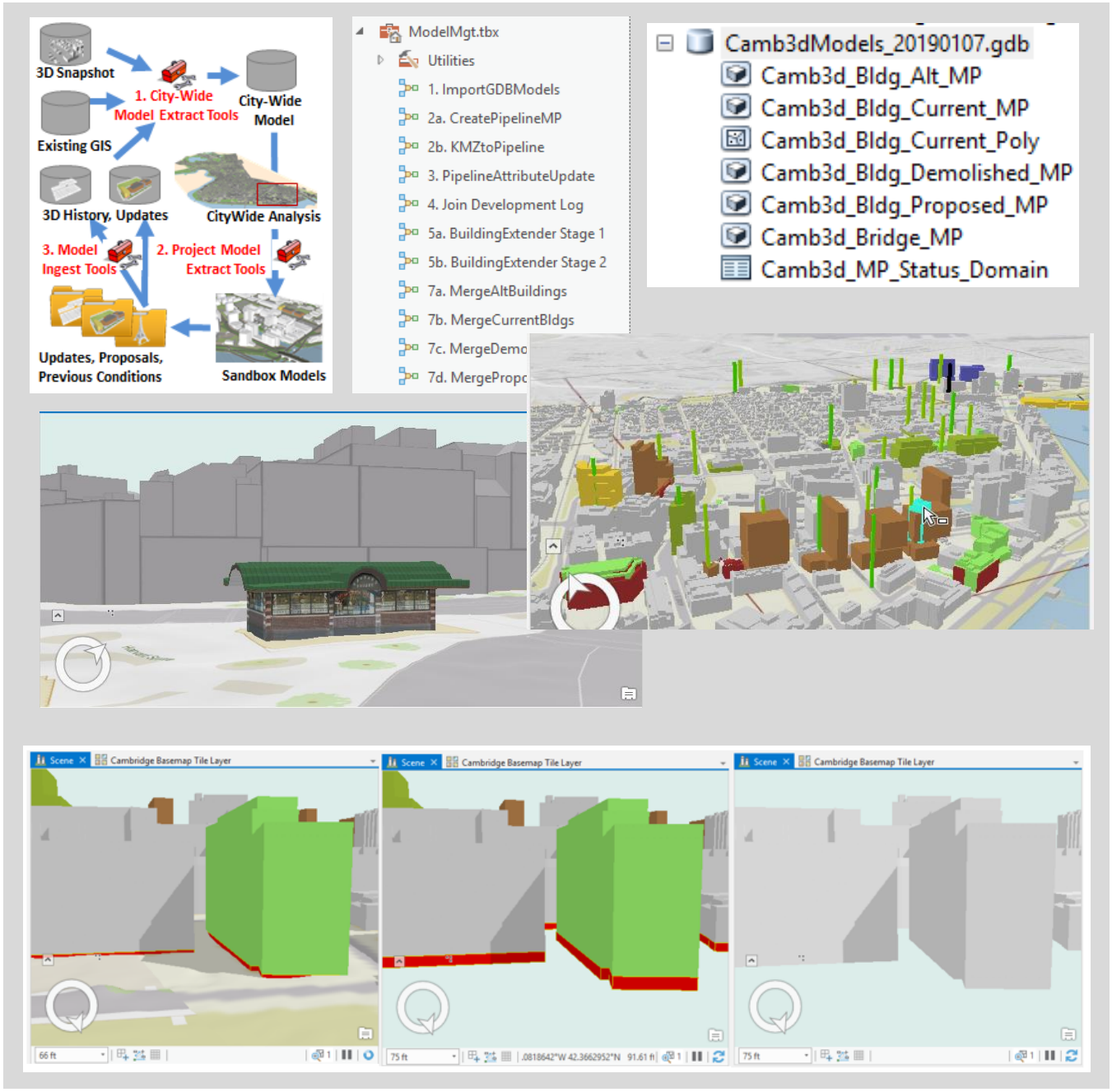
Managing your City-Wide 3D Building Model Collection

Cambridge 3D Geographic Information System



City of Cambridge, Massachusetts
Geographic Information System
January 13, 2019 Accompanying Service Pack 11.

Prepared by:
Paul B. Cote Geographic Information Services

cultivating spatial intelligence™

Managing your 4-Dimensional City Model

This user guide discusses an architecture and a set of workflows and tools for developing and maintaining a collection of three-dimensional models representing all of the buildings in a city. The data architecture incorporates measured models of existing buildings, hand-made models of building proposals, and model updates that may be compiled by photogrammetry or through the submission of individual models from developers, and wholesale exchange of buildings with cooperating independent city-modeling projects managed by independent GIS agencies within the city, like universities.

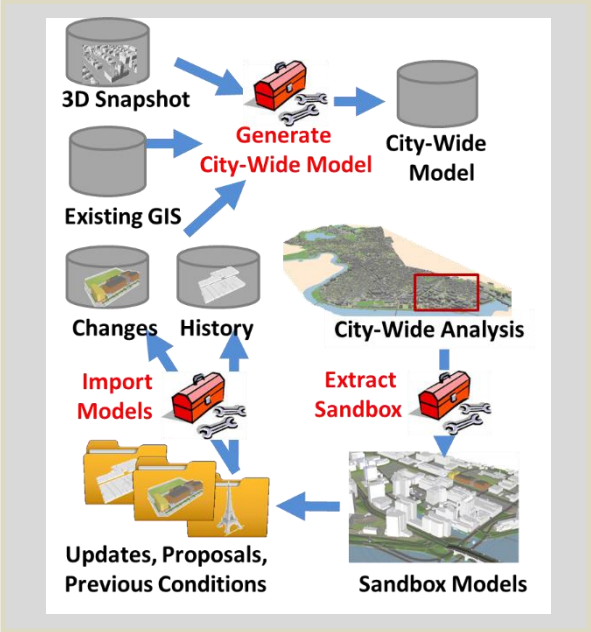


Figure 1: Cambridge 3D GIS Architecture

By providing a simple means of keeping track of building models throughout their life-cycle from proposal, through active use to demolition creates a framework for understanding the context of lots of different stuff for planning and design, the context of historical documents; and how these relate to places in the current scenario. The workflows and tools described here will assure that routine updates will preserve a historical record of the shape of neighborhoods through time.

Figure 1 illustrates the lifecycle of data in the city-wide 3D model. The city-wide model is generated from the city's existing terrain and planimetric layers and a collection of 3d models of buildings and bridges. The workflow and tools discussed here are concerned with update and preservation of building and bridge models as shown in the bottom left corner of Figure 1.

Improvements for 2019 Edition

This document supersedes the ImportModels Workflow document that was issued in July 2017. This redesign includes

Simplified file organization:

The organization of the folders and data that are necessary to work with the city-wide model management system have been separated into functional units. For example, all of he multipatch feature classes related to managing 3d models have been put into their own geodatabase: Camb3dModel.gdb.

Data sources and products that have been used to create the master model – like the original data-sets issued by our 3D vendors or source models in KMZ or sketchup format are now and the data products, like sandbox models and tiled exports have been taken out of the Master Model folder and placed in a separate Camb3dArchive folder – which may gro in size without making it more difficult to clone or back up the essentuial data-sets in the master-model.

The new file management concepts, combined with a catalog approach to model attributes make it much simpler to find the data that you need and saves time spent trying to figure out where new data should be filed (I hope!)

An Archival Approach to Model Management

As the Cambridge 3D model grows through more updates and quality control we have learned that it can be important to be able understand the lineage of individual models and to be able to easily find the source data for each model. Part of the simple rules for filing model sources makes it possible to trace individual models in a 3dModel_MP feature class to its original sources. This becomes more valuable as the city-wide collection of building models becomes a more diverse patchworks of different batches of models produced by different sources and methods.

A completely new table design for the multipatch tables.

This table design has lots of new features, including build-in links to Google Streetmap Oblique views of buildings, Pictometry Bookmarks, and links to the up-coming CDD home page for buildings that are in the Large Project Review Pipeline. The table design and tools workflows wil make it easy to implement further improvements in the Building ID system and coordination of the current Building ID system with the Assessing department's and the Cambridge Historical Commission's building identifiers. If you can't wait to see this, the **Data Dictionary** for our new **Camb3d_Models_MP table** scheme is provided in the appendix of this document.

All tools and workflows are carried out in ArcGIS Pro

There is no longer any need to carry out some of the model attribution tasks in ArcGIS desktop.

The 20180101 Updates

After applying the new table schema to the (previously) latest multipatch feature classes, pbcGIS went through a large quantity of updates. This included about 218 models created by a contractor using Satellite Imagery (GeoEye 2016). We also incorporated a wholesale swap of building models contributed by the Harvard Planning Office. And 14 new hand-made models from a variety of sources which have been submitted as Sketchup format models. Integrating these updates gave us a chance to understand some of the difficulties of quality control, of migrating models into the existing database with all of the splitting and demolition and QA QC details. All this along with some of the curve-balls introduced by the almost ready for prime-time ArcGIS Pro.

Eliminated gaps between buildings and Terrain Models

Being collected photogrammetrically, most of our building models just touch a terrain surface as interpreted by the building creation procedure. The problem with this is that a different terrain model may dip a little more in certain spots. This would be true of more detailed terrain models or even less detailed ones. The solution to this is to add basements to the bottoms of buildings so that there is no visible gap visible between the building bottoms and any reasonable terrain model. We have written a geoprocessing tool that adds an eight foot basement to each of our buildings.

Safer, saner workflows.

Our experience with the latest batch of updates provided more in-depth experience with a high volume of a variety of different types of updates and quality control situations. This gave us an opportunity to

think through workflows for checking and admitting and demolishing buildings as long transactions that always provide a means of checking and recovering from mishaps that can happen through user error or sketchy behavior of ArcGIS Pro.

Cambridge 3D GIS Model Management Data and Workflows

Take the Tour!

The documentation that follows is organized as a guided tour through the folders, data-sets and documents delivered with Service Pack 11. Once you unzip the service pack you will see the Archive and the ModeMgt folders depicted in figure 2.

New 3D Model Management Architecture

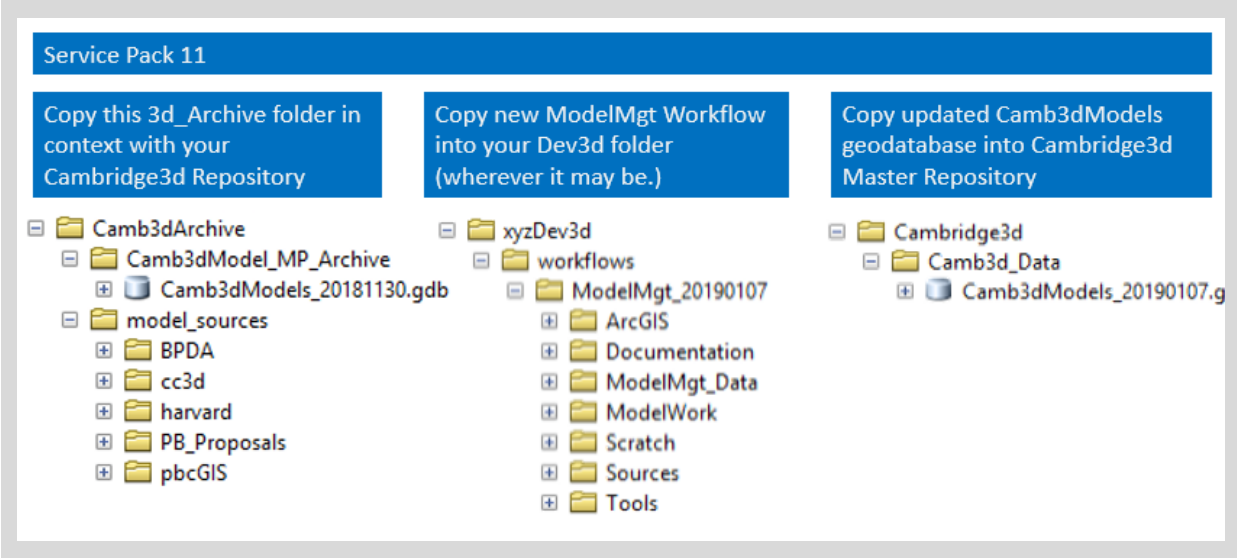


Figure 2: Service Pack 11 Items and their destinations

The **Cambridge3D Master Repository** (introduced as the "Mother Model" in the Service Pack 10) is intended to be a self-contained workspace that includes all the data, tools and documents associated with the current working versionof the city-wide 3d Model, including 3d models, terrain and groundplan resources. This modular architecture makes it easy to make a single back-up or replica of the complete city model, including terrain, groundplan and buildings with no external dependencies. The **Cambridge3d** repository provides a central place where various 3D applications can reference the most up-to-date 3D data resources.

Continuing development of derived 3d products, like sandboxes and tiled exports and managing the continuously evolving collection of models is carried out with independent **Development Workflows**, which are collections of tools and documents encapsulated as independent replicable folders that are designed to be easily forked into independent working factories called Workflows. These modular workflow folders are designed to be self-contained and could be used anywhere in the enterprise file system, but they may reference to one or two data-sets from the Cambridge3D Master Repository to

avoid needless duplication of data. From Service Pack 10 we recommend that these development workflows be managed outside of the **Cambridge3D** master repository.

The Camb3D Archive Folder

The Archive folder is another new feature introduced with this service pack. This folder is where persistent source material including archived copies of the Master model, derived sandbox models and the tiled export files can be stored. It makes sense to store these files outside of the Master repository so that the master repo will be lean and mean and easy to replicate and back up without duplicating a lot of stuff.

About Model Sources and Batches

The archival approach to managing model source material is another new feature that comes with this service pack. It is important to keep source data for the models that come into the system. In the case of models that originate or are modified using design tools like SketchUp, the editable model manuscripts contain a lot of structure like layers and grouping that are lost in the process of importing these to multipatch features. Updates that come in as geodatabases will have their own attributes and other information that will be stripped away. There is also a chance that some subtle corruption may occur in the translation process that may not be discovered until later.

The concept of model batches has been introduced as a means of referencing the source material that often issued or updated to the master model as batches that have a particular author and issue date. The **Model Sources** folder has a sub-folder for each source. Within these folders, model batches are individual folders named Source_Issue date (e.g. cc3d_20130101). The YYYYMMDD format for date strings is preferred since it forces the folders to sort chronologically.

The Camb3d_Models Geodatabase

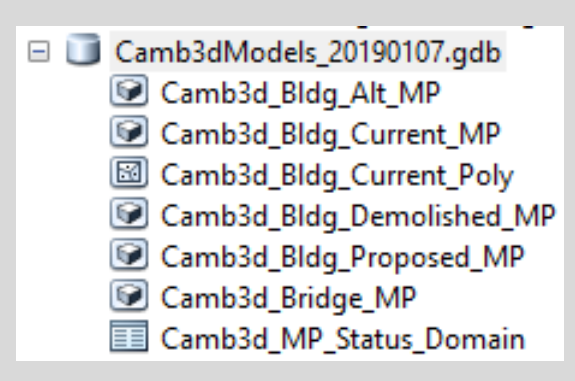


Figure 3: The 3D_Models geodatabase

One of the improvements in this update to the Cambridge 3D City Model architecture is that the feature-classes related to 3d model management are now separated from the data-sets related to the development of the terrain model and groundplan. Now, all multipatch feature classes for buildings and bridges are stored in the **Camb3dModels** geodatabase. This facilitates checking out the feature-classes for updating and checking updates back into the Master repository. This process will be explained in detail in a couple of pages. For now, it will be useful to look at the subject of the updates: The feature classes that reside in the **Camb3dModels Geodatabase.** Applying a date string to the name of this geodatabase is essentuial for managing updates and backups.

Camb3d_Models Feature Classes

Here is a summary of the concept behind each of the multipatch models in the Camb3dModels geodatabase.

Camb3d_Bldg_Current_MP: These are all of the building multipatches that would be rendered in a current view of the city. This includes buildings that have been proposed or permitted for demolition. It does not include buildings that were under construction in the latest update period. All of the models in this feature class are made from measurements of the as-built condition.

Camb3d_Bldg_Current_Poly: The footprints of the features in Camb3d_Bldg_Current MP.

Camb3d_Bldg_Proposed_MP: 3d multipatch features of any building model that was created from drawings of proposed buildings. This includes model that were submitted by developers and their architects. Even after the buildings move through the under-construction to complete status, their models remain in this feature class. This rule is designed to keep all of the proposal models in one location rather than shifting them into the **Bldg_Current_MP** feature class and then back out again. Once the as-built condition has been measured and modeled, the model of the proposal can be compared with the as-built. Over time, Camb3D_Bldg_Propsoed_MP will be an ongoing record of every building proposal (if CDD ever decides to collect models of proposals.

Camb3d_Bldg_Current_MP: This feature class is the resting place for models of buildings that have been demolished. During the update process, buildings that are removed get their Disappear Date updated and the model is copied into Camb3d_Bldg_Current_MP before it is removed from Camb3d_Bldg_Current_MP.

Camb3d_Bldg_Alt_MP: is a feature class where alternate versions of buildings can be kept. For example, the textured version of the Out of Town News kiosk is in here, and some LOD3 models of the Fogg Art museum – which has some problems that need fixing. These models can be inserted into scenes through the use of definition queries to suppress their counterparts in Camb3d_Bldg_Current_MP.

Camb3d_Bridges_MP: Holds multipatch features for bridges.

An attribute data dictionary for the **Camb3dModels** feature-classes is provided in the appendix of this workflow document. I recommend that you look through that as understanding these will be necessary for understanding the procedures described in the detailed explanation of what the Model Management tools do.

Model Management Workflows in Context

Figure 3 illustrates the big picture over-view of the Model Management workflow. To grab a term form database design, this update workflow uses many of data management patterns of a Long Transaction. The working version of Camb3dModels.gdb lives in the Master Model folder. At the beginning of the update cycle, the geodatabase is "checked out" and copied to the 3D_Archive repository. This initial state is also copied to a Camb3dModelMerge geodatabase within the ModelMgt_Data folder, along with all of the updated model where lots of stuff is done to them. Ultimatelym this results in new versions of

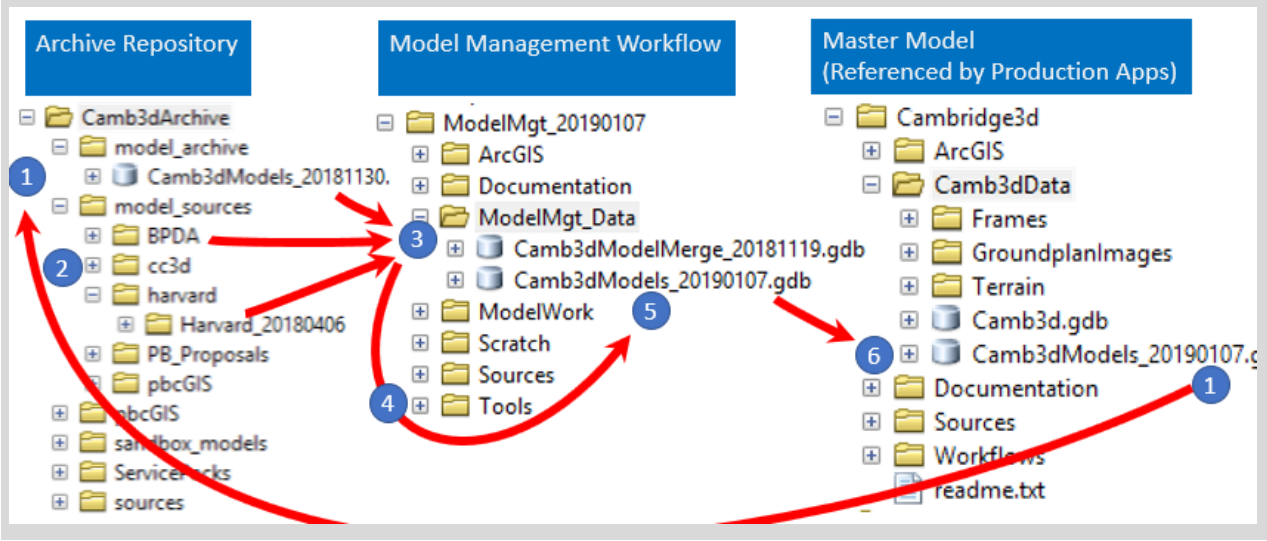


Figure 4: Lifecycle of the Camb3d_Models Geodatabase

the feature classes in the Camb3dModels geodatabse, that replaces the initial state that was checked out at the beginning.

The numbered steps illustrated in Figure 3 are outlined in more detail below:

Step 1: The existing modelgeodatabse is copied from the Master Model to the Model Archive folder. The date string on the archived models geodatabase is changed to reflect the date that the geodatabase was copied.

Step 2: Updated or improved models form various sources are copied into the **Archive/Model_Sources** folder in batch folders whose names reflect the source and the issue date of the new models.

Step 3: Source models are imported into the **ModelMerge geodatabas**e in the Model Management workspace. This step is accomplished using a few tools that will be described later.

Step 4: Lots of stuff is done to the feature classes in the **ModelMerge** geodatabase. These procedures are described in detail below.

Step 5: The result of the Model Management workflow is a new Camb3d_Models.gdb with a new date string that reflects the date that the new updates are issued. This is copied into the Master

Model Camb3d_Data folder where the updated feature classes will be referenced by production apps.

Step 6: While the models reside in the master model, there may be ad-hoc updates and status changes performed by the model administrator.

Model Management Workflow and Tools

The Model Management workflow is designed around an arrangement of folders An ArcGIS Pro document and a collection of geoprocessing tools. The folders in the Model Management Workspace are shown in figure 4. All workflow folders have a similar folder architecture which is designed to make the workflows modular and self contained and easily to clone and re-use.

- **ArcGIS folder:** holds map documents and ArcGIS Pro project documents
- **ModelMgt_Data folder:** The persistent data products of the ModelMgt workflow go here.
  - **The Camb3DModelMerge.gdb** is a staging area where the various update features and the copies of the initial 3d feature classes are brought together and flagged as described below. These data-sets are preserved as a means of making the transition form the initial datasets to the updated versions canbe easily repeated and fine-tuned.
  - **Camb3dModels_YYYYMMDD.gdb** is the final output of this workflow.
- **ModelWork folder**: holds the sketchup models that are used to repair or split models or to clean up models of proposed buildings and turn them into KMZ models for import into the geodatabase. These models will end up being copied into the Archive/Model_Sources folder when the particular instance of the ModelMgt workspace is finished.
- **Scratch Folder and Scratch.gdb:** Are were intermediate products of models or experimental data-sets are stored.
- **Tools Folder:** are where the geoprocessing tools and associated python scripts are stored.

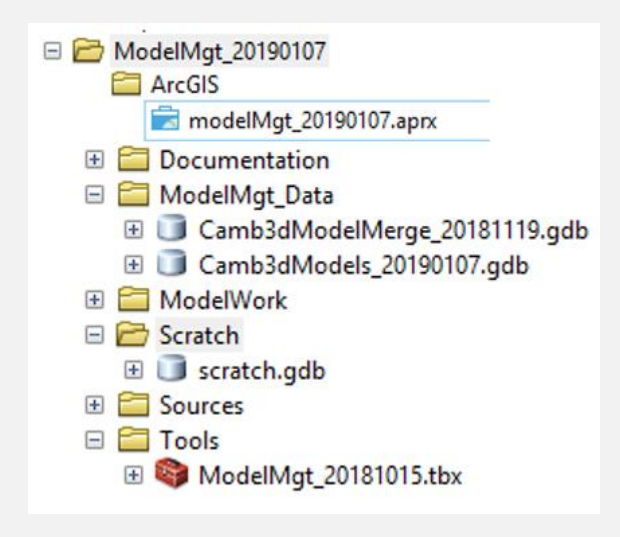


Figure 5: Model Management Workspace

This workspace and its documents and tools are designed to be fre-standing and should work no matter where you unpack this folder. Following the convention described in the Camb3d_User Guide, it makes sense to place the Model Management workflow folder in the tour Dev3D folder where you keep your other 3D projects.

The ModelMgt_YYYYMMDD ArcGIS Pro Project

The ArcGIS pro project, in the ArcGIS folder is the setting for all of the procedures necessary to update the feature classes in the Camb3DModels geodatabase. To make this project work in your environment, you may need to check the Project Options to set

- **Home Folder:** should be your instance of the ModelMgt folder.
- **Default Geodatabase:** Scratch/scratch.gdb
- **Default_Toolbox:** Tools\ModelMgt_20181015.tbx

Another critical aspect of workflow projects in ArcGIS Pro and Desktop that you should check are the geoprocessing environment variables for

- **Output Coordinate System:** Massachusetts State Plane NAD 83 Feet; with Vertical Coordinates as NAVD83 (Height_ feet.
- **Workspace and Scratch workspace:** the Scratch folderin your instance of the ModelMgt workspace.
- **Fields:** DO not use Fully Qualified Field Names

ArcGIS projects set up this way seem to be good at adapting their path references to where ever you put them using relative pathnames. Nevertheless, you may want to check out these properties, since they are critical for making sure that tools and workflows will work without problems when the **ModelMgt** project is cloned and re-used with a different folder name.

The ModelMgt Toolbox

The heart of the Model Management is the collection of geoprocessing scripts the ModelMgt toolbox. You can find the tbx file in the ModelMgt/Tools folder. This shows up wen you open the catalog tab, since it has been declared as the default toolbox of the ModelMgt.aprx arcpro project. Notice how I have made a back-up copy of the toolbox by making a copy of it and appending a date-string to it. This is a useful thing to do when you set out to hack these tools.

Full disclosure, the tools described here are made in the spirit of getting the job done. They are not slick and may require some adjustments to suit the task at hand. I generally recommend running them one step at a time the first few times you work through a model.

I am running ArcGIS Pro Version 2.2 Service Pack 4. I find that there are times when tools that ought to work do weird things such as tools that seem to run

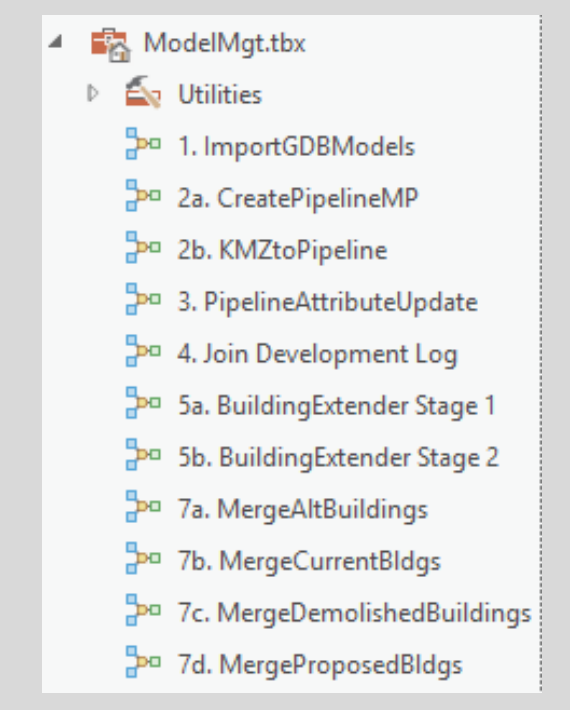


Figure 6: The ModelMgt Toolbox

and return no errors without doing anything. Other times, when a couple of rows of a table are selected and you try to calculate some field values, it can clobber the values of all of the rows. This sort of stuff is very disconcerting. Eventually one learns to double-check the check the before and after condition of every operation before you commit your edits. These lessons have also led to the update pattern described in the steps listed below.

Safe Data Handling is No Accident!

A person could manage their 3D Models geodatabase by just copying new features into it or replacing models using the ReplaceMultipatch tool. Very straightforward editing procedures like this are inherently unsafe and are likely lead to situations where irreversible changes are made to the collection of building models, and in many cases, you won't even be able to tell what those changes were, let alone have any hope of recovering the lost data.

Step-By-Step Workflow

The procedures described below are intended to provide a safe, sane workflow for making updates to the city-wide buildings collection with a means of saving source data, initial conditions and producing updated 3d models feature classes, with each step being easily reversable and with complete capability to audit all changes between versions of the 3D Models feature classes.

Check Out Initial Cmab3dModels.gdb

1. Make a copy of the ModelMgt workflow folder and change its date string to today's YYYYMMDDD
2. Make a dated copy of the current Camb3d_Models geodatabse from the Master Model and save it in the Camb3D_Archive folder with the current YYYYMMDD date string appended to its name.
3. Save another copy of the Camb3D_Models geodatabase in your ModelMgt/ModelMgt_Data folder. This should replace the existing Camb3D_ModelMerge.gdb that is in there. This **merge geodatabase** is going to become your model merge workshop where you tag re-shuffle a bunch of 3D models.

One assumption that we are going to make is that nobody will make any ad-hoc changes to the versions of these feature classes in the master model while you have these data-sets checked out.

Import New Feature Classes:

1. If you have received new updates in the form of geodatabase feature classes, you should check their normal, and fix them if necessary. The original model files should be filed in a new batch folder within a new batch folder in **Archive/Model_Updates/Source/Source_batch.**
2. These features should be merged with the Camb3D_Bldgs_MP feature class template. If necessary, you can make one of these from the latest Camb3d_Bldg_Proposed_MP feature class. The tool, **1. ImportGDBModels** facilitates this process and also makes sure that each new model is assigned its permanent model ID and several other attributes related to the model batch. The output of this model should be a new feature class in the merge geodatabase

3. For model updates that are coming to you as KMZ or COLLADA files, you will need to first create a new empty pipeline feature class using the **2a. CreatePipelineMP** tool. I would call he new feature class **handmade_pipeline** and store it in the merge geodatabase.

The preferred way of importing handmade models is form KMZ files. The tool, **2b. KMZtoPipeline** takes care of this. This model also assigns a new Model_ID and several attributed regarding the model.

QA QC and Merging Updates and Demolitions

Now you are ready to visit each of your pipeline features and do some QA/QC and set flags with merge instructions.

1. I find it useful to first do a Select by Location query to find the buildings in the Bldg_Current_MP that intersect with the pipeline features, then set a flag on these that allows you to make them a definition query layer that shoes just the intersecting buildings, and another that shows all the rest of the non-intersecting ones. These layers make it easier to flash and flag the potential demolition candidates.
2. As you visit the different updates, it is handy to clock on neighboring buildings with the Pop-Up tool and use use the Oblique_Ln and Pictom_Ln references to field check the conditions. With the Oblique_Ln, you can sometimes get a more update view by using the orange pag-man to look at the location in street-view.
3. Import the latest Development Log points file. You can use the Utilities/**ImportDevelopmentLog** to make this easier. It preserves the full names of fields form the development log. Make an extruded poins layer out of this so that it is easy to see these markers when you have the buildings showing. I find that making them 125 feet tall works pretty well.

In this run around the buildings the only changes you should be making are to the **Merge_Instr** and **QA_Issue** and **Repair_Instr** attribute fields. These are all conveniently located at the end of the table, it is best to edit them by simply typing into the fields. Be sure to hit **Save Edits** when you feel confident about this.

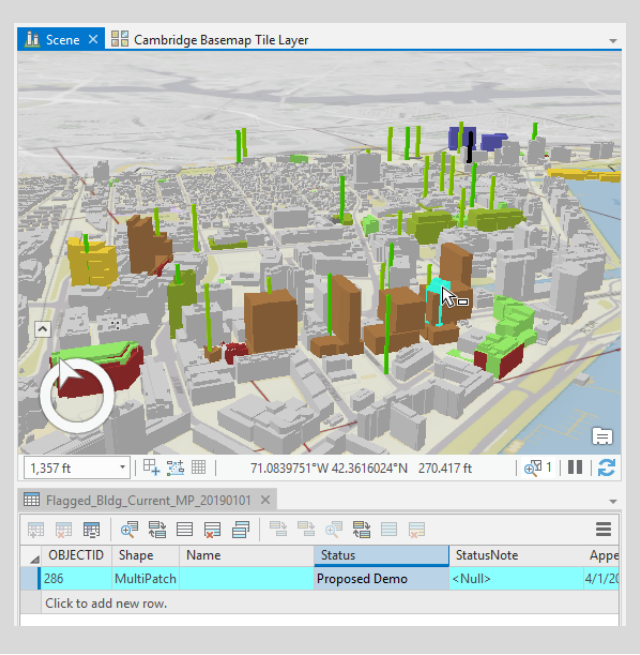


Figure 7: Flagging Buildings and Development Log Info

The **Merge_Instr** field should hold a simple reference to the model's new status or the feature class that a model should be sent to. For example, a model in the **Bldg_Current_MP** feature class that is proposed to be demolished, you should set the **Merge_Instr** to "**proposed demo**." If the model appears to be demolished, you can set the **Merge_Instr** to "**demolished**". In cases where a building model should end up in the **Bldg_Alt_MP**

feature class, you can set the **Merge_Instr** to "**alt**." Be consistent in these tags or you can clean them up later.

As you are marching around to each of the new buildings, set the **DL_Proj_ID** attribute for projects that are related to projects referenced in the Development Log. Three or four attributes will be transferred over from the development log later.

Make sure to scrutinize and adjust the location of each multipatch as you are looking at them. Use the edit tool to shift and rotate as necessary. If there are any issues that need attention, report them in the **Repair_Instr** and **QA_Issue** fields. For example, models in the from the **Bldg_Current_MP** feature class that are only partially covered by new buildings should be marked as "**split**" in the **Repair_Instr** field.

Improving the Fit of Buildings with the Terrain

Next, you may want to extend the bottoms of buildings below the terrain so that they are sure to be non-floating even on sloppy terrain models. For this, you can use the **5. BuildingExtender** model. This model uses the **Min_Z_Ft** value to create a building footprint and extrude it so that it meets the bottom of the building and extends 8 feet below the ground. It creates a multipatch from this and merges it with the initial multipatch layer. Note that you should be careful with this. Particularly if any of your models have donut holes or overhangs.

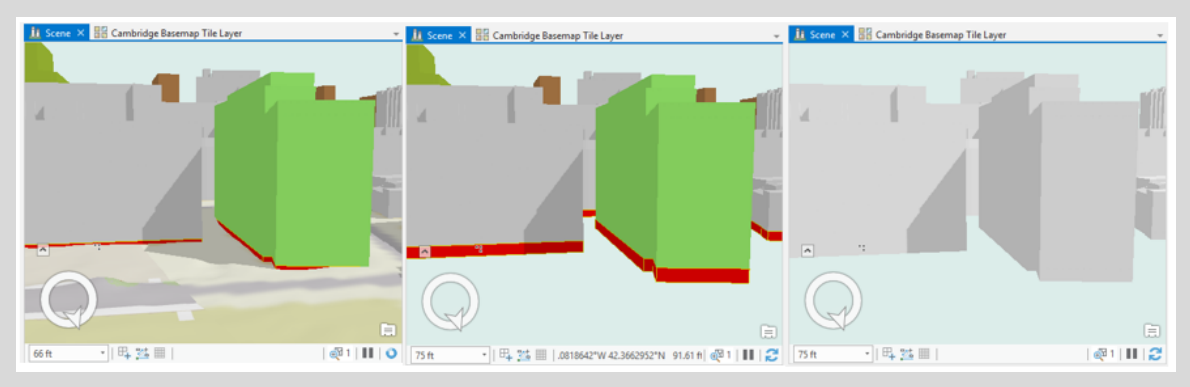


Figure 8: Fitting Buildings to the Terrain

True Story! When we ran the Extend Bottoms tool on the Bldg_Current_MP feature class, it worked, except for buildings that were part of the CCD_Pictometry_2015 batch! This made us feel good about having saved the **Mod_Batch** identifiers! For these building models, the basements were created, but the tops resisted the Union process in which the rest of the original building top models successfully fused together with their basements. This is why for tat batch of buildings, the basements are separate multipatches. A **QA_Flag** has been set for these.

Automatic Attribute Assignment

Now that you are through looking at each building and the potentially intersecting current buildings, you can go through a couple of procedures to automatically set useful attributes.

1. The model named, **3. PipelineAttributeUpdate** does a lot of spatial joins with the Cambridge BASEMAP_Buildings layer You should make sure to download the latest of these and be sure to fil in the variable that holds the date that this latest **Basemap_Buildings** layer was issued. Not when you downloaded it; but check the issue date on the Open data page. This model has a lot of parts. Be sure that you read all of the notes on the model itself – especially the one about changing the name of the output feature class at the very end.

The **3. PipelineAttributeUpdate** is a big pain in the butt. It should be re-designed after ESRI fixes a bug that makes many tools ignore the environment setting for "**Use UnqualifiedField Names**".

Add Essential Development Log Attributes to Proposed Projects

For any of the feature classes that include models that are associated with the Large Project Review process, you can use the model named, **4. Join Development Log** to update all of the fields associated with the Development Log. In this model you should also read all of the comments, and don't forget to download the newest Development Log table and update the **DL_Update** variable that reflects the issue date.

Preparing to Merge Updates

Using the **Merge_Instr** field to flag changes allows for a quicker Q/A and inspection process. We can do these quickly without too much worry because we know that we are going to check them carefully at this stage, and set all of the critical attributes like **Appear_Dt** and **Disapp_Dt** and **Status**, etc under much more careful and controlled conditions.

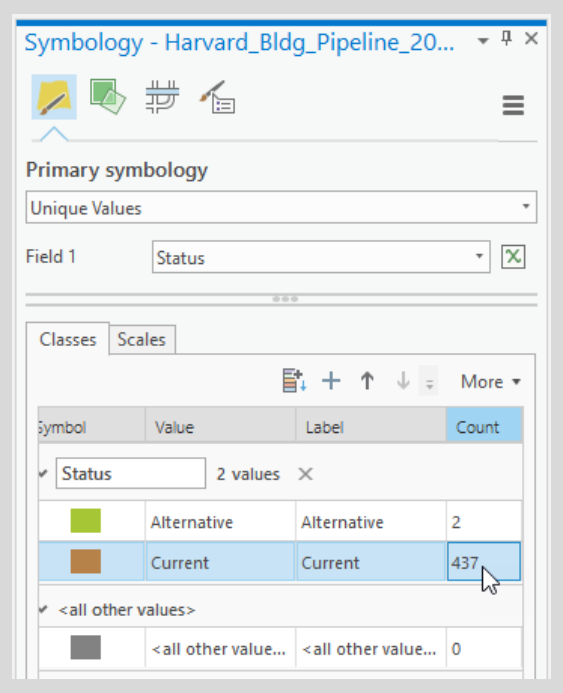


Figure 9: Check Consistency and Completeness of Status values

I recommend using WinZip to zip a back-up copy of your merge geodatabase before you begin this step.

We are now ready to start updating some of the building model attributes based on the Merge_Instr field. This will involve making selections based on the value of **Merge_instr** field and then using the Calculate Field Values tool to updates such attributres as **Status**, **Apperar_Dt**, Dissap_Dt, **Editor**, **Edit_Dt**, and so on. This is tricky and you have to be careful that the updates that you apply are affecting only the selected records, and not wiping out the attribute values for every record in the table. ArcGIS pro does weird things, and you have to watch it like a hawk and check each step before saving edits.

Checking Consistency

I recommend applying a categorical symbolization renderer to each of your merge layers, and using the Update Count feature in the symbology editor to make sure that all of your values for **Status** are

compatible with the geodatabase **Status_Code** domain (see the **Status_Domain** table in the Camb3d_Models.gdb.) And that none of the pipeline models have null for status at this point.

1. Now create a new Camb3dModels_YYYYMMDD.gdb in your ModelMgt/ModelMgt Data folder. This is going to hold all of the updated feature classes. We will call this **the Update Target Geodatabase**.
2. Take a look at the models named **7a. MergeAltBuildings** and its sisters to see how these work. For example, We start by creating a new Camb3d_Bldg_Alt_MP feature class in the update target geodatabase. Then we use a sequence of Selections and Appends to copy whatever features from the individual Merge feature classes into the new update target feature classes.

It should be noted that the select statements in these merge models are included so that you can check how many features are expected to be appended at each step. I write down the numbers of features from each layer and the cumulative total expected. Otherwise, if models are missed here, the error may not be discovered until later, when it will be really tricky to figure out where to recover them from. Again, ArcGIS pro sometimes gets this wrong, and restarting it may be the only way to get whatever problem cleared. So these merge models are designed so that if at any time you want to completely re-jigger the mix of the merge process, all you need to do is re-run the model.

Also note that the **Select by Attributes** statements in these merge models are strictly for your checking the counts. The selection expressions need to be set in the append tool, as well. So, make sure that they are the same – and check the count in the result!

Commit Updates

Congratulations! You are now ready to replace the initial copy of the Camb3dModels_YYYYMMDD.gdb in the master repositoty with your updates. Users and apps that reference the feature classes inside will have to update their references for these feature classes.

Looking Ahead to Future Features

Better Building IDs and Linking: Some of the complications in the new table schema have been made in order to facilitate future features of the Cambridge 3D model. Fist will be an improved building identification scheme which will allow us to associate building models with other building information from assessing (which includes accurate “Year Built” information and use data for each building.

Toward an open source Archival and Exchange Architecture: Currently the building management system depends on ESRI Geodatabases for model management and visualization. ESRI Geodatabases are great, but they do have a drawback in that they are not an open format. A robust archival and exchange system should have a means of reading and writing everything to open source formats. The same features that let us associate models with their source data will eventually (after fixing some python scripts) allow us to dump each building model out to an open source KMZ file. This will give us an archivally safe architecture, and also a means for users and staff to access individual building models in a format that can opened directly in SketchUp or viewed in Google Earth.

Appendix A: Camb3dModels Data Dictionary

Note to Cambridge GIS Reviewers: there are a lot of attributes in this table. Given my tendency to over-think things, this makes me a little nervous. I think that most of these attributes are worthwhile, but I welcome your feedback on these.

Most all of these attributes have their values calculated by three geoprocessing scripts that are run during the model ingest process.

All of the feature classes in Camb3D models have most of their attributes in common. The bridges feature class omits several attributes that are concerned with buildings and the CDD development log.

Field Name	Type	Description
Name	Text	Name of the building, building part, building phase, building aggregation or bridge that the model refers to. This field may default to a valid address for the building or may be left blank.
Status	Text: Coded Domain	A flag that is used to determine whether the model should be rendered in in a scene. Most values relate to a stage in the lifecycle: Proposed, Under Review, Permitting, Permitted, Under Construction, Construction Complete, Current, Proposed Demo, Approved Demo, Permitted Demo and Demolished. Values for this field are regulated by a controlled geodatabase domain, as listed below.
Staus_Note	Text	Optional elaboration on status. This field should be filled in when the status is equal to **Other**, or **Suppressed**.
Appear_Dt	Date	Reflects the date of the first appearance of the object. In most cases, we do not know when the object was built. This date reflects the first reported observation of the building.
Appear_Src	Date	Provides a reference to the source of the earliest known observation. This value could be a hyperlink.
Disapp_Dt	Date	The date of the earliest observation that the building, building part, building aggregation or bridge no longer exists.

Disapp_Src	Text	This text elaborates on how the Disappear date was established. This value could be a hyperlink or the title and date of an old map or the date of an observation made with Google Street View.
Model_ID	Globally Unique ID	A globally unique ID for the model represented by this record. This key is assigned to each skin model as it is entered into the system and if follows each record as the model moved demo Current to Demolished or other archival status. This ID makes it possible to share, edit and merge independent versions of the data-set, and to determine which features have been changed from one version of this feature-class to another.
Model_Src	Text	The person or firm responsible for creating the 3d model. This short string could be used as a short attribution for the model.
Bldg_Use	Text	Building Use: reflects the activities supported by the building part in question. Currently, the fines-grain use data that we have is provided at the Parcel level, see the **Par_Use** field.
Model Batch	Text	Models are integrated into the system in batches. A batch name is typically prepared by appending an abbreviated source name and date string (YYYMMDD). These batch names may be used to find the original source model in the Archive folder.
Model_File	Text	For models that originate as unique files -- such as hand-made Sketch-up files, this would be the name of the source model. For models that originate as geodatabase features, this would be the name of the parent geodatabase. This may also refer to the name of a KMZ file that has been exported from this feature-record.
Model_Src	Text	Model Source: A short string that could be used as a very concise citation for the model or foot-print. It will usually be the name of the person, firm, agency or department that produced the geometry of the building part. In a more elaborate scheme, this string could be used as a reference to a metadata document on the web or intranet.
Model_Dt	Date	Indicates the date that a model was made. Note that this is not always the date that is portrayed in the model. This date will be useful for understanding whether an alternate version of a model is newer or older than another.

Model_Note	Text	Any notes about the model that are not covered by more specific attributes. These are notes about the model - not the building.
Model_LOD	Text	Because building part models have been compiled from various sources, they vary in their level of detail (LOD). LOD 0 is a polygon. LOD1 is a simple extrusion. LOD2 is a straight-sided mesh with potentially non-horizontal roof details. LOD 2.5 models may have overhangs, (e.g. models created by the Urban Design Technology Group.) LOD3 includes models that portray building details such as doorways and windows. These Level of Detail designations are based on the CityGML standard. . It is important to be able to filter modes according to level of detail when doing automated procedures that may be affected by whether a model contains undercuts and overhangs. An illustrated key to levels of detail is provided appendix 2 of the the Model Management user guide.
Survey_Dt	Date	Reflects the date that the measurements were made for the model in question. For example, many of the models in this collection were surveyed in 2011 by Infotech. This includes many LOD1 models which were merely extruded roof-prints, and many LOD2 models which were created later by Cybercity3d frm the same survey data.
Survey_Src	Text	Enter a shore descriptor that identifies the individual, agency, department or firm that assumes responsibility for the survey information. In web-oriented applications this field may hold a reference to a web-based metadata record.
Centr_Lat	Double	Latitude for the internal centroid of the model footprint. Assumes the WGS84 earth model.
Centr_Lon	Double	Longitude for the internal centroid of the model footprint. Assumes the WGS84 earth model.
Z_Max_Ft	Double	The elevation of the highest vertex in the model expressed in feet above sea level.
Z_Min_Ft	Double	The elevation of the lowest vertex in the model expressed in feet above sea level. Note that it will often be the case that the lowest vertex in the model may fall substantially below the level of the ground.
Gnd_El_Ft	Double	Elevation of the ground for the lowest vertex in the building model. Calculated by projecting the building footprint onto a period-appropriate terrain model. This value

		is useful for discovering models that have floating corners. It will also be useful for padding the bottoms of the buildings should we decide to try that.
Height	Double	The calculated difference between **Gnd_El_Ft** and **Z_Max_Ft**.
Tile	Text	The tile name from the Cambridge GIS 3D model. This ID as assigned based on a spatial join with the centroid of the multipatch footprint.
Cambridge GIS Building Fields: The next three fields are populated from a spatial join (centroid-within) with the Cambridge GIS Building Polygons layer. Given the one-to-many situations, and missing IDs for many small buildings, these fields are not complete nor correct. However, they are useful for many applications, such as joining with the Building Energy Use database. It is expected that as the Building Identifier system is upgraded in the next several weeks, we will add more fields, including, hopefully, a **Bld_Address** field.		
Bld_ID	Text	Building ID: The Cambridge GIS Building ID. Assigned by a spatial join of the Cambridge GIS Building Footprint Layer. This join is not one-to-one, so don't expect this information to be perfect..
Bld_Type	Text	Building Type: From the Cambridge GIS Buildings feature class.. Assigned by a spatial join of the Cambridge GIS Building Footprint Layer. This join is not one-to-one, so don't expect this information to be spot-on.
Bld_Update	Integer	Building Update: reflects the issue date for the Cambridge GIS Buildings shape file that was used to update the values for **Bld_ID** and **BLD_Type**.
CDD Development Log fields: The Development log is posted quarterly to reflect properties and status changes to large projects. This information is useful for linking models of proposed projects to the web page for the project in the CDD's document management system bu the PB_ID. Project Use can be useful for thematically coloring models according to use.		
DL_Proj_ID	Text	Development Log Project ID. This ID links to the project ID in the Community Development Department's Development Log table.
DL_PB_Id	Text	Planning Board Special Permit ID: From the Cambridge Development Log. This code is useful for locating the documentation for each large project that is reviewed by the planning board.

DL_Name	Text	Development Log Project Name. If possible, it is good to differentiate between the individual buildings or project phases or proposal revisions.
DL_Status	Text	Cambridge Development Log Status: Reflects the Status of the project as of the date reflected in the Development Log table that was issued on the date reflected in the **DL_Update** field.
DL_Update	Text	Development Log Update: Reflects the issue date of the Development Log table that was used to update all attributes that begin with the DL prefix.
DL_Yr_Comp	Text	Populated from a join with eh Article 80 table. For buildings that have been subject to Article 80 review, this value should reflect the status at the time of the last update of Article 80 information.
DL_Use	Text	Development Log Primary Use.
Oblique_Ln	Text	Google Oblique ID: A hyperlink that should open Google Maps in Oblique view centered on the building.
Pictom_Ln	Text	Pitcometry Bookmark: A hyperlink that opens a Pictometry window centered on the building. Licensed only for internal use only. Password required.
PB_Link	Text	Planning Board Link: This URL should open the planning board's documentation page for buildings that have been through the large project review.
Editor	Text	The name of the person, firm department or agency that was responsible for the last edit.
Edit_Note	Text	A short string reflecting the nature of the latest edit operation. Your note should include the date. Append your new note to the old note so that a short audit-trail can be stored in the 254 characters of this field.
Merge_Flag	Text	Merge Flag: Used in the update process to create defined views of update or demolition candidates. See the Camb3D Model Management documentation for more details.

Merge_Instr	Text	Merge Instruction: This flag is used in the update process to mark candidates for promotion or demolition. See the Camb3D Model Management documentation for more details.
QA_Issue	Text	Quality Assurance Issue. Used in the update process.
Repair_Instr	Text	Repair Instruction: Used to flag buildings that need to be split or have their normals fixed, etc.

Appendix 2: Levels of Detail

This diagram from the Technical University at Delft,Netherlands elaborates on the City-GML concept of Levels of detail. Use this as a guide for figuring out what value to assign to the **Model_LOD** attribute. It is important to be able to filter modes according to level of detail when doing automated procedures that may be affected by whether a model contains undercuts and overhangs.