Creating vast game worlds - Experiences from Avalanche Studios

Emil Persson, Avalanche Studios
emil.persson@avalanchestudios.se

1. Issues with vast game worlds

Creating vast game environments introduces several complex issues that are not present in smaller game worlds. As the size of the world grows issues with floating point precision arise, causing objects to jitter, simulation to become unstable, and rendering to exhibit artifacts. These problems were very apparent during development of Just Cause 2 and had to be addressed. Depending on location in the game animated foliage exhibited vertices snapping up to decimeter sized increments. Trees were at times jumping instead of swaying. In the distance there were severe z-fighting issues. Shadows were sometimes unstable. The shadow acne that occurred could not be effectively dealt with using the standard depth bias approach. Additionally, sampling depth buffers using vendor-provided methods yielded insufficient precision. Several techniques were developed to reduce precision loss at every part of the pipeline.

Meanwhile the amount of data to represent such a vast environment is overwhelming. Consequently significant amount of work was invested in a high performance streaming solution to dynamically load and unload the required data. In addition techniques were developed to make the world appear interesting from huge distances, even for areas currently not loaded. Furthermore, several novel approaches were developed for better utilizing the limited memory available on current generation consoles. This allowed more objects to stay resident in memory.

Rendering a large game world with a lot of detail presents yet another challenge. The city in Just Cause 2 turned out to be particularly problematic, with a huge amount of objects to render. Several other locations were also equally challenging to render. A number of techniques were devised to deal with this problem, including a highly optimized occlusion culling system, a batching system, and a sophisticated level-of-detail system.

Post Just Cause 2 the Avalanche Engine has seen several radical changes presenting new issues which require new solutions for the newer titles under development. With the switch to deferred rendering, the addition of an indoor mode, new features such as dynamic lights casting shadows, a new set of precision issues has occurred with a new set of solutions required. With the additional memory required for deferred rendering and full HDR rendering the memory constraints have become significantly tighter.

2. Summary of solutions

To deal with precision issues we unchained matrices in shaders to avoid lossy dependencies. We eliminated large offsets by decomposing matrix transformation and merging intermittent translations. Floating point timers were reset on opportunity. Fixed point was used where appropriate. The depth-buffer was reversed and floating point depth was used on consoles. Sampling depth on PS3 was done with a custom resolve shader that preserved precision despite lossy results from texture unit and partial precision math.

Unstable shadows were fixed using a sub-pixel jitter compensation shift merged into the shadow matrix. The shadow range was dynamically adjusted depending on player elevation over ground. Resizing of the shadow range was made discreet to avoid unstable shadows due to range expansion. Objects were culled from shadow passes based on clip-space size. Shadow rendering costs remained largely constant even when expanded. The shadow cascades were store in an atlas to allow single-sample shadows. Cascade transitions were hidden using a screen-space dithering approach.

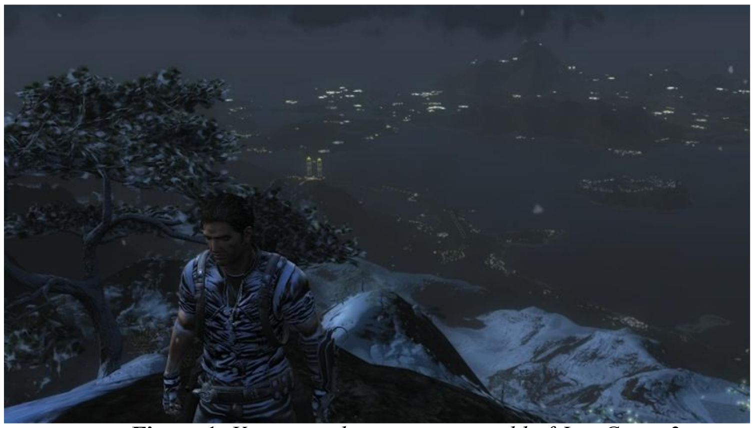
Figure 1. *View over the vast game world of Just Cause 2.*

The streamer uses a double-buffered, concurrent, out-of-order engine where reading data and creating resources happens in parallel. The data is pre-optimized on disk in gigabyte sized archives and related resources placed adjacently. The streamer automatically reorders requests to minimize disk seeks. Memory was better utilized by analyzing temporal overlap and reusing render target memory for temporally non-overlapping surfaces. Luminance textures were merged into channels of DXT textures and extracted with hardware configurable swizzles on consoles. 1ms of GPU time was traded for 6MB memory on Xenon by converting 32bit shadow maps to 16bit using a memory export shader. Vertices were compressed from float to shorts. This caused issues when vertices snapped differently for models placed adjacently, leaving ugly gaps. This was solved by quantizing models together using common settings, trading some precision for consistency. Normals and tangent vectors were compressed into single floats. Post-JC2 this was improved to store a complete tangent space (including bitangent flip-bit) in only four bytes.

The general LOD system for models was complemented with a landmark system to keep the world alive and navigable at huge distances. See for instance the twin towers of Panau Falls Casino miles away, over Rico's right shoulder in Figure 1. A distant light system was implemented to give the world life at large distances, as seen in Figure 1, especially at night, but also during daytime.

Particles were optimized using a novel particle trimming algorithm generating optimized polygons encapsulating the "meat" of the particle while cutting away as much of the empty space as possible given a target vertex count. This roughly doubled particle rendering performance on average (including clouds) without affecting visual quality at all.

BFBC (Brute Force Box Culling), a low-level asynchronous SIMD optimized occlusion culling system was implemented that operates on artist provided occluder boxes and system provided bounding boxes. Higher level systems were built on top of this where objects could be managed grid-based, as compounds or instance-based depending on particular system needs.

Post-JC2 the build system has been vastly improved with tools for running the latest build directly over http from a server. Hot-reload and direct link has been implemented to improve iteration times. Trust in the pipeline has been achieved.

SIGGRAPH 2012, Los Angeles, California, August 5 – 9, 2012.
ISBN 978-1-4503-1435-0/12/0008