Optimizations

Below are various important adivice on how to get the most out of the engine. Most of these have to do with getting it to run as fast as possible, but some have to to with visual quality aswell.

Transparency

Avoid overlap

Translucent materials are quite costly perfomance-wise to render and because they both have to read and write for each pixel. At the same time many effects that use translucent materials, like billboards and particle systems, have effects that depend on overlap. For example in a fire the movement of overlapping particles create create a nice effect. For, on screen, small particle systems this is not that much of a problem. But for larger things like smoke it can be very costly. What one needs to be is to try and mimick this sort of overlap effect in a single particle in order to reduce overlap. For example b haing animated textures (using sub textures for particle system) it might be possible to do this very nicely! So beware before just pushing tons of overlapping particles or billboards to do an effect!

Alpha in all trans materials

As mentioned above, a costly aspect is of translucency is that it both read and writes. This can be reduced quite bit by skipping pixels that are not needed. Normally particles and similar and round in shape and are on a square textures. This makes the corners of of the image have pixels that are never drawn. This can be fixed to some extent by having an alpha channel and set alpha to zero in these areas. The important part is that this works even more additive, mul and mulx2 blend modes! So where ever your material is invisible (add=(0,0,0), mul=(1,1,1), mulx2=(0.5, 0.5, 0.5)) have alpha value of zero and for the rest of the pixels 1 (gradient is not needed). This reduce the perfomance it by a particle system by almost 50%!!

Premultiplied alpha

Premultiplied alpha is not something that can increase performance, but is used to remove glitches. When using normal alpha blending it is common for some sort of images that the background (what is in the transperant area) seeps in on the more opaque areas. The reason for this is because of how the hardware works and remultiplied alpha totally removes the artifact. This video (wich features cats!) shows the issue:
http://www.youtube.com/watch?v=dU9AXzCabiM
In order to use premultiplied alpha simply copy the alpha layer in photoshop and paste it as a multiply layer on top of the rgb. Then just set the materail to use pre-multiplied alpha. Note: In order for fading to work in particles, you also need to set multiply alpha with color!

Objects

The fewer the objects, the better

One of the major performance hogs in 3D rendering is the number of object rendered at the same time. The reason is because it can take plenty of CPU power to handle each object and all modern GPUs work much better with a small number of objects.
So how many are too many? Well it depends. Depending on how many lights are rendered, particles on screen (and overlapping), world reflections present etc you will have more or less time to render

out objects. But to give sort of pointer, having 150 objects visible at a time is a good goal. Important note: if u have only made a single room so far, do not celebrate, if u have many connecting rooms later on, all objects will not be culled or slightly visible, adding to the total number of visible objects. A maximum figure would be 300 objects, but do not get up to this until really needed!

In order to reduce the number of objects there are three main ways to go about.

- First thing you need to know is that the game's engine will try and combine static objects ad efficiently as possible. This means that even though you place, say 100 static objects, this might end up being less than 10 in the final version. The way this can be done is if many of the objects share the same material (making it possible for the engine to combine). This means that static objects that are close by to one another should try and save the same material as much as possible. But do not go overboard on this. Do not rely on huge 4096×4096 textures for all static, and have illum textures, etc that are mostly black.
- While the engine can combine pretty good on its own, it does not work very well with smaller objects. Because of this you should have smaller objects as groups of objects instead, for example a group of 6 mushrooms in various configurations. It is okay to have single objects, but use the groups to place the bulk of them.
- If you have a really large room where the engine will not properly combine objects. I giant wall might become made up of 100s of smaller objects, when all it needs is one object really. To solve this you can force combine them in the editor by using the combine edit move. Simply add the sides of the wall into four different groups and they will each become a single object in the game.

Turn off IsOccluder for small objects and on for big
Static objects has IsOccluder checked as default (when first created). What this means is that they are used when trying to determine what is visible in the current view. This means that object is rendered an extra time and thus take extra time. In order to save this extra time you must make sure that the static object really occludes. For example, walls, floors, pillars, and other large objects are all important occluders. However, grass, pebbles, etc do very little to occlude the view and thus it is important that IsOccluder is turned off for these or else they will just waste performance.
At the same time, entities are not occluders by default, and you must make sure that large objects (doors, machines, etc) that are placed in way that they may block the view are set is IsOccluder. Just make sure that shelves, etc that already lean against occluders like a wall or not set, because there is already an object that blocks what ever view they would help occlude.

Keep static object polycount down or use entities
If 100 entities of the same type with 1000 triangles is placed, the memory this geometry data takes up is only equal to a single instance, meaning 1000 tris. This is because the same vertex buffer is rendered multiply times. For static objects, this is not the case. Instead every static object placed is put in a unique (often combined together with other objects) buffer. This means that a if you place 100 static objects of a type with 1000 triangles, it will take upp 100,000 worth of triangles in memory! This can be really important especially when working with consoles that lack much memory. So make sure to keep down the triangle count as much as possible, or use entities instead for high-poly models! This does not mean that they should be N64 type models, but that one should be very careful before spamming polys on static objects.

Lights

Only use spec when really needed

Specular means a lot of extra calculations for the light shader, and increases the perfomance cost of it by quite a bit. Because of that, make sure that not too many specular casting lights hit the same screen pixels (perferabbly fewer than one specular light per screen pixels). This often also makes the game look better. Lights with no specular can then at quite a low cost, be used to simulate ambient light and light bouncing. So make sure to place those specular casting lights in places where they count!

Do not use too many shadows
Shadows is another expensive feature so use it wisely. The best thing is to only have a single big shadow casting light source near the player. It is then okay to place a few low resolution shadows near by if they do not cover much screen area. The reason these lights cannot have high resolution shadow map is that the engine only has one large map to spare and it would then be swapped between many lights, losing a bunch of performance boosters.
The trick here is to cast a spotlight in a way that simulates some kind of average for all the light sources in the scene. You can often get away with quite hackish setups, so do not feel required to simulate reality here. Read this article for more info:
http://wiki.frictionalgames.com/hpl2/tutorials/level_editor/tutorial_2
Also study the games Bioshock and Dead Space 1/2 for a lot of good tricks on this sort of thing!

Make sure lights only cover what is needed.
Remember that you can easily customize the falloff from lights, so that you do not need to have a really large light in order to get a certain effect. Normally, you can often just let the end of the spotlight frustum or light sphere touch where you want light and the set the slowest falloff to get the your area lightly lit. What you do not want to do is to add a really large light where most of the area is outside of the map, instead make every inch of that light shape count!!
(This has been a very common mistake in the past, so keep it in mind at all times)

Physics

Turn off or simplify collisions of static objects
As mentioned above in “Keep static object polycount down or use entities” static objects do not instance their data but keep it unique for each object. This is also true for collision data. First thing to do, is to make sure to that no unnecessary object, like grass, pebbles, etc, has collision on. Turn it off far all small stuff or for geometry that the player can never reach (like distant tower or what not).
Next thing to do is to add mesh shapes (these are shapes that are added through special naming in the model file) for high polygon objects. This include things like pillars, high detail walls and more. Do not sure it on everything, but consider it for everything above 500 triangles or so. For some objects, like a rubble of rocks, it is better not have have it because it can be too complex to add collision shapes. For most objects it is well worth the job though. It makes the game take up less ram and also make collision testing faster (unless a lot of geometry shapes are used that is, so do not go wild)