Realtime 3D

>>> this information was originally compiled for a handout to a seminar on game graphics <<<

1.) Introduction

Realtime 3D graphics is without a doubt among the most interesting cg-disciplines of our time. Highly detailed environments have to be realised under difficult conditions, demanding creative solutions.
Early on computers featured graphics hardware and the first computer games all had the need to fill the screen with imagery without stressing the resources too much. The solution at hand were image compression and vector graphics, as few coordinates could describe complex objects and leave the drawing to comparably simple transformation matrices.

When the machines got more powerful (32 bit), new ways of threedimensional visualisation were developed, most importantly bitmap sprites in threedimensional space like in "Wing Commander". ID-Software used perspectively distorted bitmaps to create the halls of "Wolfenstein 3D". This was the predecessor to texture mapping, but the complexity of the transformation remained controllable by a twodimensional level-design and the viewport being in the middle of the vertical axis. Thus the depth sorting, line-of-sight calculations and sprite-positioning could be handled in 2D. Contrary to that approach, "Alone in the Dark" filled 3D Polygons with 2D gradients and textures without transforming them to match the geometry´s position in space.

Shortly thereafter Gorouad-Shading and true, threedimensional texture mapping got employed, still simplified in "Ultima Underworld" (viewport could tilt up & down) and in all it´s glory in "Strike Commander", featuring shaded landscapes with fractal textures for woods, rivers and citys. And finally, in "System Shock" 1994, the whole environment was textured 3D, the viewport free and even some of the villains not simple sprites, but freaky sprite-clouds.

Yet, not all games chose polygonal 3D: "Ecstacia" drew the characters as clouds of ellipsoids within painted pictures, similar to "Alone in the Dark" wich drew polygonal characters into painted settings.

Another technique new to computer games were the Voxels, short for volumetric

pixels: "Comanche", a helicopter-action-simulation, generated the most impressive landscapes of it´s time from 2D elevation and colormaps. Though hardware acceleration for voxelgraphics is available, this technique for image based visualisation has yet to make an entry into mainstream hardware, and the software-rendering was utilized by Novalogic for their action-simulations.

The latest big software-innovations were in 1995 DirectX as an uniform driver interface, first bone deformations in "Halflife", dynamic Level-of-detail and multitexturing in "Quake III". On the hardware side more and more standard-functions were integrated into the chips like 3D-transformation and lighting. The "GeForce 3" improved on this "raw power" with programmable vertex-shaders, flexible calculation pipelines allowing the programmers to assemble their own functions from predefined modules, for example environment mapping or 2D-refractions.

1.1) brief history computer games

1958	"Tennis Programming" an analog computers
1968	"Colossal Cave Adventures" for mainframes
1972	"Tennis" for the "best TV" of it´s time
1975	"Gunfight", first game for micro computers
1976	Apple I, most succesful kit-computer
1977	Star Wars hits cinemas, Apple II the stores
1978	"Space Wars" with 2D vector graphics, SSI (Strategic Simulations Inc.) founded
1979	"Star Trek", "Black Hole", "Alien", Motorola develops 68000 chip, Compuserve is founded
1980	Sierras "Mystery House", "Wizard and Princess" run on Apple II, "Cytron Masters" is the first real-time-strategy game, "Empire strikes back" on the silver screen
1981	the IBM PC, MS-DOS, "ZORK", "Wolfenstein", "Wizardry", "Donkey Kong"
1982	"Wizard and Pricess" out for PC, "Time Zone" the biggest adventure featuring 12 disks, first Microsoft games "Decathlon", "MS Flight Simulator"
1983	"3Demon" (3D Pac Man), "Dragon´s Lair" on Laserdisc, "Beyond Wolfenstein"
1984	"Terminator", "The Last Starfighter", Macintosh, IBM AT, "Kings Quest" and the Video Game Cartride Market crashes
1985	MS "Windows" flops, EGA, Amiga, Atari ST and i386 available, "Kings Quest II"
1986	"Aliens", "Star Trek IV", Compaq founded, "Kings Quest III", "Star Wars" vector-shooter, "Tetris"
1987	VGA becomes available, IBM PS/2, OS/2, "Gunship", "Metal Gear PC, "Adventure Construction Kit", "Black Cauldron" game, "Ultima" and "Larry". MS "Windows 2.0" flops
1988	"Kings Quest IV", Soundboards available, MS-DOS 4, "Manhunter",

"Falcon AT", "Life&Death", "Dungeon Master"

1989 "Star Trek V", i486, "Gunship 2000", "Mechwarrior", "Monkey Island"

1990 MS "Windows 3.0" is a success, "Kings Quest V", "Mixed up Mother Goose CD", "LHX", "Neuromancer", "Wing Commander", "Covert Action", "Dangerous Dave" (by John Carmack, later ID software)

1991 "Kings Quest V" CD with vocal sound & Windows-support, MS-DOS 5, "Terminator II" in cinemas, "Falcon 3.0", "3D Construction Kit", "Civilisation", "Lemmings"

1992 "Lawnmower Man", Windows 3.1, "Ultima Underworld", "Alone in the Dark"

1993 Pentium, "Kings Quest VI", "Day of the Tentacle CD", "Strike Commander", "Doom", "Privateer" and the CD-only games "7th Guest" and "Rebel Assault"

1994 PowerPC, "Star Trek VII", "System Shock", "Ecstacia", "Myst"

1995 Windows 95, Pentium Pro, "Toy Story", "War Craft II", "Command&Conquer", "Mechwarrior II"

1996 "Time Commando", 3dfx "Voodoo" chip, "Quake", "Bad Mojo", "Creatures", "Gene Wars", "Master of Orion 2", "Tomb Raider"

1997 "Fallout", "Blade Runner" game, "Diablo", "Dungeon Keeper", "Ultima Online", "G-Nome"

1998 "Unreal", "Halflife": scripted bots look intelligent,
"Heavy Gear II": 3D trees,
"SimCopter": game recalled because of random homosexual couples in the SimCity-streets

1999 "Outcast" all voxel, "Aliens vs. Predator", "Alpha Centauri", "System Shock 2", "Quake III"

2000 "Deus Ex" (great storytelling), "Diablo II" (high-tech 2D), "The Sims"

2001 "Pong 3D", "Black & White" (1st "Dogma 2001" game)

2002 "Morrowind" features a whole continent to explore,
Ego-Shooters in the crossfire again after the Washington sniper; movie-licenses cost too much money, so the games are mostly "bullet proof" jump&runs or shooters...booooring

1.2) other uses for realtime graphics

Of course, realtime graphics are not only relevant for gaming. Military and civil simulators, scientific visualisations for chemistry, biology and medicine, all use the unix-native OpenGL for graphics. But in these cases, "realtime" often refers to realtime simulation, not depiction. But for complex simulations it is also common to use realtime-depiction to view preprocessed data.

2.) requirements and difficulties

Realtime graphics in games is marked by the balancing of stunning visuals, depth of gaming, rapid development, easy maintance of the technology and low hardware requirements.

Since Windows 95 established a powerful multimedia-interface for game developers and the "Voodoo" chips brought arcade-quality visuals to the PC, the term "3D engine" had to be redefined. Game developers did not have to spend their time on supporting hardware or developing tricks (like the 2,5 D "Doom"), but could improve the game logic. Nowadays the "game engine" is a modular interface enabling easy, object oriented and often simulteneous work on game design, debugging, extension and programming.

Developers used the resources freed by modern graphics hardware to implement particle systems and improve the "bots" rudimentary AI for action games, adventures and RPGs refine the game system and storytelling to make the game all the more interactive and immersive, even the out-of-norm games and god-games and simulations improve, with elaborate AI and simulation complexity.

Processing Power is used mainly for game logistics like collision detection, AI (line-of-sight and pathfinding routines) and animation (bone-deformations and mesh-interpolation).

3.) solutions

3.1) modelling

The complexity of a 3D geometry in a game directly and lineary affects the performance, as every Vertex (a 3D coordinate and corner of Polygons) requires elaborate transformation matrices and every triangle has to be clipped and culled. Shading added the drawing of a gradient into a polygon, evaluating normal vectors and light-sources, textures require the additional transformation of a 2D image and drawing onto the polygon.

Considering this it became imperative for computer games to keep the triangle-count as low as possible. To reduce processing load and the amount of data, that had to be moved around in memory, early games were limited to static, geometric Objects (tanks, planes), occasionally they even used 2D-polygons (remember the camels in "LHX" ?).

Key to succesful Low-Polygon-Modelling is to identify the object-relevant details and to limit yourself to those. In addition, stick to creating triangles, as polygons with 4 or more Vertices have to be triangulated for rendering, wich means that the folding of the polygon may very well be random, creating flipping edges, distorted textures and other ugly phenomenons.

If you then take care to keep the resolution of cylindrical objects low and focus on the outline, efficient use of textures and avoid unnecessary complexity, for example by combining cloth and character into a single mesh, it is very well possible to create a humanoid character with less than 200 Polygons. If the games point-of-

view is face-height, it´d be a good idea to increase resolution towards the face, as this is where the user will focus his attention to.

Regarding animation, it may be necessary to resolve faces or limbs higher than that to enable more pleasing transitions. This additional control, of course, only makes sense if the geometry inhabits a large part of the screen. This led to the technique of "level of detail", swapping geometries between high and low resolution versions depending on their distance to the viewer.

Finally you should keep an eye on modularity within a model. For a legion of soldiers, all with the same uniform but different faces, it doesn´t make sense to save the whole texture for every soldier. Instead, store all exchangeable portions in one file per feature (like one for armor, one for faces,...). This means you have to clearly define the triangles with exchangeable textures when modelling a figure. For example you could implement edges along the hairline and neck to seperate the facial triangles for texturing more easily. Even though multitexturing is available, best limit yourself to one texture per triangle.

Some effects (like heads looking at the viewer) or animations (a robot twisting it´s torso without deforming) it might be useful to establish hierarchies of independent meshes instead of single-mesh characters, but that decision is left to the game designer, the engine, the modelling software or import file format.

A special case of modelling may be lighting: Some engines define object lighting, but also texture-transitions, on a per-vertex level before even choosing the shader. The vertex "color" then decides luminance or type of texture before a gorouad or phong algorithm is evaluated. For modelling this means he has to regard transitions in lighting and material when tesselating a geometry. Fortunately, this technique is in most cases limited to static objects and mainly used for terrain-modelling (low-res textures are tiled and masked by vertex-colors) or buildings with "baked" lighting and is often handled by proprietary tools.

3.2) texturing

Textures play an important part in the design of modern games, as the past years have made available more memory and bandwith (AGP) than raw processing power. Thus many details were included in the texture than in the actual geometry, though current graphic chips make high resolution models quite probable. In addition, as already mentioned, textures allow for a quick modification of characters without requiring the loading of new UV-coordinates, models or setups.

It´s important to notice, that, contrary to most 3D Animation packages, any triangle can only have one texture applied (with exception of special solutions mentioned earlier). Furthermore, every texture eats up memory, bandwith and performance, as it has to be stored in memory as long as it´s used. Thus it´s reconmendable to, depending on the level of modularity, combine all textures appearing together only in a unique combination into a single texture file. In that way the logistic effort is manageable for both artists and programmers.

Programmers also favour the "power of 2" when it comes to memory management and program layout. For the texture-artists this means to create square textures with height and width the power of 2 (2,4,8,16,32,64,128,...). Sometimes there are even

limits to the colors one can use, as custom color-mapping palettes and indexed colors are often used to keep the file size down.

Also quite unlike 3D animation packages, wich often limit the user to project a texture onto an object, games only work with UV-textures. UV-coordinates are an additional pair of 2D texture coordinates to the 3D space coordinates of a vertex, thereby defining the portion of an image to be drawn into a polygon. These UV-coordinates are usually stored as floating point numbers (ranging from 0 to 1) and mapped to cover the whole image file, so it´s easy to switch any texture for one with a different resolution.

Finally, the artist has only a limited number of texture-channels available to his creation, but that situation should improve in the following years. Originally designers only had the diffuse or ambient channel, only affecting the color of an object. Lighting-transitions were accomplished by gorouad-shading and transparencies were that taxing that they were only used for "3D-Sprites". It wasn´t until "Quake III" that multiple textures could be layered on an object, lightmaps added shadows to tiled textures and bump-maps increased surface detail.

The texture artist and the modeler have to start by breaking down the model: how many textures are feasible ? Where does the geometry have to be high-res, where are textures used for detail ? Which attributes or channels are related, or is it sufficient to use a simple noise on the bump channel ? Which parts have to feature exchangeable textures ? What is the engine capable of ? Is facial animation performed in-texture or by the geometry ? Which parts of the object require what texture-resolution ?

If all those issues are settled and the model is finished, the most important task in the process of texturing is taken on: UV-unwrapping. Like the map of our beloved planet earth, the geometry´s texture coordinates have to be mapped and unfolded to create a square layout of the triangles, best free of distortion and uncovered spaces. It also helps to keep exposed faces linked together in texture-space, both for ease of painting and to avoid visible edges withing the geometry, as most aliasing-techniques only interpolate within the current mesh-part.

3.3) animation

Many engines resolve animation very much like in the computer-gaming-stoneage: As threedimensional "Sprites" or geometry-sequences. In that case the animation is done within a commercial package and then each frame gets exported as a deformed mesh, together with UVs. Within the game those meshes are then blended (or morphed) or replaced throughout the animation.

Such animations are quickly done, implemented and recently hardware-accelerated, but movement is predictable (often it´s looped) and weird (like feet "sliding" across the floor). Recently a lot of that is encountered by blending various motion-cycles together to create some random, organic feel.

Recent games also use bones, occasionally even inverse kinematics for animation. The relationship of mesh and skeleton has to be adjusted by vertex attributes. With bones it´s easy to apply the same motion to different characters, it´s also more flexible (scripted feet can be planted on the floor) and can react on

player interaction (individual bones can be controlled, for example to have an arm pointing towards the player). Inverse kinematics, once hardware accelerated, may very well be the key to new kinds of artificial game intelligence that creates and modifies it´s own movements. Of course, the animator will have even less to do...

Simulation is rarely feasible in games and limited to short timed particle effects. Instead, often geometry sequences are used, for example the collapsing chairs in "Return to Castle Wolfenstein".

4.) things to come

The graphics hardware will advance, no doubt about that, but wich proprietary features will become standard and wich will not is difficult to tell. Geometries will icrease in resolution (ATI already incorporates realtime-subdivs), as will textures, and even the number of usable channels will grow. Animation will slowly shift from handmade to coded, but the realism offered by the upcoming games will require capable designers and artist to create those worlds. I only hope more research will be put into innovative gaming concepts like "Black and White", I can´t stand 3D shooters and jump&runs anymore...