A brief insight into BRTECH1

a written-from-scratch idtech3- and darkplaces-compatible 3D game engine

Who are we?

- Name: Benjamin Rosseaux
- Handle: BeRo
- Group: Farbrausch

Who are we?

- Name: Urs Ganse
- Handle: urs
- Group: mercury

Quake Arena III

Rocket Launcher
20
136

What is Quake III Arena?

- A first-person-shooter game by id Software, which was specifically designed for multiplayer.
- Q3A allows players whose computers are connected to a network to play against each other in real time per a client–server model.
- Q3A uses a engine called “id Tech 3”.

id Tech 3

What was new at “id Tech 3” ? (render side)

- Unlike most other games released at the time era, it requires an OpenGL-compliant graphics accelerator to run.
- It does not include a software renderer, as "id Tech 2" (Quake 2) did it yet.

What was new at “id Tech 3” ? (render side)

- The graphic concept of the engine is based tightly around a "shader" system, where the appearance of many surfaces are defined in text files, called "shader scripts."
- But these “shaders” have nothing to do with the today's “shader” term, instead rather with the today’s “material” term.

What was new at “id Tech 3” ? (render side)

- id Tech 3 also introduced spline-based curved surfaces in addition to planar surfaces, which are responsible for most all of the curved surfaces, which are present within the Quake 3 game.

What was new at “id Tech 3” ? (render side)

- id Tech 3 also introduced to provide support for models animated using frame-based linear interpolated vertex animation with attachment tags for to allow models to maintain separate torso and leg animations and hold weapons.
- So Quake 3 was one of the first games, where the player was able to look up and down and around as the head, torso and legs are separate models.

What was new at “id Tech 3” ? (render side)

- id Tech 3 also introduced “volumetric fog”, “mirrors”, “portals”, dynamic shadows and CPU-side wave-form-based vertex distortions.

So what is a “shader”?

- Shaders in the "id Tech 3" world are described and rendered as several layers.
- Each layer contains a texture, a "blend mode" which determines how to overlay it over the previous layer, scrolling, scaling, rotation and texture modes such as environment mapping.
- Each shader have a “sort key”.
- These shaders also can define several surface properties such as collision behaviour flags.

Shader script example

```
textures/gothic_wall/streetbricks10_shiny
{
    q3map_globaltexture
    sort opaque
    {
        map $lightmap
        tcmod turb sin .1 .1 0 0
        tcmod scale 2 2
        rgbgen identity
    }
{
        map textures/gothic_wall/streetbricks11.tga
        rgbGen identity
        blendfunc gl_one gl_src_color
    }
    {
        map $lightmap
        blendfunc gl_dst_color gl_zero
        rgbgen identity
    }
}
```

Shader & surface sorting behavior

- Sorting in the id Tech 3 Engine is an often misunderstanded concept.
- id Tech 3 sorts the surfaces neither back-to-front nor front-to-back, even not for transparent surfaces.
- id Tech 3 sorts the surfaces in order of the shaders in the shader array, and id Tech 3 sorts the shaders in the order of the sort key **and** then, indirect due to the data loading code structure, in the order of the file names of the contained .script file and .pk3 file. It’s important to know for the emulation of the id Tech 3 rendering bugs in BRTECH1.

How does id Tech3 render the sky?

- The sky at id Tech 3 is a skydome, which will be polygon-vertex-clipped on the CPU after the BSP/PVS visible surface collection process.
- The sky is rendered as first, even before the opaque surfaces.

And portals and mirrors?

- Portals and mirrors are rendered before the sky with help of OpenGL clip planes before the sky sort key, so that the actual scene is rendered just simply above it. Simple but effective :-)

And volumetric fog?

- Volumetric fog in id Tech 3 can be only axis aligned, since it is axis-plane based.
- If the camera is inside the fog volume, then the objects inside the fog volume will be overlayed with CPU-modulated fog textures.
- If the camera is outside the fog volume, then the fog volume itself will be rendered as flat axis-aligned plane with a CPU-modulated fog texture.

And the light grid?

- The light gris in id Tech 3 is a voxel-style 3D array with the average light direction, light diffuse color and ambient color per voxel.
- In id Tech 3 it will be interpolated trilinearly on the CPU.

Darkplaces

17:46
44

So what is Darkplaces?

- Dark Places is a heavily modified version of the Quake engine (id Tech 1) by a developer with the name Forest "LordHavoc" Hale for Linux, Mac OS X and Windows.
- It was extended by Quake2 and Quake3 BSP map support, bloom, extended real-time lighting and shadow effects, as well as particle effects, bump mapping, GLSL support, 32-bit alpha channels and improved image representation for realistic explosions, blood spatter and cartridges representations.
- Famous games, which uses Darkplaces: Nexuiz (the original first version of it) and Xonotic.

BRTECH1

BRTECH1

- A written-from-scratch id Tech 3 asset compatible engine with Darkplaces extension support.
- Written in Object Pascal. Compilable with Delphi >= 7 and FreePascal >= 2.6
- Main development time: July and August Summer 2012

BRTECH1 renderer concept

- BRTECH1 renderer is a light-prepass / deferred lighting renderer. So it’s basically a forward renderer combined with the deferred shading concept idea. So it can be nearly fully optical compatible with id Tech 3 but it can have also newer deferred rendering concepts at the same time.
- BRTECH1 supports multiple dynamic realtime light sources including soft shadows (point lights, spot lights, directional lights and with dirty tricks as such as hidden light blocker faces also (fake) area lights)

BRTECH1 renderer concept

- The vertex data for the vertex buffer object(s) are resorted and merged for modern GPUs with different approachs (selected per GPU detection):
    - One big VBO for each material and BSP area just with material-splitted frustum culling for each whole VBO per BSP area. (better for newer GPUs) (and a BSP area is defined by the map designer, so that are no portals, instead more a disposable one-way optimization hint for the render, eg. at id Tech3: while door closed => don’t render things behind it, door opened uniquely => kill the hint flag for the rest of game time, and many maps have no or just one BSP area for the whole map, but that here is only a very rough description, what BSP areas are)
    - One VBO for each BSP leaf and material with normal PVS and frustum culling as id Tech 3 do it also. (better for older GPUs)

BRTECH1 renderer concept

- The vertex data for the deform_* stuff (eg. deform_autosprite and deform_autosprite2) will be also restructured / reprocessing for the vertex shader in special ways for static VBOs. Each vertex has a one more vec4 attribute called Middle, which contains for example for deform_autosprite the middle of a face, and for deform_autosprite2 also the dominant edge length information in the w component of the vec4 attribute. The vertex shader can emulate the original id Tech 3 behaviour of deform_* perfectly with this extra information per VBO vertex.

What makes BRTECH1 different?

- The most important difference: It uses no fixed-pipeline OpenGL. It’s completely GLSL-based.
- All id Tech 3 material shader texture modifier und vertex deforms are processed on the GPU, not on the CPU, including deform_autosprite and deform_autosprite2 !
- **All** vertex data are uploaded exactly **once** in one or more complete static OpenGL vertex buffer objects. All further dynamic content processing stuff are tasks for the GPU itself (with help of vertex, geometry and fragment shaders)

BRTECH1 material shaders

- BRTECH1 converts all id Tech 3 material shaders to each one, or to more multiple if material-shader-stage-collapsing fails, GLSL shaders.
- BRTECH1 has also a GLSL-shader cache for to minimize the count of the GLSL program handle counts.
- You can also freely add own GLSL-code to your material shaders at BRTECH1.

BRTECH1 shadows

- Dynamic hard shadows are rendered with help of shadow maps, where the caster objects will be grouped in own shadow map render groups and sorted by viewer depth, a. k.a. very cheap to implement but working quasi-cascaded shadow map variant.
- Optional soft shadows are postprocess-rendered by blurring the hard shadows in screen space with help of PCSS-like blocker search.

BRTECH1 wireframe rendering

- Wireframes without fixed-pipeline with help of barycentric coordinates

```
#extension GL_OES_standard_derivatives : enable

vec3 d = fwidth(vBC);
vec3 a3 = smoothstep(vec3(0.0), d*1.5, vBC);
float  edgeFactor = min(min(a3.x, a3.y), a3.z);
```

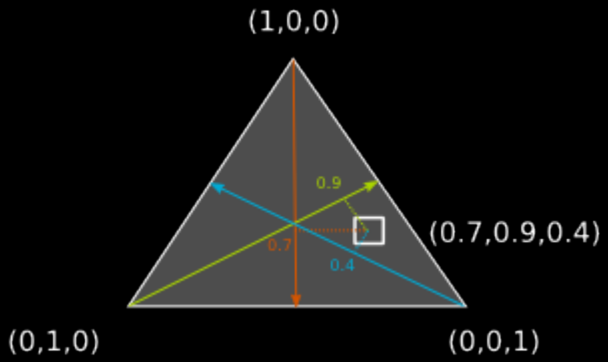

Image source: codeflow.org

BRTECH1 dynamic model lighting

- For additional lighting of models, BRTECH1 converts the light grid from the .BSP file to a voxel-style 3D texture.
- Each voxel is splitted in two 3D RGBA8 pixel/voxel in two 3D texture height halfs:
    - Ambient color and light direction longitude (y: 0.0 … 0.5)
    - Diffuse color and light direction latitude (y: 0.5 .. 1.0)
- These informations are then used for the per-pixel-lighting in the fragment shader.

BRTECH1 portals and mirrors

- Portals and mirrors will be rendered as separate FBO render-to-texture passes before the actual render pass (with projection matrix manipulation based clip plane for to avoid hidden overdraw) and then displayed as normal surface textures in the actual scene.

BRTECH1 fog

- Volumetric fog is rendered almost in the same as id Tech 3 do, with the exception, that it is fragment-shader-based instead CPU-based fog texture pixel content modulation.

BRTECH1 sky rendering

- The sky will rendered after all opaque surfaces instead before all opaque surfaces as Id Tech 3 did it.
- Together with stencil buffer masking of the sky-mark-draw-mask-faces to avoid sky-shows-through-artifacts at cracks and micro-polygon-holes on the opaque surfaces.

BRTECH1 sound

- The BRTECH1 sound engine has 3D HRTF stereo rendering, doppler effect, underwater simulation...

Demo time!

or also: make a demo about it! :-)

One more thing

If you do want to use BRTECH1 for your game (commerical or freeware), because you do want to use the GPL’ed GTKradiant and the other GPL’ed idTech tool chain but avoid the original id Tech license fees, then you can just contact me, and I will resume the BRTECH1 development for the yet missing network code...

But attention, the game logic has to be implemented in (Object) Pascal. :-D

Thank you!

Questions?