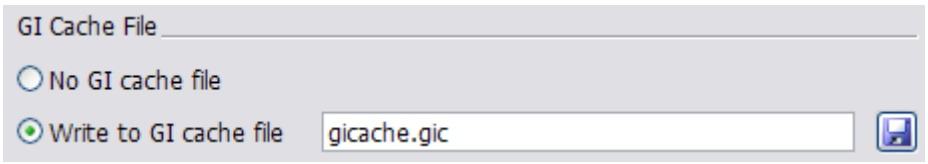
3.1) Still image project using a Global Illumination cache

You must compute the GI cache on your system and include it with your project. It would make little sense to compute it on the RANCH, since:

- this operation cannot be parallelized, so we would not be much faster than your system, and you would still have to pay for it.
- the GI cache computing is very fast compared to the rendering part, and you can often compute it at a lower definition than the final image to save additional time.

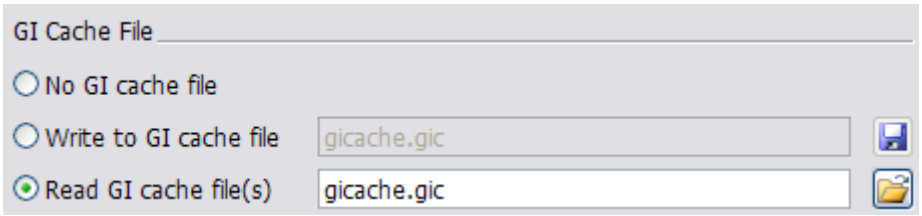
1) First go to the render options, click on the **GI Settings** button

and choose the option **Write to GI cache file**:

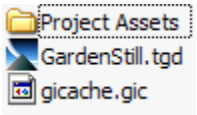


enter the name (without any path) of your GI cache and click on Render Image.

2) When it's done, go to the GI Setting once again. This time, select the **Read GI cache file(s)** option instead with the name of the GI cache, and save the scene.



3) include the GI cache file in your project, next to your .tgd scene.



You can now prepare your project as indicated in section 2.

3.2) Animation project, method 1: included precomputed GI cache files

With this technique, you compute the GI cache files yourself on your system and include them in the project. Using the 'sparse cache' technique (generate a GI cache every n^{th} frame) will give you the best image quality.

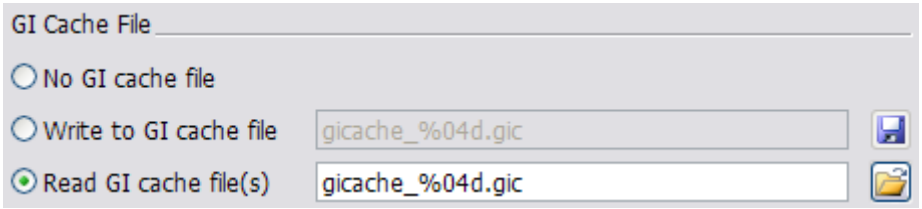
Advantages

- you will only have to upload and render one project on the RANCH (instead of two with the method described in 3.3).
- the final rendering will be slightly faster as all the GI files will be stored on each node and will not have to be fetched on the fly across the network.

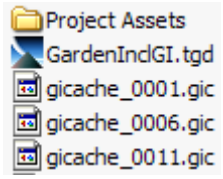
Drawbacks

- Unless you have a fast upload speed, the size of the project archive may become too big for you to upload it to the RANCH in a reasonable time, because of the included GI cache files.
- it may take a while to render the GI cache files on your system, depending on their number and the settings you choose.

The GI settings of your RANCH project must be configured as below:



Important
Make sure that all the included GI files have the name **gicache_xxxx.gic** (e.g. gicache_0001.gic, gicache_0002, etc.):



otherwise the server won't recognize the GI files and the project will be rendered without them.

3.3) Animation project, method 2: the Prepass/Rendering technique

With this technique, the animation project is done in two steps, that is two RANCH projects:

- The first project is used to compute all the GI cache files
- The second project does the actual rendering. Each rendered frame uses as its GI solution an interpolation between the previously rendered GI cache files.

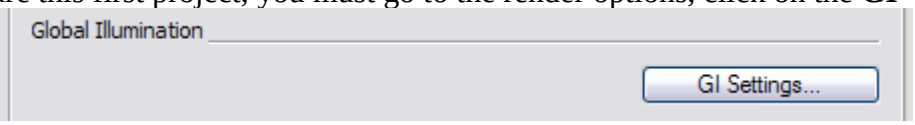
Advantages

- The size of the projects to upload is smaller: you don't have to include the GI cache files.
- The speed of the RANCH is used to render the GI cache files.
- You will be able to reuse the already computed GI cache files for several projects (or one big project split into several smaller projects).

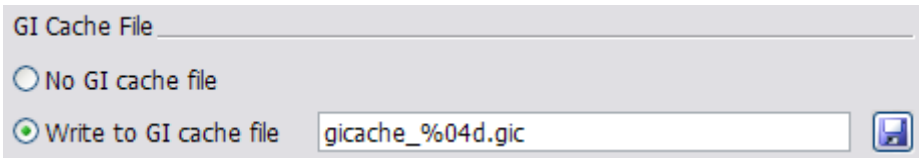
Drawbacks

- You will have to upload and render two projects on the RANCH (instead of one with the method described in 3.2).
- The final rendering will be slightly slower as several GI files will have to be fetched on the fly across the network for each frame to render.

First project: GI cache generation (*Prepass*)

When you prepare this first project, you must go to the render options, click on the **GI Settings** button 

and choose the option **Write to GI cache file**:

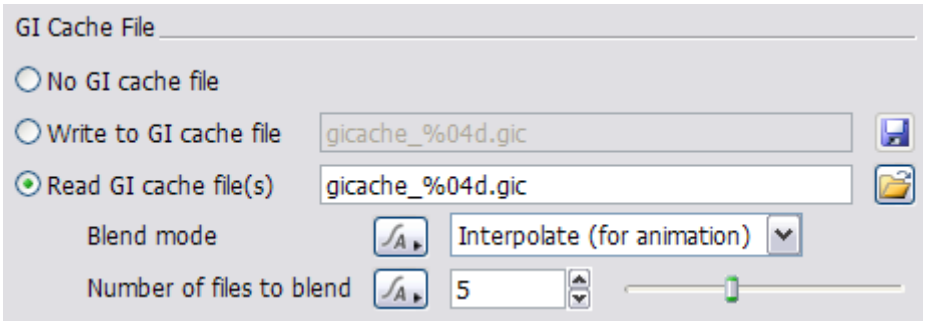


Note that the RANCH will force the output name of the GI files to **gicache_%04d.gic**, whatever the name you enter here (in order to easily find them for the rendering phase).

The RANCH supports the 'sparse cache' technique, which gives the best image quality. You just have to configure your scene to generate a GI cache every n^{th} frame (every 5 or 10 frames for instance) by entering this value in the 'Sequence step' field of the "Sequence/Output" tab:

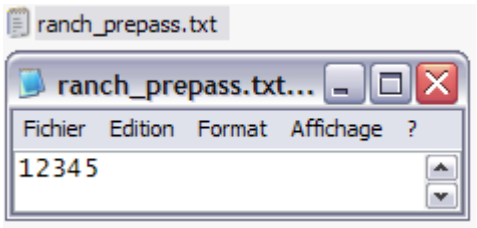
Second project: Rendering

For the rendering project, select the **Read GI cache file(s)** option instead, choose **Blend mode Interpolate (for animation)** and then the **Number of files to blend**: values of 3 or 5 are generally sufficient to achieve a very good image quality with the 'sparse cache' technique.

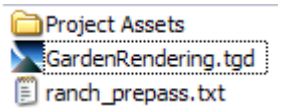


Note that, for a Rendering project, the "Sequence step" parameter is always forced to 1 by the RANCH (it is not possible to use a frame increment greater than 1).

Then - **very important** - create a text file called **ranch_prepass.txt**. This file must contain a single line: the ID number of the RANCH project used to generate the GI cache files. For instance if the first project ID was 12345, then your **ranch_prepass.txt** should be:

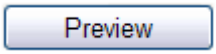


Include this file in your project archive, next to the .tgd scene. That way, the automated server will know where to find the GI cache files:



You can now prepare your project as indicated in section 2.

Appendix A : frame preview

The RANCH offers you a very handy feature for checking visually your animation project when it is being rendered. It displays 256-pixel wide thumbnails of a large sample of rendered frames on a web page specific to your inprogress project. To access this page, you just have to click on the 

button which appears when your project is being rendered (and of course if there is something to preview: if each frame of your animation takes 30 minutes to render, obviously there will be nothing to see during the first 30 minutes :)

Below is an example of what you will be able to see when you click on Preview (the preview image is always around 1900 pixels wide, its height depends on the number of thumbnails).

Notes:

- At the end of the render, the preview image is also copied in your project's directory.

- The preview function may not work with some graphics formats; in that case it will display black frames or nothing at all. That does _not_ mean that your project has a problem of course. You can still check its progress in % in the queue.

Appendix B: high resolution still image rendering - technical info

When rendering an animated sequence, each node renders a different frame at a given time. But the process is very different when rendering a high resolution still image: the image is split into many bands (or strips), and each band is rendered by a different node. When all the bands have been rendered, they are all gathered across the network and recombined to form the final image.

Estimating the total render time of an animation is relatively easy if you know the number of frames and the average render time per frame. But for still images, it's not that simple because the total render time depends on the repartition of the complexity in the image, which can vary considerably with each scene.

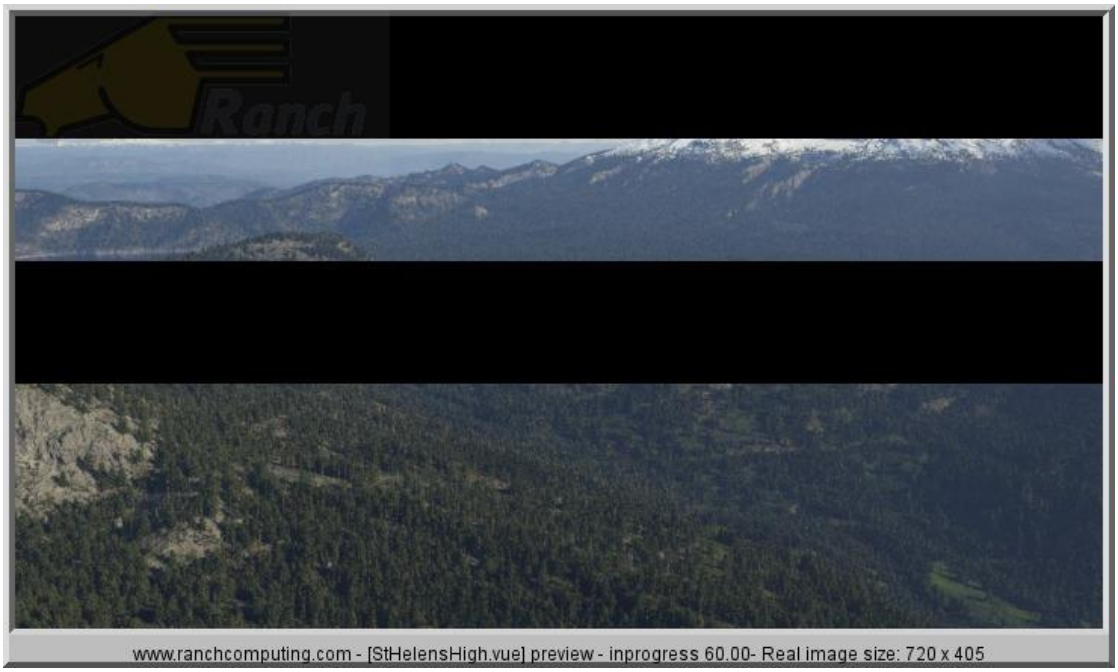
1) Concept: things to know

The RANCH is able to render Terragen high resolution still images using MultiBand rendering, a technique which has several advantages over traditional distributed rendering, and that we also use to render high res stills with 3ds Max and Vue xStream.

- it scales linearly with a large number of nodes, when classic Distributed Rendering quickly shows diminishing returns when the number of render nodes is increasing. With MultiBand rendering, each still image is efficiently processed by all the RANCH Runner nodes.

- it allows automatic stitching and gathering of partial images if the project is interrupted (manually or by an automated time limit). In that case, you don't lose everything and can complete the missing parts on your system by rendering regions and compositing them with the partially rendered image.

- it lets you observe, in near real time, the progress of your render by clicking on the **Preview** button. The preview image is updated by the automated server every minute or so.



Still image preview while rendering (here at 60%)

2) Preparation: things to do

Please read the following instructions thoroughly to prepare your still image project. It is important to respect a few simple rules in order to achieve the best performance and image quality. Please be aware that there will be no refund if the result is not what you expect because you did not follow these rules.

- As a general advice, do not send a still image project to evaluate the render time of animation frames: you wouldn't be able to deduce anything from it. In still image mode, each node computes a part of the image, and all these parts are recombined to form the final image. In animation mode, each node computes an entire frame, so two very different processes.

- MultiBand rendering on the RANCH works with the following bitmap formats: **TIF, BMP, EXR (16 bits/channel)**. If you choose another format, the project will be rejected. Please note that even if you choose EXR with 32 bits per channel, the final image output from the RANCH will always be at 16 bits per channel.

- If you use Global Illumination (GI), it is **essential that you include your GI cache file in your project** (see section 3 of this guide). If you don't, each node will compute its own GI solution, the result being a bad image with lighting discontinuities (banding effect).

- The higher the complexity, the more performance gain you can expect. Avoid sending too simple scenes - even as a test - as they won't reflect the real performance of the RANCH (much of the time would be used by overhead such as network traffic rather than actual rendering).