An Introduction to Surfaces and Terrains, Part 2

Transcript

Colin: Hello, and welcome to the ESRI Instructional Series Podcast. This broadcast is entitled, *Terrains, Terrain Datasets: Some Best Practices*. I am Colin Childs from the Educational Services at ESRI in Redlands, California, and today I'm joined by Clayton Crawford, project manager for the ArcGIS 3D Analyst team. Clayton is going to answer some questions on terrain datasets, and share some best practices with regard to terrain datasets with us during this broadcast. This discussion is tailored to users of 3D Analyst who have to deal with large amounts of data representing surface information, and who want to learn more about terrain datasets, and how best to create and set them up.

So, before we begin, let's have a little recap on what we learned earlier in our previous podcast. So, the terrain dataset is unique to ArcGIS, and is a collection of participating feature classes that live inside a feature-dataset-like structure. You establish the role of each of these feature classes, and what that role will be within the terrain, and how each data source is going to be used. So, I'm going to lean over, and ask Clayton a little more. So, Clayton, could you tell us briefly a little bit more about the structure of a terrain dataset? Is it akin to a feature dataset? Is it different? Is it similar? Maybe you'd like to tell us a little bit more about that structure itself.

Clayton: Alright, Colin. Well, a terrain dataset lives inside a feature dataset. It's like a topology, in that it's an entity inside a feature dataset that has participating feature classes associated with it, and rules tied to each feature class defining how it participates in the terrain; and you will see it listed as a data entity in ArcCatalog's Table of Contents tree. It looks like a dataset there. You can rename it. You can copy it; preview it. You can add it as a layer to ArcMap, for example.

Colin: Okay. So, like a topology, it has rules, and you define what happens to point data, what happens to line data, and so on, and the roles that it plays within the actual terrain dataset itself.

Clayton: Yes. Good.

Colin: So, what's the typical workflow for creating a new terrain dataset? Is there a method? Is there a workflow you need to follow through?

Clayton: Yes, there isn't one specific workflow. It all depends on where you're getting your data from, what kind of data you have; but there are a few typical things that people go through. First starting off with the source data (typically comes from a data provider, we're talking about usually photogrammetric data, lidar data, sonar data), so these come in ASCI format files. With lidar, there is a standard binary format now, called LAS, so people often get collections of these files, and usually for larger projects there can be many files involved. So, there's this data loading or importing process that comes into play, and we have a couple of importers that can be used in our geoprocessing toolset to load that information into the geodatabase. So, those create your feature classes. We identify which feature classes are going to be used to define the terrain, and for each one, indicating if they have Zs (usually they do), where the Z values come from (could be from the shape field itself), if it's 3D geometry, or an attribute field; and also the role that the feature class plays in defining the terrain. And we call that the surface feature type, or SF type. Is it a mass point, break line data, clipped polygon data, etc.

Colin: Okay. So, Clayton, tell me. Are there some things that you would recommend considering when creating a terrain dataset? You know, what sort of things would you recommend to people when they get to the point where they're ready to create a terrain dataset?

Clayton: Well, the first thing I would look at is the amount of data involved, making sure that I have enough disk space to stage and load the data, store the data, build the terrain from it. So this comes in first, just knowing, kind of having a rough estimate, of the number of points, and what format the data is coming in, to make sure that I have the ability to load the data into the database properly; using the tools that we have available (the spatial reference of the information, both the horizontal and vertical datums being used, and associated with that). You know, we have a resolution property that is used to store data in our geodatabase that impacts storage requirements. The higher resolution or the more detail you want to be able to store and resolve with your data, is going to take up a little more disk space. So, for large projects, those kinds of things become more important.

There's also (for very large products), you may consider doing things through different stages. It can be that the terrain might take days to build, and you could be uncomfortable with something that's going to take five days to construct a terrain from, if you've got billions of points that you're dealing with. So, it's possible to do it in smaller chunks, where you take a day, really at a time, and so if there's an unplanned power outage, or a hardware failure of some sort, you don't lose everything. So that needs to come into play, and so I would always recommend to do a dry run, using a subset of the data, and using the same workflow. You know, have a planned workflow, and try that on the subset of data. Make sure everything works as expected, and then you can feel more secure in going and processing the larger dataset.

Colin: Just returning back to spatial reference. So I'm assuming that spatial reference is really important. Is it applicable to build the terrain on data that has, say, geographic coordinates? Or should your data really be in a projected coordinate system?

Clayton: The data should be in a projected coordinate system. The terrain relies – it's a TIN-based surface, and our TINs use a Delaney triangulation. Delaney triangulation is based on proximity of points, and expecting that the X and Y distance are constant (Euclidean distance operator), so to have a valid Delaney triangulation, we need the coordinates to be projected.

Colin: Ah. Interesting. Very good to keep that in mind. So, tell me, I find these terrains really, really interesting, and so what sort of methods have you deployed in the terrain dataset to facilitate the speed and scalability, and the ability to deal with these huge amounts of data (as we obviously can in a terrain dataset)?

Clayton: There are a number of things that we combine to enforce together to help us out. First off, we do have an internal tiling scheme. So, a horizontal tiling structure, where it's kind of a regular tile system that gets applied to the data so that we can create some spatial coherence, and chunk it up into manageable pieces, such that each tile has a manageable amount of data in it that can fit into memory and be processed as a unit, and that facilitates scalability there. We also, in addition, add the concept of pyramiding, which is similar to raster pyramids that, I think, a lot of people are already familiar with, in that taking a smaller sub-sample of the data to be used when being viewed, or used at smaller scales.

You don't want to try to access the full resolution data when you're accessing it at a very small scale and cannot possibly resolve the detail on the display screen, for example. So we do something similar with terrains, except that it's different: we have vector pyramids. So, we're actually taking the source measurement data, and separating them based on their importance in defining the terrain from a vertical perspective.

So the user can define the number of pyramid levels that we have, and each pyramid level has a Z tolerance resolution associated with it, and that is, that represents the vertical accuracy of that pyramid level relative to the full resolution source data. So, you get less and less accurate with smaller-scale access, or lower levels of detail, but that requires less measurements. And so basically, what we have is that we only need to retrieve the source measurements based on Area of Interest and level of detail requirements. We view, maybe, the full terrain at a small scale, but at a low level of detail, so we just pick a subset of measurements as we zoom in, (into a small area), we gather the full resolution set of points, but for a small area, so we're never overloaded with points. So we have the horizontal tiling system, and the vertical tiling, per se, in pyramids. And also, we keep the source measurements as is. We don't actually build a TIN.

It's a TIN-based surface that gets used in the end. You see, the surface representation is TIN-based, but we don't actually store the TIN. We just store the measurements and build TINs on the fly, on demand, as needed.

Colin: With the number of sample points at that time?

Clayton: Yes.

Colin: Sounds like a very efficient way of dealing with a large amount of data. So, just to summarize: database pyramids, and data partitioning in the vertical domain. Very interesting! Potentially, you have very, very large numbers of data points used in a terrain dataset. Maybe you could tell us a little bit more about what you do with this potentially large amount of data that you're working with as well. Do you use point clustering? Or what sort of technique do you use to deal with that?

Clayton: We found that we could not dedicate one database row per point. We are working in a database. Other systems use file-based techniques for things, but we want to provide database management tools for our surface measurements, and that's a very good strength of ours. But a potential weakness is giving a database row per points, when we have hundreds of millions or billions of points. That doesn't scale properly. So, we do have a multi-point shape type that the system supports, both for low-level needs and higher-level needs. So, terrain internally clusters points (or groups them) into multi-points, where we can put thousands of points into one database row, and store them as a multi-point with an individual row. So, that helps things scale quite effectively.

Colin: So reading and writing becomes a lot more efficient because you're not reading individual records; you're just reading a whole bunch of them (a cluster).

Clayton: Right. A read of one row gives us thousands of points.

Colin: Excellent! That's excellent! So tell us, how do these terrains interact in ArcMap, and, let's say, in ArcGlobe, for example? Are there any unique ways they interact, or do they work the same as any raster surface, or TIN surface?

Clayton: The terrain rendering most closely resembles TINs in terms of the drawing. So, you would add a terrain to ArcMap as a layer. So we do have terrain layer type, and the initial display of that will look pretty much just like a TIN (color, hill-shaded display). If there are break lines, they'll be turned on. You can go to the terrain layer properties. The Symbology tab there provides different options. You can display it based on slope or aspect, or a single-color hill shade value, just like a TIN.

Colin: Alright! So, the same kind of analysis tools are there that you would use for a TIN, you can use on the terrain?

Colin: It's exactly the same.

Clayton: Yes. The same, with one distinction: that the interactive analysis tools that we have on the 3D Analyst toolbar (line of sight, steepest path, contour, interpolate shape tools), they will only enable when the terrain is being displayed at its full resolution by default.

Colin: Interesting!

Clayton: So it's sensitive. Those tools are sensitive to the resolution that the terrain is being displayed at (what pyramid level is being used for the display), and you can go and say that you want those tools to start enabling at a certain scale level (or accuracy level) that the terrain is being displayed at. So, if it's really small scale display, the terrain isn't highly accurate. You don't want to enable some analysis tools by default, and so they're sensitive there. That's the only difference.

Colin: Okay. So, as you're zooming in/zooming out, you're obviously redrawing or re-creating the surface, and so those tools are enabled or disabled depending on what scale resolution you're at.

Clayton: Yes.

Colin: Excellent! So, Clayton, let's finish up a little here with a little talk about performance and size estimates, and interesting issues and things that have come up, maybe.

Clayton: Well, people are first concerned with getting the data loaded into the geodatabase. So, in terms of the raw loading of points, whether it's from ASCI files, or the lidar LAS binary format, we can bring in about a hundred fifty million plus points per hour into the geodatabase.

Colin: Wow!

Clayton: That's the first part. And then, once the terrain is defined in terms of which feature classes are participating, and you define the pyramiding scheme (the schema for the terrain), there's a *build* process to build the terrain pyramid, and for that we can process somewhere between ten to twenty million (maybe upwards of thirty million points per hour). And I know that sounds like kind of a wide range of capability there (ten to twenty or thirty millions of points per hour), but it depends a lot on the nature of the data that you have. Is it relatively flat or rough data compared to your pyramid (the values you use to define your pyramid)? Are you using a very fine resolution value, or not? So there's potentially a wide range of work that needs to be done in the build process.

Colin: I guess this is why you're also recommending people try out with a smaller subset of the data first; find out what is optimum for them before they commit to building a complete terrain out of a very large dataset, right?

Clayton: Exactly. And once you have an estimate for the smaller size, adding more data (it tends to grow), it's a linear growth in terms of time, so it's not some exponential, or other, or logarithmic curve, so you could get a reasonable estimate using a smaller subset of how long it will take to build a larger subset.

Another thing is the storage requirements (lots of data). People need to figure out how much disk space is going to be required. And we found that it takes about 1 GB of storage for a hundred and fifty million points or so, and that will vary based on the resolution parameter that's set for the feature classes, or the feature dataset when you define that (how much storage space will be needed), and then you may or may not need to double that requirement depending on whether you are building a terrain using what we call *referenced feature classes* or *embedded feature classes*.

Colin: Very interesting. So, just let's finish up here with this podcast, and do you have a couple of things that you'd like to point out that people should sort of look out for? A couple of limitations maybe, or things to be aware of, and things to watch out for when creating a terrain dataset?

Clayton: Well we offer different kinds of geodatabase support. We have personal geodatabases, file geodatabases, SDE. Personal geodatabase does have a 2 GB limit in terms of its overall size.

So, there is a significant size constraint there. If you're dealing with lidar data, I would avoid personal geodatabases for large lidar projects. The pyramiding scheme that we use for terrains uses a Z tolerance filter to do the point thinning or generalization, and that works best with Bare Earth lidar data. So, while terrains can be built from first return lidar, which includes tree canopy and building tops, the pyramiding is not so effective with that form of data. So, you can use first return lidar. Just expect to access it at large scales, and not expect the pyramiding to work great for you at small scales. And then, also in terms of the coordinate system, you need to use projected coordinates for your data.

Colin: Okay. Any resources that you could recommend people consult to learn a little more about terrains?

Clayton: The help system has a lot of information available in it (conceptual discussion) covering some of what we've talked about here: about what a terrain is, and how it works, and how best to build a terrain. So, I think that's the first resource to go to. Also, the Terrain Build wizard; the other geoprocessing tools that can be used with the terrain. The mechanisms on the user interface all have context help associated with them, so certainly, use the context help. And then for starters, we do have a 3D Analyst tutorial with an exercise that's specific to importing and building, and using a terrain. So I would recommend going after that exercise in the 3D Analyst tutorial.

Colin: Excellent! Clayton, thank you very, very much for helping out, and coming to answer some questions during the podcast! Alright everybody, for further resources, as mentioned, check out the ArcGIS Desktop help topics on terrains (surfaces, and terrains). Also remember to consult your ESRI software product pages, and the tutorials.

Thank you for tuning in to this session of our ESRI Instructional Series Podcast. Stay tuned for future broadcasts.

