



Client : European Commission

Project : FRESH

Project N°: FP6-516059

Project Number: FP6-516059

Document number: 3.2.3

Document Title: Algorithms and UML diagrams to convert CATIA files into PIVOT

Document status: Confidential



The main algorithms developed for converting the information from CATIA files into PIVOT structures focus on the process of extracting electrical information from the mock-up. This report describes first the algorithms used for analyzing the virtual mock-up and, then, the methods for converting the information obtained during the analysis into PIVOT files and for reading these files and converting them back to PIVOT structures.

1.- Stages of the analysis

The analysis of the mock-up obtains electrical information from the geometrical description of the mock-up. The whole analysis process is divided in four steps:

- The first one is the spatial partition. This method consists in the division of the scene in a mesh of volumetric units and the assignation of the triangles that constitute the elements to each one of the volumes that contain them. This stage prepares the structures necessary for the rest of the algorithms. Regarding the volumetric units, the initial option was to use a mesh of regular cubic units, the voxels, but this technique was later combined with an octree structure in order to achieve a greater precision of the analysis in less time. This combination of octree and voxels is what gives the name of "hybrid analysis" to the whole algorithm.
- The following step is the connections analysis. This algorithm stores the voxels that contain triangles of more than one part in one list for each couple of parts and then, taking the center points of the voxels boxes as references, calculates the medium point that would correspond to the connection between the two parts. This process is quite fast and precise, but it may become ineffective if the mock-up is very complex and its configuration very dense, because the size of the voxels is then bigger than the separation between the components and the methods extracts additional false connections. In order to correct this, a complementary function was added to this method. This function analyzes the triangles contained in the candidate voxels and determines if they truly intersect. This way the ambiguities are avoided and the false connections disappear.
- Once the connections among the parts have been obtained, it is time to calculate the axis of the bundle segments or paths. The bundle segments may have more than two connections and more than one path (one for each couple of connections). These paths represent the wires that may cross the bundle segment. The extraction of these paths is performed by three methods. The first one classifies the voxels that contain the volume of the analyzed part in different layers. The second method then traces the path between the voxels that contain the two connection points using a modified version of the A* algorithm. Finally, a third method checks that the extracted path crosses through the center voxels of the bundle segment section and adjust it if necessary.
- The union of several paths that put in contact two electrical devices is what we call a wire. The purpose of the route analysis is just to calculate these collections of paths. This is a combinatorial problem because sometimes the bundles have branches that multiply the number of possible routes between two different devices. The algorithm may choose automatically the shortest of these routes as the correct one, but it has been decided that it is preferable to obtain all of the routes and let the user choose which one is the best option.

2.- Mock-up analysis algorithms

Two algorithms have been developed in order to analyze the geometrical information of the mock-up. The first one, the standard hybrid analysis, corresponds to the algorithm explained above and applies the spatial partition and the other steps of the analysis to the whole mock-up at once. The second version, the individual hybrid analysis, performs the division of each element and, in consequence, the other steps of the algorithm part by part.

2.1.- Standard hybrid analysis

The standard hybrid analysis is able of analyzing all types of mock-ups: small, medium, complex, with dense or sparse configurations. In general, the method is fast and gives excellent results when calculating the paths and connecting points of all of them. The only difficulties appear when analyzing complex mock-ups with a dense configuration, because a higher level of detail is required and that means an important increase of the runtime cost.

The functions and variables used not only for the spatial partition but also for the rest of the steps of the analysis are contained in the class HybridStructure. The partition technique that has been used builds first an octree structure of six levels, classifies the leaf nodes in interior, exterior and border octants and then performs a voxelization in those of the interior and the border. The result is a 3D mesh of voxels that contains the volume of the components of the mock-up. The triangles that constitute the elements of the mock-up are assigned during the spatial partition, first, to the octants that contain them and, then, to the voxels in which the leaf nodes are divided.

The leaf nodes that are partially full, this is, the border octants, are stored in PartMap, a structure of HybridStructure. This way, we will know which region of the space is occupied by the mock-up and, therefore, is susceptible to be analyzed. Similarly, the interior leaf nodes are saved in PartInterior, also a structure of HybridStructure.

HybridAnalysis is the root function connected with the interface module that will call all the other functions. It does not only call the methods that perform the spatial partition but it also calls those ones for calculating the connections points and the axis of the bundle segments. The functions for obtaining the wire routing from the paths of the bundle segments are not included because, as they may use information contained in the complementary XML or XLS files, they are called by different interface features. From HybridAnalysis, the first function to be called is BuildNewHybridTree, which is the function responsible of performing the spatial partition of the 3D scene. HybridAnalysis prepares the necessary variables and then calls IncreaseHybridLevel that will build the different levels of the octree structure. During this process, the leaf nodes that are partially full are stored in the PartMap structure.

Once this process is finished, BuildNewHybridTree calls LookForInteriorHybrids in order to classify the octants in exterior, interior and border. This is necessary for knowing which leaf nodes need to be voxelized.

With the interior and border leaf nodes separated from the exterior ones, BuildNewHybridTree will call the Voxelize function for voxelizing them. Finally, the last two functions called by this method are ObtainNeighbourVoxels, which will assign to each voxel a reference to their neighbour voxels and LookForInteriorVoxels, which will classify the voxels of the border leaf nodes in exterior, interior and border. Both operations are very useful for tracing the paths of the bundle segments, the first one because it helps to check the transitions between the voxels and the second one because it helps to classify the voxels that contain a part in different layers depending on how interior they are.

After all these operations, the structures necessary for the rest of the steps of the analysis are already prepared and it is possible to begin extracting information about the topology and the routes of the mock-up.



2.2.- Individual hybrid analysis

The main reason for developing the individual hybrid analysis is that it offers a better accuracy in the critical sections of the mock-up. This analysis is a variation of the hybrid analysis that ~~takes~~ considers separately the space region containing each part. The extraction of connections and paths is performed by the same functions used in the hybrid analysis, but the size of the voxels is smaller and the final precision is, therefore, much better even when the specified number of voxels per leaf node is smaller. However, as it requires performing all the analysis steps for each part selected and as it analyzes multiple times the same space region when the parts are superposed, the drawback is that it is much slower.

A secondary benefit of this method is that, as the connections and paths analysis are applied individually for each part, the mesh of octree and voxels and other intermediate structures may be deleted before starting the analysis of the next part and this reduces the amount of memory needed. This is especially useful when working with large, complex mock-ups.

The functions associated to this analysis are quite similar to those ones of the standard hybrid analysis.

3.- Connections analysis

The connections analysis is the algorithm used for obtaining the connection points between the elements of the mock-up. It basically consists in checking which voxels contain triangles of more than one part. When one voxel satisfies that condition it is stored in a list containing all the intersecting voxels between these two parts. Once all the voxels have been examined, the center points of these voxels are summed up and divided by the total number of voxels in order to extract the medium point. This point is an approximation quite close to the connecting point between the two parts and, therefore, it is saved in the corresponding lists of each part.

This algorithm has a weakness and a limitation. The weakness refers to those cases in which the distance between two separate parts is smaller to the size of the voxels. An option for solving this problem consists in performing a second test that checks if at least a couple of triangles of each part intersect.

As regards the limitation, as all the voxels are stored in the same list, the algorithm may fail if two parts have two or more connection points between them, because the method would consider that only one exists.

Although the simulation process developed by WP5 only needs the connections between a bundle and an electrical device or between two electrical devices, in this step the case of connections between electrical bundles is also taken into account because these data are needed in the paths analysis.

The function CalculateContacts is the method that checks which voxels contain triangles of more than one part and stores them in separate lists for each couple of elements. In order to solve the detection of false connection points, the user may select an option that activates the function CheckTrianglesCollisions in CalculateContacts. This function adds a second test of the triangles contained in the candidate voxels that checks if they truly intersect.



When all the voxels have been checked HybridAnalysis calls the function CalculateConnectionPoints, which will use the center points of the voxels stored in the different lists for calculating the medium points. This point is a very precise approximation of the connecting point between the parts and, thus, the function will store it in the connections list of each part.

4.- Paths analysis

The paths analysis is the method used for obtaining the axis of the wires. It may analyse any type of element of the mock-up, but it only makes sense to use it for analysing the bundle segments. One thing that must be clear is that the paths are not the same as the wires. The paths are simply routes inside each part that put in contact two of the part connection points. Additionally, the paths may never cross voxels that are exterior to the part they belong to. The wires are always composed by one or more paths.

The algorithm is divided in two steps. In the first one, the internal and frontier voxels of the part are assigned a value depending on the interior level they occupy. Once the voxels have been classified, the connecting points of the part are studied in pairs. Taking the voxels that contain the parts and using the A* algorithm a route is traced through the most interior voxels of the part. The following heuristic rule is used for determining which is the correct path: The transitions between a voxel of a more exterior level to another of a more interior level have a lower transition value than the transitions between voxels of the same level and these others have also a much lower value than the transitions to a voxel of a more exterior level. This favours the routes through the center of the bundle segments which would correspond to the axis of the wire.

This technique, however, may present some imperfections in the critical sections: The extreme curvature of the bundle segment may cause that the algorithm prefers more external voxels despite their higher transition cost, just because the path is shorter. An optional checking function was added to the algorithm. This function takes the voxels used for the current path and analyzes for each one of them the column and row, of the correspondent bundle segment section, in which they are contained. In these arrays of voxels, it selects the ones in the extremes and extracts the points and intersections with the box of the triangles contained in them. With these data, it calculates four medium points corresponding to left, right, up and down. If the bundle segments are circular, which is the general case, it does not matter if the coordinates of these points coincide with the extremes of the section. The intermediate coordinates of left-right and up-down points will give us respectively the horizontal and vertical coordinates of the center point of the section, what will allow us to adjust the path to the axis of the bundle segment.

Finally, before adding the new path to the part its route is smoothed. As the path has been traced using rectilinear segments, it is possible that sometimes the aspect of the path may not fit well with the shape of the part. The final result is not a curve but a polyline with shorter segments. However, high values of the smooth parameter produce approximations very close to a curve.

The paths analysis is the last step of the hybrid analysis. It is performed after the spatial partition and the extraction of the connection points between the different elements of the mock-ups.



FindExtractionPath is the function that really contains the methods for calculating the paths. It focuses on a concrete path between two connection points. First, it calls AlternativeMark that will examine the border and interior voxels of the mesh and group them in different layers depending on the interior level in which they are. FindExtractionPath then explores the interior and frontier voxels seeking the route with the lowest transition costs.

Finally, two functions are involved in the process of storing the extracted path. The first one is AddPath that is called by FindExtractionPath for saving the data adequately and the second one is SmoothAxis that is called by AddPath for smoothing the routing of the path. The level of smoothing is determined by the value specified in the Smooth box of the interface window.

5.-Route analysis

Most mock-ups are designed in such a way that a wire must cross several bundle segments before putting in contact two electrical devices. The route analysis was developed: for calculating all the possible routes between two electrical devices, even those ones that cross several bundle segments.

These routes define the path that the wires of the mock-ups follow and, as wires may cross more than one part, what the routes structure stores is the list of bundle segments that the wire cross, the total length of the wire and the identifiers of the connecting pins of the electrical devices.

In a first step, one of the devices is chosen and a new route is created for each bundle segment connected to it. Then, the bundle segments that are connected to the last element, and that are not yet included in the route, are added to each one of these routes. The process is then repeated for each route until the other electrical device is found.

This analysis completes the information obtained with the standard and individual hybrid analysis, giving a more precise and realistic description of the wires. It is executed separately once the connections and paths of the elements of the mock-up have been extracted.

RouteAnalysis is the function called by the interface when the user wants to calculate the routes. However, what this function makes is simply to give the possibility of working in a parallel thread while the user performs definition tasks and calls GetWiresFromPaths, which is the function that really prepares the variables and calls the other functions.

The function GetRouteForCouple is called in order to extract all the possible routes for each couple of electrical devices. Three are the functions necessary in this process.

- The first one, GetAllRoutes, analyzes all the bundle segments connected directly or indirectly with the first electrical device and calculates the maximum number of possible routes between both electrical devices.

GetAllRoutes is called for each connection of the first electrical device.

- The second function called by GetRouteForCouple is ShortenRoute. When using GetAllRoutes for extracting all the routes, it is possible that the algorithm include loops or unnecessary elements that make the route longer than it should be. ShortenRoute checks each route looking for intermediate elements that may be skipped and removes them.
- Finally, CheckRepeatedRoutes is called for completing the process. A problem of ShortenRoute is that it may produce routes that already existed, this is, the shortest versions of some routes may overlap other routes that had been already obtained. CheckRepeatedRoutes checks all the routes and when it finds two with the same list of bundle segments it eliminates one of them.



At this point all the possible routes for the chosen couple of electrical devices have been calculated. The function CalculateRouteLength is called then for obtaining the length of each one of them. Finally, the function AddRouteStruct is called for saving the routes in the class RootScene and for changing any associated XML or XLS files.

6.- Generation of PIVOT files

The generation of a PIVOT file is the process with which the information contained in the structures product of the analysis of the mock-up is, first, converted into PIVOT-like structures and, then, stored in a file.

This algorithm is much simpler than the previous ones, because it only has to put together all the data and save them in a file following the structure defined by the PIVOT DTD.

GeneratePIVOT121Click is the function with which starts the algorithm. It is a member of TForm1 and it is called directly by the interface. This function prepares the necessary variables and calls the functions responsible of the different steps of the conversion process.

We may divide these functions in two groups: those which collect the data from the result structures of the analysis and build the Model structure and those which, once the Model structure is complete, pass the content to the PIVOT file.

In the first group we may include AddRoutesData and AddPIVOTModel. The first one builds the PIVOT structures that contain the information associated to the bundle segments and wires while the second one builds the structures that define the electrical behaviour of the electrical devices. These methods also call the functions AddWireGeometry and AddElementGeometry that, respectively, will attach the structures containing the geometry.

In the second group, the two main functions are WriteModelInfo and AddXLSInfo. WriteModelInfo writes the Model structure that represents the mock-up into the PIVOT file. Finally, if a XLS file has been used for adding information of the mock-up, the function AddXLSInfo opens it and stores in the appropriate cells the lengths obtained for the wires

All the previous functions are members of TForm1.

7.- Reading of PIVOT files

This process consists in reading the information contained in a PIVOT file from a previous analysis and converting it into the structures used by the application. These data include the geometrical description of the components of the mock-up, their associated attributes, the definition of their types, the connections between them and the routing of the wires. This operation allows us to visualize the results that we had obtained and apply little modifications, complete the analysis of the mock-up if it had not been finished, or perform again the whole analysis if necessary.



The algorithm may be divided in two parts. In the first one, the information of the files is read in order to build the correspondent PIVOT structures, while in the second one, the PIVOT structures are converted to the structures that the application needs for showing us the mock-up and performing the analysis.

The first of these functions is ReadModel. This method builds the Model object that represents the mock-up using the information contained in the XML structure. Two functions are called by ReadModel for complementing it: AddGeometry and ReadModelContent. When an element of the mock-up is being read AddGeometry converts the associated geometrical description to the appropriate structures of the Scene module that will allow us to visualize the physical objects in the application window. ReadModelContent will read the attributes and characteristics of the element. There is a list of attributes that ReadModelContent expect to find for each element. If new attributes are going to be added we will have whether to change this function or to add them as generic properties. All these three functions are defined in the PIVOT module.