MARCH 9TH, 2009

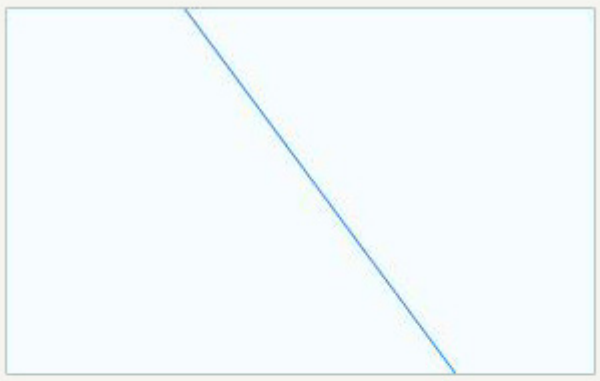
QUAKE ENGINE CODE REVIEW : RENDITION (4/4)

Quake renderer is the module that took the most work during the initial development. It is extensivly described by Michael Abrash book and John Carmack's .plan files.

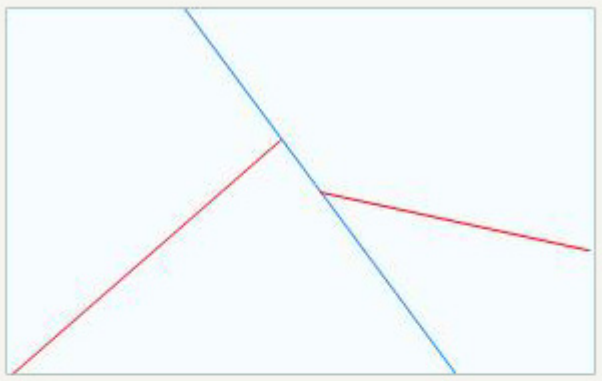
This article is in four parts :

Rendition

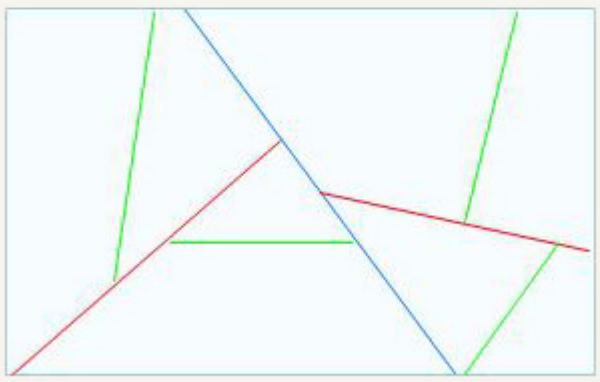
The scene rendition process revolves around the map's BSP. I invite you to read more about Binary Space Partitioning on Wikipedia. In a nutshell, Quake maps are heavily pre-processed ; The volume is sliced recursively as in the following drawing:



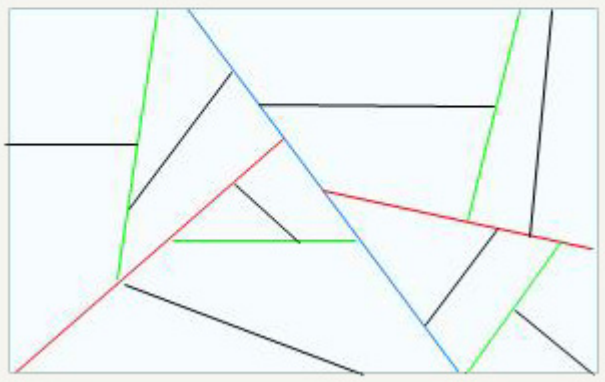
Iteration 1



Iteration 2

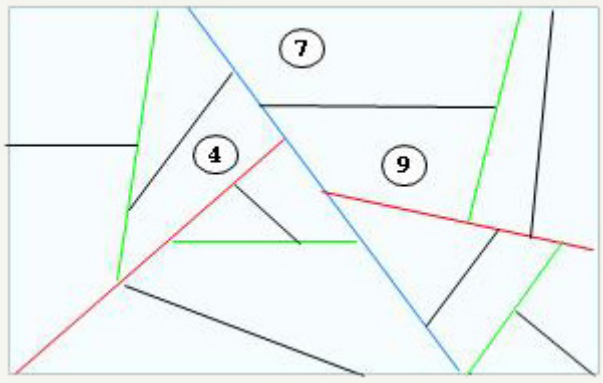


Iteration 3



Iteration 4

The process generates a leafy BSP (the rules are: choose an existing polygon as splitting plan and choose the splitter cutting the less polygons). After the BSP is generated, for each leaves is calculated the PVS (Potentially Visible Set). As an example: the leaf 4 can potentially see leave 7 and 9:



The resulting PVS for this leaf is stored as a bit vector:

Leaf Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

PVS for leaf 4	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0

This resulted in a global PVS size of about 5Mb, way too much for PC in 1996. The PVS is hence compressed via delta length compression.

Compressed PVS for leaf 4	3	2	1	7

The Run-length encoded PVS only contains the number of 0s between the 1s. Although it may not look like a very efficient compression technique, the high number of leaves (32767), combined with a very limited set of visible leaves, brings down the size of the entire PVS to 20kB.

Pre-processing in action

Armed with the precalculated BPS and PVS, the engine's map rendition routine was simply:

- Traverse the BSP to determine in which leaf the camera is positioned.
- Retrieve and decompress the PVS for this leaf, iterate through PVS and mark leaves in the BSP.
- Traverse the BSP near to far
- If a Node is not marked, skip it.
- Test the Node Boundary Box against the Camera Frustrum.
- Add the current leaf to the rendition list

Note: The BSP is used multiple times ex: to walk the map near to far for each active light and tag the polygons of the map.
Note2: In software rendition, the BSP is walked far to near.

Code analysis

The rendition code can be summarized as follow:

```
SCR_UpdateScreen
{
                GL_BeginRendering
                SCR_SetUpToDrawConsole
                V_RenderView
                |               R_Clear
                |               R_RenderScene
                |               |               R_SetupFrame
                |               |                               Mod_PointInLeaf
                |               |               R_SetFrustum
                |               |               R_SetupGL
                |               |               R_MarkLeaves
                |               |               |               Mod_LeafPVS
                |               |               |                               Mod_DecompressVis
                |               |               R_DrawWorld
                |               |               |               R_RecursiveWorldNode
                |               |               |               DrawTextureChains
                |               |               |               |               R_RenderBrushPoly
                |               |               |               |                       DrawGLPoly
                |               |               |               R_BlendLightmaps
                |               |               S_ExtraUpdate
                |               |               R_DrawEntitiesOnList
                |               |               GL_DisableMultitexture
                |               |               R_RenderDlights
                |               |               R_DrawParticles
                |               R_DrawViewModel
                |                       R_DrawAliasModel
                |               R_DrawWaterSurfaces
                |               R_PolyBlend
                GL_Set2D
                SCR_TileClear
                V_UpdatePalette
                GL_EndRendering
}
```

SCR_UpdateScreen

Calls:

1. `GL_BeginRendering` (Set variables (`glx,gly,glwidth,glheight`) later used in `R_SetupGL` to setup viewPort and projection matrix)
2. `SCR_SetUpToDrawConsole` (Decid console height : Why this is here and not in the 2D part ?!)
3. `V_RenderView` (Render 3D scene)
4. `GL_Set2D` (Switch to ortho projection (2D))
5. `SCR_TileClear`
6. Optionally draws a lot of 2D stuff, console, FPS metrics etc...
7. `V_UpdatePalette` (name fit software renderer, in openGL this set the blending mode, according to damage received or active bonus etc making screen red or bright etc...). Value is stored in `v_blend`
8. `GL_EndRendering` (Swap the buffer (double-buffering)!!)

V_RenderView

Calls:

1. `V_CalcRefdef` (No idea sorry :) !)
2. `R_PushDlights` Mark polygons with every light with effect on them(see note)
3. `R_RenderView`

Note:

R_PushDlights calls a recursive method (`R_MarkLights`). It uses the BSP to mark (using an int bit vector) polygons affected by lights, BSP is walked near to far (from the light's POV). The method checks if light is active and if in range. `R_MarkLights` method is particulary noticable because we can find here Michael Abrash's direct application of distance point-plan article "Frames of Reference" (`dist = DotProduct (light->origin, splitplane->normal) - splitplane->dist;`)).

R_RenderView

Calls:

1. `R_Clear` (Clear GL_COLOR_BUFFER_BIT and/or GL_DEPTH_BUFFER_BIT do only what is needed)
2. `R_RenderScene`
3. `R_DrawViewModel` (Render player model is we are in spectator mode)
4. `R_DrawWaterSurfaces` (Switch to GL_BEND/GL_MODULATE mode to draw water. Warping is done via sin and cos lookup table from `gl_warp.c`)
5. `R_PolyBlend` (Blend the entire screen with value set in `V_UpdatePalette` via `v_blend`. This is to show when we take damage (red), are under water or under bonus boost effect)

R_RenderScene

Calls:

1. `R_SetupFrame` (Retrieve the BSP leaf where the camera is, store it in variable "r_viewleaf")
2. `R_SetFrustum` (Setup the `mplane_t frustum[4]`. No near and far plane
3. `R_SetupGL` (Setup GL_PROJECTION, GL_MODELVIEW, viewport and glCullFace side, also Rotate Y and Z axis as Quake z axis and x axis are switched with openGL.)
4. `R_MarkLeaves`
5. `R_DrawWorld`
6. `S_ExtraUpdate` (Reset mouse location, take care of sound issues)
7. `R_DrawEntitiesOnList` (Self-explained)
8. `GL_DisableMultitexture` (Idem)
9. `R_RenderDlights` (Light bubbles, light effects)
10. `R_DrawParticles` (Explosions, Fire, Static, etc)

R_SetupFrame
Notice the line: `r_viewleaf = Mod_PointInLeaf (r_origin, cl.worldmodel);`
This is where the Quake engine retrieves the leaf/node the camera is currently positioned in the BSP.

Mod_PointInLeaf can be found in model.c, it runs through the BSP (the BSP root is at model->nodes).

For every node:

- If the node is not splitting the space further, it's a leaf and hence it's returned as the current node position.
- Else the BSP splitting plane is tested against the current position (via a simple dot product, that's the usual way BSP tree are visited) and the matching children is visited.

R_MarkLeaves

Variable `r_viewleaf` holding the camera location in the BSP (retrieved in `R_SetupFrame`), lookup (`Mod_LeafPVS`) and decompress (`Mod_DecompressVis`)the Potentially Visible Set (PVS.)
Then iterate on the bit vector and mark Potentialy visible nodes of the BSP with: node->visframe = r_visframecount.

R_DrawWorld

Calls:

1. `R_RecursiveWorldNode` (Walk the BSP world front to back, skipping Nodes not marked earlier (via `R_MarkLeaves`), populate `cl.worldmodel->textures[]->texturechain` list with the appropriate polygons.)
2. `DrawTextureChains` (Draw the list of polygons stored in texturechain: Iterating through cl.worldmodel->textures[]. This way, there is only one switch per material. Neat.)
3. `R_BlendLightmaps` (Second pass used to blend the lightmaps in the framebuffer)

Note:

This part uses the openGL infamous "immediate mode", at the time it was probably considered "start of the art".

`R_RecursiveWorldNode` is where most of the surface culling is being done; A node is discarded if:

- The content is solid.
- The leaf was not marked via the PVS (`node->visframe != r_visframecount`)
- The leaf fails the frustrum clipping.

MDL format

The MDL format is a set of fixed frames, Quake engine does not interpolate vertex location to smooth animation (so higher frame rate do not makes models animation look better).

Some elegant things

Elegant leaf marking

The naive approach to mark a leaf of the BSP to be rendered would be to use a boolean `isMarkedVisible` and before each frame:

1. Set all boolean to false.
2. Iterate through the PVS and set visible leaf to true.
3. Later, test leave with `if (leave.isMarkedVisible)`

Instead of this, Quake engine uses an integer to count the number of frame rendered (`r_visframecount` variable). This allow the step 1 to be skipped entirely:

1. Iterate through the PVS and set visible leaf `leaf.visframe = r_visframecount`
2. Later, test leave with `if (leaf.visframe == r_visframecount)`

Recursion avoidance

Found in `R_SetupFrame`, instead of going with a quick and dirty recursion to walk the BSP and retrieve the current position: a while loop is used.

```
	node = model->nodes;
	while (1)
	{
		if (node->contents < 0)
			return (mleaf_t *)node;
		plane = node->plane;
		d = DotProduct (p,plane->normal) - plane->dist;
		if (d > 0)
			node = node->children[0];
		else
			node = node->children[1];
	}
```

Minimize texture switchs

In openGL, switch texture via (`glBindTexture(GL_TEXTURE_2D,id)`)is very expensive. In order to minimize switchs numbers, every polygon marked for rendition is stored in an array chain, indexed on the polygon's texture material.

`cl.worldmodel->textures[textureId]->texturechain[]`

Upon culling is done, the texturechains are drawn in order, this way there is N texture switchs where N is the total number of texture visible.

```
	int i;
	for ( i = 0; i < cl.worldmodel->textures_num ; i ++)
		DrawTextureChains(i);
```

Return to main Quake Source Exploration page.