Creating an HVE™ Vehicle Geometry from Orthographic Diagrams

James P. Sneddon
Baker Sneddon Consulting

Copyright © 2004 Engineering Dynamics Corporation

Abstract

The vehicle geometry file provides a graphic visualization of a vehicle and damage profile in HVE™ simulations. Many users prefer specific vehicle geometries over generic for use in video exhibits. An increased need for specific vehicle geometries occurred with the release of the DYMESH® option in SIMON®. Unlike EDSMAC4™, the vehicle geometry is an integral part of the collision simulation in DYMESH.

Currently, there are over 150 vehicle geometry files included with the HVE system software, and additional models are added with each release. Most users will periodically require a vehicle that is not included in the database. If a generic vehicle geometry will not suffice, the user can order the vehicle from EDC, purchase the vehicle geometry from a third party, or build the vehicle geometry file using 3D modeling software.

Building a vehicle geometry requires three dimensional point data. This data is best acquired by using a digitizing arm, survey instrument or photogrammetry. However, some vehicle configurations can be built from orthographic diagrams. This whitepaper will provide an introduction to this method.

Introduction

Building vehicle geometries is a process of varying difficulty. Vehicles with planar or flat surfaces, or with curved surfaces of uniform radii can easily be built. A semi-trailer is a prime example of this type of vehicle. Often these types of vehicles can be built from a few measurements, or diagrams. Figure 1 depicts a trailer built from field measurements. The van body was created very quickly, with the greatest amount of time spent on the underbody details.

The free form surfaces common to passenger vehicles greatly increase the difficulty of modeling a vehicle. Figure 2 is a geometry file of a Ford Mustang Convertible downloaded from an online 3D model exchange site.

The downloaded file was modified and converted to the .h3d format, and imported into

Figure 1. *The planar surfaces of this trailer make it one of the easiest vehicles to build.*

Figure 2. *The intricate details of the free form surfaces of this Mustang are nearly impossible to build from an orthographic diagram of the vehicle.*

the vehicle editor. The subtle contours of the hood and fenders and intricate details of the air scoop are nearly impossible to build from the typical top, front, back and side orthographic views. These vehicles are best modeled from digitized point data.

Between these extremes there are many vehicles that can be modeled from orthographic diagrams with reasonable effort. Examples of these vehicle types include buses, construction equipment, panel trucks and trailers. This whitepaper will outline the procedure used to build the vehicle geometry file for the TMC RTS 4400 Transit Bus depicted in Figure 3.

The geometry file was built using Rhinoceros® (a.k.a. Rhino); a three dimensional NURBS modeling program. Rhino's NURBS (Non-Uniform Rational B-Spline) geometry provides excellent flexibility in creating and editing free-form shapes and surfaces.

Figure 3. *This Chicago Transit Authority TMC RTS 4400 Transit Bus was built from the orthographic diagram in Figure 4.*

Orthographic Diagrams

Orthographic diagrams of the vehicle can often be obtained from the manufacturer or dealer. The owner of a fleet vehicle, such as the transit bus, may have these diagrams on file. The diagram in Figure 4 was used to build the bus geometry.

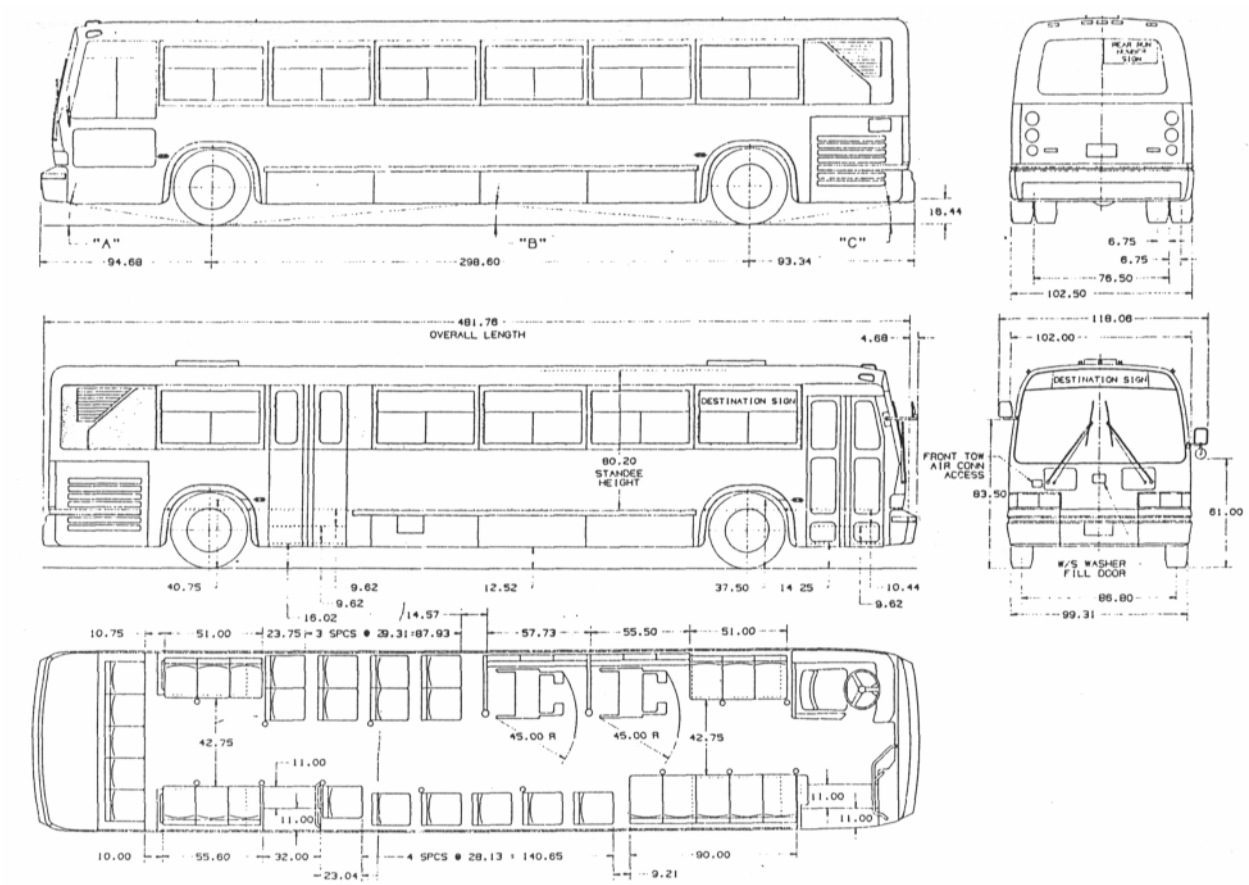

Figure 4. *The TMC RTX 4400 Transit Bus was built from these five orthographic views.*

A minimum of four views (top, front, rear and side) of the vehicle are required. Since the sides of the transit bus are not identical, a fifth view is required.

The diagram was scanned and saved as a bitmap. Using the *BackgroundBitmap* command, the image was imported as a background in Rhino. The size was specified such that the image was approximately to scale.

The background image was used to create the curves from which surfaces were constructed. For this vehicle, all curves were constructed in the top view, and rotated as needed. Alternatively, each view can be scanned and placed separately in the corresponding viewport.

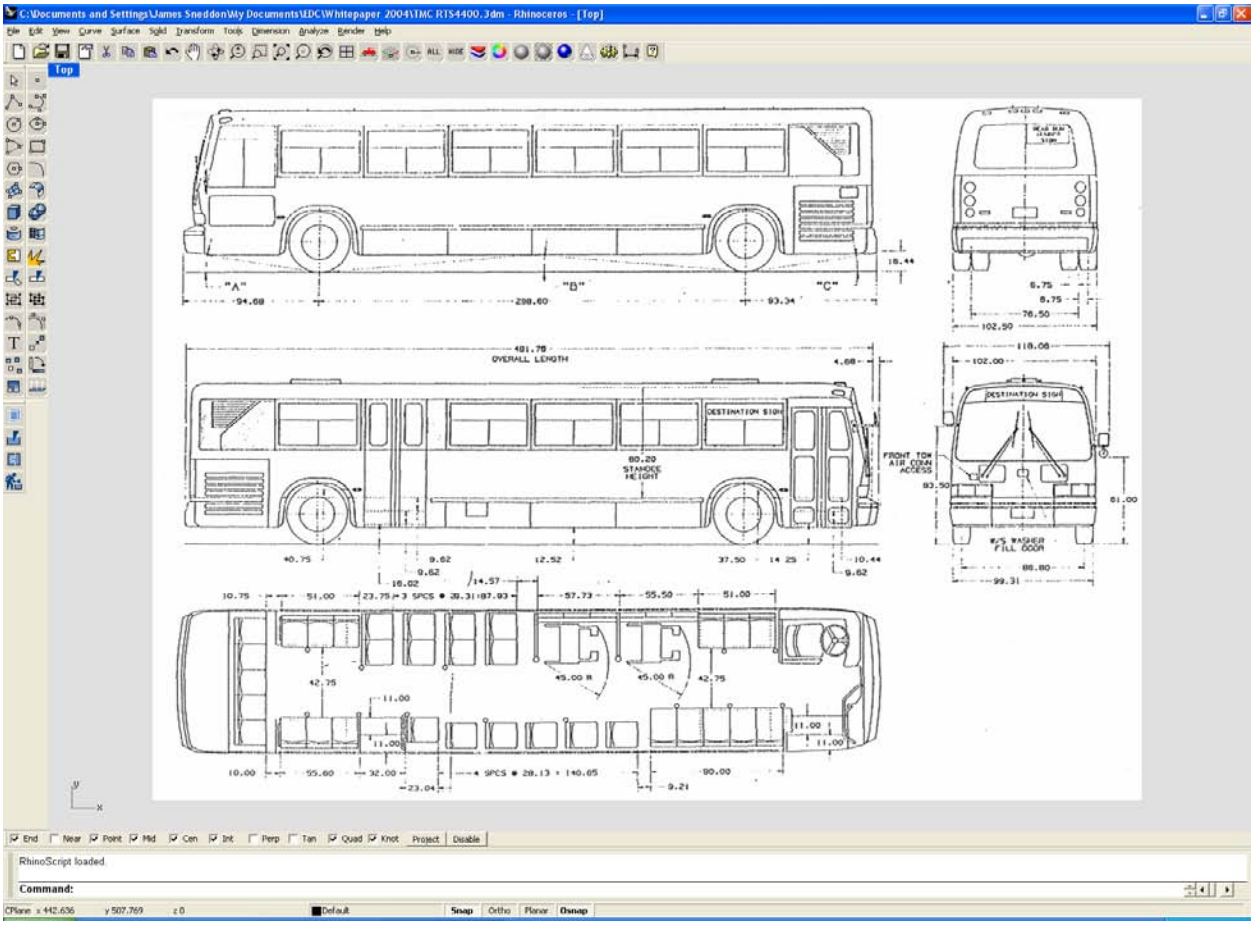

Figure 5. *The background image placed in the top viewport.*

Symmetry

Most vehicles are bilaterally symmetrical relative to the *xz* plane. The transit bus is symmetrical with the exception of the curb side doors, and driver's side window.

Symmetry reduces the number of points required to build the geometry. Points and curves used to build the body are mirrored about the centerline. Care must be used to ensure that curved surfaces are not only symmetrical, but tangent across the centerline.

A centerline was established in the end and top views of the bus. In the front view, a polyline was drawn along the roof of the bus, ending at the centerline. Using the *Mirror* command, the polyline was mirrored about the centerline. An *interpolated curve* was then drawn through the vertices of the polyline. This will ensure that the roof line is both tangent and symmetrical about the centerline (Figure 6).

Wire Frame

Curves constructed from the orthographic views are used to build a wire frame, or skeleton of the vehicle. From this wire frame, NURBS surfaces can be constructed to form the vehicle exterior.

In the front view, a curve along the right side was built as previously described. Use as few vertices or points as needed to match the curve. If too many vertices are used, the curve will appear irregular. If this occurs, the curve can be redrawn with fewer vertices, or large variations in curvature can be removed with the *Fair* command.

The small radius curve between the roof and sides was not built with an interpolated curve. Instead, the right side was mirrored about the centerline, and the *Fillet* command was used to build the connecting curves. This will ensure that the curve is tangent to the top and sides.

The perimeter curves were built in the top view, ensuring that symmetry and tangency were maintained about the centerline as before. The side profile view was created in the right orthographic view. Fillets were used to connect the roof line to the front and rear curves in the side profile view.

The height of the side profile and the width of the perimeter must match the profile curve drawn in the front view. Figure 6 shows the front, right side and top profile curves in red. The centerline in the top and front views is in blue. The bumper curves were not included as they will be added later.

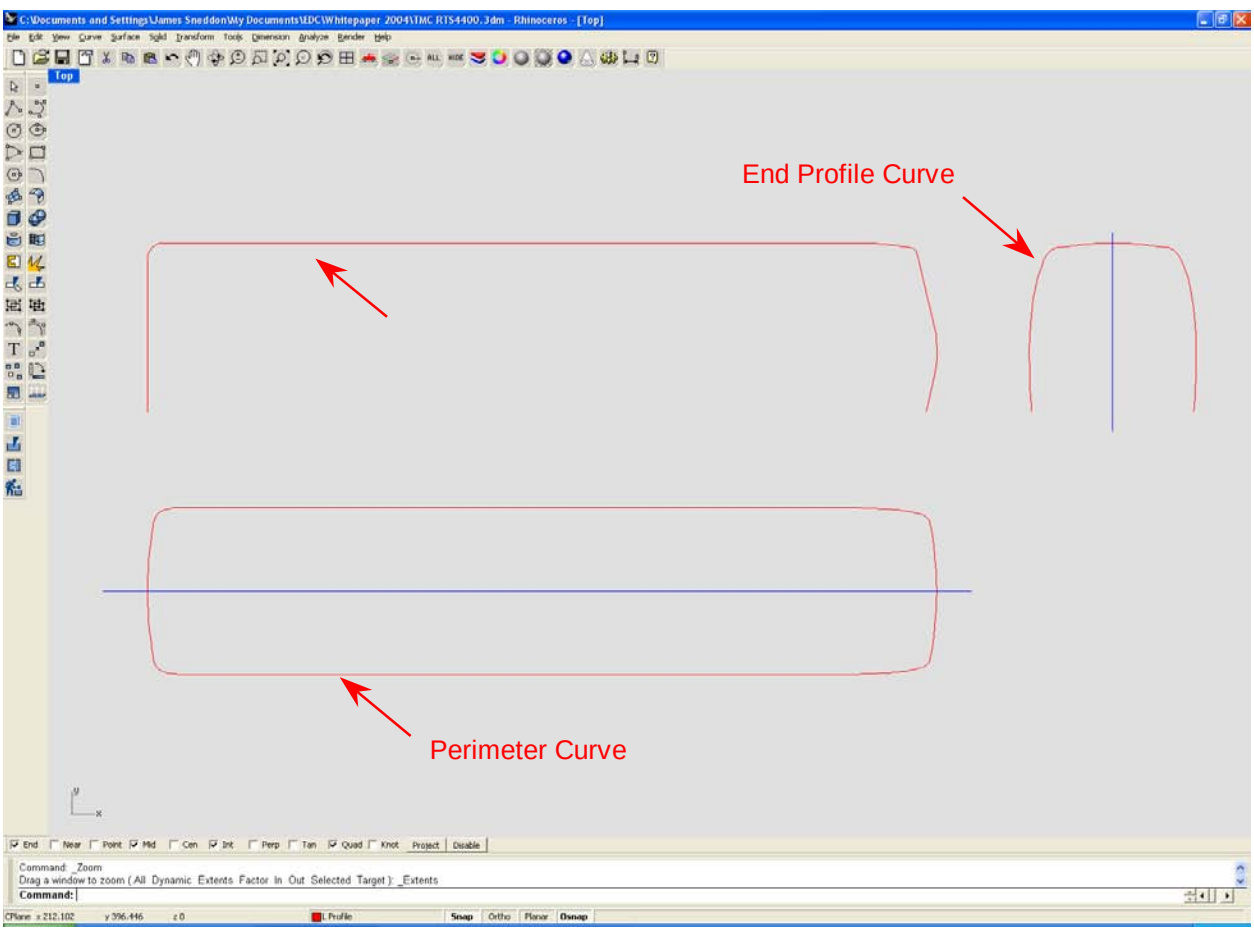

Figure 6. *The curves in red establish the front, side and top profile of the transit bus. The blue centerline in the top and front views is used to maintain symmetry about the xz plane.*

The end and side profile curves were rotated so that they are properly oriented to the perimeter curve in the top view. The end profile curve was moved so that the widest point was aligned vertically with the perimeter curve. It was positioned in the center longitudinally. Then the side profile curve was moved so that it was aligned with both the perimeter curve and end profile curve. The intersection points of all curves must be coincident. Figure 7 shows these curves properly rotated and aligned.

The perimeter curve tapers at each end of the bus. A copy of the front profile curve was placed at the start of the tapered section at each end. The perimeter curve was copied to the bottom of the profile curves, and a second copy placed approximately midway up the side of the bus. These curves were adjusted so that the intersection points with the end and side profile curves were coincident. The completed wire frame is shown in Figure 8.

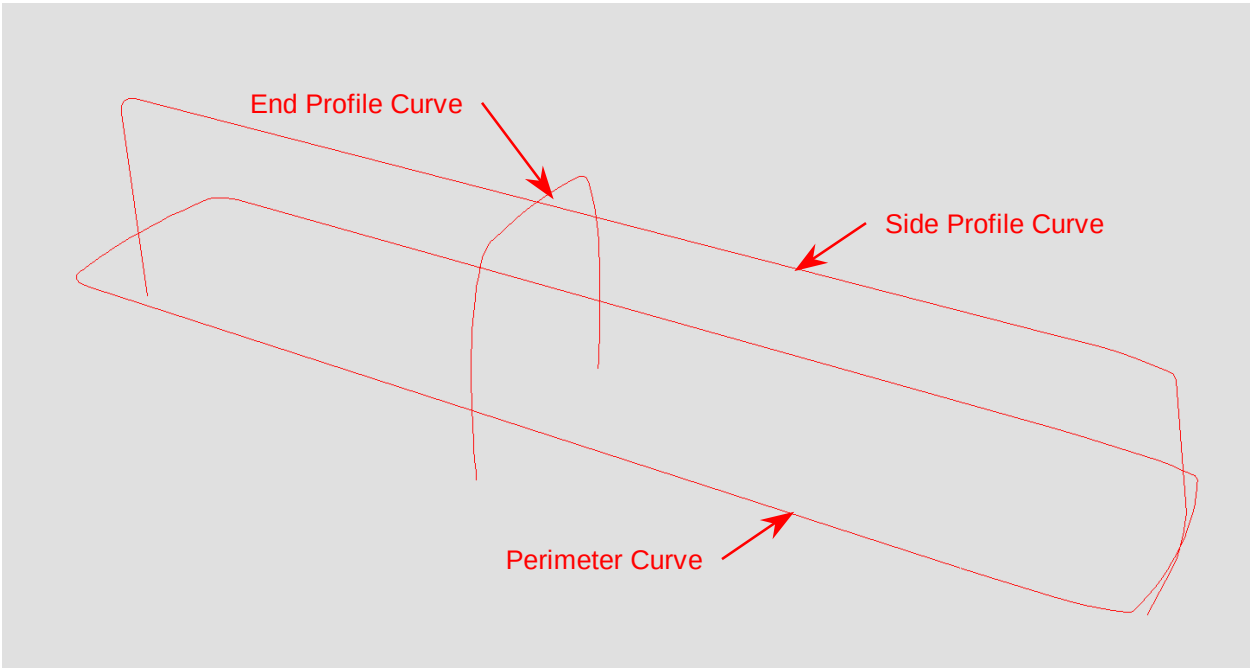

Figure 7. *The front and right side profile curves have been rotated and moved relative to the top perimeter curve. The points of intersection must be coincident.*

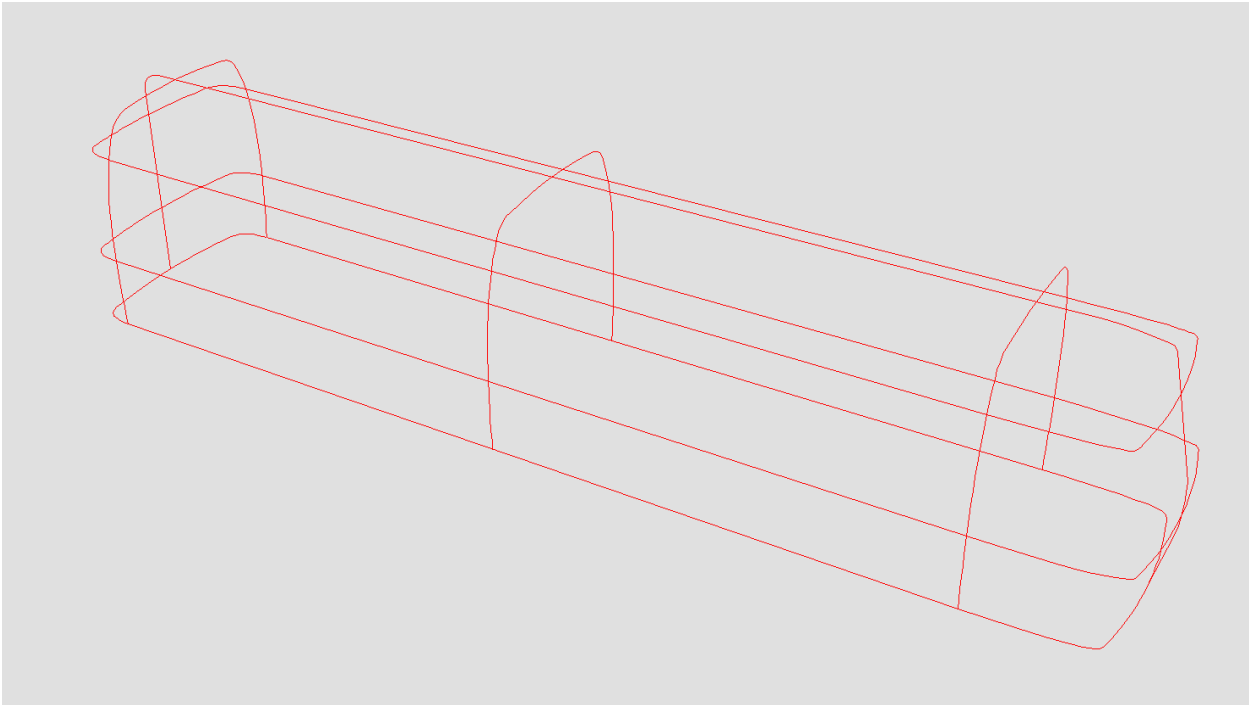

Figure 8. *Copies of the end profile and perimeter curves are added to complete the wire frame. The intersection points between the curves must be coincident.*

Surfaces

Numerous surface tools are available in Rhinoceros, which enables the same surface to be constructed by several means. The methods described herein were used by the author to build the transit bus. They were selected for accuracy and ease of use, but are not the sole means by which a vehicle geometry file can be created. In some instances, an identical surface can be built with a different surface tool.

NURBS surfaces were constructed from the completed wire frame to form the bus body. Specific details, such as windows and doors, were split from the initial surfaces. The final product is a polygon mesh created from the joined NURBS surfaces.

The bus body is built in separate sections and joined later. The first section is constructed from the end profile curves. The *Join* command is used to join the individual entities together in each of the end profile curves.

A NURBS surface is constructed with the *Loft* command. The surface is lofted through the three end profile curves. This surface, shown in Figure 9, forms the sides and roof of the bus. The bottom will be built separately.

The ends of the bus were built with a curve network. The perimeter and side profile curves were joined. Since curves in the same direction of a curve network cannot cross, the end profile curve must be split at the intersection with the side profile curve. The ends are built separately from each curve network with the *NetworkSrf* command. The curve network used to build the front end is shown in Figure 10, and the constructed surface in Figure 11. The rear of the bus is constructed in the same manner.

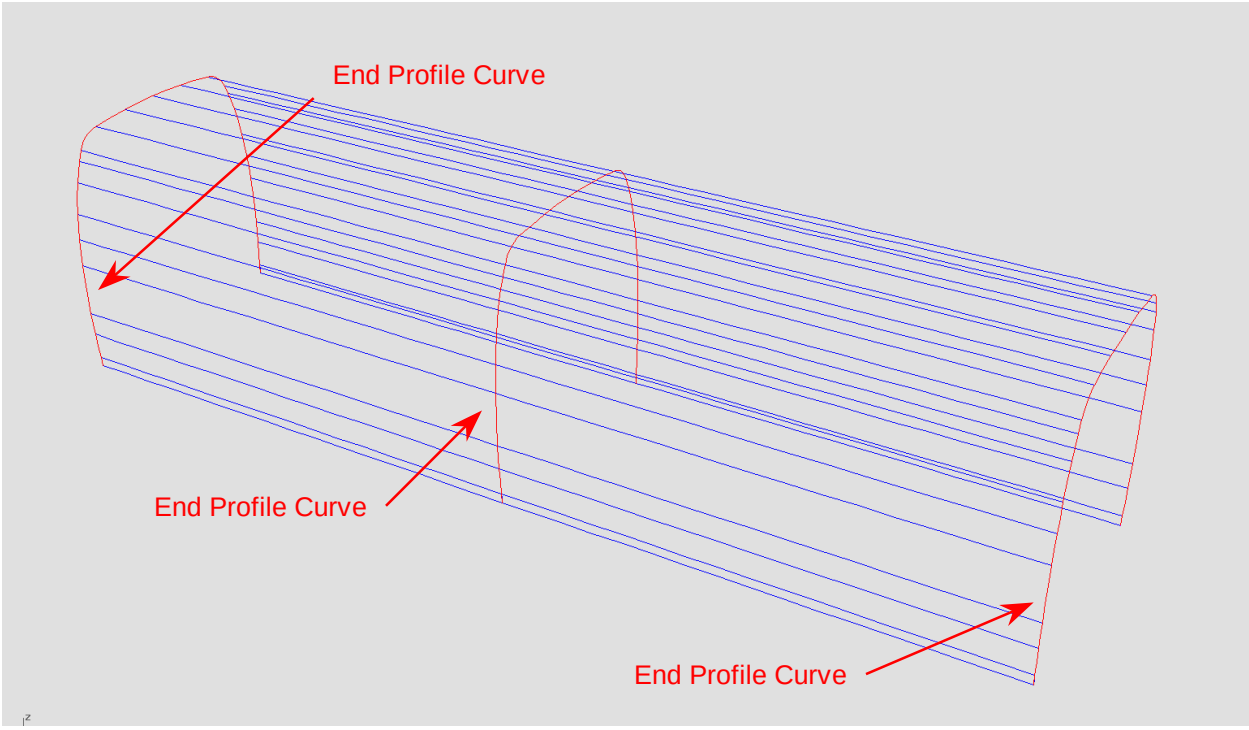

Figure 9. *The blue surface was lofted through the red end profile curves. This forms the sides and roof of the bus.*

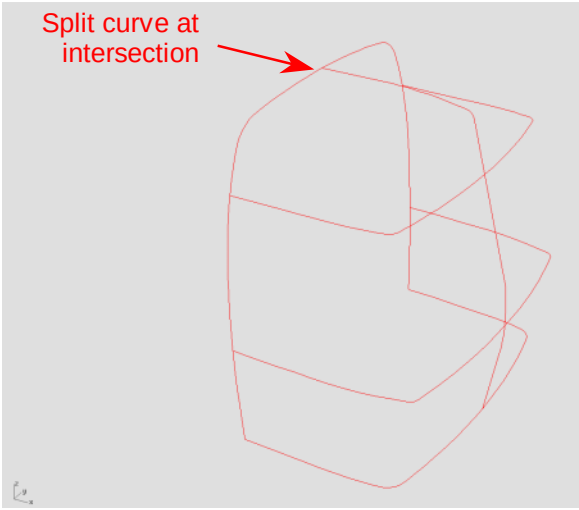

Figure 10. *The front end of the bus was constructed from the network of curves illustrated here. The end profile curve must be split at the intersection point with the side profile curve.*

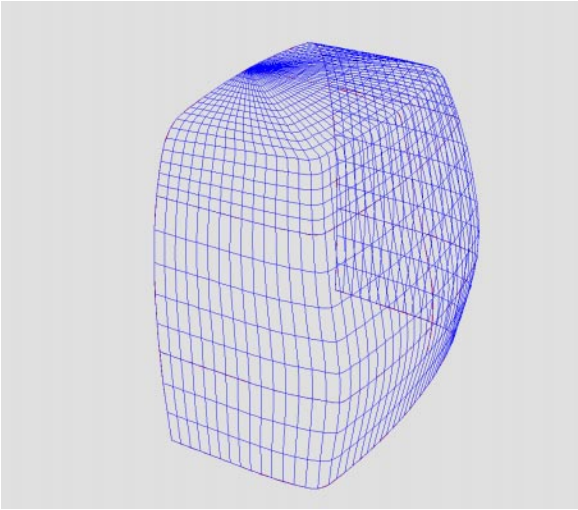

Figure 11. *The NetworkSrf command was used to construct the surface forming the front of the bus from the curve network shown in Figure 10.*

Surface Normals

Much like HVE environment files, surface normals apply to the vehicle geometry too. It is important to properly orient the surface normals. Particularly with DYMESH events where reversed normals may result in ignored interaction between colliding surfaces. The surface normals of a vehicle geometry must be pointed toward the outside.

After creating the exterior surfaces, the *Dir* command was used to check the surface normal orientation. Reversed normals were corrected with the *FlipNormal* option in the *Dir* command. Figure 12 shows the surface normal display.

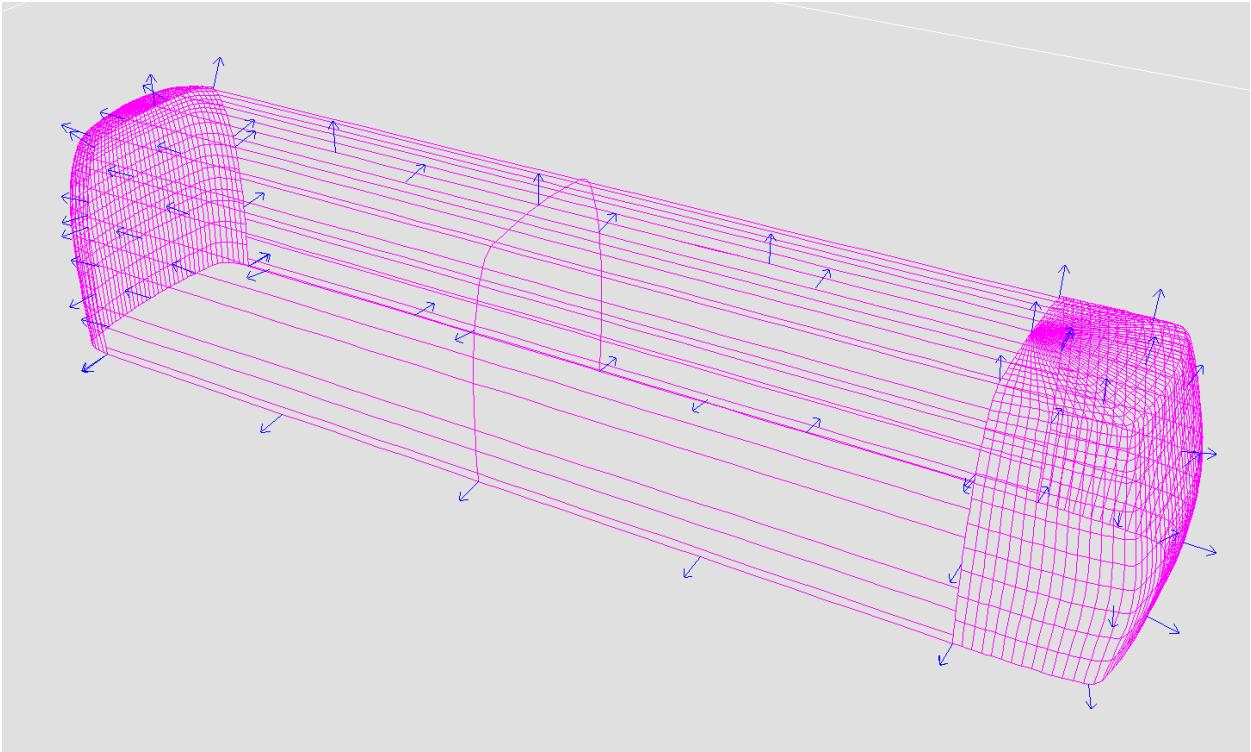

Figure 12. *The Dir command can be used to check for proper surface normal orientation. The surface normal should point toward the outside of the vehicle.*

Details

Vehicle details can be split from the initial surfaces forming the exterior body. The NURBS geometry permits a surface to be split using curves or surfaces as cutting edges. Using the orthographic diagrams, curves were drawn along the edges of the windows, doors, wheel openings and engine grills. Additional lines were drawn to create the color scheme used by the Chicago Transit Authority. Figure 13 shows the lines and curves used to split the details from the left and right sides of the bus.

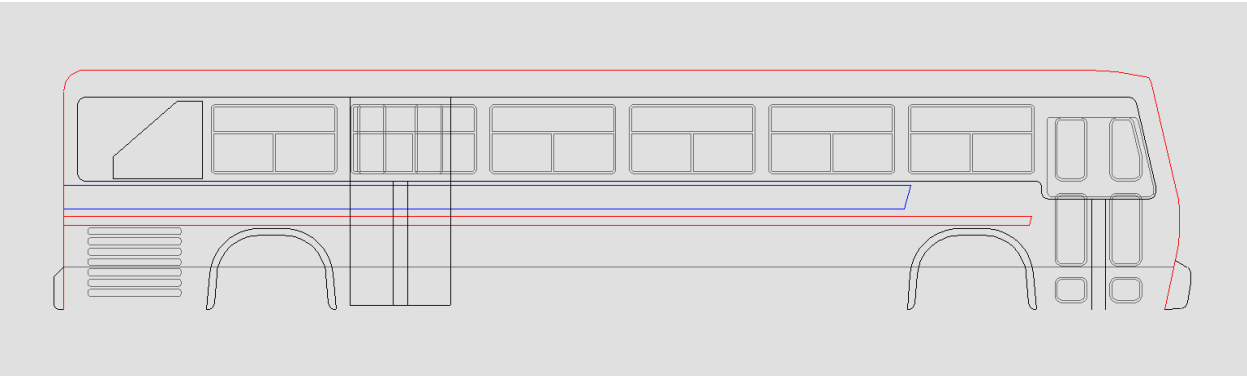

Figure 13. *The windows, doors and wheel openings can be trimmed from the exterior shell using lines and curves drawn from the orthographic views.*

These lines and curves were rotated and aligned to the exterior surfaces. Using the *Split* command, each detail was split from the exterior surfaces. Details, such as the doors, are present on the right side only. To prevent the split command from splitting both sides, a surface was built from the detail with the *Extrude* command. The extruded surface passes through the right side only. Using it as a cutting edge, it splits only the side through which the extruded surface passes. Figure 14 shows these details split from the exterior surfaces.

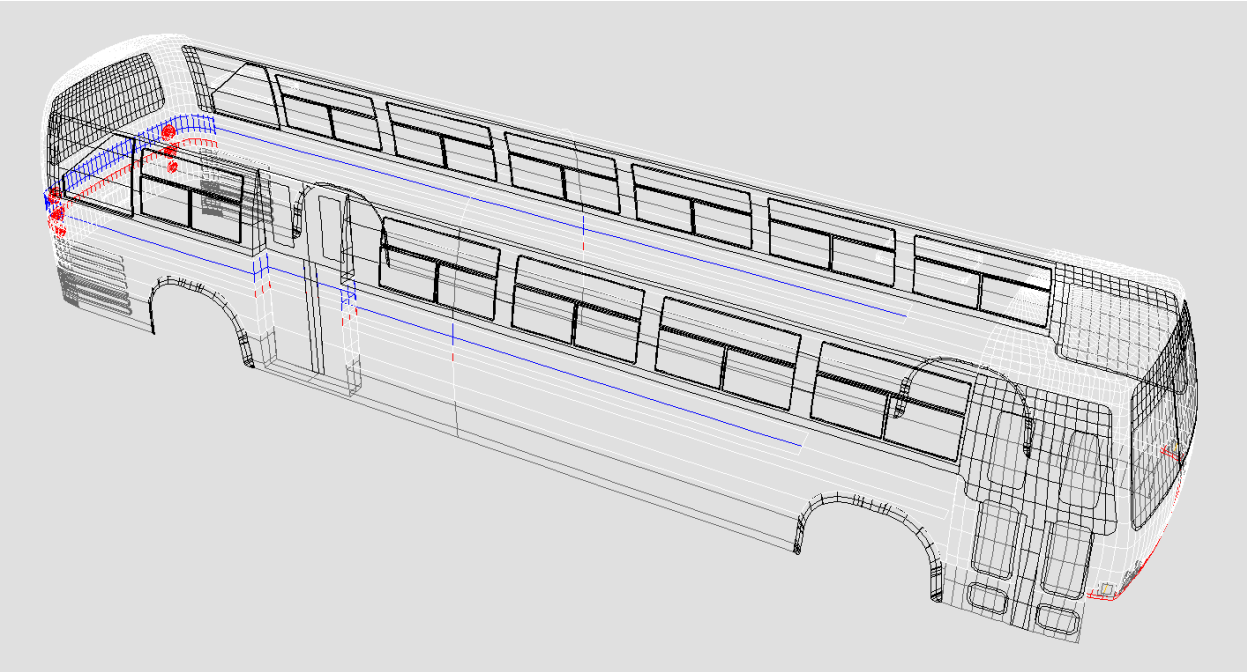

Figure 14. *The vehicle details were split from the initial surfaces. The color attributes were changed to better visualize the split surfaces.*

To give the model more depth, the windows were moved inboard approximately one inch. The gap between the body and window was filled with the *EdgeSrf* command. The same process was performed on the engine grill. The wheel openings were split twice, and flared outboard. The opening for the rear bus door was deleted, and replaced with a vertical planar surface. The gap filled as previous. Figure 15 shows a close up of the rear door and windows.

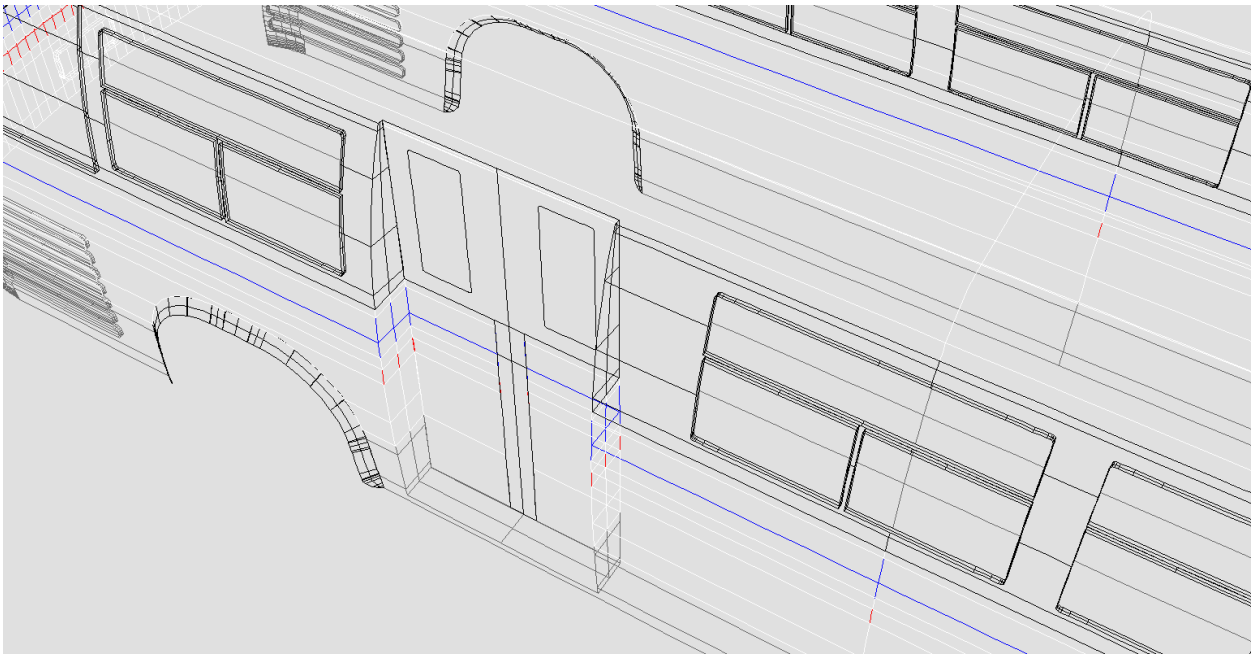

Figure 15. *The windows were moved inboard to give the model depth. The rear door surface was replaced with a vertical planar surface. The EdgeSrf command was used to fill the gaps between these details and the exterior surfaces.*

The bumpers were constructed from a network of curves. Profile and perimeter curves were created from the orthographic diagrams. These curves were copied to complete a wire frame of the bumper as in Figure 16.

The bumper surface was constructed using the *NetworkSrf* command (Figure 17). To prevent overlapping surfaces, the *Trim* command was used to remove the surfaces behind the bumper (Figure 18).

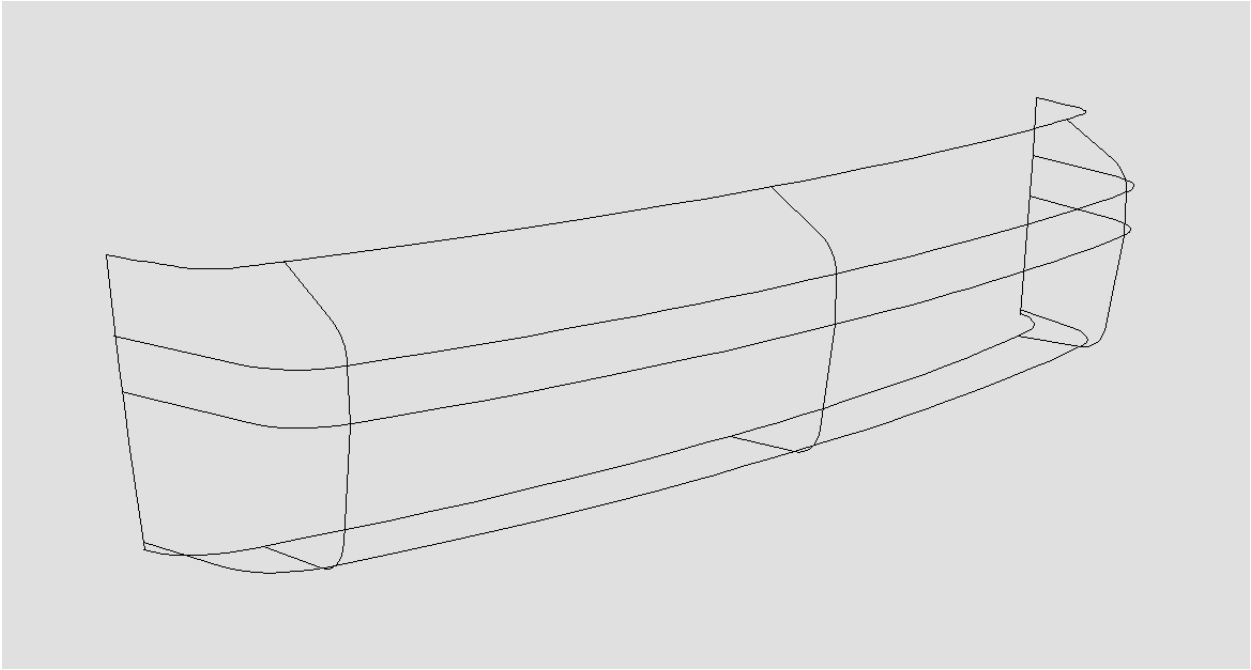

Figure 16. *A wire frame of the front bumper was drawn from the orthographic diagrams. This forms a curve network from which the bumper surface was constructed.*

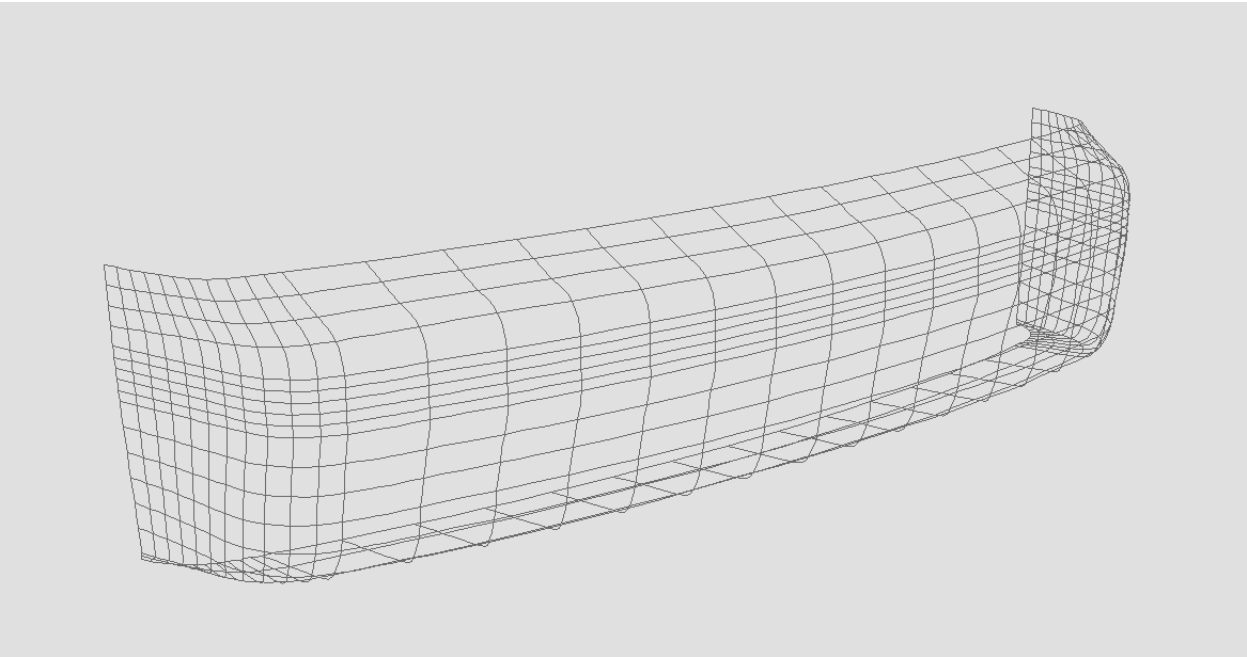

Figure 17. *The bumper surface was created with the NetworkSrf command from the curve network in Figure 16.*

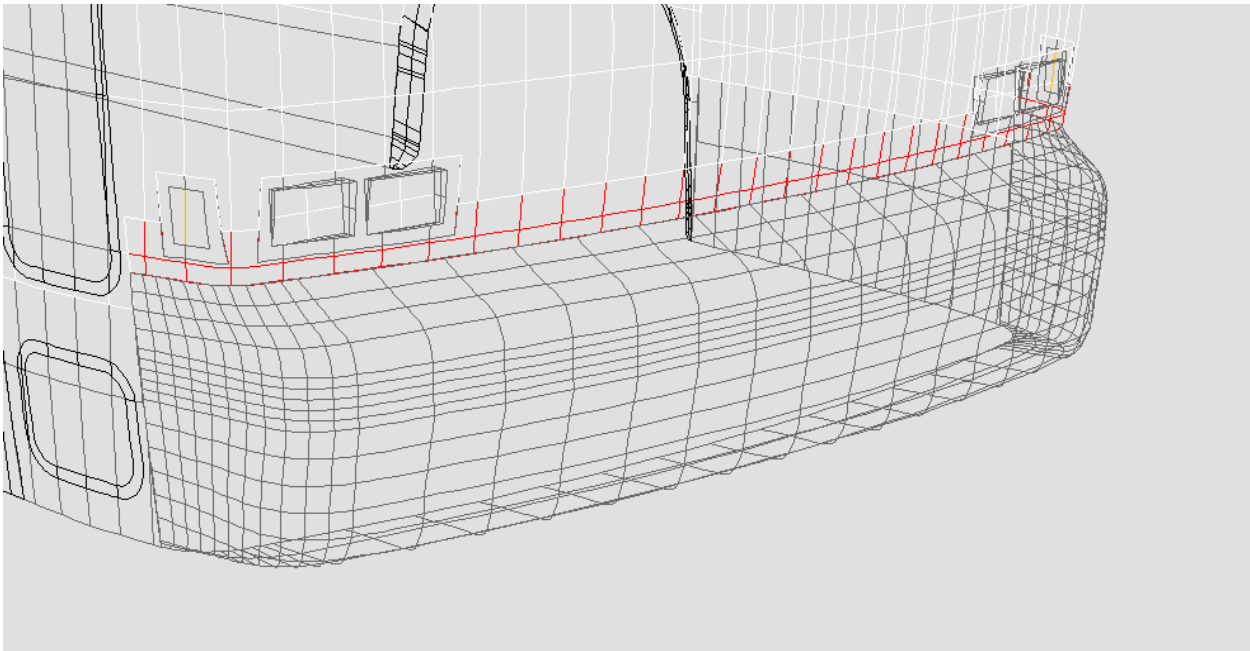

Figure 18. *The surfaces behind the bumper were trimmed to prevent overlapping surfaces.*

The bottom of the bus is planar, with the exception of the wheel wells. An initial surface was built with the *Plane* command in the *xy* plane. The edges of the surface extended beyond the perimeter of the body, and were trimmed by the perimeter curve along the bottom of the wire frame. The surface was moved to the proper elevation.

The edges of the wheel openings in the sides of the body were used to create the wheel wells. A surface was built between the wheel openings on opposite sides of the bus with the *EdgeSrf* command. The back wall of the wheel well was built with a vertical surface parallel to the *xz* plane. This surface was offset from the centerline approximately two feet to provide sufficient room to house the tires once the geometry is imported into HVE.

Finally, the bottom and interior wall surfaces were trimmed by the extended surfaces. The final surface is shown in Figure 19.

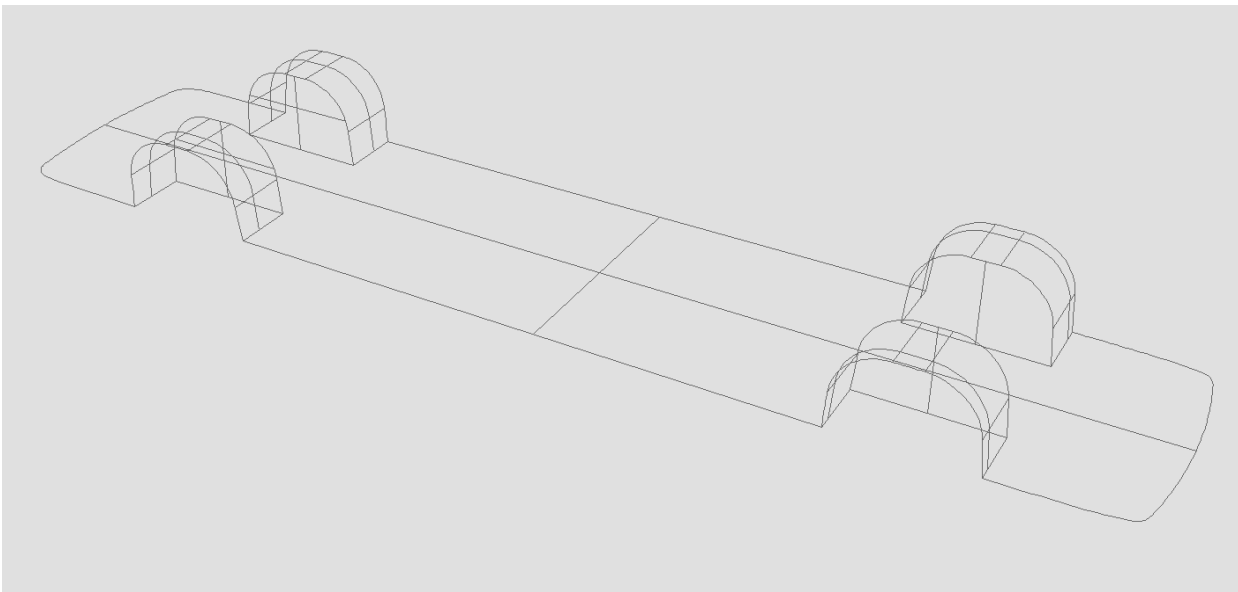

Figure 19. *The bottom of the bus was formed by a trimmed planar surface. The wheel wells were built from the edges of the wheel openings in the sides of the bus, and trimmed by the vertical surfaces forming the back wall. All surfaces were trimmed such that no surface extended beyond the adjacent surface.*

Meshing

The NURBS surfaces built in Rhino must be meshed before exporting to HVE. The seams between contiguous surfaces must be watertight. Gaps in the mesh can result in unexpected damage profiles in collision simulation models. In DYMESH events, these gaps may lead to the collision routine ignoring interaction with the detached surface.

By joining the NURBS surfaces before meshing these gaps can be avoided. The *Join* command was used to join the individual surfaces into one polysurface. The *ShowEdges* command was used to check for naked edges in the model. A naked edge will indicate a gap between surfaces. Figure 20 shows naked edges appearing along the roof and bumper.

Since the bus is a solid, there should be no naked edges within the model. The *JoinEdge* command can be used to close the gaps in adjacent surfaces. It overrides the tolerance which prevented the successful joining with the *Join* command.

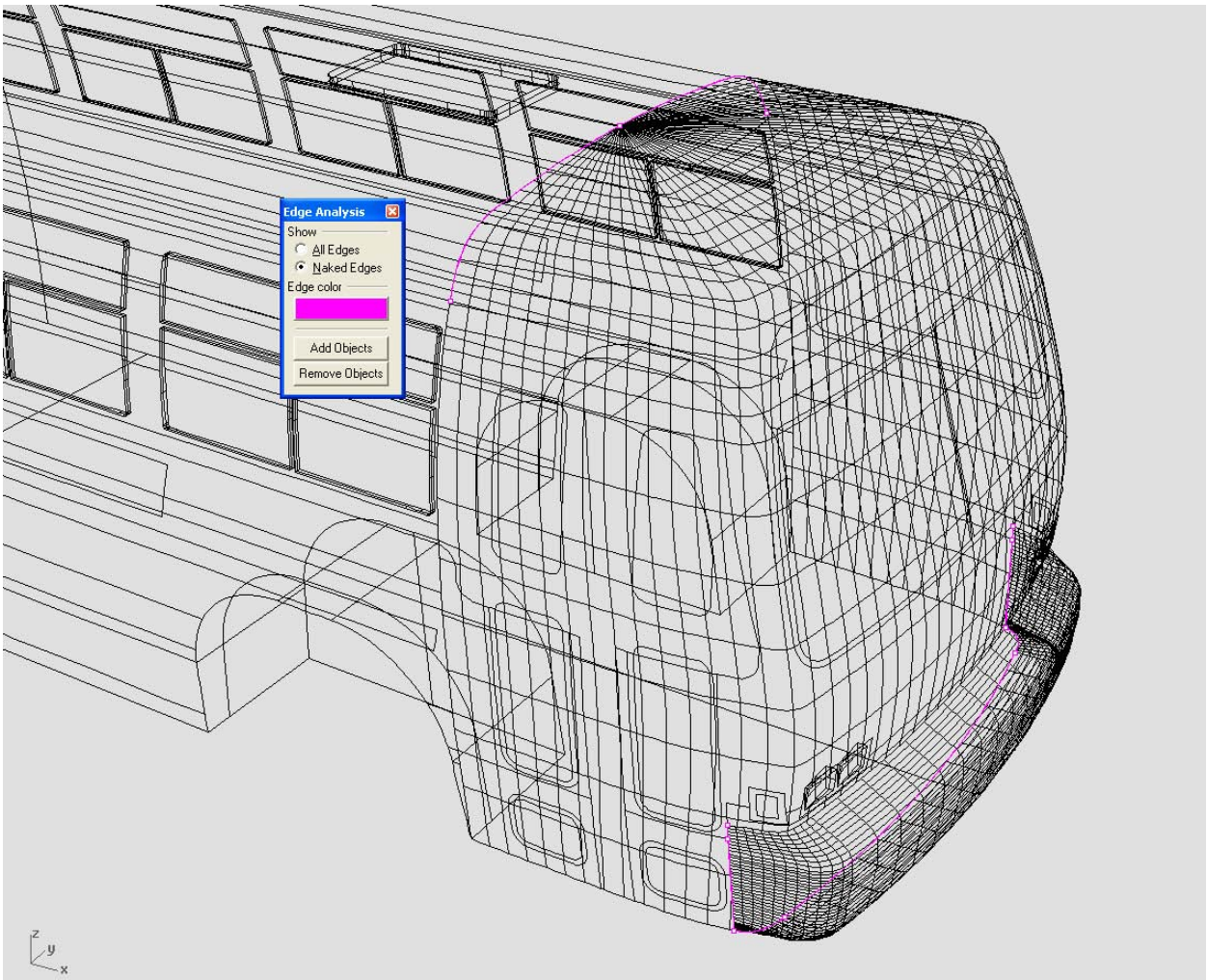

Figure 20. *The ShowEdge command displays naked edges in the selected polysurface. The magenta lines indicate gaps in the model. The JoinEdge command was used to close these gaps so that the mesh is watertight.*

The *Mesh* command creates a polygon mesh from the NURBS surfaces based upon user entered parameters. The vertices of contiguous surfaces will be coincident.

The mesh density is controlled by the Polygon Mesh Options dialog, and is a compromise between the accuracy of the model, and the ability of HVE and the user's system to manage the geometry file size. Decreasing the *maximum angle* in the detailed options dialog will increase the polygon count and the accuracy of curved surfaces. The *maximum edge length* can be used in lieu of the tessellation option in HVE. Figure 21 shows the Polygon Mesh Detailed Options dialog with typical settings.

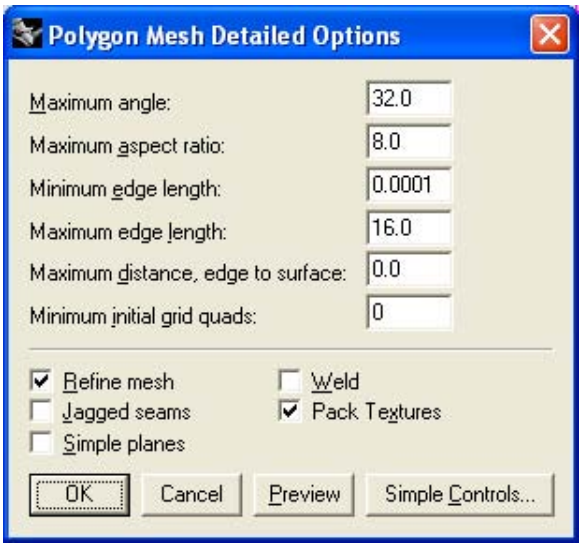

Figure 21. *Mesh density is controlled by the Polygon Mesh Options dialog.*

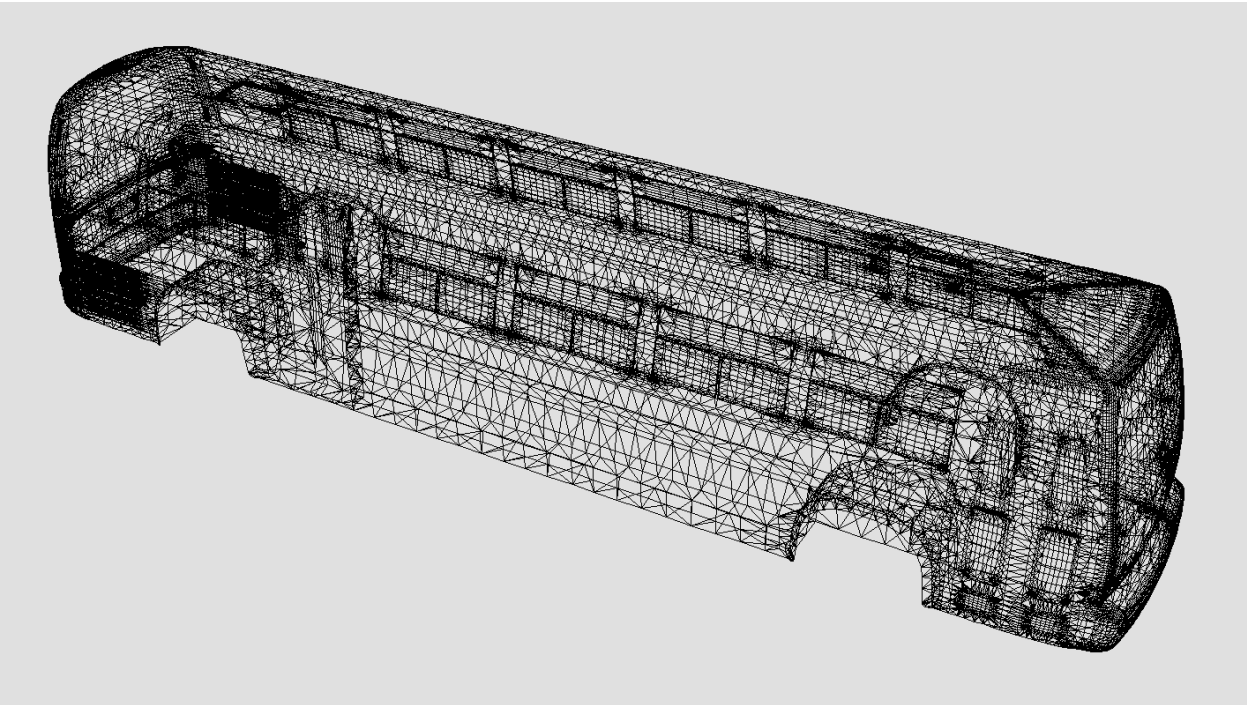

Figure 22. *The polygon mesh was created from the joined polysurface with the Mesh Command.*

The polygon mesh is a separate entity from the NURBS polysurface used to create it. Figure 22 shows the polygon mesh of the transit bus. At the time of creation, the mesh is one object. The *Explode* command was used to separate the mesh into individual meshes corresponding to the joined surfaces. The exploded meshes were moved to individual layers. The attributes were set by layer to match the color scheme of the bus (Figure 23).

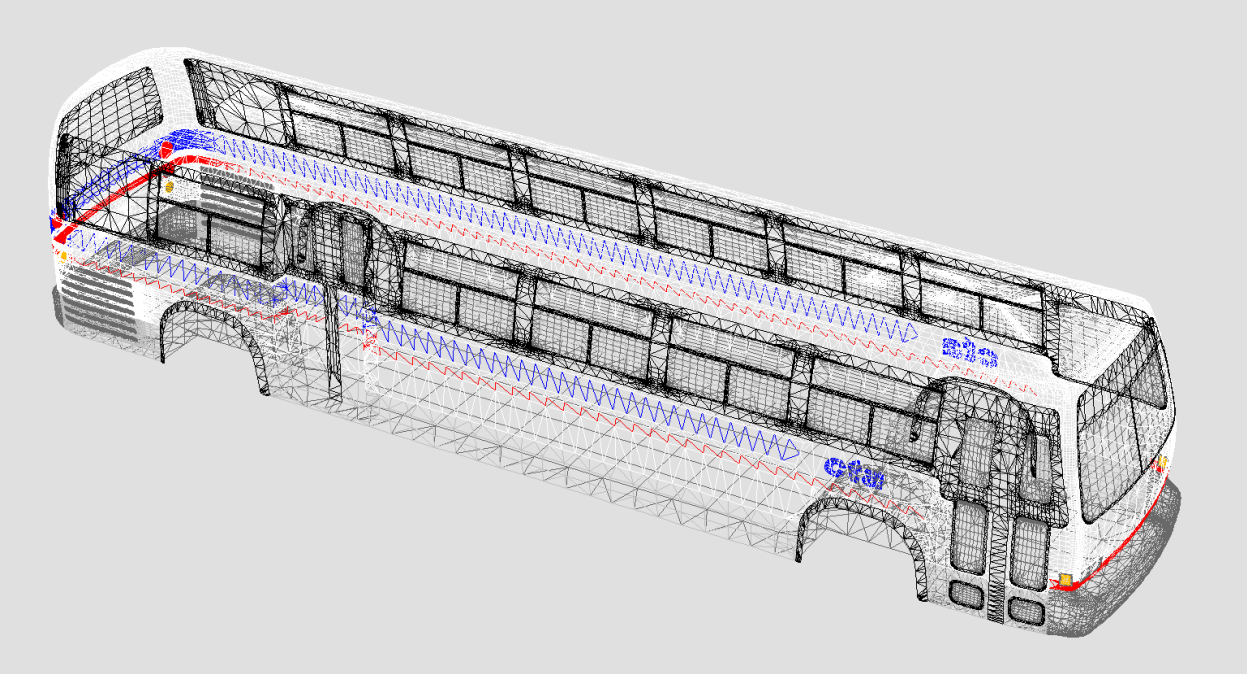

Figure 23. *The exploded meshes were moved to individual layers and the attributes set to match the bus's color scheme.*

Rhino's material properties control how the mesh will appear when rendered. The basic color setting controls the color of the mesh, and corresponds to the diffused color setting in HVE. The gloss finish (reflective finish in Rhino, Version 2.x) corresponds to the ambient color setting in HVE. This should be set to black for the most desirable results.

In Rhinoceros, Version 3.0, the ambient color of a mesh may not translate correctly when imported into HVE. The surface may appear "washed out." The ambient color can be edited in the 3D Editor in HVE, or with a text editor such as Notepad. This does not occur with version 2.x.

The transparency setting for the windows or translucent surfaces can be set in this dialog too. Figure 24 shows the material properties for the window layer of the transit bus.

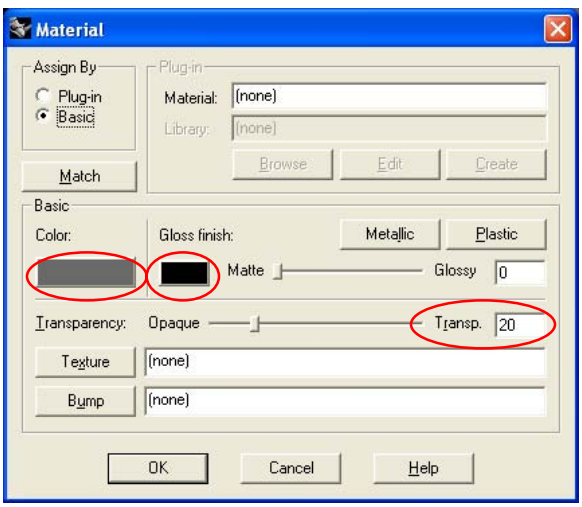

Figure 24. *The material properties control the appearance of the polygon mesh. The Basic color setting controls the color of the rendered mesh. The Gloss finish setting should be set to black. The Transparency setting can be used for glass or translucent surfaces.*

Export

The vehicle geometry was prepared for export. Because the background bitmap was placed approximately to scale, adjustments may be required. The *Distance* command was used to compare the length, width, height and wheelbase to the dimensions listed on the diagrams. Adjustments to these dimensions were made with the *Scale* command. If necessary, Rhinoceros can scale objects in one or two dimensions only using the *Scale1D* and *Scale2D* commands respectively.

The origin of the vehicle coordinate system in HVE is located at the vehicle's center of mass. Using dimensional and inertial data, the origin of the geometry file was relocated. The geometry file must be rotated about the *x* axis to correspond to the SAE coordinate system used by HVE. If the drawing units were in inches, no scaling is required.

The preferred file format for importing into HVE is VRML (.wrl). This format retains the material properties and surface normal orientation of the polygon mesh. Rhinoceros exports this file type. Only the polygon meshes are exported. Using the *Export* command, the meshes were exported as a VRML (.wrl) file. The VRML file is ready for import into HVE.

Conclusion

The methods presented herein provide a means of building geometry files for select vehicle types. These vehicle types are not predominate in the Engineering Dynamics Vehicle Database, or readily available from other sources. This provides the user with an option to digitizing a vehicle. This methodology can be used to build vehicle components that can be added to existing vehicle geometries (Figure 25).

Figure 25. *This ambulance was created by modifying an existing vehicle geometry.*

References

HVE Physics Manual, Engineering Dynamics Corporation, Beaverton, OR, 2004.

Rhinoceros Users Guide, Version 3.0, Robert McNeel & Associates, 2003.

Sneddon, James P., *Introduction to Creating HVE Environments with Rhinoceros*, 2002, WP2002-2, Engineering Dynamics Corporation, Beaverton, OR.

Cheng, Ron K.C., *Inside Rhinoceros*, 2002, Onword Press, Albany, NY.

HVE Operations Manual, Engineering Dynamics Corporation, Beaverton, OR, 2000.

Trademarks

HVE and EDSMAC4 are trademarks of Engineering Dynamics Corporation.

DYMESH (US Patent Number 6,195,625) and SIMON are registered trademarks of Engineering Dynamics Corporation.

Rhinoceros and Rhino are registered trademarks of Robert McNeel & Associates.