# Right-sizing and optimizing the use of images.

## Overview

A common question for most decorators is how to best use images as textures. Obvious there is the need to balance performance with awe inspiring graphic experience. This document will try to bring some reasons and method to the table.

First we need to proper distinguish in concept of Lag and a discussion of FPS and Delay, two somewhat different measurements. Then it seems beneficial to explain a little of what happens to the graphics engine that allows us to see a believable 3d world.

Finally it would be useful to come with a number of procedures and suggestions on how to best organize images.

While the document strives to be readable for everyone, it would certainly help to have a fair exposure to decorating.

## RLC is a special Chat server.

A regular heard misconception is that RLC is some sort of TV with a big server to play the world. It really is not. RLC can best be described as a chat server for computers. What happens is that when we arrive at an instance we do get a list of props and list of images locations sent. Our computer tries its best to recreate the scenery based on that info. Then all moves, all effects, animations are performed local in our own PC. Regularly when we move or engage in a different animation (dance or walk) this information is sent across to other people in the same location and our graphics engine is trying to recreate that.

A few observations that you might have experienced:

- After a repair you see half people or get only partial filled props or see default textures while everyone else in the room can see all.
- You might see a significant better quality image (no black filling lines in not optimized images, or better shaped especially under oblique angles.)
- A friend on Skype screen share has similar events however not at the same moment as your screen.
- Arriving at a zaby your flash player or music playlist / mixpod is playing an entire different song. Butterflies and other animated objects are not even remote in the same area.
- A person arriving at a busy club seems to float in the air nearby, then get clothes and or legs and then poofs to the entry point.

## Granny Engine

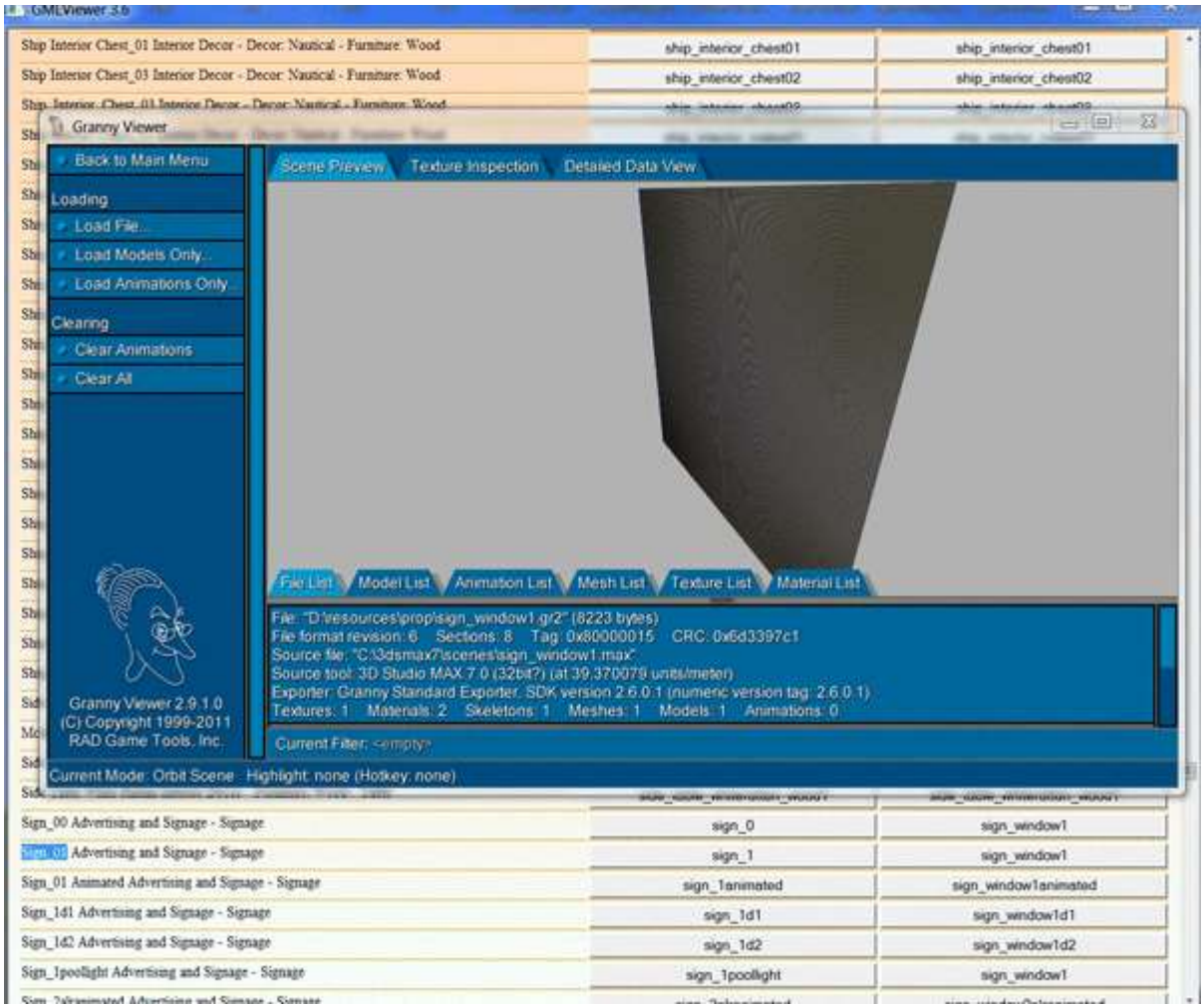
Writing code that relative simple allows to place objects and animate them is far from simple. Currently only a few companies exist who have spent millions of dollars into creating a library that can be used to create a believable 3d world. There is Direct3D from Microsoft and OpenGL from Silicon Graphics. On top of those are then game engines and render engines… pieces of code to bridge the gap towards a relative simple prop. One of the oldest and best known is Granny from RAD Game Tools http://www.radgametools.com/granny.html. Other engines exist… noticeable Unity3D

on which the next RLC client will be based (http://unity3d.com/gallery/demos/live-demos Do have a look at Tropical Paradise and Substance demo… droooool).

For the technical inclined: I am aware that the term engine is loosely used in this document. I make no clear distinction in what happens in what part of which CPU or memory bank, nor will I talk about bus type, pipelining etc. Partial that is because no longer keep track of the rapid changes in ability of graphics card and CPU. In all openness I no longer care and am too old. If you like to know about 286 chipset, 386 and in lesser amount 486 and things like bitblitting, sprites etc… I am your man… however I really have not kept up with the latest programmable vertex shaders and CUDAs. Also: it does not seem to matter for the topic of understanding how to improve our ability to deco.

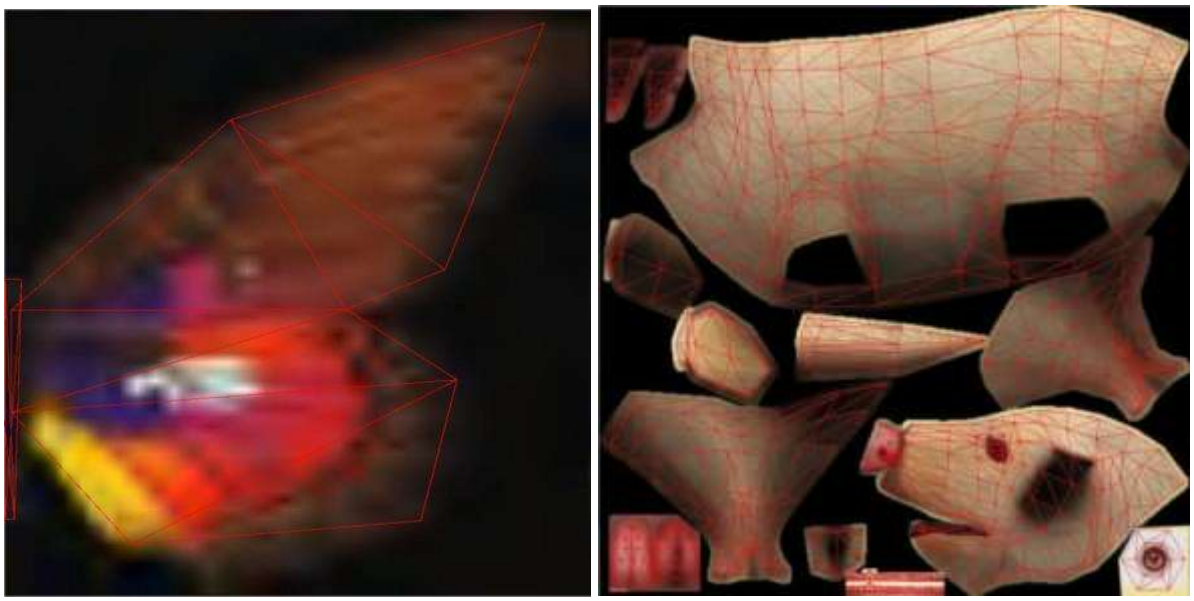
## A prop

In RLC we use props. This can be thought of as a simple definition of something which (like a human avatar) has bones (animations work on bones), some sort of mesh or chicken wire around those bones (like skin tissue) and one or more images that are strung or mapped on top of that mesh.




On reason why this viewer is really nifty is that it allows us to see how the map is wrapped around the mash. As is visible from the butterfly… not all the image is used, only the triangles as indicated by the UV map. (In case you like to know why it is called UV… the space… mesh that is… is defined in X,Y and Z. Since it would make every meeting about graphics twice as long if we had to add… "I mean the X of the Mesh instead of the X of the Map… smart dudes used the letters UVW. Since in a 2d

map there is no depth or W axis this became hence the UV map. Also… notice that the UV map is NOT pixels. It simply is a percentage like number (single or float).



So what the granny engine does is trying to keep up with loads of props like furniture and avatars, some of which are animated and move location. The speed or lack of it can be seen as the Loading lag… the initial time we have to wait when we enter a new instance and the drawing lag, as how fast, responsive and fluent the image appears to be while we are in the instance.

To my knowledge the viewer is no longer available at the RAD website. That said it still is on several download mirrors. Without claiming any right or ownership and under the impression that such is not against the original intention: http://sifupetertools.com/Deco/gr2_viewer_setup.exe

## Initial Delay Lag … as expressed in delay time when we enter an instance

From the previous we realize that our computer has issues to solve when recreating the environment in a believable fashion. It needs to get instructions on what props to display, then on where to get all the textures from and get each and every one of those from internet or cache. And in addition it needs to move props or avatars by scripts. All this greatly depends on things we do not have much influence on, such as internet speed, type of connection to the internet and hardware configuration.

### Internet speed

Much of the speed in which that happens depends on the speed from the internet. A reasonable way to get some indication would be to use some speed tests available on the internet. There are several (avoid things that look like System Checkup or Computer speedup kits, etc… that are evil programs. Look careful for just the speed test). http://www.speedtest.net/

To read it what is important is to understand the difference in bit and Byte (notice capitalization). A byte is like a letter or number. A bit is the computer way to describe On/Off… 8 bits is a Byte and can be thought of as a way to represent a letter. (Just in case you are a geek: 01000010 is the letter B)

Since internet providers like to sound more rather than less they commonly express the speed of the line you pay for in bits per second. This means that if my internet speed is 10 Mbits per second that it can download about 1.2M Byte per second. Then there is something like framing and retransmissions. This further reduces the actual usable bandwidth even more. In general: divide the speed of your internet by 10… whatever the number… that is about how long it will take for you to get the image, song, flash etc. Files sizes are expressed not in bits but Bytes.

### Wireless issues.

Electricity does not do so well with air. Over time we have grown familiar with wireless radio and TV and our mobile phones. However in such devices we can live rather well with lack of precision. Even a second drop of music or temporary disconnection from mobile we can survive or fill in. No such luck exists on transfer of files. There is no point in sending a file that is partial damaged. So retransmission will be significant higher when your internet is wireless. If you experience terrible load times and constant crashing… consider loaning an Ethernet cable between your router and your PC or laptop and comparing the behavior. You might be massive surprised.

### Caching and hardware.

With the regularity on the forum the question pops up about clearing the cache. While the sometimes heated debate is more forum style rhetoric than real hostility, there is a point to be made: files do NOT suddenly go sour past some date nor does somehow our computer decides to do things today different from yesterday. The reason why internet explorer works reasonable fast with places we often visit is that it strongly relies on keeping files local in a large cache. If you would open an HTML file you would see links called URL to images. The browser has a large index and can quickly figure out if that file is already somewhere on the disk. With speed of Hard Disk several million times faster than the internet there is clear benefit in using as much caching as we can. Obvious everything is cacheable, or we would see the same news as first time we ever went on CNBC. Browsers understand directives that tell how quick content should be considered outdated.

The discussion on how exact RLC handles caching is outside this topic and a tad sensitive. Simply said, it has folders that contain all props and most of the images in use. Access to that is regulated by instructions when we arrive at an instance.

In the editor there is a **script** called **Cached Web Image**. Caching can be set to be Never… which means: do delete the image at first opportunity, preferable when leaving the place. Or per Session: delete the images as soon as a person is booted and or logs of and gets a new session. Or Forever: keep the image till next repair. The sensitive part revolves around the question… can an evil person access those images and in effect steal what you created. The sad answer is yes, however he or she will use a much easier but non disclosed method. So while correct there really is little point in punishing yourself and other zaby or VWW visitors by a vain attempt to make it difficult by clearing cache.

Tip: **get an SSD**… its speed is mindboggling fast compared to normal hard disk. A 100G will cost about 100 dollar. And easily handles both your OS and the Utherverse Client Software

### Delay or Ping time

Another concept that we should understand is that the amount of data sent is not the same as the time it takes to get sent. Compare a postman who might be very efficient and do large geographic areas and loads of letters yet takes much more time than a courier service which is far less efficient yet does it shorter.

This delay time is measured in milliseconds. Higher numbers mean that it takes more time to get from your place over the internet to RLC and back. This number is important because all the communication wants some sort of confirmation that it arrived properly and will wait if not received in time.

One cause of crashing in RLC is that the chat server demands that all PM and local is properly handled. You can try by sending a PM to someone just arriving online. The computer might be soo busy with other things that it does not establishes a correct chat communication and the person boots again.

## Ways to improve loading speed

All the above was intended to clarify that certain factors cannot be influenced by deco. The visitors Internet speed, delay time, the speed of hard disk, the existence of SSD or the type of computer are factors outside the control of the decorator. There is however a fair bit of things we can do to create a reasonable fast loading experience.

### Have your own server

If you can afford a few dollars per month… get a server with sufficient bandwidth and faster download speeds:

http://www.hosting-review.com might give some suggestions.

While not many servers have the elegant features of Photobucket or Imageshack, many do outperform the non pro account significant. I personally measured difference in servers ranging from about 0.3 seconds for the complete arrival of an image to close to 10 seconds.

## Use proper sized images

A common issue is that the images in use are significant larger than ever will be visible. This is effects greatly the initial lag experienced when arriving at a place. Further down is a more complete discussion how size effects Lag by low FPS. There are many images that are essential effects in a zaby and many that are not noticed. By measuring the most efficient size you should be able to achieve a balance between beauty and loading lag. Most image editors have settings to save. They often do so in an elaborate format required to allow future editing. Photoshop has an export to Web that does a great job in compressing and sizing it to minimize download speed. Other programs like RIOT might also be beneficial.

## Integrate images

Say you want a bathroom wall with a towel, several mirrors, light switches etc. you might consider building all that together in your drawing program. This could also allow you to use shadow, soft edging and other techniques that make a huge difference in loading time and performance. Here a single PNG which fits in a bookshelf. Instead of six PNGs.

One of the first templates I used. A setup with normal PNG layer and depending on the client some window screenshots, and some 30 or so props that could be switched off and on to create several unique looking fronts.

Seven props into a single jpg… loads faster, looked much better .

Here some PNG artwork was offset against the wall. Shadow is added. The result is considerable better and faster than the original. In addition: There is not texture filtering between props. This makes an edge always look rather ragged. Notice the following image. On top are 4 pixels already in the color of the ceiling. This makes the jagged edge between ceiling and wall optical disappear.

Another advantage is that we could use light and shadow way before RLC allowed that.

## Use the proper format

Images that can be used in RLC are PNG and JPG. The oldest type and special created to allow for transferring photos of real objects is JPG. The default format is however destructive in the sense it does contain less information than the original. Best avoid re-editing a JPG… it looks quickly really crappy. There is really no substitution for the wonderful images JPG can provide unless you have to use cut outs.

The PNG file is from much later date and was mostly developed to allow for non-rectangular shapes required by buttons. As result it does have ability to have transparent areas. Images you create yourself from some relative simple set of colors / Adobe Illustrator / Vector images might be surprising small. So if you design like a texture for a club that has some wild yet limited colors in it… try certainly PNG. Example … JPG is 3113 Bytes,  PNG is 387 Bytes.

## Use the proper URL

Servers do need to check the full URL … so links directly from the address  bar often contain session information. It takes the server time to figure out that the session info no longer exists before it finally gives us the correct image. Also, often servers have a browse interface… where you have all the upload features… and a much faster backend.

In the case of Photobucket… both following links will work… obvious the last one is significant faster.

http://s471.beta.photobucket.com/user/LightSensei/media/SekaBeachTowel8h.png.html?sort=4&o=19#/user/LightSensei/media/SekaBeachTowel8h.png.html?sort=4&o=19&_suid=135441221387505994651871418999

http://i471.photobucket.com/albums/rr79/LightSensei/SekaBeachTowel8h.png

Notice the S471 and I471 host name … the one with an “I” is just spitting out images as fast as it can.

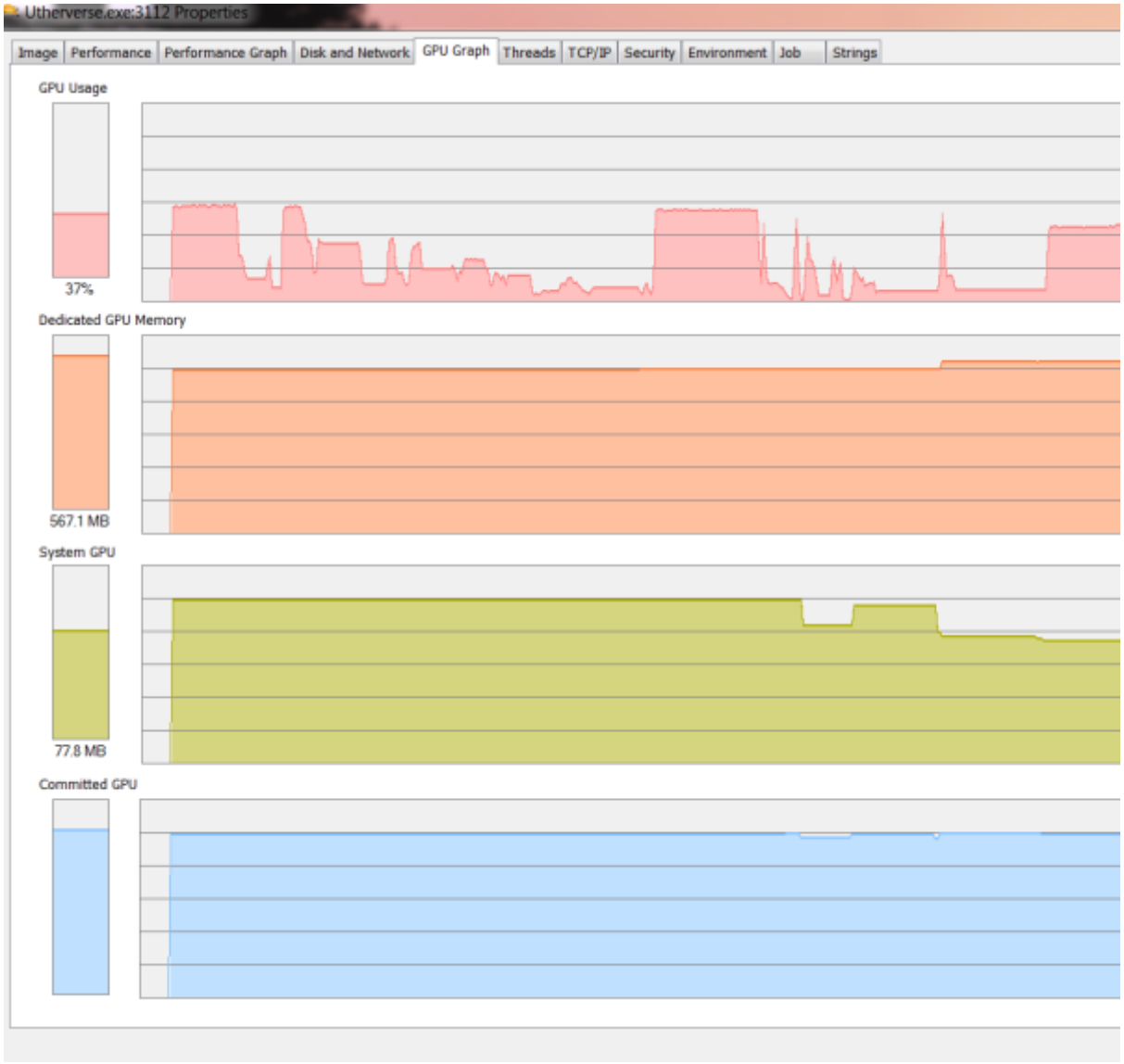
In short… avoid to burden a server with additional instructions… anything in the URL that looks like a ? or & will require inspection by the server.

## Measure your own creation

From all the tools available for decorators I personal enjoy the little gem that comes from this Microsoft website: http://technet.microsoft.com/en-us/sysinternals/bb896653

This tool allows you to see several key parameters and might provide some lovely new understanding of RLC. In class we will have a look at how to use it.

The tool provide great info on how much data made it to your PC, how a zaby loads, how much bytes are involved and then has info on how those combined images are making demands on Graphics card memory and calculation power.




This tool also shows how the CPU deals with the load. What is further useful is that it allows to take a time capture. This is useful in understanding how small tunes can have beneficial impact on deco.

## Lag as expressed in slow graphics.

Next we will look at an entire other type of lag… one dealing with how fluent the PC appears to operate and handle the graphics it needs to display.

Again there are some parts we as decorator have little influence about like the type and speed of the PC a visitor is using and the type of graphics processor. There is however a fair bit we can do to optimize the experience as much as possible.

We will begin with understanding some ideas on how a graphics engine works and see how we can use that to our benefit.

### FPS or Frames per Second

The measure of speed expresses how quickly the combination CPU and GPU together deal with creating a single Frame. For each Frame the avatar is considered fixed … and in some state of animation like dance. Each subsequent frame we could move and animate altering some of the calculations done on the previous frame. The faster your PC is… the quicker it crunches all the drawing stuff and the more Frames you have per Second.

### Drawing, how does it work

Think of the following. Say you want to make a painting of a nice sky, snow topped mountains in the background, trees nearer by, a log cabin with a lovely lady in front. At first glance we might set out to make a sky… then paint all the mountains on top… then paint all the trees on top of the sky and mountain… then paint the cabin… and keep on painting objects closer by till we are finished.

We would quickly realize that we could save a lot of work if we would begin the drawing with some outlining… as in we really need not paint the sky behind the mountains behind the trees etc. Obvious outlining has its limits. Some parts of sky and trees behind the cabin might be visible through the windows.

In order to make a realistic cabin we will have to use perspective. While we know the wall might be rectangular we still draw it as a trapezoid… the near end is larger than the back end.

We also notice that while we might spend a week on making the snow look realistic we after some time realize that most people would not have the eyesight to see the details that are so far away. So there would be benefit in spending time based on distance… nearby and larger objects get more attention.

Such principles also exist in computer graphics. Names that are often used in 3D graphics are Z-Buffering, Occlusion Culling, Binary Space Partitioning, Transformations and Zoning and some other important stages required to make the entire scene.

RLC is based around the Granny engine from RAD Game Tools.  An SDK exist as well as a viewer of granny objects. This is useful as it allows us to look at the original textures and UV lines.

## Transformations

The process of drawing begins with creating a list of all props and their positions, scale and rotation. Based on the view direction a quick object culling then filters out objects that are outside viewpoint.

## Perspective correction

To make a believable 3d world visible on a flat screen some nifty stuff is involved. While the object might be rectangle and the corresponding triangles nice and regular… those need to adjust in order to create a bend corresponding to what we experience in real life. This btw is the reason why objects are expressed into triangles… much easier to transform those than other shapes like rectangles. After this step there is a long list of triangles. That sound daft at first yet has a good reason. Instead of having to imagine how each pixel part of a body has to behave with light and texture, by reducing the objects to loads of triangles the computer saves considerable time by only having to do the calculations per triangle. So effective all dots made up of the same triangle will have similar light and bending effects.

## Binary Space Partitioning

At first glance each object is the same amount of triangles as it is designed in the granny engine and this would mean that the computer simply adds up all the triangles of all the props and starts drawing. That however is not the case.

Firstly we can omit all triangles that face away or are behind us, greatly reducing the amount of work to be done.

This process is called Binary space partitioning. You can experience its impact when you first turn around in an instance. The significant delay is from creating a list of all the objects and all their surfaces and then reducing them into those visible and those behind us.

Once the process has established a list of triangles from each object and sorted them into visible and nonvisible much of the work for the scene is done.

There will however still be triangle surfaces that are partial visible. We need not draw/consider those fully, since some parts will be outside view or behind other objects. This requires those triangles to be split up, adding additional work.

## Zoning or Level Of Detail

Clearly things that are close by and larger have more effect on the image we see as things far away. The graphics engine uses this principle by reducing greatly the amount of triangles an object consists of. You can experience zoning or lodding easy by walking away from a palm tree… at some distance it will vibrate its fronds flapping from nicely curved to flat. As it falls outside the zone it will also reduce the exactness of calculation. As result it seems to jump back again.. inside the zone.. where it then detects it should be outside…

As of today zoning effects can be observed at around 6000 away, 11000 away and beyond 42300 all triangles are no longer considered. For those longer in RLC they might know in the past the maximum view horizon was closer by: 36900 . However with the arrival of large foundations it was possible to lose props from the opposite corner.

## Z-Buffer

After we divided the space up into loads of bend triangles we could now draw all those starting with the furthest away ending with the one on the front. This however is not the smartest. We could check each pixel and see what for that pixel the nearest triangle is. By quickly going through the list of triangles we keep replacing further away with closer by till we have checked all triangles. As result we then have a list with each screen pixel and its triangle that influences it.

There is obvious a danger… if we know the triangle we might not know that the image on that triangle is transparent… e.g. a window in a cabin still needs some trees and mountain visible through it. Years back in RLC the only way to avoid this issue was by electing props = triangles to not use the Z buffer. As result the graphics engine would draw all non-transparent objects/triangles and then finally paint the NoZBuffer items on top of that. Not fast… but better than the default image of Sydney night sky.

Currently we can use an optimized version of translucency. This works by doing the normal BSP and Z Buffer techniques and then if the image is applied and drawn, it will check if it needs to draw triangles behind by looking at the number… if the transparency or Opacity is below that given number it will display the texture of the triangle behind.

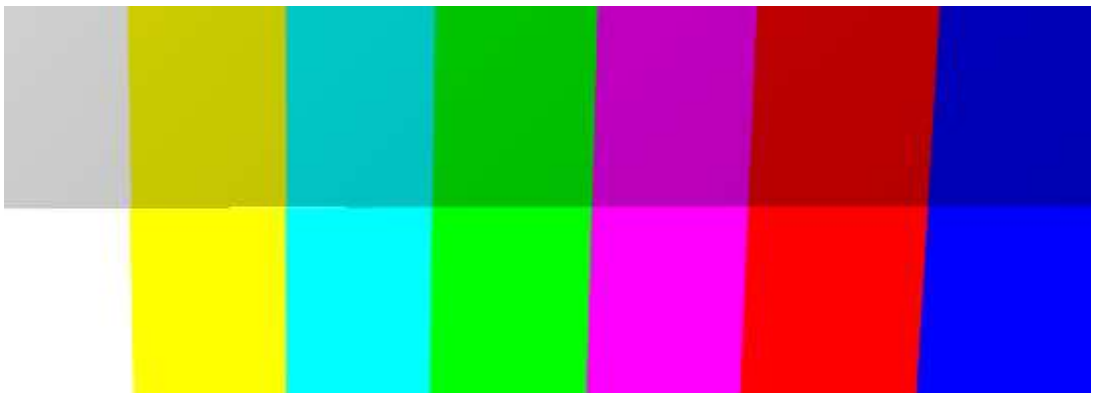
## Light calculation

After all the visible triangles are sorted out, the graphics engine will do a calculation for each of those triangles to determine the light/tint. This calculation can be switched off in the state of the editor by selecting Unlit. All triangles that the prop consist of will then not need to be considered. A light calculation is an averaging from light sources. While other courses deal with the exact implementation of light, there is a fair bit of work involved with calculating the distance to each light source and its angle and the effect it will have on the image. In the state we can further adjust the effects by altering how strong the effect will be. There is ambient light… which does not consider the angle or distance at which the light hits the triangle… and we can adjust how much that light will be taken. There is also another calculation for directional light and a corresponding parameter called Diffuse. Final there is a setting that alters the entire amount of light a surface seems to emit. At 255 it in effect seems similar to Unlit.

The granny engine is not exactly the latest on light. In comparison the Unity 3d engine uses a much faster method and far more intuitive. To still be somewhat valid and cause not too much lag the

granny engine does not use inheritance and only limits the amount of light sources to 10?? (Edit: I recently was surprised to notice that the number has gone up from 5 (plus directional and ambient).

By default each zaby has some single ambient light. I added the comparison image.



## Image mapping

The next step, after the math, is cutting up the supplied Cached Web image or default map. Per screen pixel we know which triangle to use and under which light conditions that needs to be painted. At first sight it would be sufficient to pick a pixel from the corresponding location of the supplied image. That however would create a rather crude image for two reasons. First is that the amount of pixels visible on the screen rarely is the same as the image. That is: say an image is 2048 pixels by 1024 pixels. If that covers a wall somewhere in the distance then on the screen it might just be 30 x 15 pixels (and that is if we are lucky.. one side might be 15 pixels.. the other side might be 40 because it is nearer by). The second reason is that it is highly unlikely that a single pixel from the image is representative for that area.

So some clever trick needs to be done to create something that looks realistic.

## Bilinear texture filturing

What is needed is that for each pixel some interpolation needs to be done that comes close to the required color. How that is performed depends on the Setting Texture Filter. (For the geek… I used the word pixel above simply because it is a common used word… feel free to use the word texel where appropriate)

Imagine standing straight in front of 4 dots of color. If you would stand in the exact middle we would simply take the average of red, green and blue and use that as color.

This corresponds with the setting Bilinear filtering. This process works well with images that are between half the size and double the size of the original picture. Beyond that it gets rather coarse. This effect can be observed by walking towards an image. At some stage suddenly the image seems to disintegrate.

## MIP Map

Continuing with the above example of an image that takes only a small size of pixels at the screen while being rather large we quickly realize that this means a lot of work for each subsequent frame or step that we get closer. As we move large sections of the image needs to be checked.

The solution to this problem is called Mip mapping. It works by pre-sizing an image once into a succession of smaller images and then using the nearest by image to interpolate the correct color for a screen pixel.

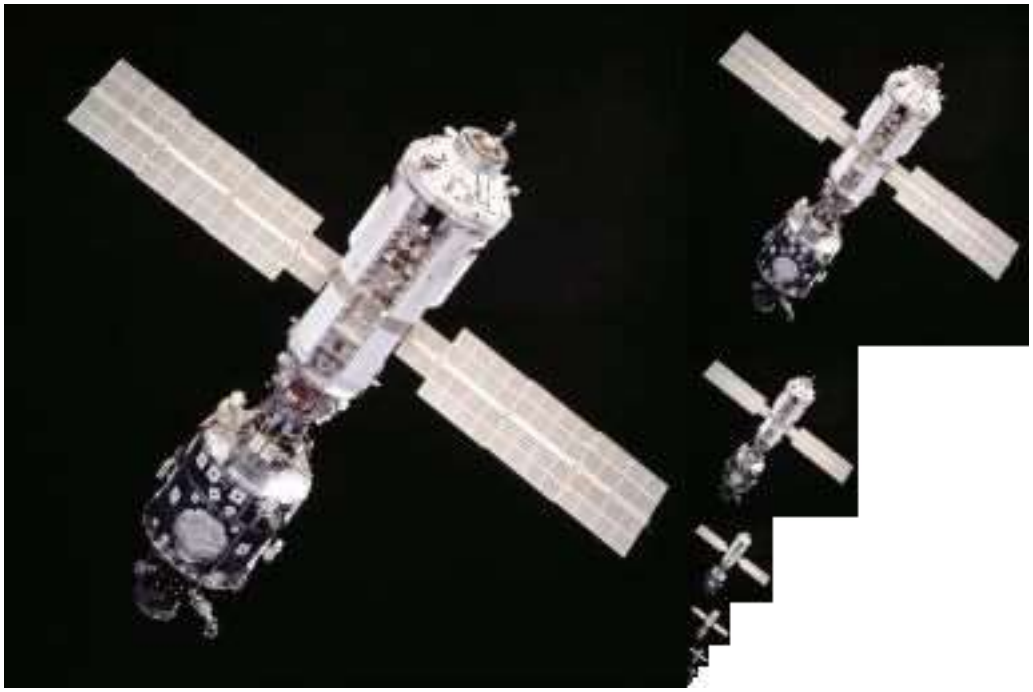
Hence... each image exists in memory in full size… then the graphics engine creates a combination of 4 pixels into 1 making the image half the size. Several steps of images... each half the size of the previous are created. This process greatly favors images that have nice computer numbers… in decimal numbers: 64, 128, 256, 512, 1024, 2048, 4096, etc.

Historic in many graphics engines like DirectX and OpenGL required images to be square and of proper size to be able to benefit from MipMapping. That requirement however no longer exists. As result Maya and other 3d mapping tools no longer create by default square and computer happy sized UV maps and texture maps.

That said… when possible … do use mip map ready numbers. It will reduce initial lag as each images gets mipmapped. However there is no longer need to be religious about it. There are many props that really require non square maps to look best.

Also notice that in state you can switch off mipmap. Use this if you know relative sure that the image will be viewed from a limited fixed range. A Mipmap only is beneficial with items that are viewed over a large range… as in… when you walk all the way to a castle … mipmap is good. If you sit in the same spot viewing the same stuff that already is proper sized… very likely you might benefit from the lower memory requirements with no mipmap.

From wiki an example of a how a Mipmap 'looks' in memory.



### Trilinear Texture Filtering

As we come closer to an object it appears to grow and when the texturing takes place it then will at some stage swap to the next mipmap. This jump is visible at the screen… sudden increase in quality as we come nearer to an image. To soften this jerk in image we can use trilinear filtering.  There we

as we move from one mipmap to the next in size, the screen pixel will be a mix of both mipmaps softening the transition.

### Anisotropic texture filtering

Both Bilinear and Trilinear allow us to see images steady grow as we walk towards them and do a fair job on making something believable. There are however issues with objects that are not straight in front. Or objects that have sides under oblique angles… like plants. The problem is that mipmapping really enjoys rectangular math… and with objects seen from the side we have the issue that at one part they are nearer to us.

For this many game engines including RLC use anisotropic filtering. How many samples are used is expressed in a number 2x, 4x, 8x and 16x. If you never had the setting at 16x please do first make a screenshot of some plants and other fine items preferable under high angles and then alter the setting… close and restart RLC … enjoy… and make a screenshot to compare how much better it looks!

If you are into Skyrim… great discussions exist on the internet on how to tune the graphics card to handle LOD and Texture filtering to create an even better experience.

### Hardware

Clearly it helps to use a powerful computer, yet we cannot expect everyone to have a large wallet. Certainly there is a wide range of CPU and GPU available. Some graphics cards cost more than the rest of the PC together. Notice that a fast card does not alter initial download lag. It will however greatly help in providing a smooth Frame Rate

## How to reduce graphics lag

### Reduce the amount of props by splitting non visible parts of to another zone

The matrix design system is firmly based around this principle. Objects that are in the same nearby area must be considered and cause a lag. By splitting your zaby into sections that are beyond the view horizon of 42300 you in effect reduce the work the graphics engine has to just those props around you. This works great in castles. Many a decorator started enthusiastic on digging moat and erecting walls quickly to find that performance decreased dramatic as detail in each room went up.

We understand while standing in one room that the rest of the castle is invisible… sadly the game engine does not have that kind of Occlusion Culling so we need to implement it ourselves by whisking a person back and forward when they click at a door to a room.

You could in the older zabies still use space by thinking of the Rubik cube or matrix. The center of each block is about 90k away from any other world providing 27 subdivisions inside your zaby. In each block you need to provide some sort of a believable doorway to another part.

### Integrate images

Often we have already an image from say a mirror or painting… it looks quicker to just grab it and continue decorating. Yet the issue is: that is a background image… and a PNG image with transparency on... both will be cut into multiple triangles and clipping regions. With a little work extra in Photoshop, Corel or GIMP we would have suddenly a much better looking image… JPG… without transparency issues… and simpler triangle work. Often a bit of integration greatly reduces

the complexity for a graphics engine.

## Reduce double sided texture setting

While it is often by default on, we do well in reducing the texture double sided. Most props respond to that by no longer painting certain triangles. Not all do however… some furniture does stay texture double sided despite it being switched off. Think of it this way… do you really want to live inside the couch and be able to sit inside… or would you just want to see the outside. Very often cabinets, ceilings, floor, grass etc. all has texture double sided on. Worst of all might be that the flash script does switch texture double sided on by default. Many expensive flash SWF files are also unnecessary visible from the other side.

## Reduce 3d props into flat props

A wall has 6 sides and is by default double sided texture. A sign_01 is just one side by default. A wall needs 3 sided transform of the image… a sign just one. Also … with the way light is implemented there really is no way to have a wall that properly is lit on both sides if there is inside and outside light. A sign .. showing the room texture can be in the room light area and an adjacent sign with a corridor texture might be in a different light area.

## Use repeating prop maps

Most props use a map image only once. Here is a list of exceptions

| Image reuse | Prop |
|---|---|
| **1x2** | ZabyPartition,  Double Door (small edge section on sides) |
| **1x4** | Transport Metal Roof Small(strong folding makes this only useful for 3d textures) |
| **3x3** | 3X3Sign |
| **6x7** | Brick Wall |
| **25x25** | Court Floor |
| **30x1** | RunwayLights (can be used sideways when image is rotated) |
| **60x60** | flr_dance (plane01) |
| **10x1** | FishTank Glass Top |

It is true that repeating textures do not respond to light very well. Also they are not always applicable and seamless images have their own limits. Yet the performance enhancement is substantial.

### Avoid Flash to display still images

Few things look better than a lovely fireplace or a wonderful dance floor. The cost penalty is however significant. Despite all optimizations the tested performance of Flash using images is about 34% more lag compared to Cached Web Images.

### Right size the image

Measure the amount of screen pixels used when looking at the prop. Find the nearest larger nice number in the list of 64, 128, 256, 512, 1024, 2048, 4096 etc. While this takes a little extra planning, many visitors will be very happy that you took that time. This benefits both the initial loading lag as well as the FPS lag. Clearly there have to be moments where you use a sky of 8192 by 1536 bytes… but not many. Nor does a tiny something on the toilet has to have an image of 1024x1024… probably nobody will ever notice any quality difference at 32x32.

### Go naked

Sorry… had to put this in… yet without joke… I have seen people with multiple clothing items like jewelry size 2k by 2k … I know it looks great… in theory… however a party with 20 of those people is impossible with a laptop.

### Avoid clipping

One of the more complex issues with non-static images is that fancy lists to avoid doing too much are quickly outdated. Creating and keeping such lists up to date can become more work than it is saving. Make objects touch… not needless stick through floors as this quickly increases the amount of triangles. Avoid 'radar reflectors'… this are settings where many objects intersect. The amount of triangles that need to be considered by the graphics engine can quickly grow into the hundreds.


Notice that 6 planes cost as much processor time as 1 Tree crown. That is because most props quickly are split into visible and non-visible triangles. The crown is designed by an amateur. Notice that newer trees do not have this issue as they properly avoid intersecting the same plane.

## Emitters

Another issue is the use of emitters. While wonderful avoid them heading through floors… remember that time to live X speed per second = distance a particle travels in its lifetime

So if 1000 rain particles are generated at a speed of 800 units per second and lives for 5 seconds it needs to be 4000 + 200 or so above the floor. Otherwise 1000 particles per second X 5 seconds = 5000 raindrops that need cutting by the graphics engine when they hit the floor.

## Use light correctly

There is an entire training on light. It is probably besides the change of Cached Web Images, the biggest boost RLC deco ever had. It comes at a price. Use it where effective. Consider painting light on larger surfaces in Photoshop / Corel / GIMP. While all zabies have light effects in them by default there is benefit in using modest amount, properly placed lights.

## Conclusion

With this document I hope to have been able to provide some background. It seems beneficial to split Lag into Initial Lag which is related to getting all the info to your computer and Display Lag which is related to the computer being not as fluent and responsive as we like due to complexity of drawing a 3D environment on a flat screen.

Initial lag depends a lot on factors outside our control. It limits us to making a balanced guess between spaces that are fast loading and empty and spaces that have thousands of beautiful props yet take 10 minutes to load. Where the balance is depends a lot on function of a space… clubs visitors do not like to wait… and on your decision to allow caching. Further there could be gain great gain in focusing on certain elements to be of high quality and hence large image sizes. Then many images might be integrated reducing the amount of individual files to be loaded. Final a proper choice in JPG, PNG and compression is important for an acceptable delay in loading.

Display lag is far more complex. We have little control over the make, type of GPU and available memory in the PC of a visitor. We can however do a fair bit to minimize lag by avoiding common pitfalls. While see-through can be great, there is benefit in limiting the amount of props that have to be looked at. Matrix style building can make a big difference in splitting the amount of props that are together in the same viewhorizon space. Further will correct prop use, integrating image and right sizing images make a sizable difference in lag.

May you find as much join in deco as I did.