Solid Earth Discuss., 6, C554–C562, 2014
www.solid-earth-discuss.net/6/C554/2014/

Solid Earth
Discussions
Open Access

Interactive comment on "Plate tectonic raster reconstruction in GPlates" *by* J. Cannon et al.

J. Cannon et al.

john.cannon@sydney.edu.au

Received and published: 5 June 2014

We thank the reviewers and the editor for taking time to critically evaluate our submission. The manuscript has been improved on the advice of the reviewers. Our responses to specific comments are below. The updated manuscript is attached as a supplement to the Reviewer 1 response for referral - all manuscript modifications are in bold typeface.

Reviewer comment:

It may be worth revisiting their opening sections (e.g. section 3.0 and 4.0) and try to clearly set a stage for the reader to appreciate the problems and solutions discussed in the upcoming sections.

Response:

Yes, this is a good idea. Especially section 3.0 which is a good place to give an overview of the steps involved in multi-resolution raster reconstruction.

We have added the following text to section 3.0...

" The rendering framework described so far can render an unreconstructed raster either directly to the computer screen when the user has chosen not to reconstruct a raster or as the first stage of the raster reconstruction process when the user does choose to reconstruct a raster. This section describes our approach to visualisation of a reconstructed present-day (or time-dependent) raster by building a multi-resolution cube map framework on top of the raster rendering framework described so far. Our raster reconstruction process involves a multi-resolution cube map, a set of tectonic polygons and a rotation model. By imprinting the raster data into the overlapping tectonic polygons and then independently rotating the polygons across the globe, using the rotation model, we are essentially reconstructing present day raster data to its past geological configuration. First we introduce the concept of a cube map as an efficient way to capture raster data in a representation that is decoupled from the raster's geo-referencing and inbuilt raster projection. We then extend the cube map with multi-resolution tiles to support level-of-detail and visibility culling necessary for efficient rendering. We show that any tile, in the cube map, can be generated by rendering the geo-referenced raster using the method described in Sect. 2. With the generation of a multi-resolution cube map covered, we finally describe in detail how raster data are imprinted onto tectonic polygons and rotated with them across the globe. The multi-resolution cube map facilitates this by enabling the raster imprinting, of pre-rendered cube map tiles onto tectonic polygons, to be accelerated by common graphics hardware. "

...note that we didn't give the major step of section 3.4 (see "we then finally describe in detail..." above) proportionately extra attention here (even though it is the largest step in section 3) because section 3.4, in turn, has many sub-sections and section 3.4.0 already gives a reasonable overview coverage of those sub-sections.

We have added the following text to section 4.0 to give an overview of the various enhancements to raster reconstruction...

" The rendering framework described so far can render a reconstructed raster directly to the computer screen in the standard 3D globe view. This section describes various enhancements to raster reconstruction that build on the approach described in Sect. 3. Visualising reconstructed rasters in the 2D map projection views discusses how interactivity is maintained in the 2D map views. Instead of rendering the reconstructed raster directly to the computer screen (as done in the 3D globe view) it is first rendered into an intermediate cube map (called the reconstructed cube map) and then rendered directly to the computer screen via a map projection. Improved raster reconstruction using age grids makes use of a second raster containing pixel values representing the age of oceanic crust. Replacing the usual per-tectonic-polygon age test with a per-pixel age test restores the missing sections of reconstructed raster data near oceanic ridges. Revealing raster detail with surface normal maps modulates the colour of a raster with the surface lighting generated from a second raster. This enables patterns in the second raster to be visually transposed onto the first raster. And finally, analysing reconstructed numerical raster data adds support for reconstructing floating-point raster data (in contrast to colour raster data). This enables numerical raster data to be analysed in the data-mining front-end of GPlates and also enables export to numerical raster file formats. "

Reviewer comment:

It is asserted that 'the vast majority of desktop graphics hardware manufactured in the last decade is more than capable of displaying raster data at interactive frame rates...' I'm not sure how to interpret this given that the increase in GPU power over the last decade has been quite remarkable (perhaps 20-100x speedup).

Response:

We can see that this is a little confusing. Essentially we were suggesting that even ten

year old graphics hardware can render texture data at interactive frame rates provided the amount of texture data is limited (and that, by inference, more recent hardware is also capable of this). So we've removed reference to GPU capabilities and changed the text to the following...

"Interactive exploration of raster data, at the fixed resolution of the computer monitor screen, requires the user to pan and zoom the view in order to expose desired raster regions and details. Since the imported raster data can have arbitrarily high resolution we must employ visibility culling and level-of-detail (LOD) techniques. These enable the raster to be efficiently rendered at the highest detail level permitted by the monitor resolution and the user's zoom level."

...note that this also includes a change requested by Reviewer #2.

Reviewer comment:

Section 2.2-3 discusses the problem of data streaming. Solid-state disk technology is also significantly faster than HDD. Modern 64-bit computing has a maximum of 64 GB RAM for standard desktops. Also, I wonder if RAMDISK technology can be coupled with GPlates. RAMDISK storage (treating RAM as storage) may also be able to utilize >4 GB RAM in a 32-bit operating system environment. Are any of these resources significant game-changers here? Also, is the cache file generated (797:21) stored on the HDD or in memory?

Response:

Actually the raster cache file is stored on disk (instead of in memory) and contains the entire raster (including its down-sampled level-of-detail versions). There is another cache containing texture resources, but this resides in memory (instead of on disk) and essentially contains only the currently visible subset of the raster. We've made a small note of this in section 2.3 and section 2.4. The texture cache will typically contain much less data than the entire raster and therefore work well on memory-

limited systems. The disadvantage of streaming is the potential for stuttering due to the difference between the time when a tile is requested and the time when that tile is available in the texture cache, which brings us to the Solid State Drive (SSD)...

A Solid State Drive (SSD) will improve streaming performance due to its low latency random access reads compared to hard disk drives (HDD). We've noted this in section 2.2. In other words, as the user pans and zooms there will be less stuttering with an SSD. However by storing our raster tiles on disk in a Hilbert space-filling curve arrangement (as mentioned in Section 2.4) we were able to mitigate much of the cost of high-latency disk accesses in HDDs (the main storage devices in use at the time when we first implemented raster reconstruction in GPlates). It turns out that, when using an SSD, profiling reveals much of the streaming time (for non-colour rasters) is taken up by CPU (Central Processing Unit) cycles spent converting the floating-point pixels to colours (using a colour palette). Since the actual SSD read latency does not appear to contribute much to profile we believe a RAMDISK (with even lower latency than an SSD) will not make a significant improvement. To reduce this secondary stuttering we will, in the future, move the colour conversion code from the CPU to the GPU (Graphics Processing Unit) where it is significantly faster for graphics hardware that supports it. Another future example of a step in the streaming pipeline is decompressing a raster tile (that is stored in compressed format on disk). The current raster cache file is not compressed (even though the original imported raster might be in a compressed format such as JPEG) and hence can consume many gigabytes of a user's disk space (for high-resolution rasters and time-dependent rasters). So instead we plan to store compressed data in the raster cache file and decompress it on-the-fly during streaming. In addition to saving users' disk space this benefits streaming performance by reducing the amount of data to be read from disk. However more processing time is then required to decompress that data. So we plan to use the Open Computing Language (OpenCL) to efficiently execute decompression on the CPU (or GPU where possible). There is also the possibility of predictively loading tiles that are just outside the visible area (and at one level above and below the current level-of-detail). So we are hoping

that a relatively modern system with an SSD combined with these future improvements will provide for a smooth streaming experience with very little or no stuttering.

Reviewer comment:

The LOD algorithm utilized here is in principle quite conventional in the 3D gaming industry. I wonder if there is a promising, significantly different avenue of future development for a scientific environment such as GPlates. For example, much of the visibility culling work of the algorithm presented is complicated, ad hoc, and even still somewhat lossy or overcomputed (even if much improved on previous efforts). For instance, in 798:2, the authors indicate that for a 1650x1050 display a raster of dimensions 2700x1350 can be used. This means that two times as many pixels are rendered than can be displayed. An alternative approach might be to find the global coordinates corresponding to each pixel in the field of view, evaluate its properties, and display its color. "Euclideon Geoverse" uses an algorithm of this kind. In this case, no LOD analysis is necessary.

Response:

As you noted, at times more pixels are rendered than can be displayed. This also can happen as a result of the non-uniformity of the cube map as noted in the Discussion section. The quality doesn't suffer but there is a higher rendering load. There are conventional methods that provide a closer match between rendered and displayed pixels. However we found the extra pixels to be an acceptable tradeoff that enabled us to implement a comparatively simple multi-resolution tile structure (based on the cube map and the quad tree) and still provide interactive raster reconstruction on a wide variety of systems. In our experience modern graphics hardware is extremely fast and can easily deal with some extra rendering load above the optimal load (provided visibility culling and LOD are still doing a decent job).

The "Euclideon Geoverse" looks very interesting. It seems they can interactively render a very large point cloud. They must have a clever spatial indexing scheme to enable

them to so quickly find which points in the cloud the screen pixels project onto (from any viewpoint). With a computer screen of around 1,000,000 pixels (eg, 1,680 x 1,500) their per-pixel test to find a point in the cloud must somehow minimise the number of disk accesses per screen draw because they are apparently not pre-loading any point data. Presumably each disk access brings in some data (say 4K) that can be shared by adjacent screen pixels due to their indexing, otherwise you'd have one million IO operations every frame (which wouldn't be possible at 15 frames/second).

Something similar to Euclideon Geoverse could remove the need for conventional visibility culling and level-of-detail techniques in a scientific environment such as GPlates. Just as a quick example, in our case it might look something like casting rays into the globe for each screen pixel to find intersection location on the globe, then finding which reconstructed tectonic polygon that includes, then performing the reverse of that tectonic polygon's reconstruction rotation to get present day position, then looking up very high resolution raster on disk (and repeating for each screen pixel). Though if you're always sampling the raster at its original (very high) resolution then I'd imagine there'd be aliasing issues (ie, raster details flickering as the view moves). And this is what level-of-detail normally addresses through filtering because a single screen pixel could cover a potentially very large number of raster pixels which would then need to be filtered/averaged to avoid aliasing. So this is just for raster data on a spherical surface which would appear to still need pre-generated LOD to be feasible in which case on-demand tile streaming plus GPU polygon rendering is starting to look more appealing. We are no experts in point cloud rendering so we are not sure if aliasing is a problem there or not. In regards to 3D data, GPlates can render isosurfaces and cross-sections of 3D scalar fields (to be covered in another paper) but those datasets are not very dense compared to point clouds and can fit relatively easily in GPU memory (at least on recent desktop graphics hardware), or at least still look good when down-sampled to fit, and then super-fast GPU ray-tracing does the rest. But certainly for very large 3D datasets an approach similar to Euclideon Geoverse would be worth looking into.

Reviewer comment:

Those unfamiliar with plate tectonic rotation modeling may be confused by the continued reference to plate tectonic motion as dictated by a 'rotation model'. It may be worth briefly mentioning that rotation models refer to biaxial rotations in 3D space of a rigid surface across a spherical shell.

Response:

We have added the following text near the beginning on the Introduction...

"The rotation model provides rotations for each tectonic plate over a period of geological history. Each rotation, consisting of an axis passing through the globe centre and an angle, rigidly rotates a tectonic plate across the spherical surface of the globe."

Reviewer comment:

The continued reference to 'reconstruction' in the work was initially confusing to me. It may be worth a slightly more forward explanation in the introduction (about 794:25), indicating that mapping present-day geospatial data onto the globe is essentially a construction, and that modifying the rasterized data and tectonic configurations amounts to a 'reconstruction' of the tectonic state over geologic history.

Response:

We have added the following text near the beginning on the Introduction...

"Typical geospatial data consists of present day observations on tectonic plates. Due to the movement of plates throughout geological history this data must be reconstructed from its present day configuration to its spatial arrangement at past geological times before spatio-temporal exploration can occur."

Reviewer comment:

796:8 rending should be rendering?

Response:

We've changed this to 'rendering'.

Please also note the supplement to this comment:
http://www.solid-earth-discuss.net/6/C554/2014/sed-6-C554-2014-supplement.pdf

---

Interactive comment on Solid Earth Discuss., 6, 793, 2014.