Brender v0.5 alpha
User Manual

Document is work in progress !

Intro
Brender was first developed as an in-house organic render queue management tool. For this very reason it does keep some obscure and unstable things, use at your own risk. We will make all our possible to help you, but some things are weird :)

Important Remarks:
Security: Brender web interface was not designed to be accessed from internet, please keep it on a trusted local network. There is absolutely no security guaranteed.

1. Installation procedure

You will need apache mysql access on the machine you plan to use as server.

- Create a network drive/folder on the server machine that all computers will be able to access, for the rest of the document we will call this folder BRENDER

- Unzip files to *BRENDER*

- Point apache documentRoot to BRENDER folder, or create an alias

- Go to your mysql administration and create a **new user** called brender, make sure this user can access mysql remotely, from any host (it is needed by the clients who do not access the mysql database locally, but remotely)

- Create a **new database** called brender.

- Visit the webpage to continue the installation. It should detect and create the database structure needed. It should also create the following folders in your brender root :
    - blend (where the blend files will be placed)
    - blender (blender install folder)
        - mac (this folder will contain the blender.app file)
        - lin (for linux binary)
        - win (for windows binary)
    - thumbnails
    - render
    - logs

2. Using brender

setting the machines up :
In the web interface go to the *clients* tab, and add a new client. Enter a unique name for the client.
On your server, in a terminal start the server by typing *php brender_server.*

In the client machine, start also a terminal and type *php brender_client.php CLIENTNAME*
Back to the web interface check to make sure the client is in the client list, and is idle.
You can click on disable to `disable` it, then click `enable` to re-enable it.
Make sure you have installed blender in the BRENDER/blender folder with the mac/linux/windows version you want to use. Or use the custom blender path option on each client.

managing projects:
In the web interface, go to *settings* tab and click the *manage projects* button at bottom.
You can now create a new project. Give it a title and type in the different paths.
The *blend path* is the exact path needed to access the .blend files of your project from the BRENDER root folder. That means that for instance on a mac you can access files on a remote volume by typing /Volumes/my_remote_volume/my_cool_project/blend/scenes in the mac_blend

adding a job to render :
Click the *new job* button in the menu.
Choose a project from the list.

Type the name of the scene (in the sintel example it would be *01*)
Type the name of the shot (which is actually the blend file name) so in our example it would be *01_01_sintel_enters.*
Choose fileformat, config, chunks, starting frame, ending frame.
If *Directstart* is thicked, the job will directly start rendering, otherwise it will be set to waiting.

Example structure :
Typical 3d project folder structure that works well with brender) :

```
lib/
    chars/
        sintel.blend
        dragon.blend
        dog.blend
    props/
        weapon.blend
        shaman_hut.blend
scenes /
    01/
        01_01_sintel_enters.blend
        01_02_guardian_attacks.blend
    02/
        02_01_dragon_dances_the_lambada.blend
        02_02_sintel_runs.blend
render /
    01/
        01_01_sintel_enters
        01_02_guardian_attacks
    02/
        02_01_dragon_dances_the_lambada
        02_02_sintel_runs
```

Notes :
jobs priority: lowest number get rendered first (you have to think like this “this job is my priority number 1, so it renders before job with priority 10”)

F.A.Q.
Why are there 2 connect.php ?
The one in the *brender* root folder is accessed by *brender_client.php* and *brender_server.php* . These 2 scripts can happen to be run through network drives, and so the connect.php data should point to the mysql server that brender is running on.
The other one is in /web, and is access by the php/apache system, usually a localhost access is enough in that case.

How does brender dispatch job tasks? is there load balancing ?
To be short, brender is just a render queue manager. That means no load balancing. It will render first the job with highest priority. If during the rendering of job A , someone adds a job B with even higher priority, the job A get halted until job B is finished (or paused/canceled)

why is this so complicated with the project paths ?
this was a feature to be able to have the projects files on a machine, while the server runs on an other machine... and it allows to render and output directly to for instance the compositing machine. This also ensures that if you can render the file locally, it will render on the farm as well... There is no need between the server and client to send or recieve files/textures/sim baking/cloth baking/external libraries. Everything is “locally”.

What is progress status :
This is a little indication that can be used to show the production progress of the specific shot. We tried to include the most obvious ones.

How can I run the scripts in debug mode ?
just add the word “debug” at the end of launching command.
Example :

```
./brender_server.php debug
php brender_client.php macpro debug
```

How does the config file system work ?
The config files are python script files that are launched before the rendering. You can use them to set the desired render output dimensions, enable Ambient Occlusion or whatever you want. One nice use of them is to switch a characters model to HighRes before rendering, so your animators can continue to animate LowRes, while the file gets rendered in full blown beauty. For more infos please check the sample.py files in the /conf folder

Is brender working with compositing nodes and file output nodes ?
Yes you can use the file output nodes to for instance output your different passes. You must use relative path though.
In the example project, shot *01_01_sintel_enters* could have one pass for Sintel, and one for the background. Just create 2 File Output nodes and make their output path like this

“../../render/01/01_01_sintel_enters/sintel_pass.exr” and “../../render/01/01_01_sintel_enters/background.exr”.
You could even add one additional folder “../../render/01/01_01_sintel_enters/sintel_pass/sintel_pass.exr”. This allows you to keep image sequences in folders, and not have all your pass sequences in the same folder

How does it work internally and other technical remarks:

There are 3 main components :

1. the database
   this is where all the information is stored : the list of all available clients, the projects and jobs.
2. the *brender_server.php* script
   this is the main php script that will check clients, and job to be rendered. It will then dispatch rendering task to idle clients. To do so it will add a render order to the *orders* table. This order will have a *rem* that is the actual blender command line to be run in background mode (*rem* includes the path to the .blend file and the path to the output)
3. the *brender_client.php*
   is a php script that is run by a client. it will loop through the orders until it finds one that is adressed to himself. orders can be of type enable,disable, render, benchmark etc. Once this order is executed, the *brender_client.php* will erase that order(as it is finished)

The render order syntax :
  basically it is :
  $blender_path -b $project_blend_path/$scene/$shot.blend -o $project_output_path/$scene/$shot0001.$FORMAT -P $config.py -s $current -e $current+($chunk*$number_of_processors) -a