

D3PLOT 18.0

D3PLOT 18.0 – Contents

- [D3PLOT Viewer](#)
- [MP4 Movie Support](#)
- [Animations in REPORTER](#)
- [Cut Sections](#)
- [Material History Variables](#)
- [Colour](#)
- [Export Geometry to PRIMER](#)
- [Contour Bar Limits](#)
- [Blanking Contact Segments](#)
- [Attached](#)
- [Copying Settings between Windows and Models](#)
- [JavaScript API](#)
- [JavaScript GUI Builder](#)
- [JavaScript Engine Upgrade](#)
- [Checkpoint Files](#)
- [Miscellaneous](#)

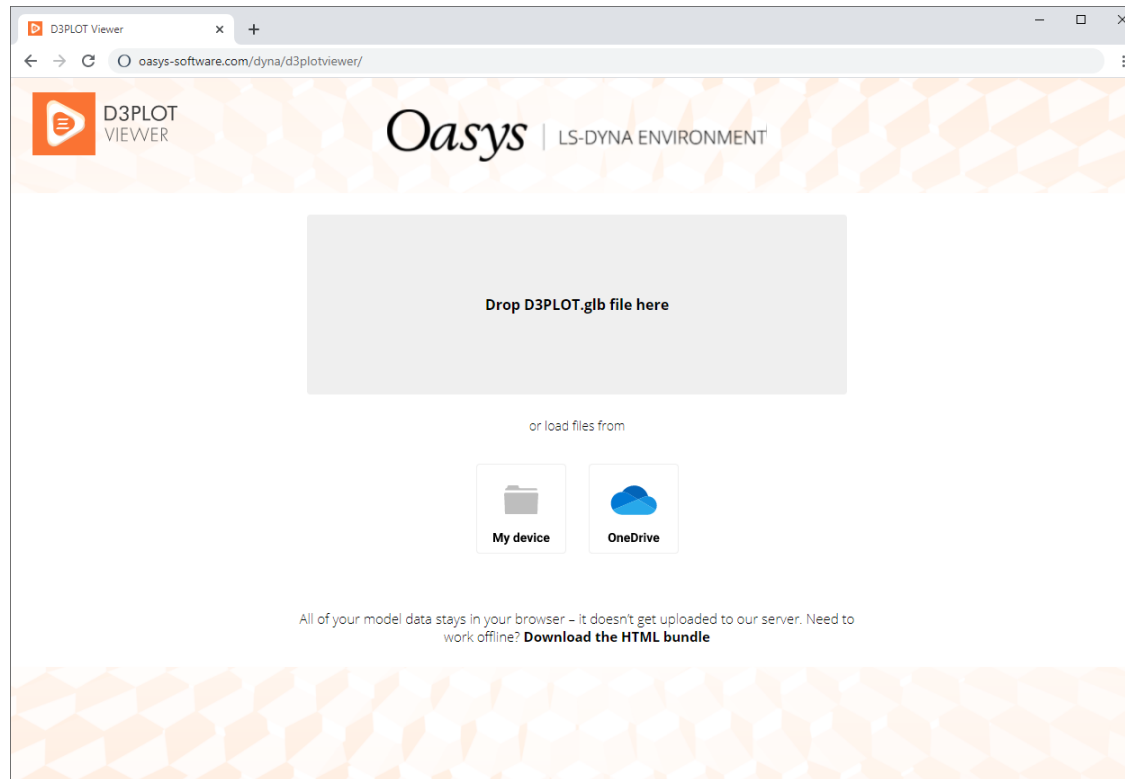


D3PLOT
VIEWER

3D results on the web, for **everyone**

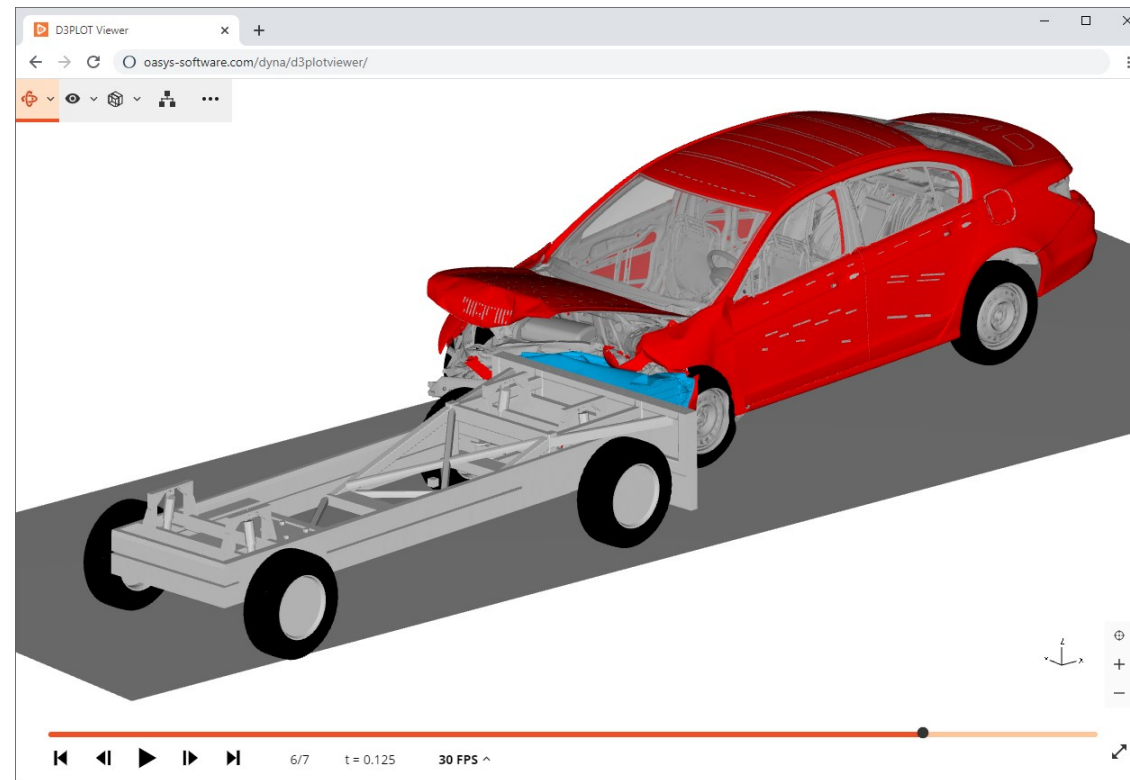
D3PLOT Viewer

Alongside D3PLOT 18.0 is the launch of [D3PLOT Viewer](https://oasis-software.com/dyna/d3plotviewer/), a brand new 3D web viewer for LS-DYNA simulations.



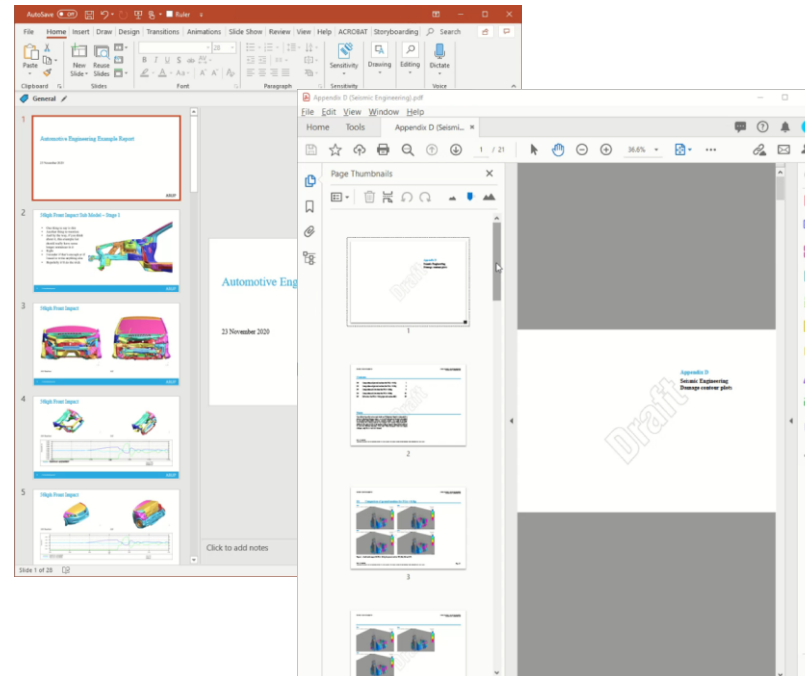
D3PLOT Viewer

D3PLOT Viewer has the power to transform the way that you review, communicate, and deliver engineering analysis.



D3PLOT Viewer

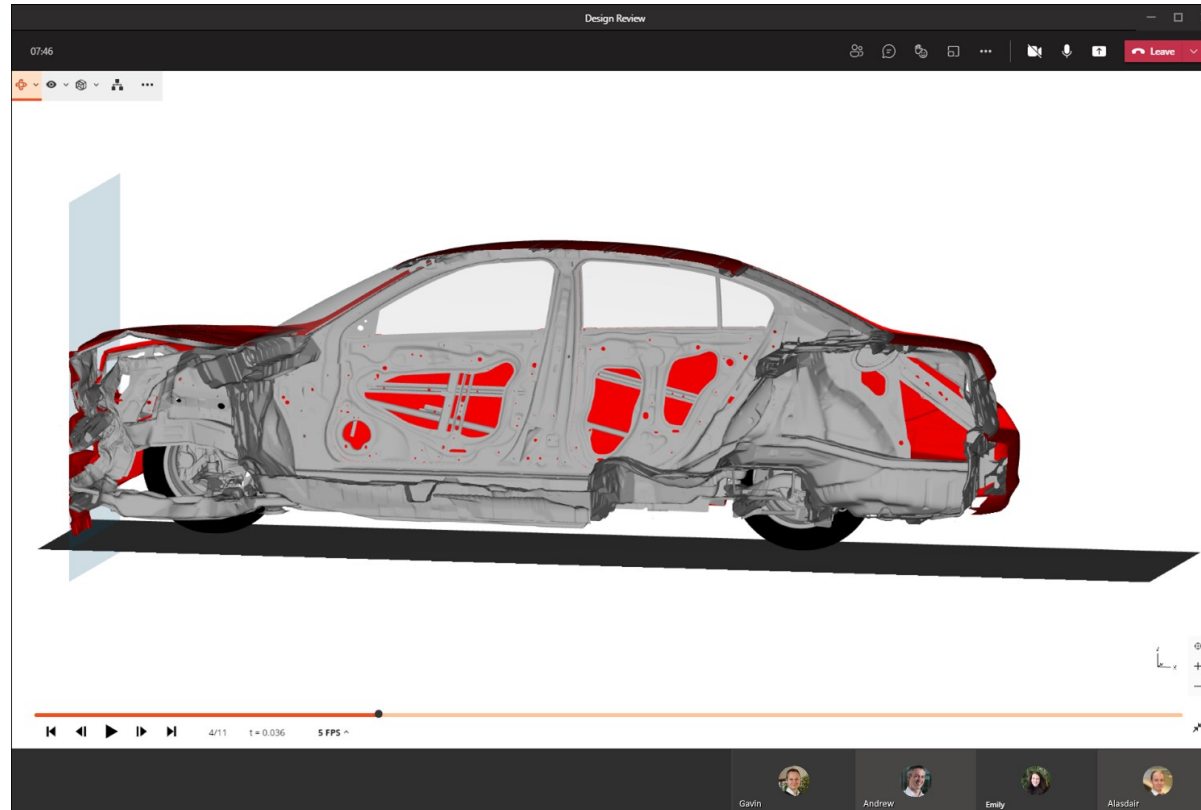
Break free from the 2D confines
of traditional reports:



With D3PLOT Viewer, you can...

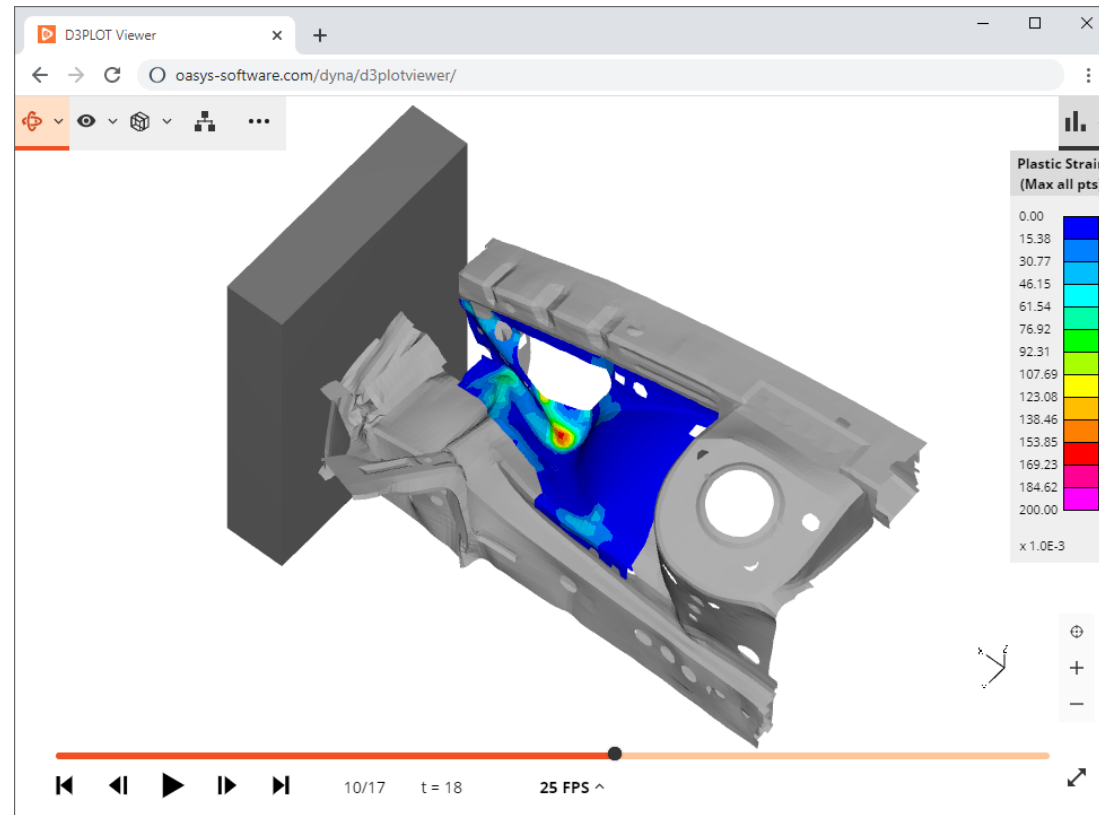
D3PLOT Viewer

Improve productivity in team meetings and design reviews by exploring results together in 3D...



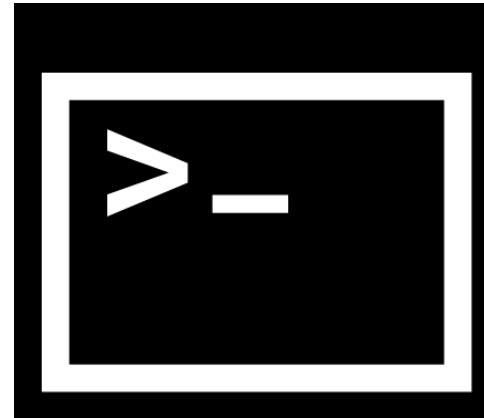
D3PLOT Viewer

Share animated 3D models with designers and suppliers to help them improve the performance of parts and assemblies...



D3PLOT Viewer

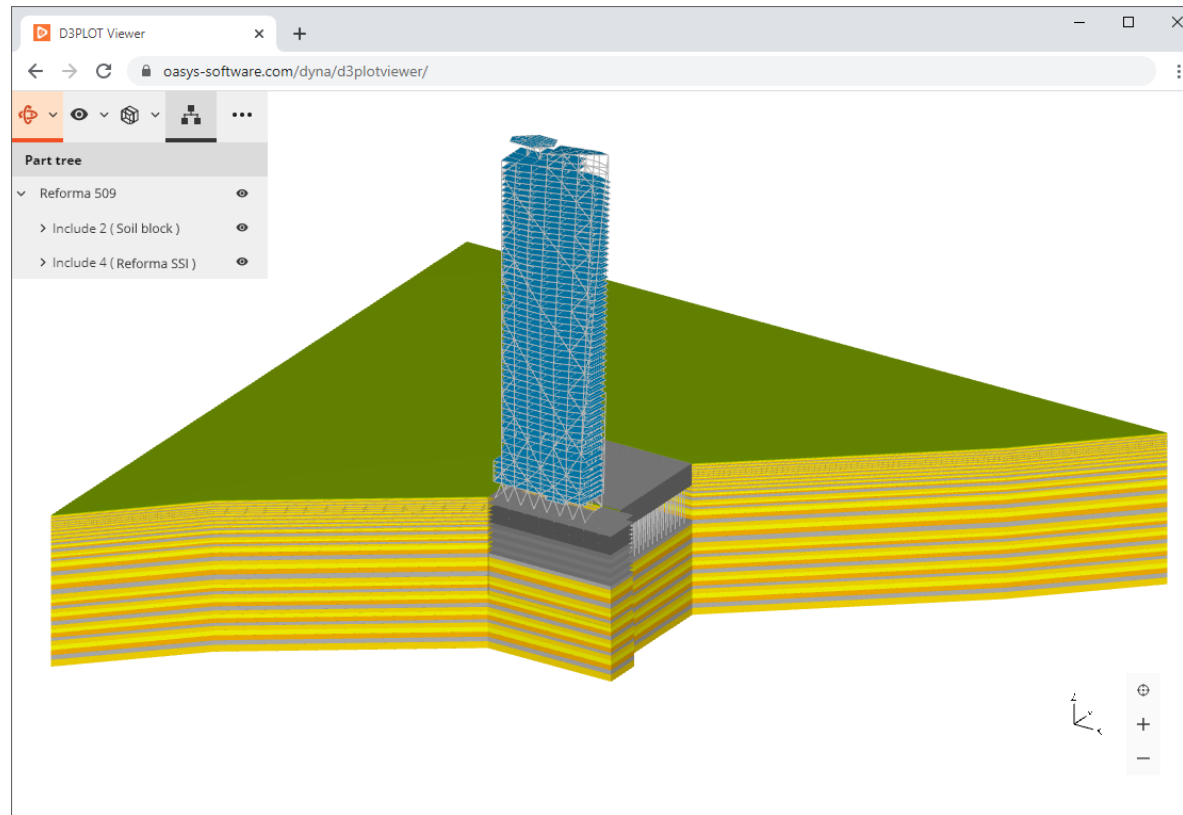
Automatically generate the output files as part of your LS-DYNA job submission – then use D3PLOT Viewer to make a quick check of results as they complete...



```
d3plot18_x64.exe -glb=D3PLOT.glb -states=even -frame_rate=5 d3plot/results.ptf
```

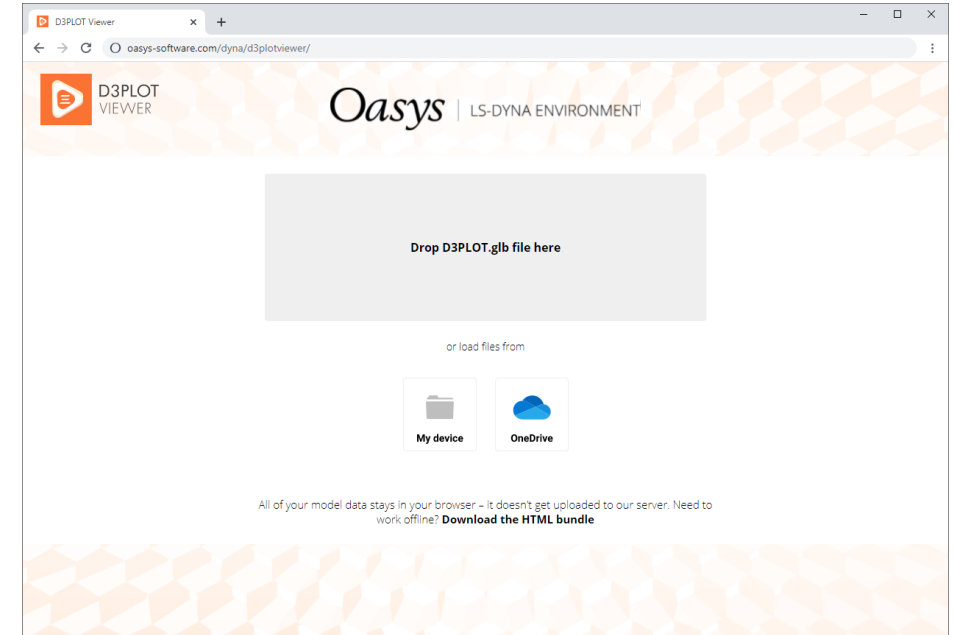
D3PLOT Viewer

Give your clients added value with 3D project deliverables...



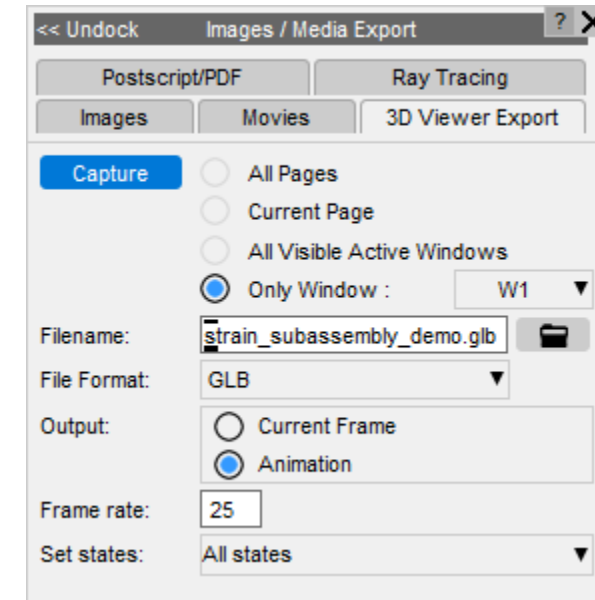
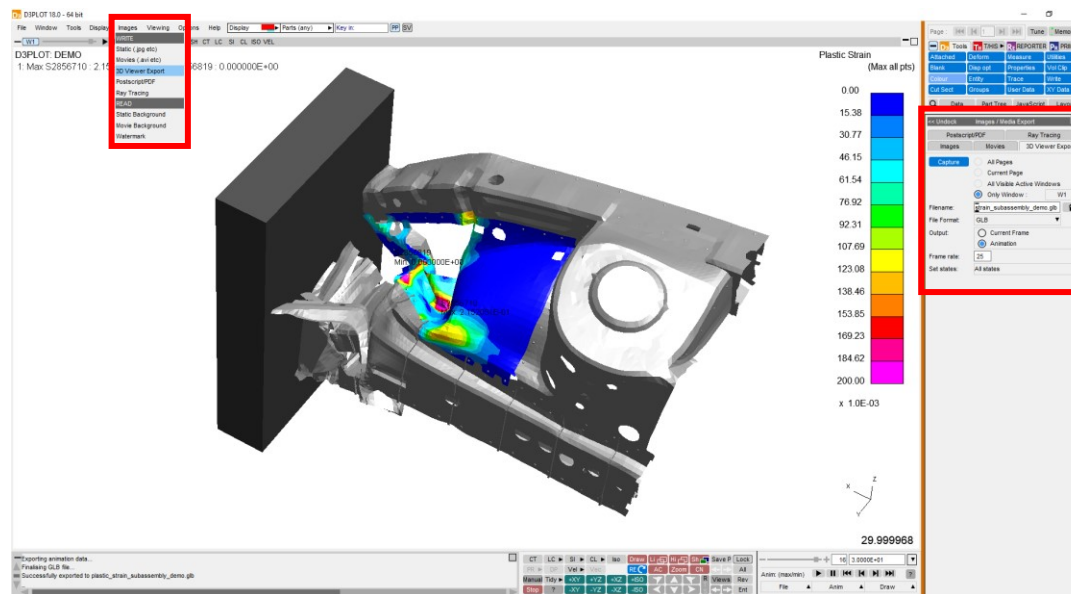
How it works

- D3PLOT Viewer is free to use – free for your team, your partners and your clients
- No installation, no registration, no setup required – simply load models and start exploring in 3D
- D3PLOT Viewer reads GLB files, which are generated by D3PLOT 18.0
- When you open a model in D3PLOT Viewer, all of your model data stays in your browser – it doesn't get uploaded anywhere
- If you need to work offline, you can download the HTML bundle



3D Viewer Export

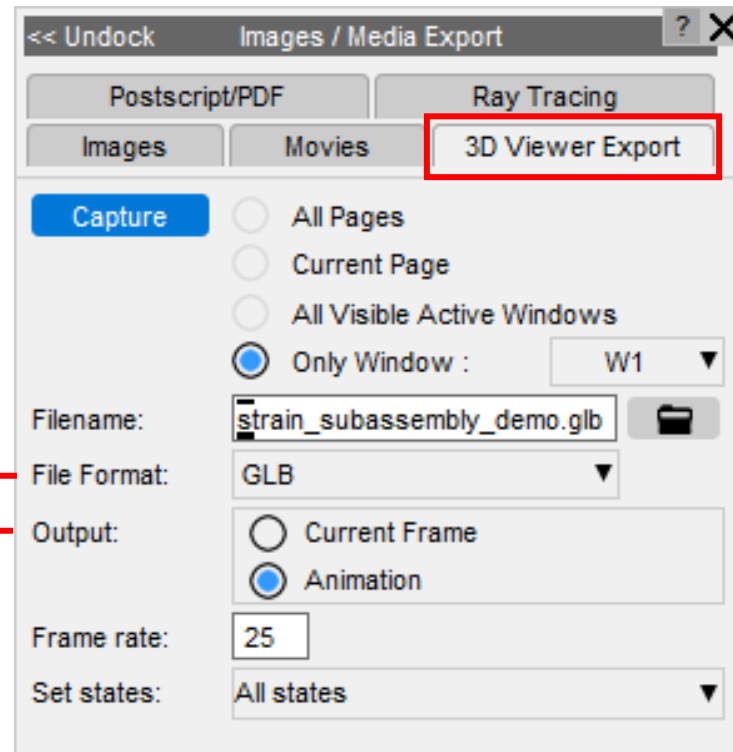
- D3PLOT can now output models in 3D as GLB files
- Export is controlled from the **3D Viewer Export** tab on the Images / Media Export panel (**Images** → **3D Viewer Export**)



3D Viewer Export

File format is compressed by default (5-10 times smaller files), but can be set to uncompressed (faster to read and write, can be read by PowerPoint)

Export either a single frame or an animation

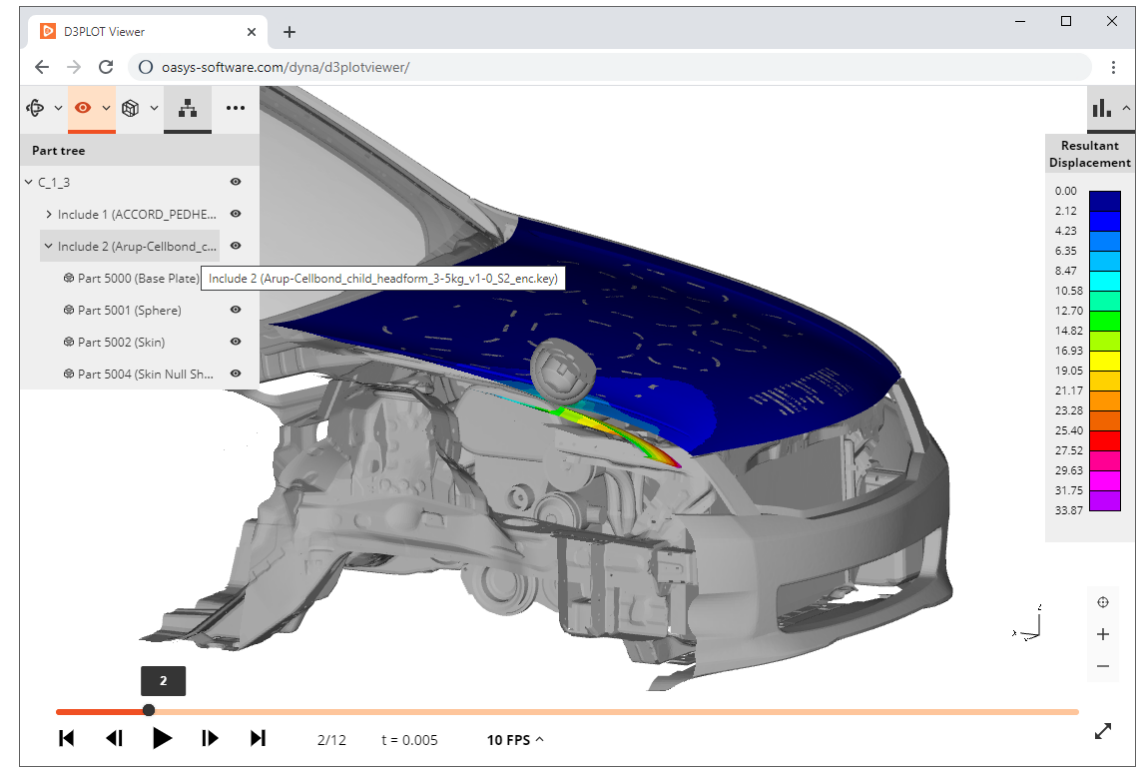


The first model from a single window is exported; select the window using the drop down menu

For animations, you can control which plot states are exported, as well as the default playback frame rate

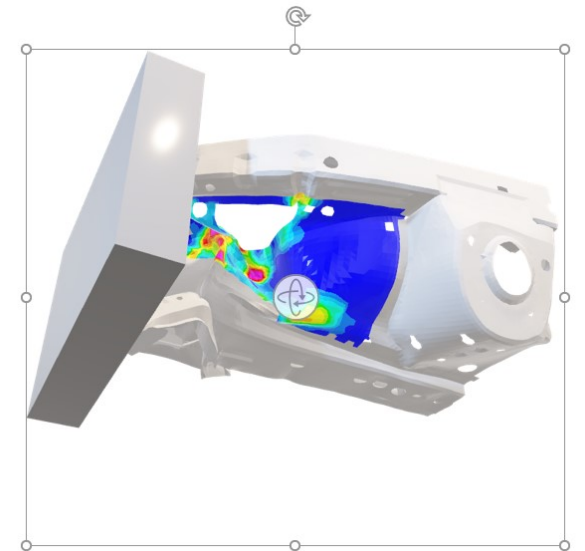
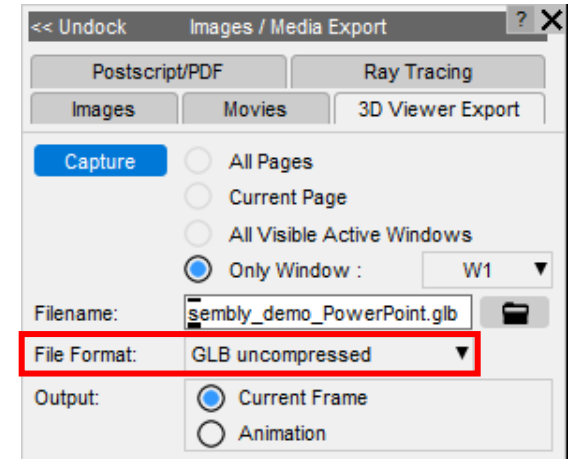
3D Export

- D3PLOT exports specially-customised [GLB files](#), best viewed in [D3PLOT Viewer](#) – but they can be opened in other viewers (or inserted into Microsoft PowerPoint slides) with limited functionality.
- D3PLOT Viewer includes features that are not available in other viewers:
 - Animations
 - Part Tree
 - Contour bar
 - Blanking parts and includes
 - D3PLOT-like navigation controls



Compatibility with Microsoft PowerPoint

- If you want to export GLB files compatible with Microsoft PowerPoint, select the **GLB uncompressed** file format (PowerPoint does not yet support Draco compression)
 - When viewed in PowerPoint, only the first plot state will be shown (GLB files from D3PLOT are not yet compatible with animation of 3D Models in PowerPoint)
 - Despite these current limitations, we think that D3PLOT's 3D Viewer Export will be useful for inserting 3D models of individual parts and subassemblies into PowerPoint presentations.
- Want to see more supported in PowerPoint? [Let us know](#).



File sharing and Cloud Storage

- We want to make the process of exporting and sharing LS-DYNA results as simple and seamless as possible, which is why we are in the process of integrating various cloud storage solutions with D3PLOT Viewer.
- Soon, you will be able to upload your GLB files to your own organisation's cloud storage provider, and then share links with your team, so that they can open models directly in D3PLOT Viewer with a single click.
- We are starting with Microsoft SharePoint/OneDrive integration. Once that is operational, we will look at adding other providers (e.g. Google Drive, Dropbox). [Let us know](#) which cloud storage provider your organisation uses so that we can prioritise development.

Current limitations

We have worked hard to pack as much as we can into this first release of D3PLOT Viewer but, for now at least, 3D export from D3PLOT has the following limitations:

- Solid, thick shell, shell and beam elements are supported; other entity types are not yet supported
- SH, CT and SI plot modes are supported, other plot modes revert to SH plot
- Contour values are output as a single value at each node, so only low and medium resolution contouring is supported; unaveraged and high resolution contouring reverts to medium resolution
- Exporting cut sections from D3PLOT is not yet supported (although you can add a cut section in D3PLOT Viewer)

File size

- The maximum size of model you can open in D3PLOT Viewer is limited by web browser constraints. At the time of D3PLOT 18.0 release, D3PLOT Viewer has the following limits:

File format	Maximum file size
GLB (compressed; default)	~ 4 GB uncompressed data – will typically result in a GLB file of < 1 GB
GLB uncompressed	2 GB

We will continue to work on ways of increasing these limits.

- **What does this mean?** You might not be able to view 50+ plot states of a full vehicle crash simulation. But you will probably be able to view 10-20 plot states of a full vehicle, or more plot states of a subset of the model. It all depends on the model size, and whether you include contour data. We encourage you to explore what is possible for you, and send us feedback!
- Are file size limitations causing you problems with D3PLOT Viewer? [Let us know.](#)

MP4 Movie Support

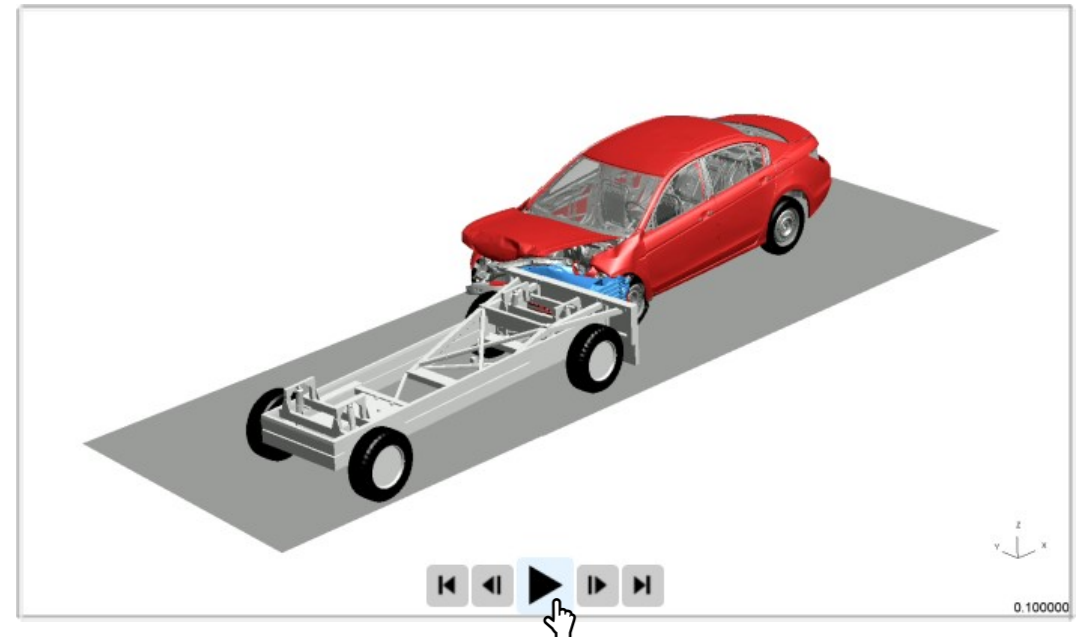
Reading and writing MP4 files

MP4 Movie Files

In D3PLOT 18.0, we have added support for MP4 movie files using H.264 Advanced Video Coding.

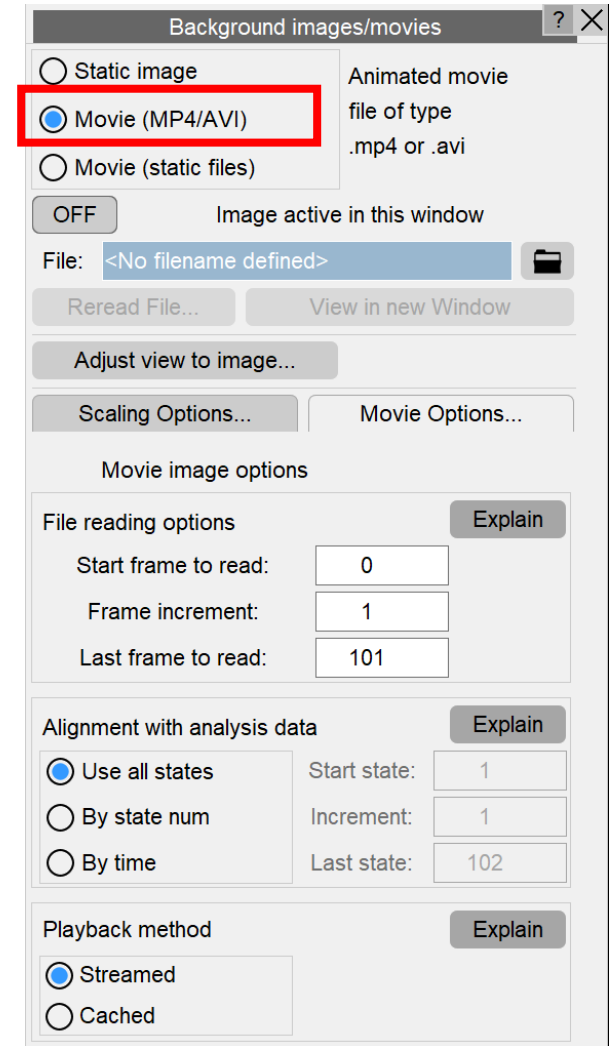
This allows you to export high-quality animations of your LS-DYNA results in a modern format compatible with Microsoft PowerPoint, video players, web browsers, and the latest business communication platforms.

Bring your LS-DYNA visualisations to life with MP4 Movies from D3PLOT...



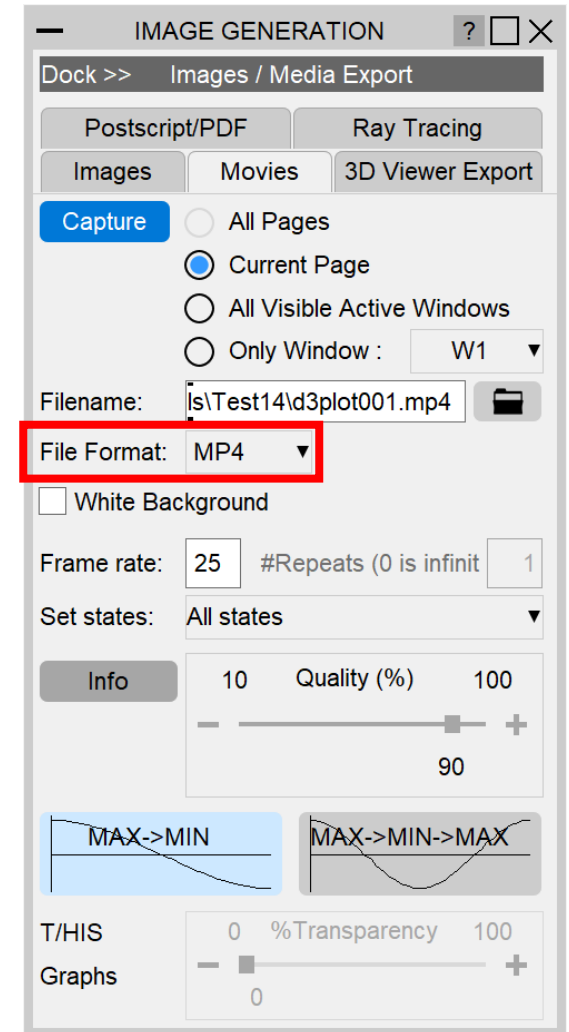
Reading MP4

- In previous versions of D3PLOT, it has been possible to read an AVI movie into the background of a model window for synchronised playback. For 18.0 we have expanded this to also include MP4 files.
- The image on the right shows the panel found under **Images** → **Read** → **Movie Background**, where you can configure MP4 video playback.
- The `IMAGES` dialogue commands `MOVIE_READ` and `MOVIE_DISPLAY` can also be used to read an MP4 file and toggle its visibility in a given window.



Writing MP4

- D3PLOT is now also capable of exporting your animations to an MP4 movie file. This is now the default file format option when writing movies.
- The encoder we use writes MP4 files using H.264 Advanced Video Coding, producing high-quality videos with a small file size.
- The image on the right shows the panel found under **Images** → **Write** → **Movies**, used for writing MP4 files.
- The `IMAGES` dialogue command `MP4` can also be used to write MP4 movies (e.g. `/IMAGES MP4 d3plot001.mp4`).

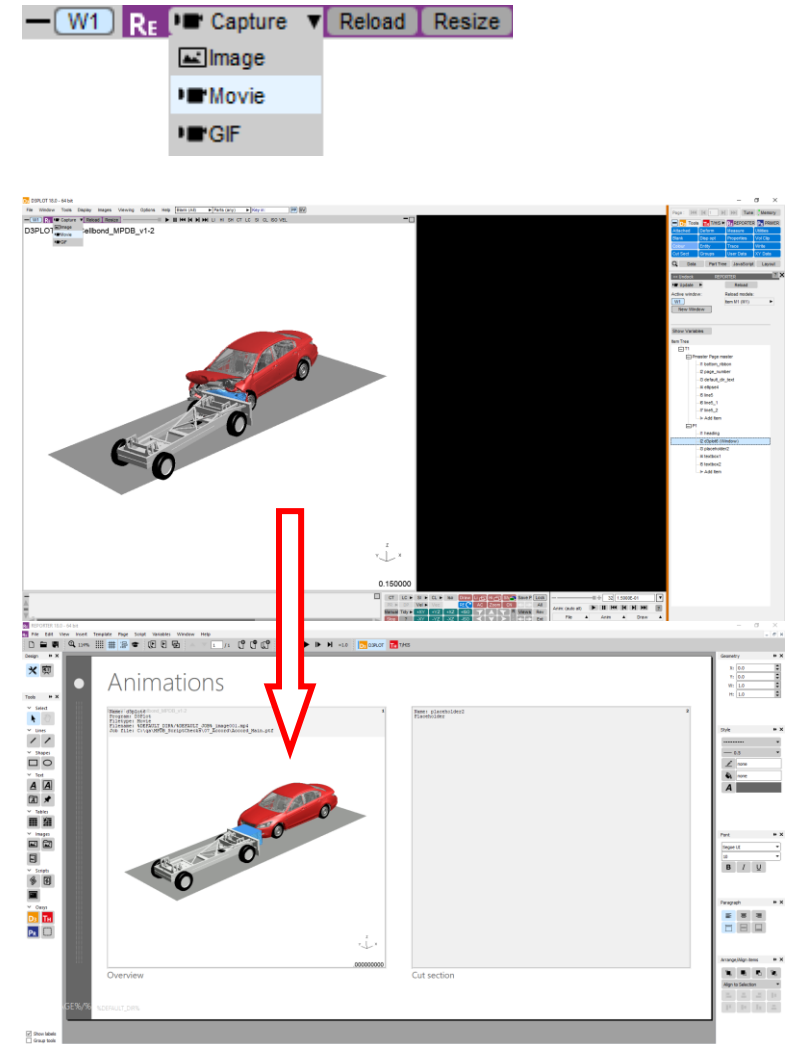


Animations in REPORTER

Capturing MP4 and GIF

Animations in REPORTER

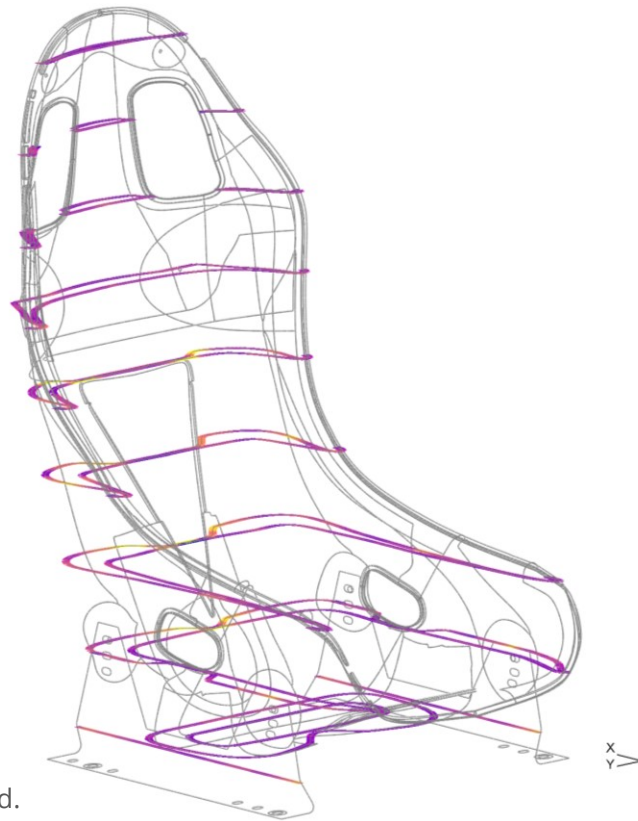
- In addition to adding MP4 support to D3PLOT, we have also added movie playback functionality to REPORTER.
- D3PLOT Items in REPORTER are now capable of Capturing an MP4 movie or an animated GIF.
- The REPORTER Capture button in D3PLOT has been updated to reflect this and now features a drop-down menu with three options: Image (.png), Movie (.mp4), or GIF.
- Existing D3PLOT Items with the Image type can be overwritten with a Movie or GIF when updating your REPORTER Templates.
- When selecting a D3PLOT Item from the REPORTER panel Item Tree, the Capture button will update to match the current type of the D3PLOT Item.



Cut Sections

Multiple parallel cuts

For a given cut plane, multiple parallel cuts can be defined by a (uniform) spacing and numbers of planes in each direction.



Seat design by Reverie Ltd.

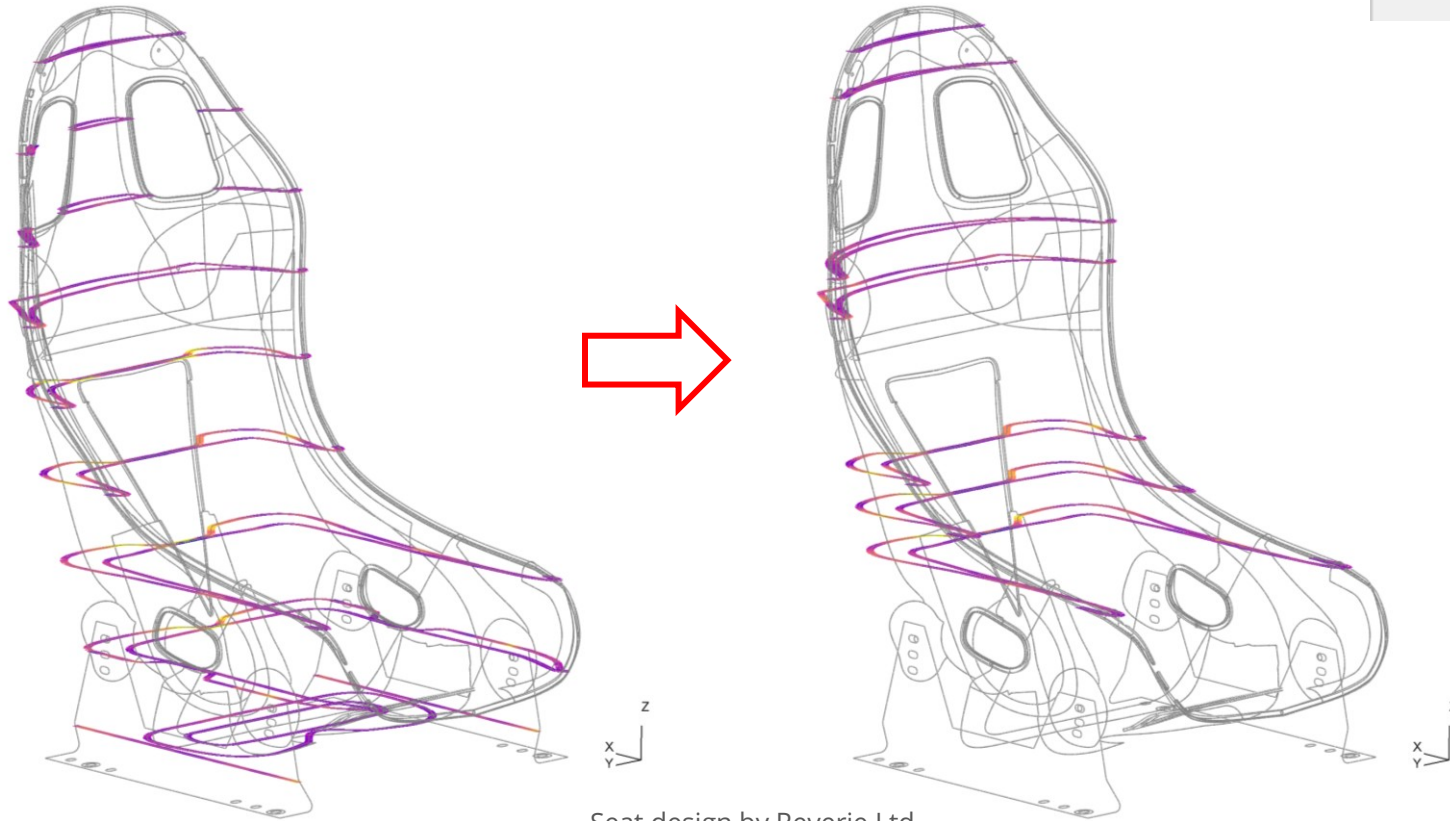
<input type="checkbox"/> Thick cut	?	<input checked="" type="checkbox"/> Multiple cuts	?
Thickness:	100.0	<input type="checkbox"/> Custom spacing	
		Spacing:	100.0
		+/-Np:	999 999

Custom spacing

In D3PLOT 18.0, the spacing can now be customised to any spacing (not necessarily uniform).

☐ Thick cut ?
Thickness: 100.0

☒ Multiple cuts ?
☒ Custom spacing
Edit spacing



Seat design by Reverie Ltd.

CUT SECTION CUSTOM SPACING

Positions relative to local cut origin Done

Number of parallel planes: 8

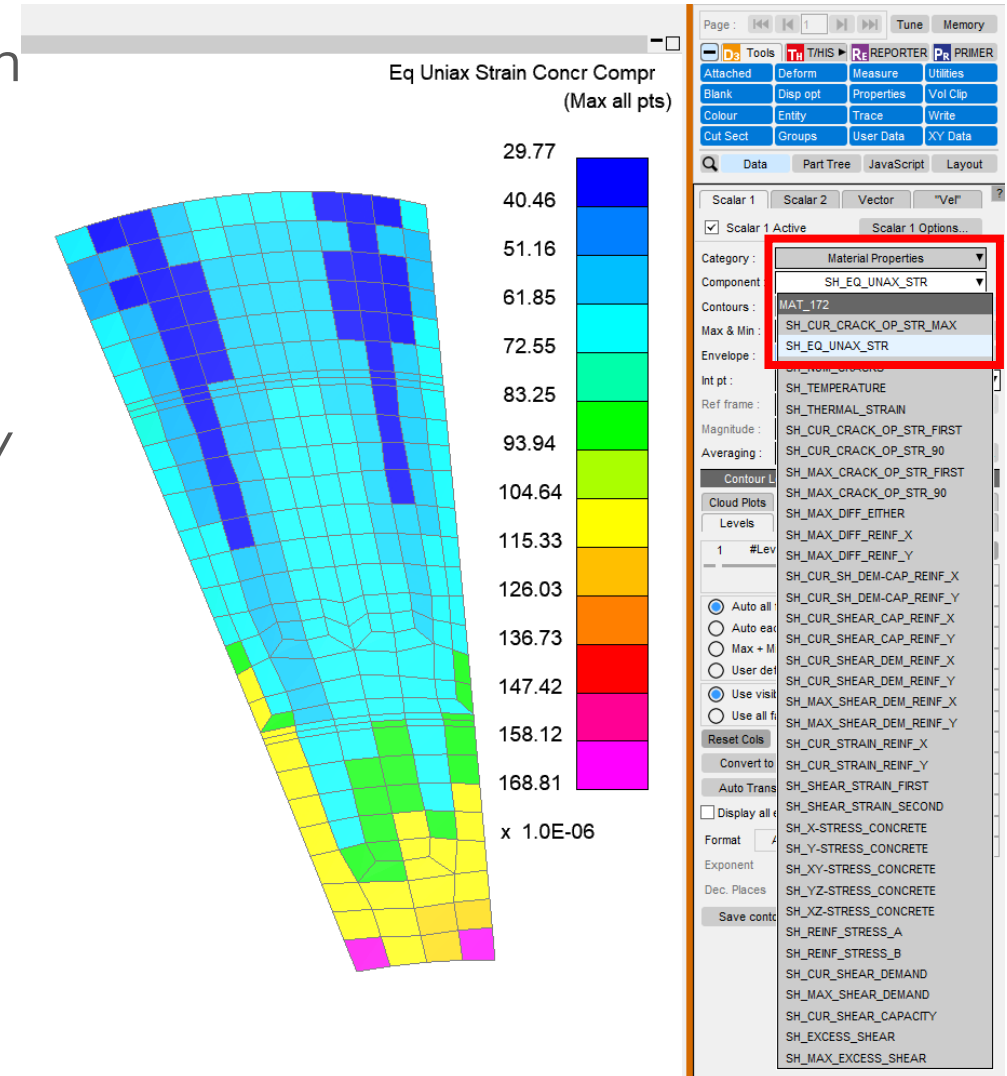
Add position:

1 ▶	-100.0
2 ▶	-50.0
3 ▶	0.0
4 ▶	200.0
5 ▶	250.0
6 ▶	450.0
7 ▶	500.0
8 ▶	600.0
9 ▶	undefined

Material History Variables

The *mat_prop.csv* file

- D3PLOT offers history variables for materials in the **Data** → **Material Properties** menu based on a file *mat_prop.csv* included in the installation package.
- In D3PLOT 18.0, we have updated *mat_prop.csv* to include the latest history variable information.
- With the new preference **d3plot*mat_prop_location**, a different file with user-defined variable names can now be used instead of the default *mat_prop.csv*.

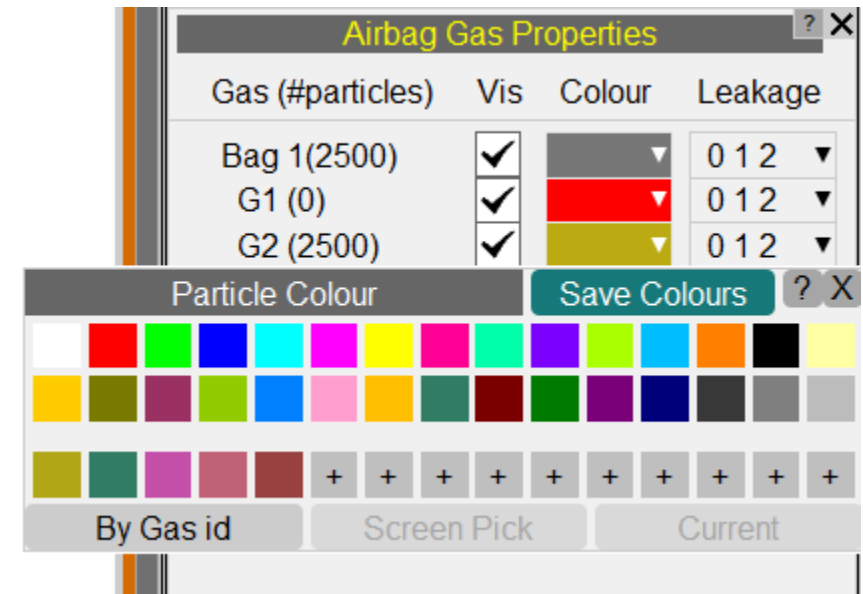
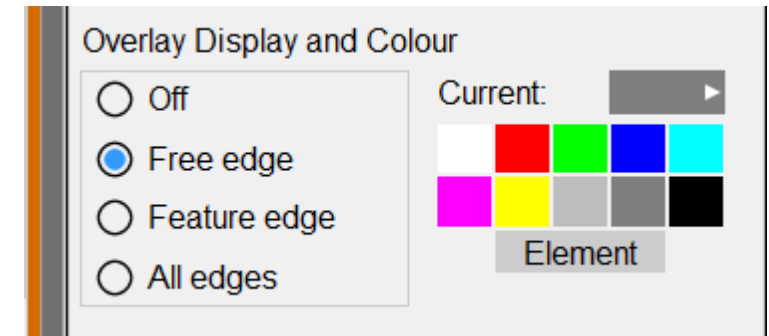


Colour

User colours available

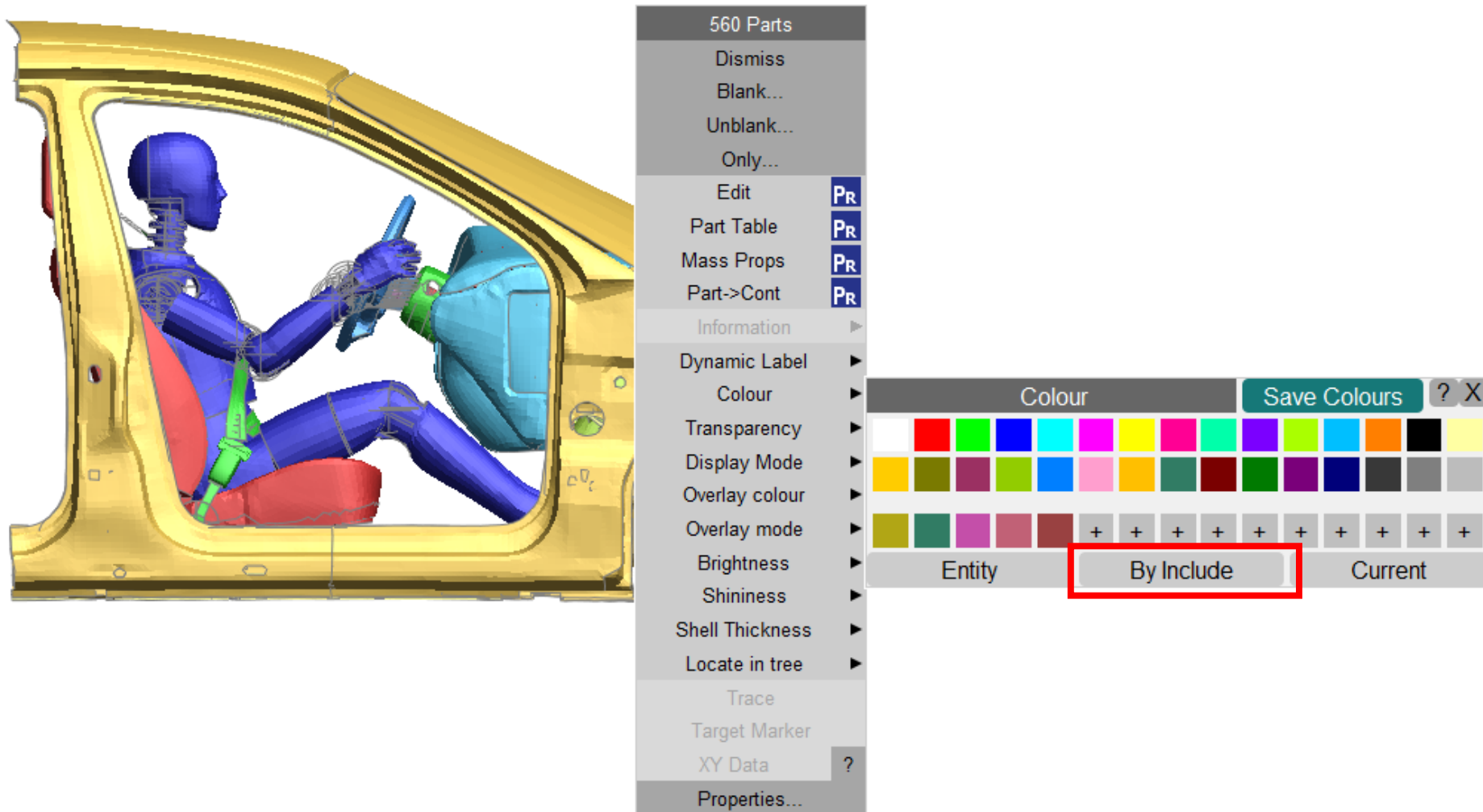
The full 30 core colours and any user-defined colours can now be accessed in the following menus:

- Overlay current colour
- Airbag particle colour
- Hidden colours for PR, LC and Vector plots
- Undeformed geometry colour
- Edit Window model colour
- External data blob plot symbols
- Streamline group colour
- Watermark colour



Colour by Include

Under quick pick, there is the option to colour entities by their include file.



Export Geometry to PRIMER

Export Geometry to PRIMER

- The **Write** → **Keyword data** panel in D3PLOT now has an option to export deformed geometry – along with initial stresses – directly to PRIMER.
- This is useful for applying information from the end of one simulation to an existing keyword model, for further analysis.

Write KEYWORD data

Apply Selected 2817 Parts

Parts Beams Shells

Thick shells Solids Nodes

Ztf file present

☐ Export to a keyword file

☒ Export deformed geometry to PRIMER

File : egration\Example_model\d3plot027.key

Data component

☒ Nodal coordinates

☐ Write constraints

☒ Apply deform/Fixed node adjustments

☐ Include Deleted Elements

☐ Elements topology

☐ Shell thickness

☐ Initial stresses

☐ Extra history variables All

☒ Use shell int pts info from ztf file

☐ Use user defined coords 3

☒ All Hughes-Liu Beams

☐ Resultant beams present

☐ Initial strain

☐ Initial nodal velocity

Done

D3PLOT

Export Geometry to PRIMER

- Select **Export deformed geometry to PRIMER**. Select the required options for the data that needs to be exported (nodal coordinates, initial stresses, initial strains, etc) from the panel and click **Apply**.
- A PRIMER integrated session will be opened containing the corresponding keyword file that relates to the results.
- The data will be sent to PRIMER and the “Node import” panel will be displayed. Click **Apply** in PRIMER to import the data.

Write KEYWORD data

Apply Selected 2817 Parts

Parts Beams Shells
Thick shells Solids Nodes

Ztf file present

☐ Export to a keyword file
☒ Export deformed geometry to PRIMER

File : egration\Example_model\d3plot027.key

Data component

☒ Nodal coordinates
☐ Write constraints
☒ Apply deform/Fixed node adjustments

☐ Include Deleted Elements
☐ Elements topology
☐ Shell thickness

☐ Initial stresses
☐ Extra history variables All

☒ Use shell int pts info from ztf file
☐ Use user defined coords 3

☒ All Hughes-Liu Beams
☐ Resultant beams present

☐ Initial strain
☐ Initial nodal velocity

Done

D3PLOT

Export Geometry to PRIMER

Write KEYWORD data

Apply Selected 785 Parts

Parts Beams Shells

Thick shells Solids Nodes

Ztf file present

☐ Export to a keyword file

☒ Export deformed geometry to PRIMER

File : ration\Int12\BLOCK_2\d3_tmp_019.key

Data component

☒ Nodal coordinates

☐ Write constraints

☐ Apply deform/Fixed node adjustments

☐ Include Deleted Elements

☐ Elements topology

☐ Shell thickness

☒ Initial stresses

☐ Extra history variables

☒ Use shell int pts info from ztf file

☐ Use user defined coords

☒ All Hughes-Liu Beams

☐ Resultant beams present

☐ Initial strain

☐ Initial nodal velocity

Done

D3PLOT



PRIMER

Node import

Help

Which data do you want to import (if present)?

Previous **Apply**

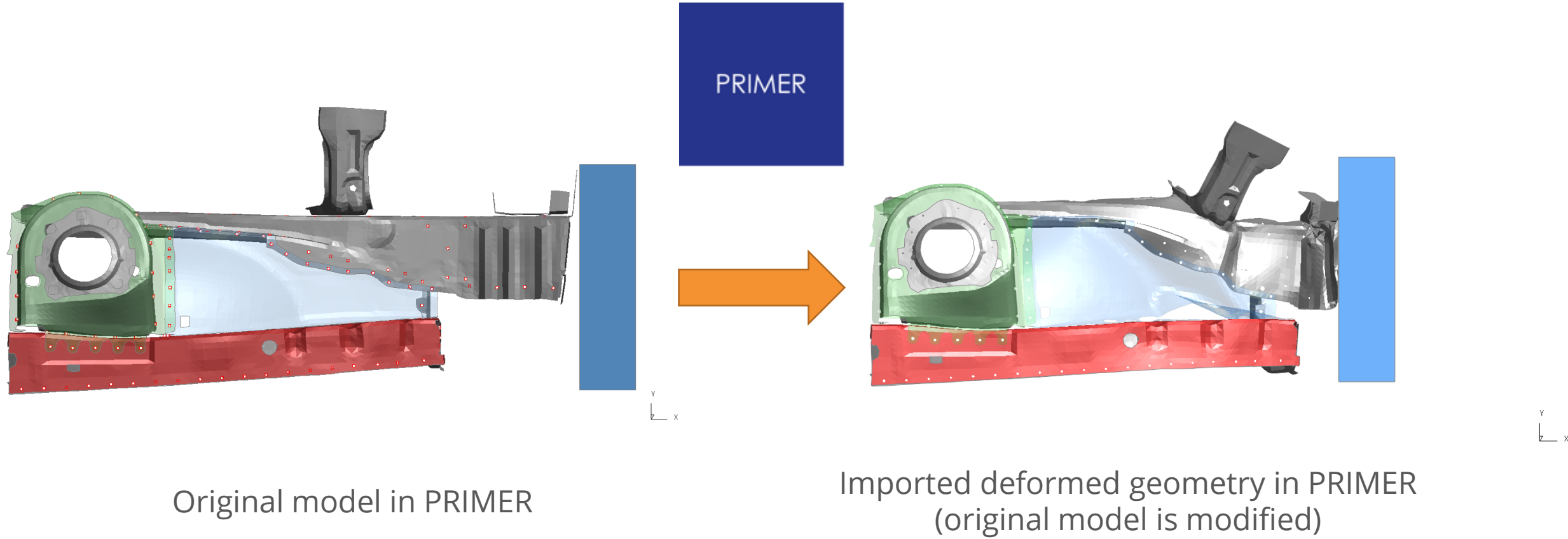
	Import	Delete
New nodal coordinates	<input checked="" type="checkbox"/>	
*INITIAL_STRESS_SOLID	<input checked="" type="checkbox"/>	<input type="checkbox"/>
*INITIAL_STRESS_SHELL	<input checked="" type="checkbox"/>	<input type="checkbox"/>
*INITIAL_STRESS_BEAM	<input checked="" type="checkbox"/>	<input type="checkbox"/>
*INITIAL_STRAIN_SOLID	<input type="checkbox"/>	<input type="checkbox"/>
*INITIAL_STRAIN_SHELL	<input type="checkbox"/>	<input type="checkbox"/>
*INITIAL_VELOCITY_NODE	<input type="checkbox"/>	

☐ (Re)create *INITIAL_FOAM_REF_GEOM

☐ Remove existing *INITIAL_FOAM_REF_GEOM

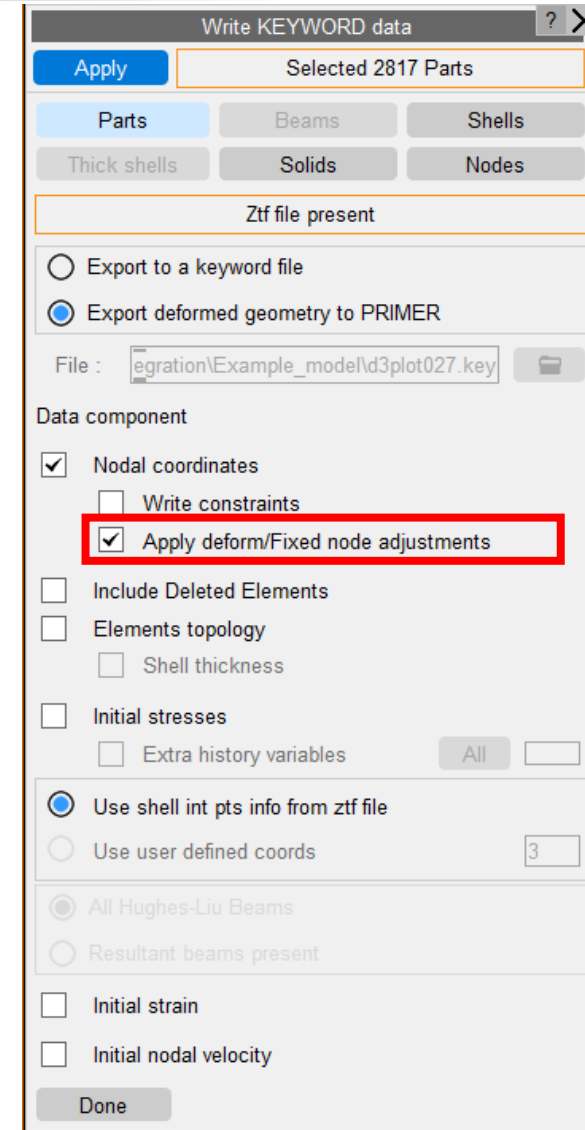
☐ Remove any IALEGP reference when importing *INITIAL_STRESS_SOLID card

Export Geometry to PRIMER



Export Geometry to PRIMER

- Relative nodal coordinates can also be exported to PRIMER.
- Go to **Deform** → **Fix node** or **Deform** → **Shift def** and turn on relative nodal coordinates using the **Fix node** or **Shift deform** methods.
- Then go to **Write** → **Keyword data** and tick “Apply deform/Fixed node adjustments”.
- Click **Apply** to export the relative nodal coordinates to PRIMER.



The screenshot shows the 'Write KEYWORD data' dialog box. At the top, there is a blue 'Apply' button and a text field showing 'Selected 2817 Parts'. Below this are several tabs: 'Parts' (selected), 'Beams', 'Shells', 'Thick shells', 'Solids', and 'Nodes'. A section labeled 'Ztf file present' contains two radio buttons: 'Export to a keyword file' and 'Export deformed geometry to PRIMER' (which is selected). Below this is a 'File' field with the path 'egration\Example_model\d3plot027.key'. The 'Data component' section contains several checkboxes: 'Nodal coordinates' (checked), 'Write constraints' (unchecked), 'Apply deform/Fixed node adjustments' (checked and highlighted with a red box), 'Include Deleted Elements' (unchecked), 'Elements topology' (unchecked), 'Shell thickness' (unchecked), 'Initial stresses' (unchecked), and 'Extra history variables' (unchecked). There is also a 'Use shell int pts info from ztf file' radio button (selected) and a 'Use user defined coords' radio button (unchecked) with a value of '3' in a text field. At the bottom, there are radio buttons for 'All Hughes-Liu Beams' (selected) and 'Resultant beams present' (unchecked), and checkboxes for 'Initial strain' and 'Initial nodal velocity'. A 'Done' button is at the very bottom.

Oasys
D3PLOT

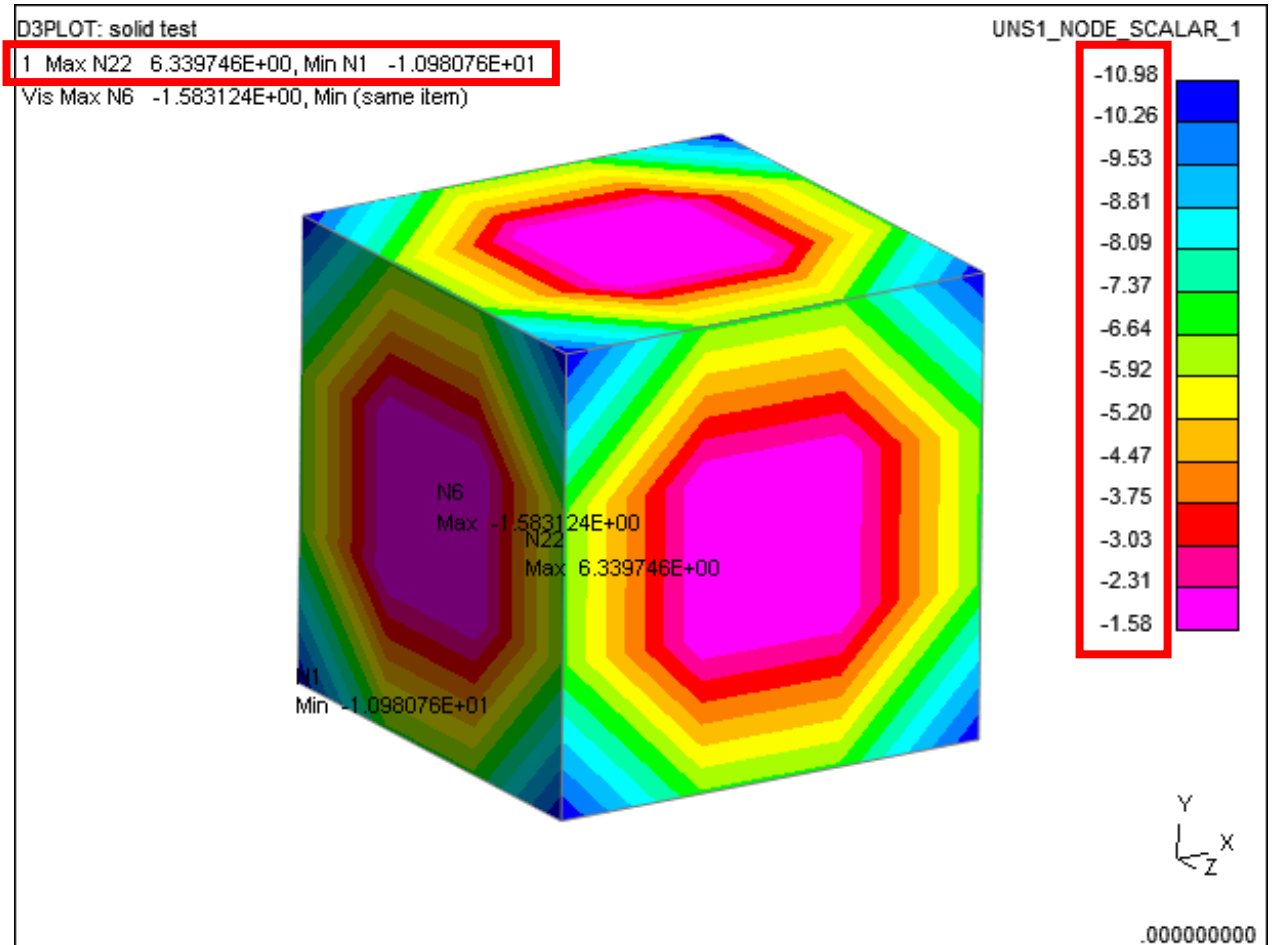
Contour Bar Limits

Contour Bar Limits

By default, D3PLOT only uses the values on the visible faces of 3D elements when using AUTO or AUTO EACH to calculate the contour bar limits for SI, CT or LC plots.

If an internal face has a higher or lower value then it is reported in the top left of the window but the value is not used for the contour bar range.

(If internal faces are turned on then the contour bar includes the values)



In version 18.0, a new option has been added to the data menu that makes D3PLOT use the internal face values when calculating the contour bar range without having to turn on internal faces.

D3PLOT: solid test

1 Max N22 6.339746E+00, Min N1 -1.098076E+01

Vis Max N6 -1.583124E+00, Min (same item)

N1
Min -1.098076E+01

N22
Max 6.339746E+00

N6
Max -1.583124E+00

Y
X
Z

.000000000

Contour Bar Limits

By default, this new option is not set and only visible faces are used, so version 18.0 of D3PLOT produces the same plots as previous versions.

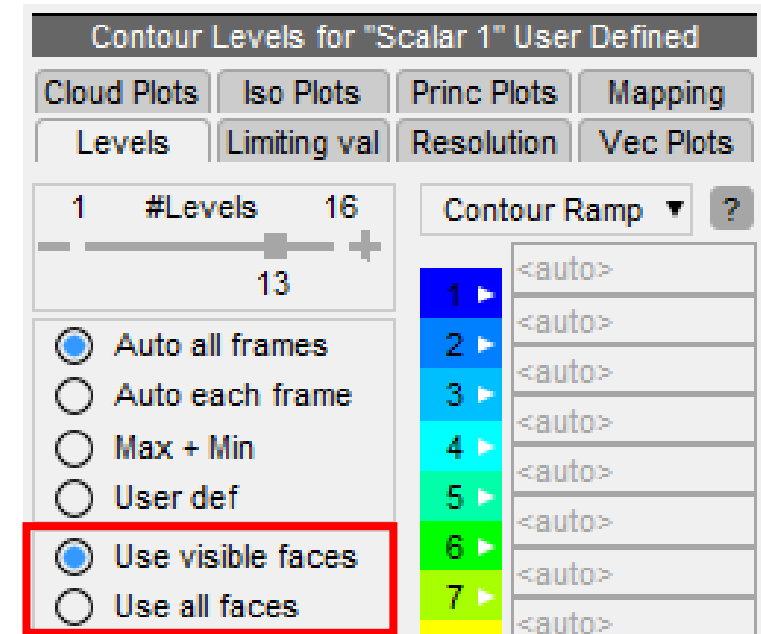
A new preference option has also been added that can be used to change the default behaviour

d3plot*contour_bar_3d_faces

This preference has 2 valid options

VISIBLE *(the default)*

ALL



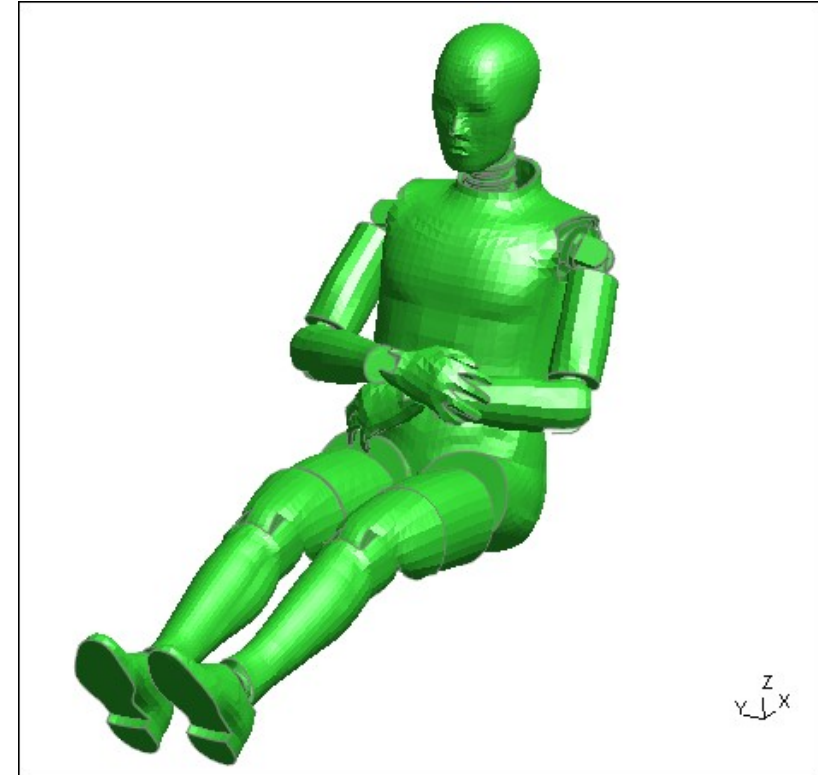
Blanking Contact Segments

Blanking Contact Segments

In LS-DYNA it is very common to define a single contact surface across multiple parts.



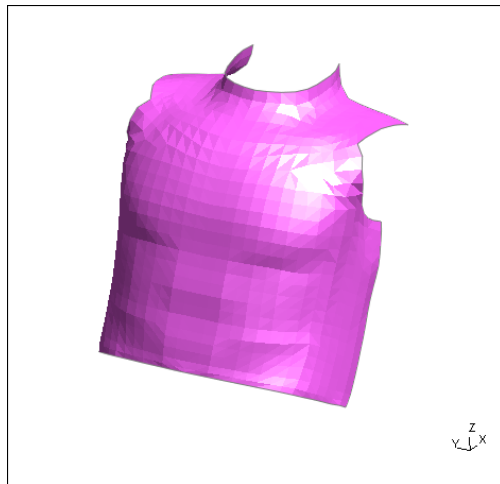
Parts



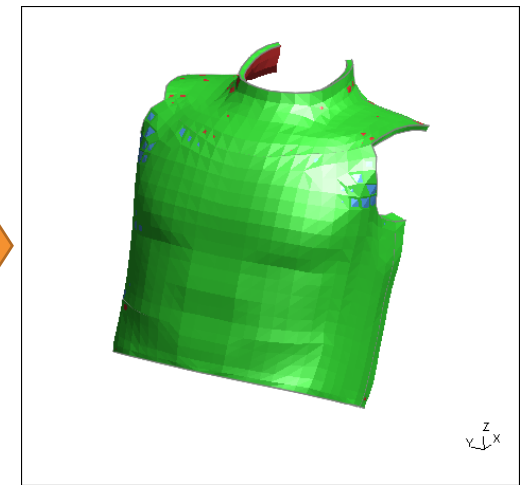
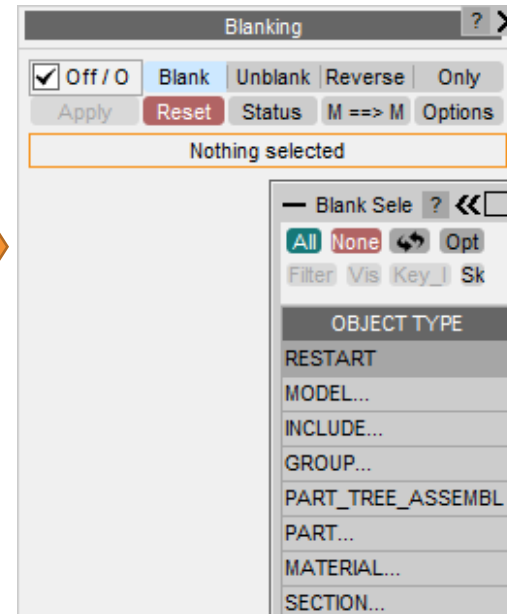
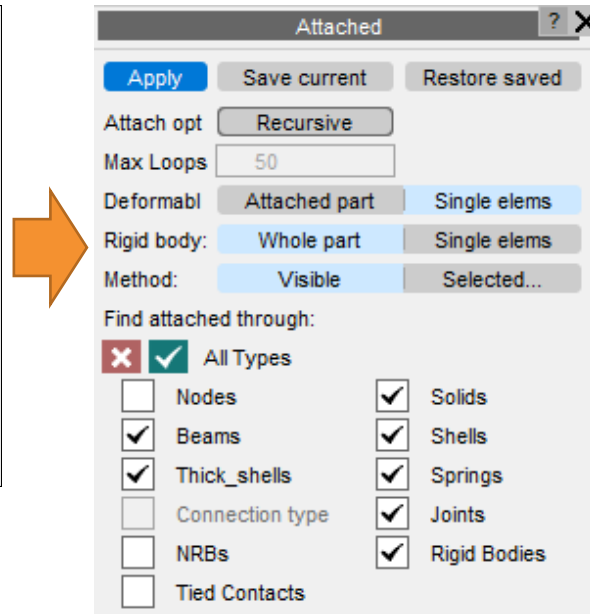
Contact Surface

Blanking Contact Segments

In previous versions of D3PLOT it was possible to view just the contact segments on a subset of parts using a combination of Attached and Blanking.

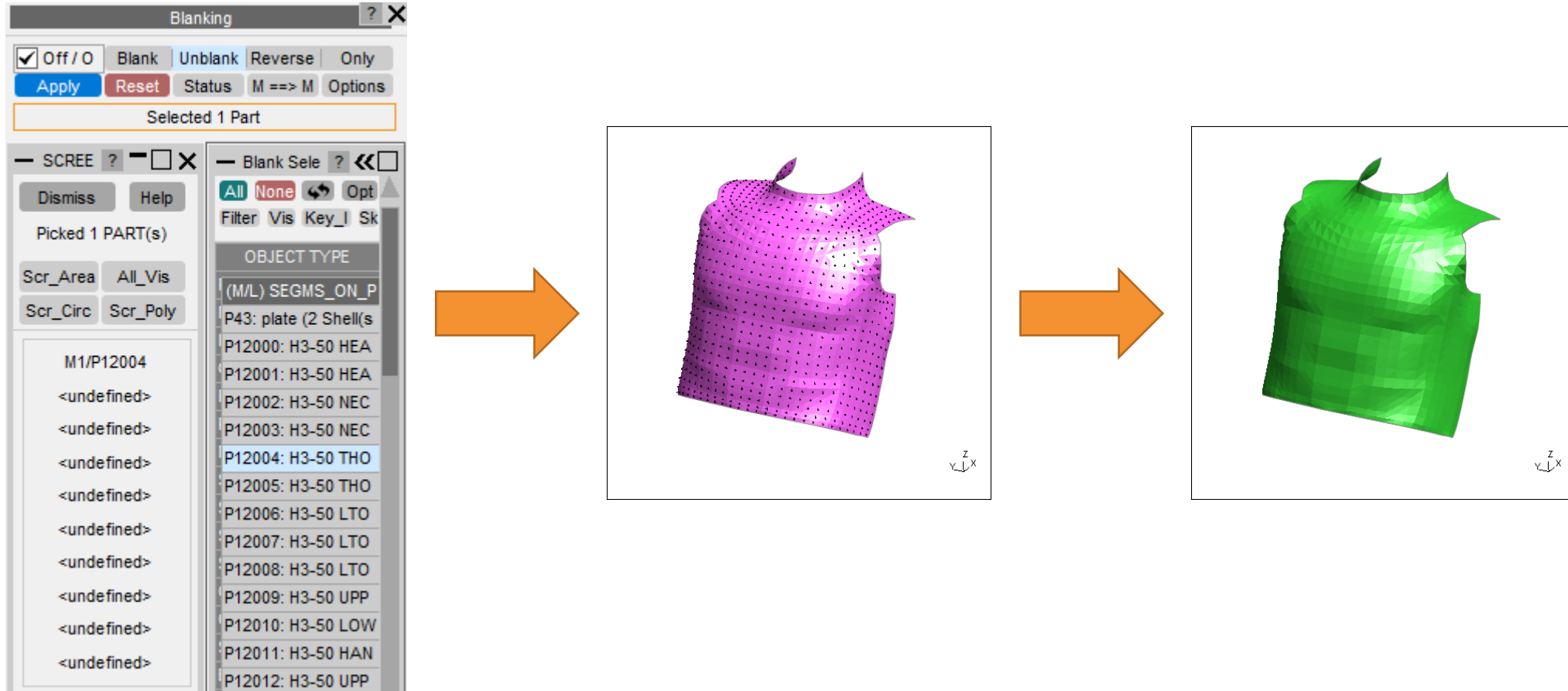


Unblank Part



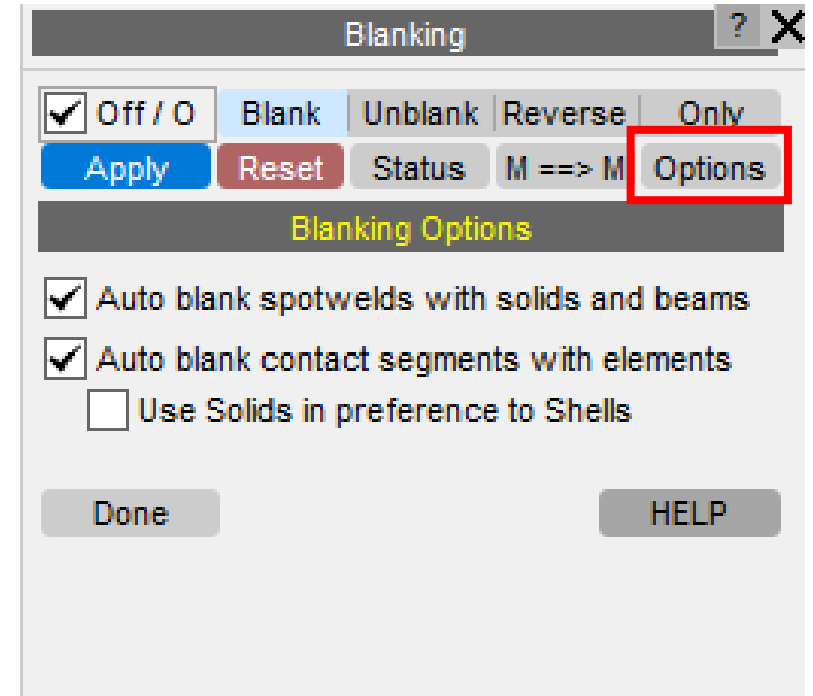
Blanking Contact Segments

In D3PLOT 18.0, the blanking menu has a new option “SEGMS_ON_PART” which can be used to directly unblank all of the contact segments on a Part.



Blanking Contact Segments

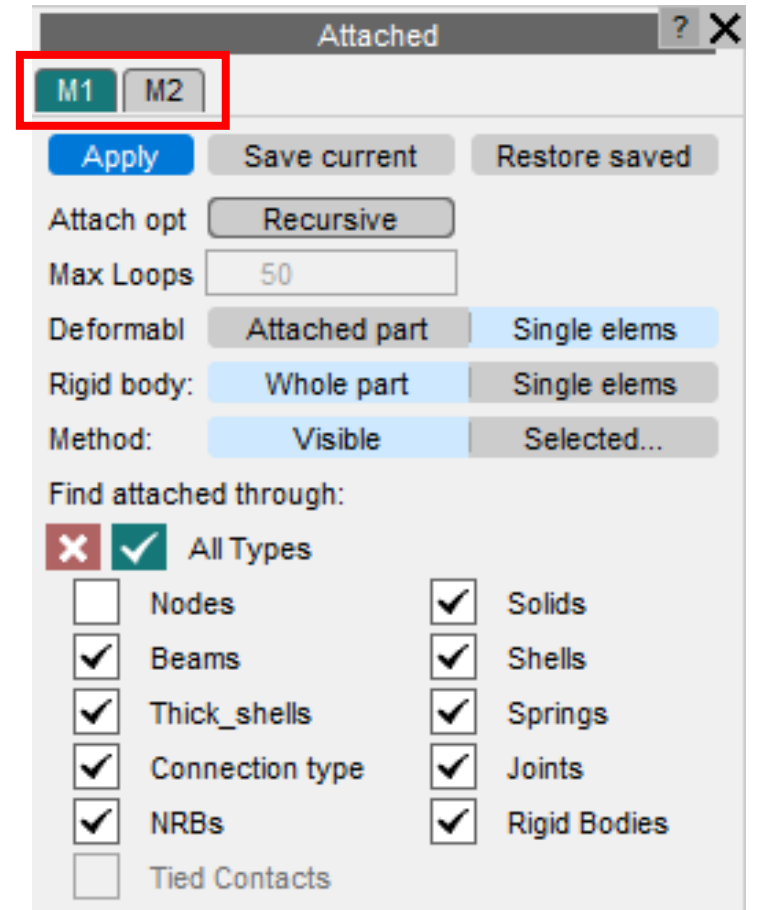
- By default, in version 18.0, contact segments are automatically blanked and unblanked when their parent element is blanked/unblanked.
This behaviour can be turned off via the new Options panel in the Blanking menu.
- If a contact segment is located on top of two or more coincident shells then the contact segment's parent is the shell with the lowest ID.
- If a contact segment is located on a shell that coats the face of a solid then by default the contact segment's parent is the shell but this can be changed if required.



Attached

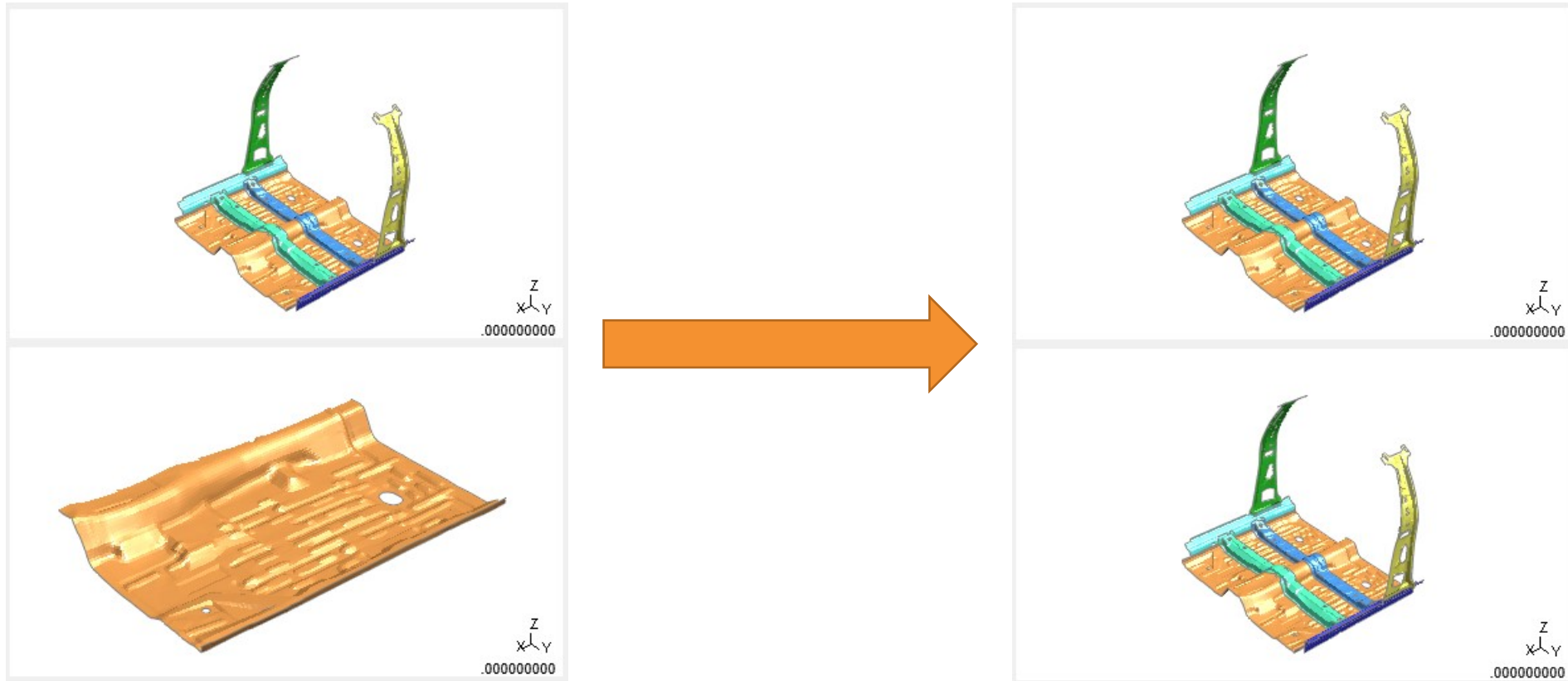
Attached

In previous versions of D3PLOT the Attached menu worked on a single model which was selected using the Model Tabs at the top of the menu.



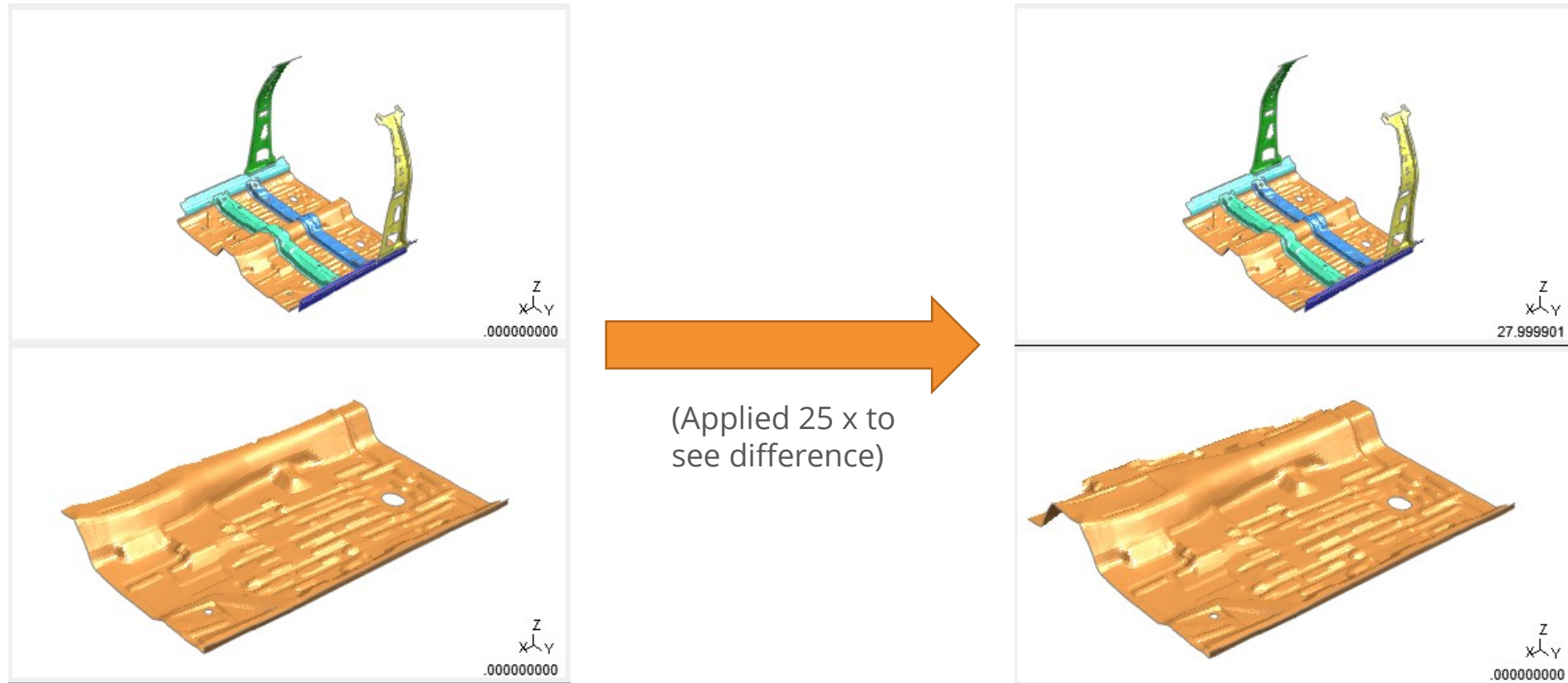
Attached

If the selected model was loaded in more than one window then Attached always used the items in the first window as the source and items were unblanked in all windows.



Attached

In D3PLOT 18.0, Attached has become a per-window operation and now applies the attached operation to each selected window independently.

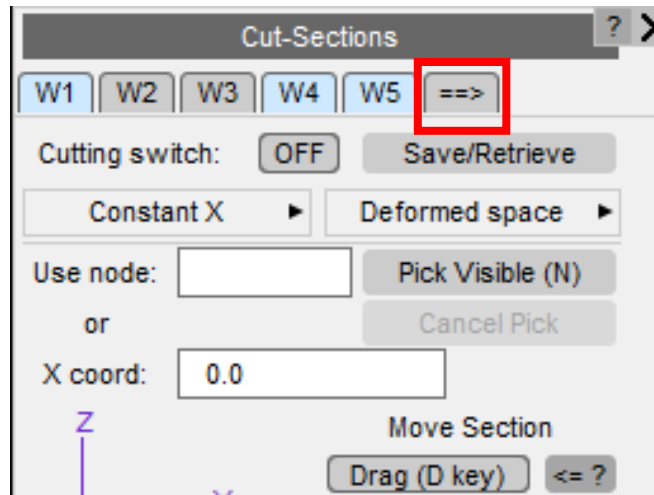


Copying Settings between Windows and Models


Copying Settings Between Windows

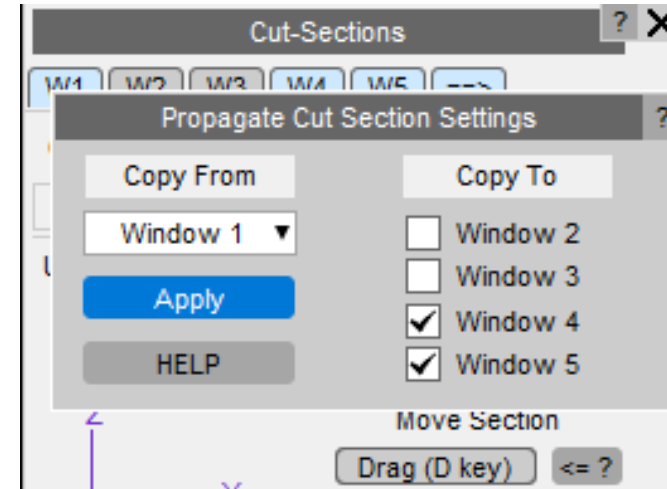
- When multiple windows are being used in D3PLOT, many menus contain an additional tab at the top of the menu that can be used to copy settings between Windows.
- In previous versions of D3PLOT, this option would always copy settings from the first active window to all the other active windows.

So in this example it would copy from W1 to W4 and W5:



Copying Settings Between Windows

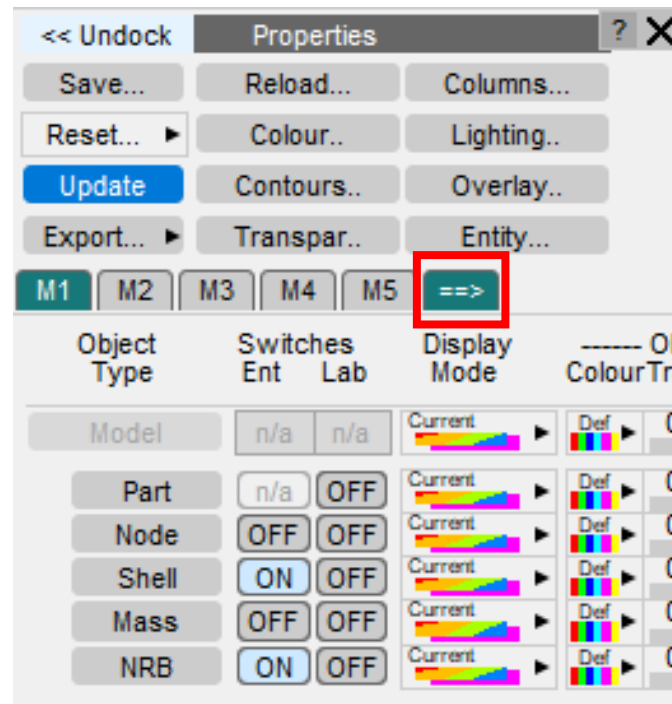
- From D3PLOT 18.0 onwards, the  tab now displays a new “Propagate” menu that allows the user to select which window to *copy from* and the windows to *copy to*.
- For backwards compatibility with previous versions then by default the first active window is selected as the window to copy from and all the other active windows are selected as the windows to copy to.




Copying Settings Between Models

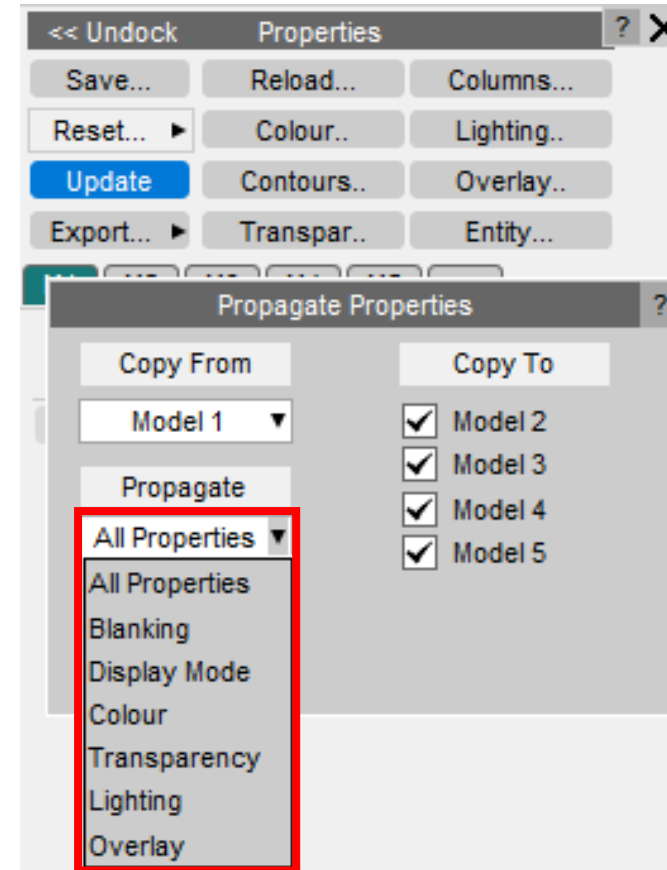
- Some menus in D3PLOT apply settings to models not windows. In previous versions of D3PLOT the additional tab in these “Model” based menus would always copy settings from the active model to all other models.

So in this example it would copy from M1 to M2, M3, M4 and M5:



Copying Settings Between Models

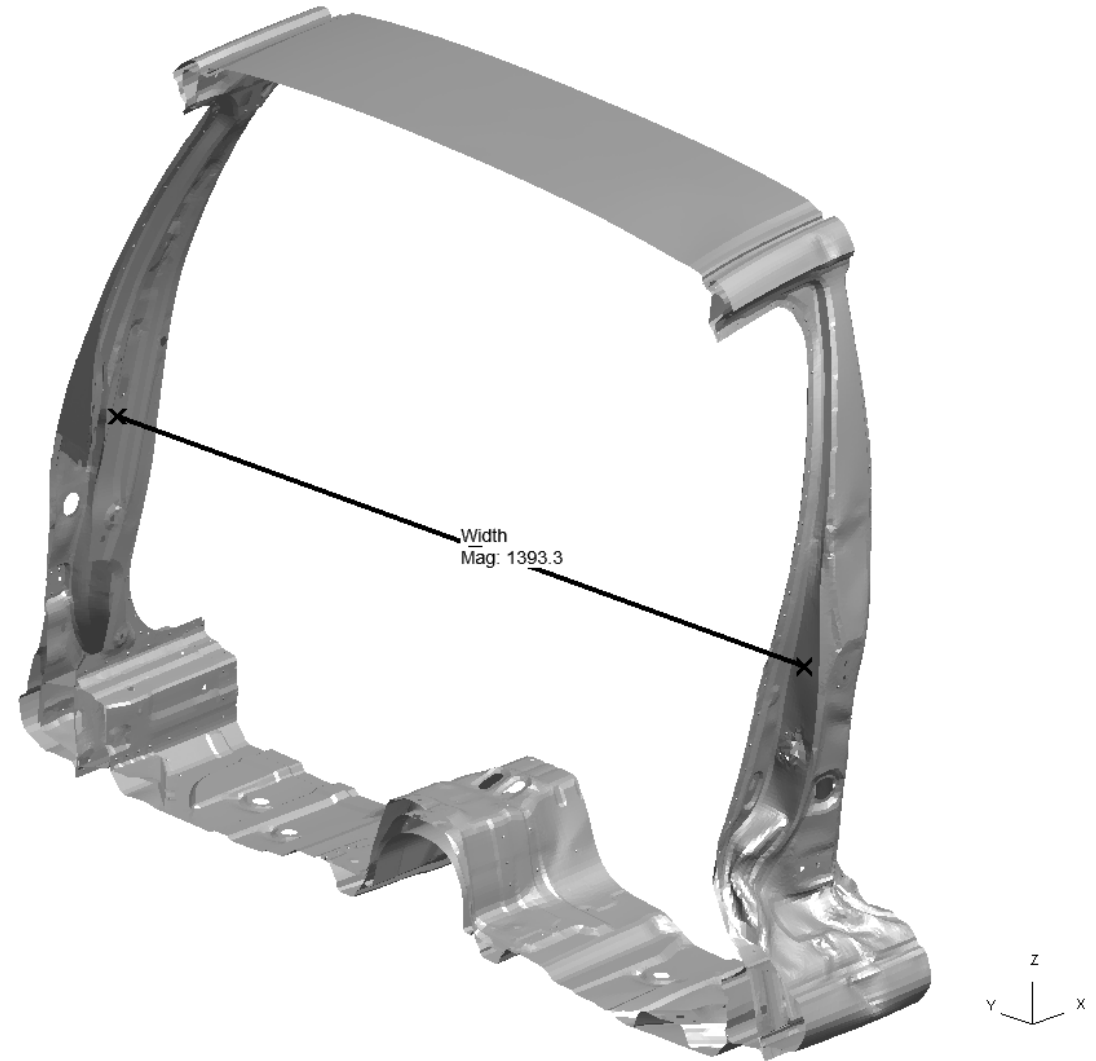
- From D3PLOT 18.0 onwards, the  tab for Models works just like the Window one and allows you to select both the model to *copy from* and the Models to *copy to*.
- In the Properties menu the new “Propagate” menu has an additional, optional selection, because often you only want to copy a subset of the properties between models.



JavaScript API

Measure Class

A new Measure class has been added to the D3PLOT JavaScript API. It allows you to create measurements and to draw them in the graphics windows with similar options to those available on the interactive Measure panel.



T/HIS Link

The following functions have been added to the D3PLOT JavaScript API to control the T/HIS link:

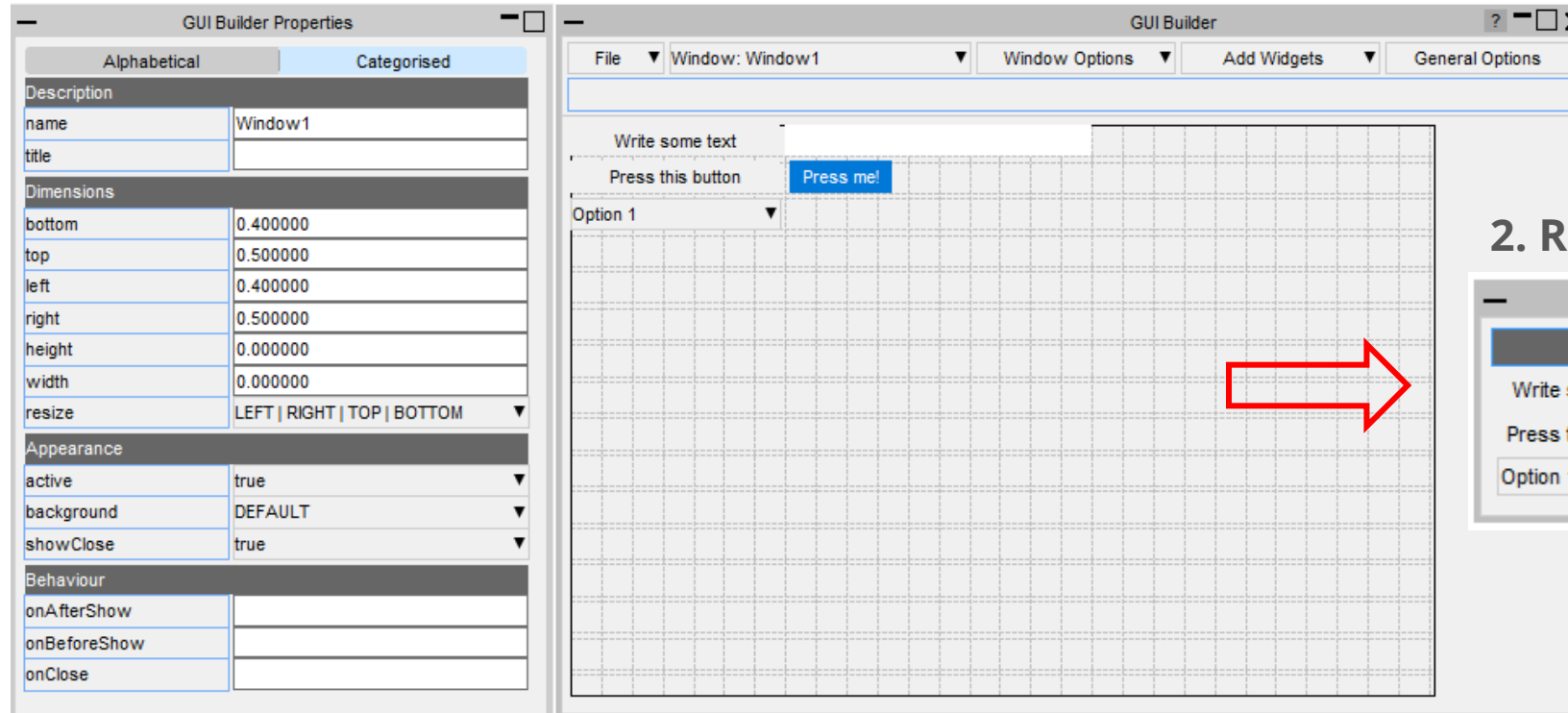
Function	Description
StartThisLink()	Starts the T/HIS link from within a JavaScript.
ExitThisLink()	Stops the T/HIS link from within a JavaScript.

JavaScript GUI Builder

JavaScript GUI Builder

An interactive GUI Builder has been added to PRIMER, D3PLOT and T/HIS to make it easier to build JavaScript GUIs, removing the need to write code to create windows and widgets.

1. Design and Save your GUI to a file

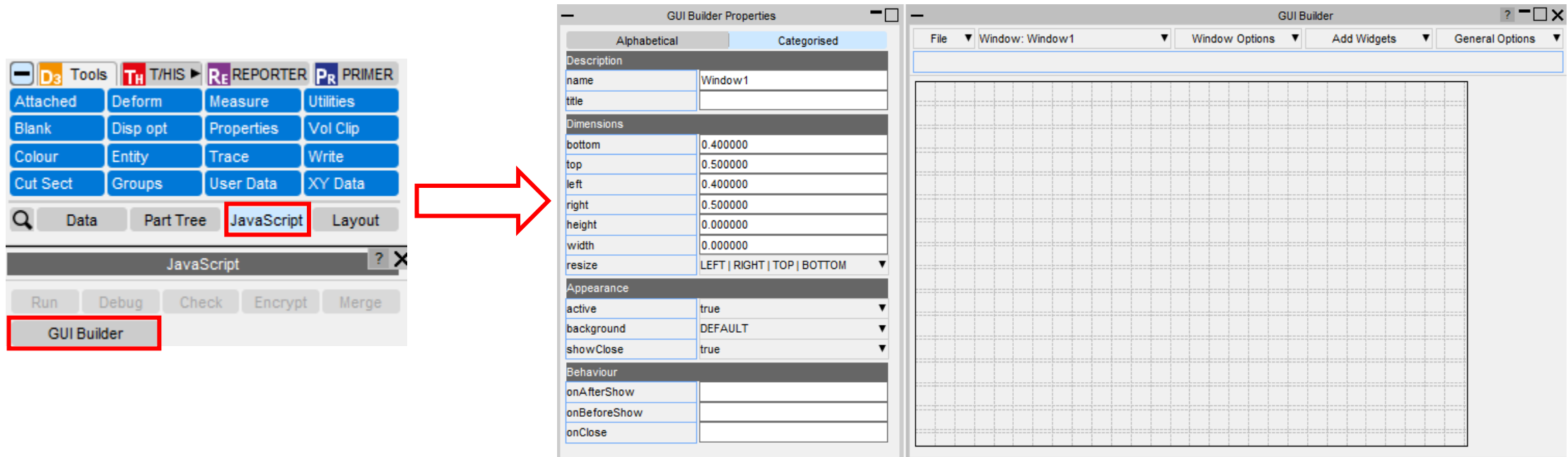


2. Read the file in your script



JavaScript GUI Builder

To open the GUI Builder in D3PLOT go to JavaScript → *GUI Builder*



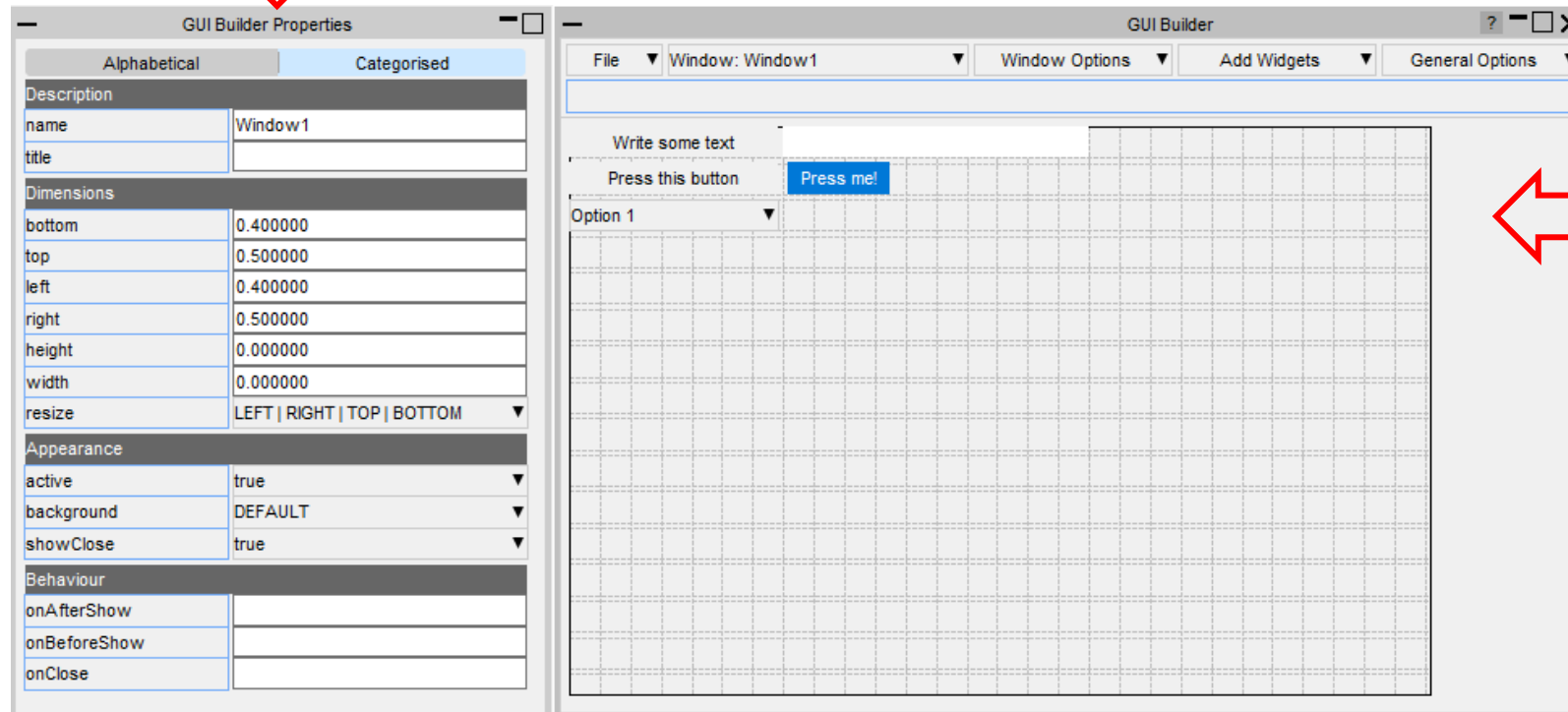
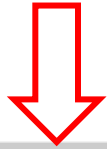
JavaScript GUI Builder

How to use the GUI Builder to build a GUI

JavaScript GUI Builder

Properties Window

The properties of widgets and windows are set here.



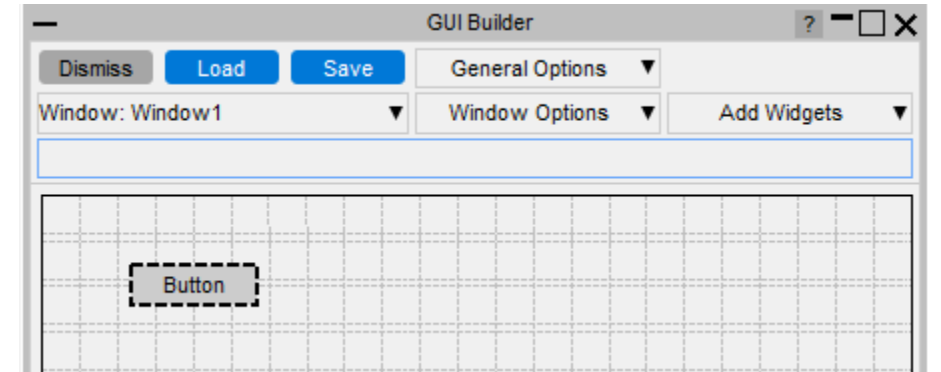
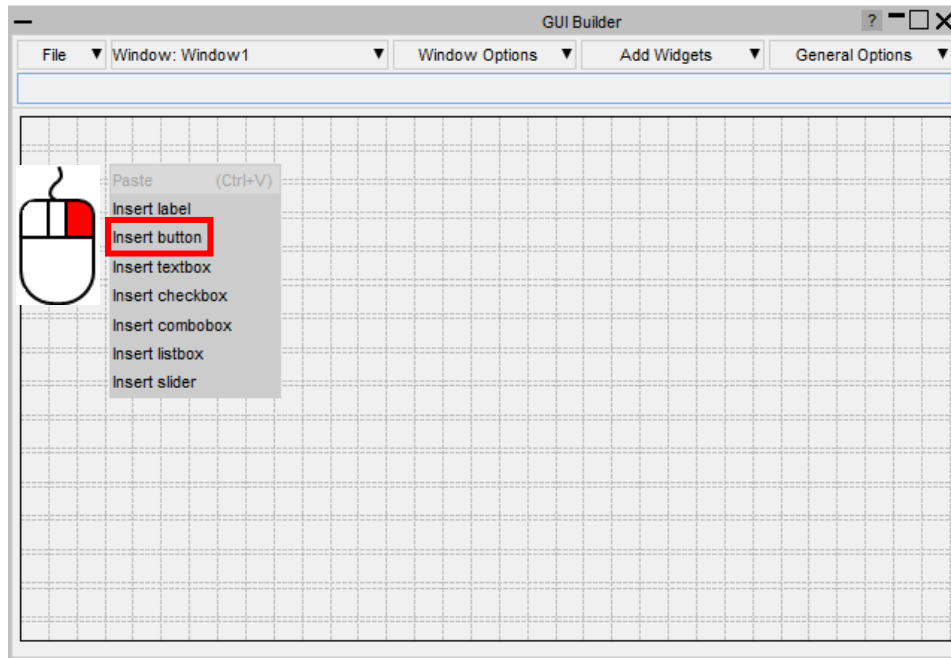
Design Window

Widgets are added, positioned and resized here.



JavaScript GUI Builder

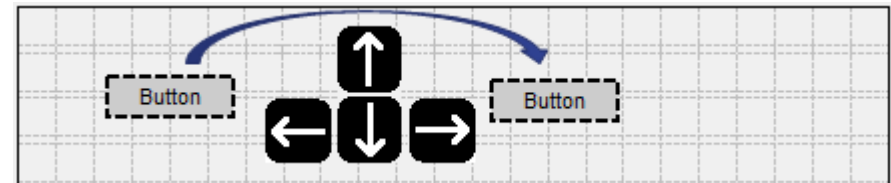
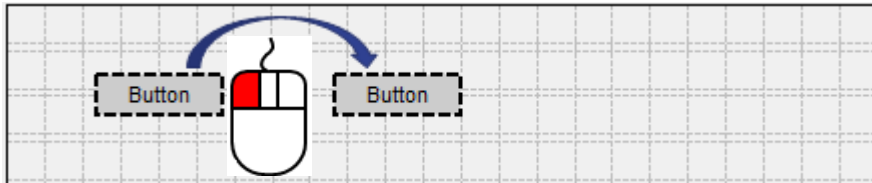
Widgets can be added by right-clicking on the design window and selecting the widget type to add.



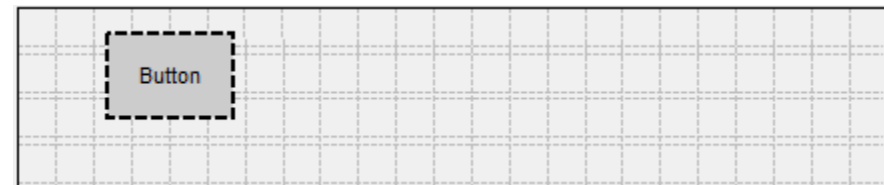
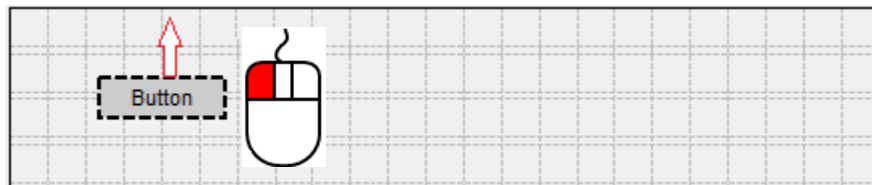
The widget will be added with default properties and highlighted with dashed lines to indicate that it's the current widget.

JavaScript GUI Builder

Widgets can be moved by left-clicking on them and dragging, or by using arrow keys

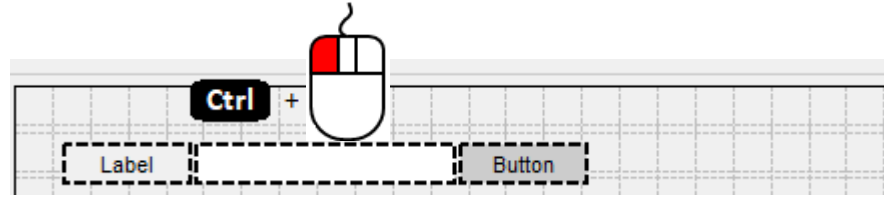


They can be resized by left-clicking on their border and dragging

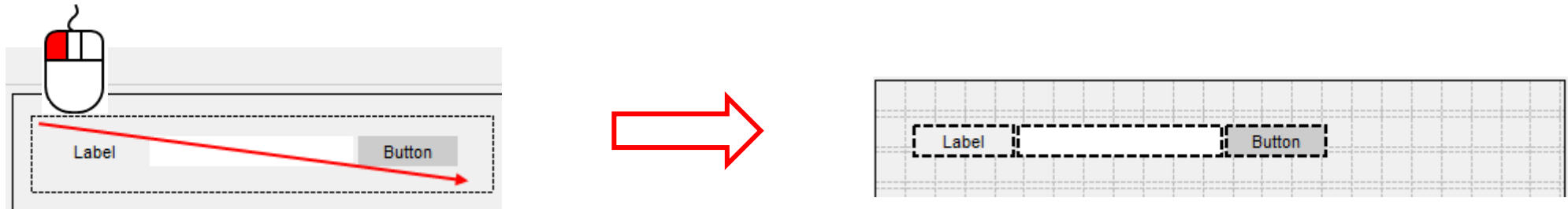


JavaScript GUI Builder

Multiple widgets can be selected by holding the Ctrl or Shift keys and left-clicking

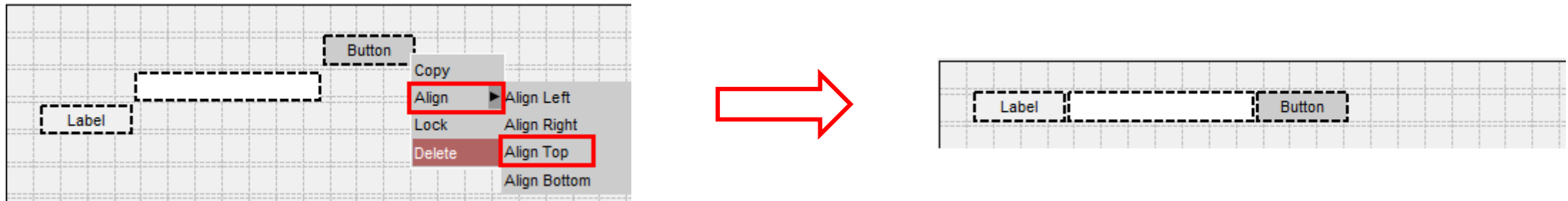


Alternatively a box can be dragged around the widgets you want to select



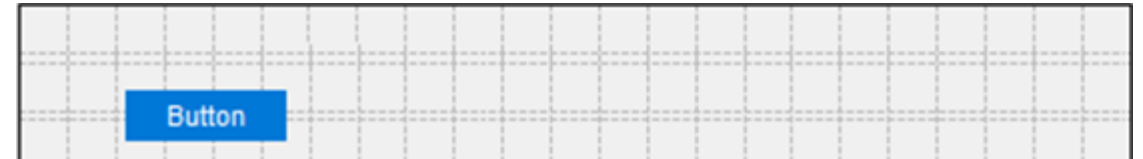
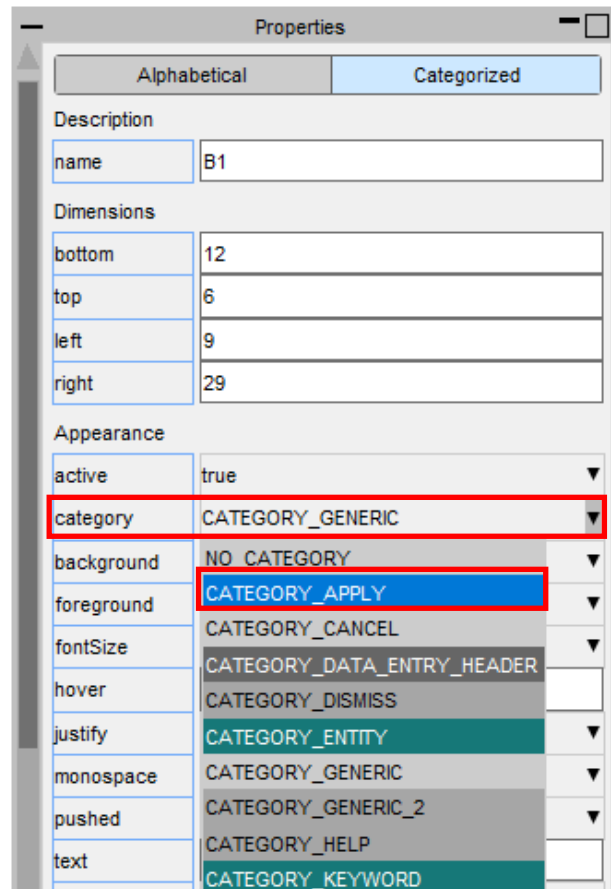
JavaScript GUI Builder

When multiple widgets are selected the borders can be aligned by right-clicking on the widget you want to align the other widgets to, and then selecting how you want them to be aligned:



JavaScript GUI Builder

The properties of a widget can be modified in the properties window, e.g. change the category to CATEGORY_APPLY:



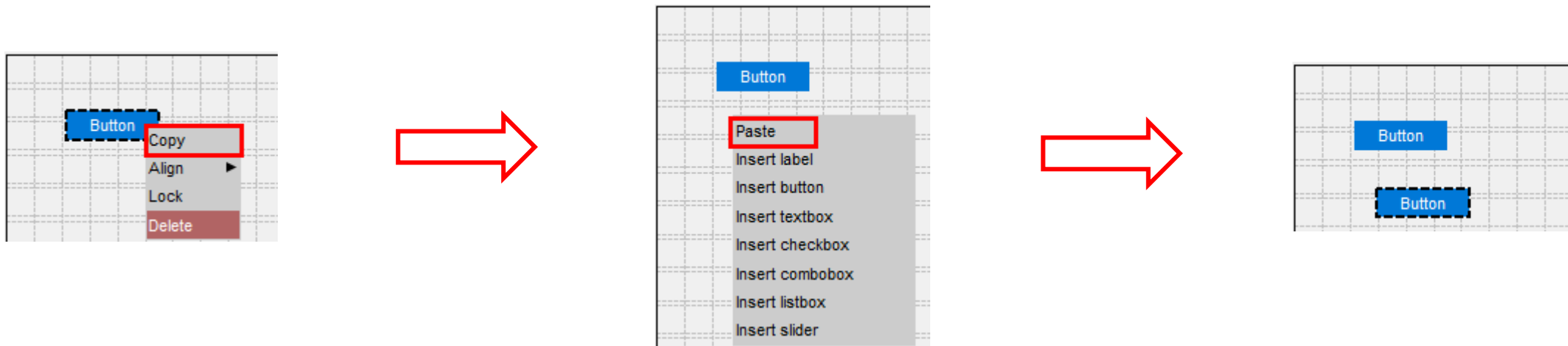
The appearance of the widget will update in the design window.

If multiple widgets are selected the property will be applied to all the selected widgets

JavaScript GUI Builder

You can copy and paste widgets by right-clicking on them and selecting 'Copy' and then right-clicking on the window and selecting 'Paste'. The new widget will have all the same properties as the copied widget.

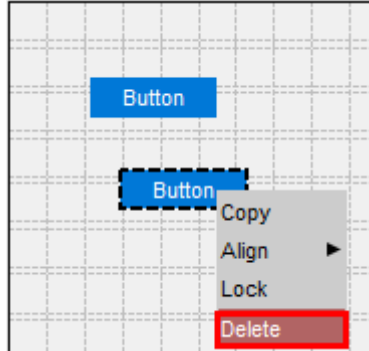
Alternatively you can use the shortcuts Ctrl-C and Ctrl-V.



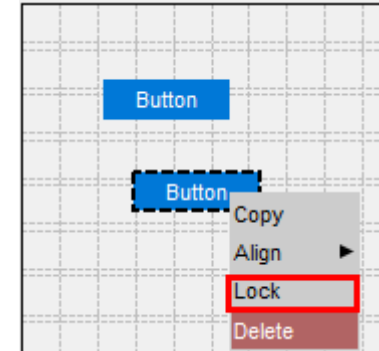
JavaScript GUI Builder

To delete a widget, right-click on it and select 'Delete'.

Alternatively you can press the Delete shortcut key.



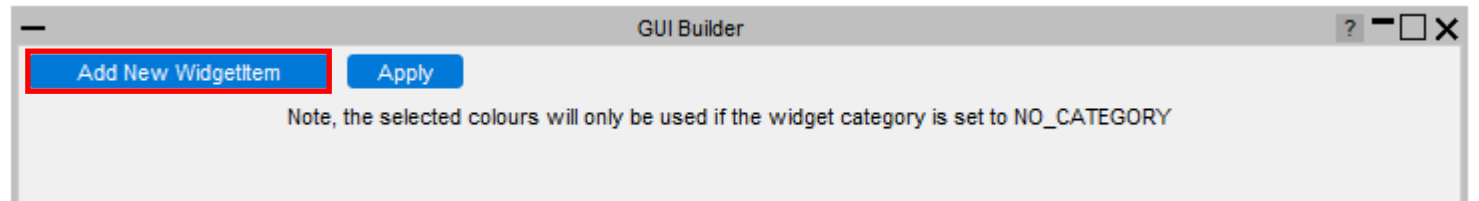
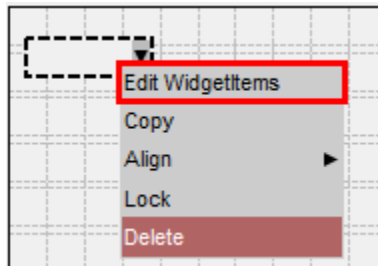
To lock the position of a widget so it can't be repositioned or resized, right-click on it and select 'Lock'. To unlock it again, right-click on it and select 'Unlock'.



JavaScript GUI Builder

To add WidgetItems to a Combobox or Listbox, right-click on it and select 'Edit WidgetItems'.

This will update the design window where you can add WidgetItems by pressing the 'Add New WidgetItem' button.

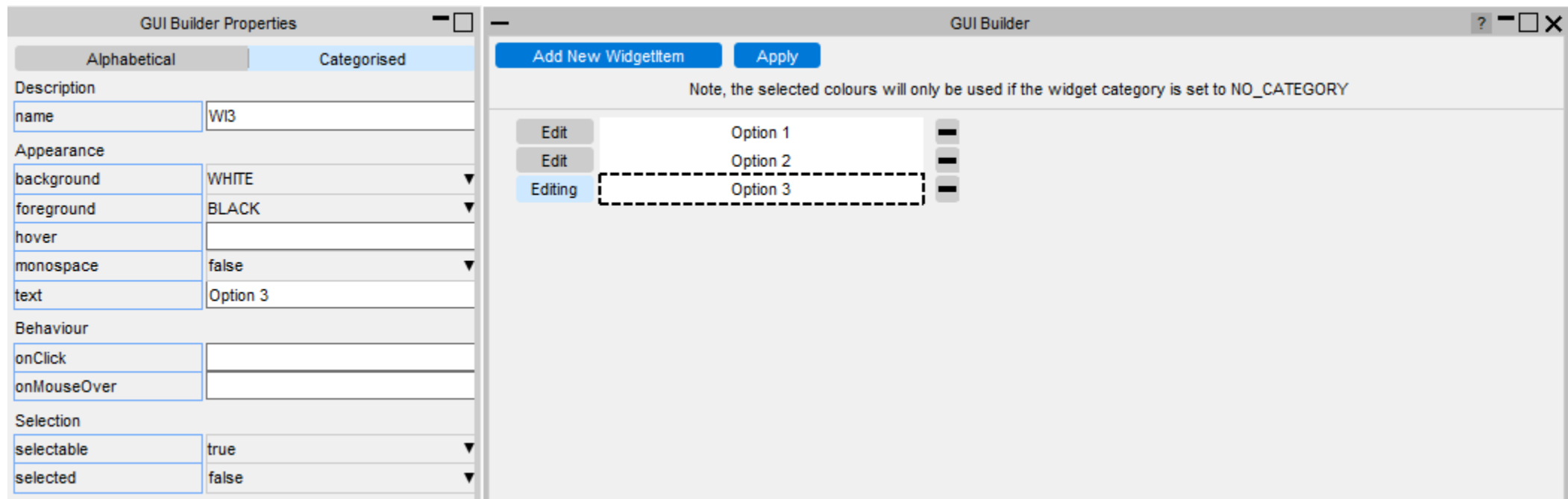


JavaScript GUI Builder

The appearance of the current WidgetItem can be modified in the same way as Widgets by clicking on the WidgetItem and updating its properties.

To delete a WidgetItem, click on the '-' on the right hand side.

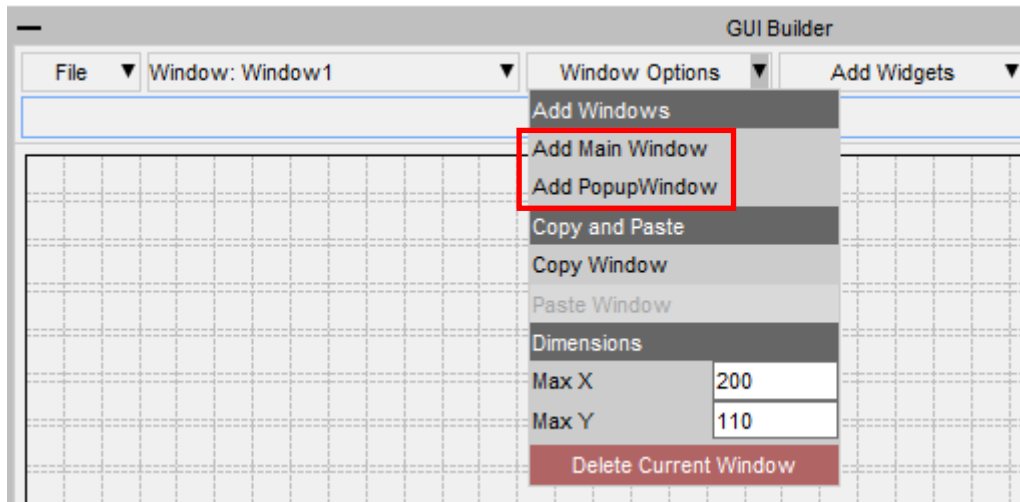
Once you have finished, press 'Apply' to return to the normal design window.



JavaScript GUI Builder

Additional windows can be created by clicking on the Window Options dropdown menu.

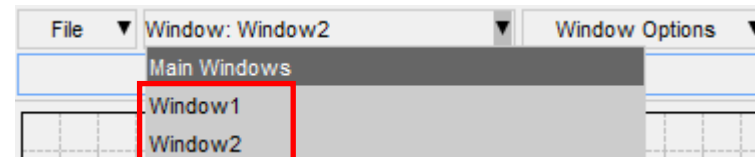
You can add either a Main Window or PopupWindow.



The name of the current window is displayed in the Window selection dropdown menu

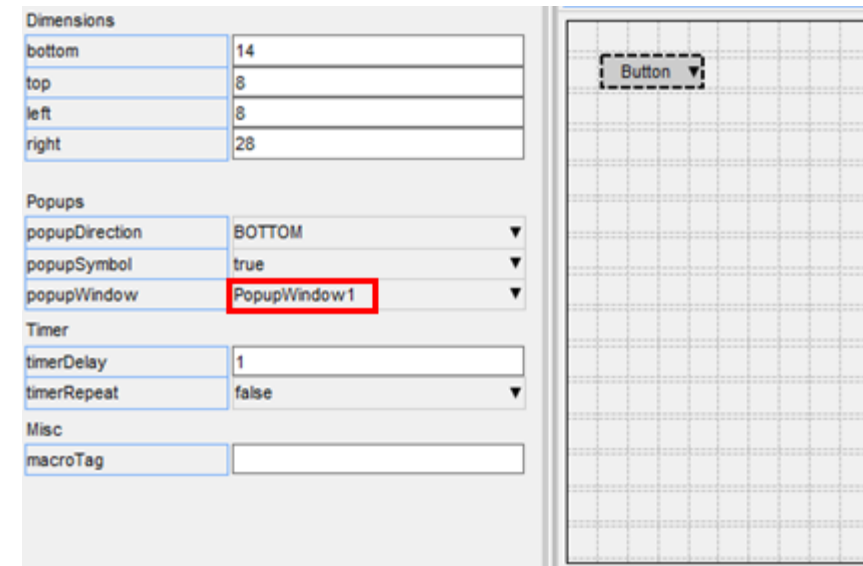
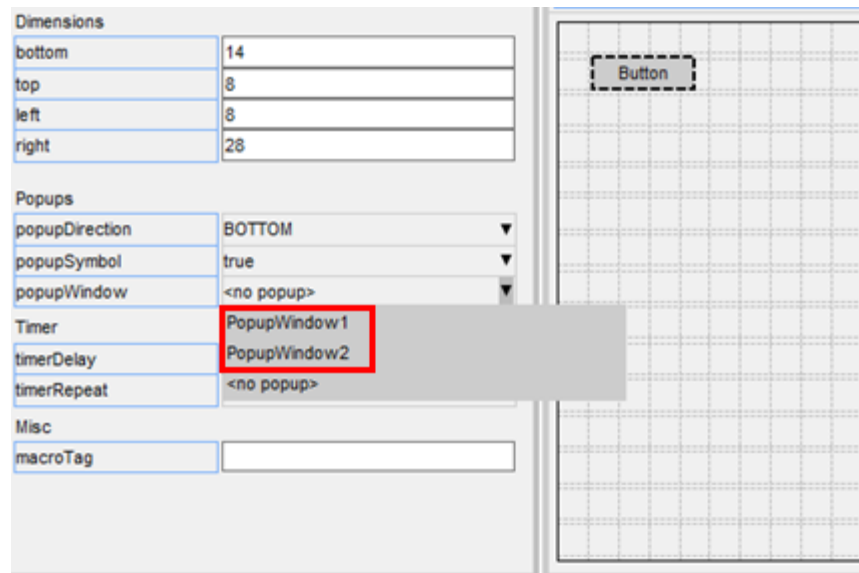


To change to a different window, select it from the dropdown menu



JavaScript GUI Builder

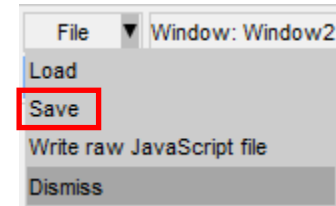
PopupWindows can be linked to widgets by setting the popupWindow property.



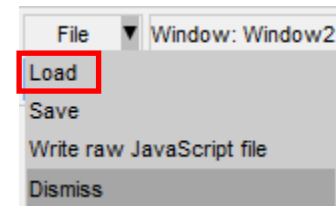
To remove a PopupWindow linked to a widget, set the popupWindow to <no popup>

JavaScript GUI Builder

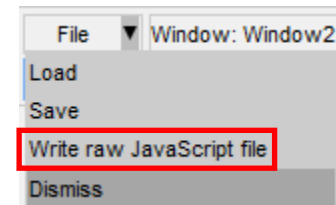
The GUI can be saved to file by pressing the 'Save' button and then selecting a file. The saved file is a JavaScript file containing the window and widget definitions in a JSON string, and a call to `Window.BuildGUIFromString()` which builds the GUI when the script is run. Further details are given in the next few slides.



It can be reloaded by pressing the 'Load' button and selecting the file to load.



The GUI can also be saved as a raw JavaScript file, with the calls to create and position the windows and widgets, explicitly defined, rather than using `Window.BuildGUIFromString()`. This cannot be loaded back into the GUI Builder, however it may be useful for creating GUI's to run in versions prior to v18 that don't have the `Window.BuildGUIFromString()` function.



JavaScript GUI Builder

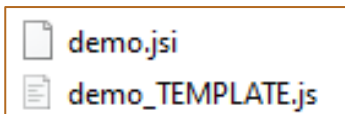
How to use the GUI in a script

JavaScript GUI Builder

The GUI is saved to a JavaScript file, containing the GUI definition in a JSON string and a call to `Window.BuildGUIFromString()`. It is saved with the extension `'.jsi'` to indicate that it should be included from another file. You should not need to edit this file.

A `*.js` file is also written to demonstrate how to include the `*.jsi` file and display the GUI. This can be used as a template to follow and modify.

It is written to the same folder as the `*.jsi` file and named '`<jsi_filename>_TEMPLATE.js`', e.g. if the `*.jsi` file is called '**`demo.jsi`**', the `*.js` file will be saved as '**`demo_TEMPLATE.js`**'



The following slides explain what is in the file and how you can reference the Windows, Widgets and WidgetItems in the script.

JavaScript GUI Builder

To read the GUI in a script you need to include the file using the Use() function.

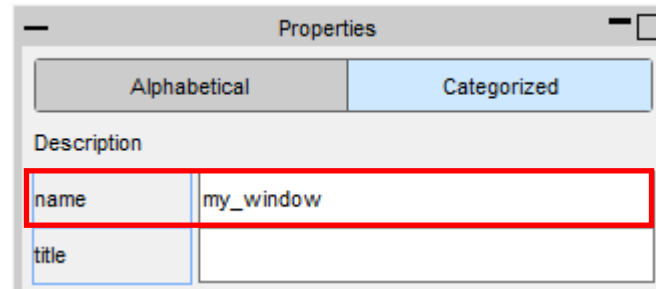
This will create a global variable ('gui' by default) containing all the GUI objects. The name of the variable can be changed in the GUI builder menu under General Options.

For example, to build the GUI saved in C:\my_gui.jsi:

```
Use("C:\\my_gui.jsi");
```

JavaScript GUI Builder

The GUI Window objects are stored as properties on the global object. The name of the property is whatever was defined in the properties window in the GUI builder:

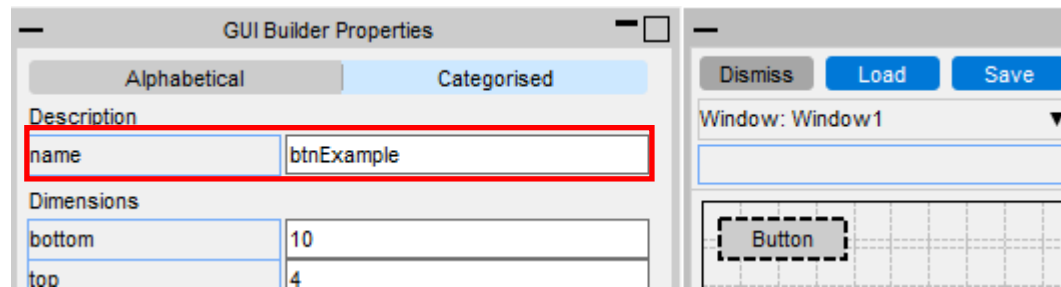


To display the Window called 'my_window' use the Show() method:

```
if (gui) gui.my_window.Show();
```


JavaScript GUI Builder

Similarly, each Widget object is a property of the Window object. The name of the Widget property is whatever was defined in the properties window in the GUI builder:



For example if the window is called 'my_window' and the widget is called 'btnExample', the Widget object can be accessed and modified with:

```
var btn = gui.my_window.btnExample;
```

```
btn.text = "Test";
```

JavaScript GUI Builder

WidgetItem objects are a property of the Widget.

For, example if the window is called 'my_window', the widget the widget item is on is called 'cbxExample' and the widget item is called 'wi1', it can be accessed and modified with:

```
var wi = gui.my_window.cbxExample.wi1;
```

JavaScript GUI Builder

Callback functions (onClick, onChange, etc.) can be assigned to the window and widgets in the properties window, by adding the name of a function to call.

For example to set the onClick property of a widget so it calls a function called 'pressed':

Functions	
onClick	pressed
onPopup	
onTimer	

This function then needs to be defined in your script:

```
Use("C:\\test.jsi");  
  
if (gui) gui.my_window.Show();  
  
function pressed()  
{  
    Message("You clicked me!");  
}
```

JavaScript Engine Upgrade

JavaScript engine upgrade

- For D3PLOT 18.0 the JavaScript engine used in D3PLOT has been significantly upgraded.
- In D3PLOT 17.0 and earlier the engine only supported [ECMAScript 5](#) features.
- In D3PLOT 18.0 the engine now supports [ECMAScript 6](#) (ES6) and many newer features.
 - The engine we use is [Spidermonkey](#) provided by Mozilla from the Firefox web browser.
 - For D3PLOT 18.0 we are now using the current 'Extended Support Release' version (ESR78)
 - Future releases will continue to use the latest ESR version available.

JavaScript engine upgrade

- The primary reason for upgrading is to give access to newer JavaScript features
- In some cases newer JavaScript code people obtained/learned from books and/or the web and tried to use in D3PLOT did not work in D3PLOT 17.0 as we only supported ECMAScript 5.
- Upgrading the engine allows the latest ECMAScript 6 (ES6) language features to be used.
 - Which ES6 (and newer) features are supported by the engine can be viewed at <http://kangax.github.io/compat-table/es6/#firefox78>
- Additional benefits to upgrading as well as ES6 support are outlined on the following slides.

JavaScript engine upgrade – ES6 features

- Upgrading the JavaScript engine gives access to lots of significant new ES6 (and newer) language features such as
 - [class](#) keyword
 - Block scope with [let/const](#)
 - [Promises](#)
 - [Arrow functions](#)
 - [Default parameters](#), [rest parameters](#) and [spread syntax](#)
 - [Set](#) and [Map](#)
 - [Iterators](#) and [generators](#)
 - [Symbol](#)

And many more

- A few examples follow but read a good book (e.g. JavaScript: The Definitive Guide) or look online for more details

JavaScript engine upgrade – example ES6 features

- class keyword
 - ES6 makes it much easier to create classes using the new class keyword and syntax

ES 5

```
function Circle(radius)
{
    this.r = radius;
}

Circle.prototype.area = function()
{
    return Math.PI * this.r * this.r;
}

var c = new Circle(5);
Message("Area of circle with radius " +
        c.r + " is " + c.area() )
```

ES 6

```
class Circle
{
    constructor(radius)
    {
        this.r = radius;
    }

    area()
    {
        return Math.PI * this.r * this.r;
    }
}

var c = new Circle(5);
Message("Area of circle with radius " +
        c.r + " is " + c.area() )
```


JavaScript engine upgrade – example ES6 features

- let statement
 - The let statement in ES6 allows you to create variables with block scope (variables declared with var have scope for the containing function which can be a source of bugs)
 - Accessing variables defined with let before they are initialised is an error (helps trap bugs)

ES 5

```
function test()
{
    Message(x);      // undefined

    var x = 1;
    {
        var x = 2;    // same variable!

        Message(x);  // 2
    }

    Message(x);      // 2
}
```

ES 6

```
function test()
{
    Message(x);      // Error

    let x = 1;
    {
        let x = 2;    // different variable

        Message(x);  // 2
    }

    Message(x);      // 1
}
```

JavaScript engine upgrade – example ES6 features

- Spread operator
 - The spread operator expands an array into the list of values in the array. It can be useful when array values are needed in a function.

ES 5

```
// Get the maximum value from data
var d = GetMultipleData(DM, NODE, 1, 1000);

var mval = -Number.MAX_VALUE;
for (i=0; i<d.nr; i++)
    mval = Math.max(mval, d.data[i])
// or
var mval = d.data.reduce(function(a,b)
                        { return Math.max(a,b); }
                        );

// draw a rectangle on a widget
var pt1 = [0, 0];
var pt2 = [100, 100];
widget.Rectangle(Widget.RED, true,
                pt1[0], pt1[1],
                pt2[0], pt2[1]);
```

ES 6

```
// Get the maximum value from data
var d = GetMultipleData(DM, NODE, 1, 1000);
var maxval = Math.max(...d.data);

// draw a rectangle on a widget
var pt1 = [0, 0];
var pt2 = [100, 100];
widget.Rectangle(Widget.RED, true,
                ...pt1,
                ...pt2);
```

JavaScript engine upgrade – other benefits

- Memory consumption
 - JavaScript uses 'garbage collection' to manage any memory that needs to be used for a script.
 - Every object, array or string you use needs to store a small amount of data to be able to do this.
 - This storage in D3PLOT 18.0 is approximately 2/3 of the size in D3PLOT 17.0.

With the default memory size of 25Mb

- D3PLOT 17.0 could create ~350,000 objects.
- D3PLOT 18.0 can now create ~500,000 objects



JavaScript engine upgrade – other benefits

- Speed
 - Scripts which do a lot of mathematical operations will be faster (~ x3.5 speed increase in our tests).
 - String manipulation in scripts is faster (~ x3 speed increase in our tests).
 - Regular expressions in scripts are faster (~ x2.5 speed increase in our tests).
 - Several other features may see some speed increase from these and other improvements.

JavaScript engine upgrade – other benefits

- Debugger
 - The implementation of the debugger has also changed with the new engine.
 - Stepping through code using 'Step' and 'Next' is now significantly faster compared to D3PLOT 17.0, especially for scripts with many lines and/or functions.
 - In D3PLOT 17.0 `try` and `catch` did not work properly in the debugger. For example, the following script would always fail with an exception in the debugger instead of 'catching' it. This now works correctly in D3PLOT 18.0.

```
var o = {};  
try  
{  
    o.UndefinedMethod();  
}  
catch(err)  
{  
    Message(err);  
}
```

JavaScript engine upgrade – other benefits

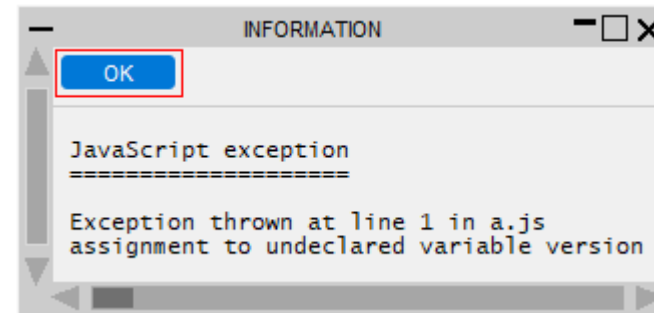
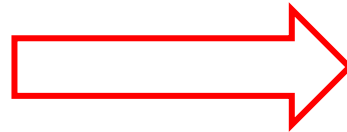
- Debugger



- Strict mode

- By default the debugger now works in 'strict mode' (see https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode for details) as this helps to find potential errors.
 - This has changed since D3PLOT 17.0. In D3PLOT 17.0 the 'strict mode' checkbox actually added some more checks to the debugger. It did not enforce 'strict mode'. This has been corrected for D3PLOT 18.0.

```
version = 18.0;  
Message(version);
```



- This is equivalent to running the script

```
"use strict";  
  
version = 18.0;  
Message(version);
```

- This behaviour can be turned off if required using the checkbox.

JavaScript engine upgrade – other benefits

- Better checking
 - The new engine has better checking. For example, in the following code an error will be given when compiling that some code is unreachable (as there are { } missing so the return is not part of the if block and is always evaluated).

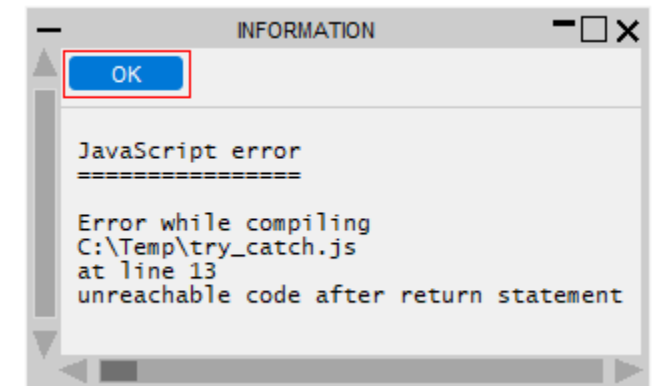
```
var vector = [ 1.0, 0.5, -0.2 ];
var length = vectorLength(vector);

function vectorLength(v)
{
    var l = 0;

    if ( !(v instanceof Array) )
        ErrorMessage("vectorLength not called with array");
        return null;

    for (var i=0; i<v.length; i++)
    {
        l += v[i]*v[i];
    }

    return Math.sqrt(l);
}
```



JavaScript engine upgrade – important changes

- ES6 Modules have not been implemented yet.
 - Upgrading the JavaScript engine has enabled ES6 (and newer) features to be used.
 - Modules are one ES6 feature that require significant changes in our software to implement and we are still resolving these.
 - For D3PLOT 18.0 we want users to benefit from all the other ES6 features so have released the new engine without module support instead of waiting until we resolve this.
 - Support for ES6 modules will be added in a future release.

JavaScript engine upgrade – important changes

- hasOwnProperty() bug in D3PLOT 17.0 and earlier
 - The JavaScript engine from D3PLOT 17.0 (and earlier) contained a bug which meant that for the classes we define, object properties that were inherited from the object prototype appeared to be own properties of the object.
 - For example a Window object inherits properties title, left, right, top, bottom etc. from its prototype.
 - In D3PLOT 17.0 this bug makes these properties appear to be an own property of the window as well as the prototype.
 - If you relied on this feature (unlikely) you will have to modify your code

```
var w = new Window("Test", 0.8, 1.0, 0.5, 0.6);  
w.dog = "Bark";
```

```
Message(w.hasOwnProperty('title'));           // false. w does not have own property title. true in 17.0 (bug)  
Message(w.hasOwnProperty('dog'));             // true. w does have own property dog
```

```
Message(w.__proto__.hasOwnProperty('title')); // true. title is inherited from prototype  
Message(w.__proto__.hasOwnProperty('dog'));   // false. dog is not inherited from prototype
```

JavaScript engine upgrade – important changes

- Script encoding differences

- In D3PLOT 17.0 the default encoding used for scripts by the engine was 'Latin-1'.
- However on Windows this was actually implemented using the default encoding, which for many countries is Windows-1252.
- In D3PLOT 18.0 the upgraded engine now compiles scripts as UTF-8 instead by default.
 - For scripts that just use ASCII (English) characters this will make no difference.
 - However the Windows-1252 encoding also contains special characters such as special quotes , apostrophes, en-dash and em-dash characters (‘ ’ “ ” – —) and these are incompatible with UTF-8.
 - If your script contains these characters it will no longer compile as 'Latin-1' (and it would also not have run on Linux in D3PLOT 17.0 and earlier). Either remove the characters or save the script in a different encoding using your editor.
- Setting a specific encoding for a script such as Shift-JIS or UTF-8 is unaffected.



JavaScript engine upgrade – important changes

- Extra checking **may** occasionally mean old scripts that ran in D3PLOT 17.0 no longer compile in D3PLOT 18.0.
 - As the updated engine has better checking (such as the check for unreachable code mentioned earlier) in some rare cases it may mean that a script which worked in D3PLOT 17.0 will fail to compile in D3PLOT 18.0 until the error is fixed.

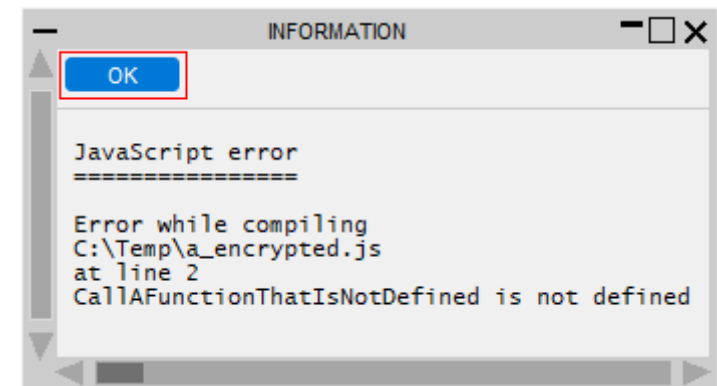
JavaScript engine upgrade – important changes

- Error messages have been enabled for encrypted scripts
 - In D3PLOT 17.0 if a script was encrypted no error messages would be given when compiling/running.
 - For example if the following script was encrypted

```
Message("Starting...");  
CallAFunctionThatIsNotDefined();  
Message("Done.");
```

no error message would be given when the script tried to run the undefined function. This could make it very hard to determine the cause of a 'released' script failing.

- As the upgraded engine has better checking and there may be some rare cases when scripts don't run we have now changed this for D3PLOT 18.0 so error messages will now be given for encrypted scripts.



Checkpoint Files

Regulate Read/Write of the Checkpoint Files

Checkpoint Files

The following preferences will regulate the checkpoint files read/write.

The D3PLOT also has the command-line options with the same names.

- **d3plot*write_checkpoint_files**: Enable/disable the recording of the checkpoint files upon D3PLOT startup. Valid values are TRUE/FALSE (default is FALSE)
- **d3plot*show_checkpoint_files**: Enable/disable the reading of the checkpoint files upon D3PLOT startup. Valid values are TRUE/FALSE (default is FALSE).
 - If the writing of the checkpoint files is disabled, the reading will also be automatically disabled.
- **d3plot*checkpoint_dir**: Specify the folder path to write the checkpoint files to and also read the checkpoint files from.

Miscellaneous

Keyboard Shortcut Additions

- **Shift + Left Arrow** : Highlights from the current cursor position to the left by one character
- **Shift + Right Arrow** : Highlights from the current cursor position to the right by one character
- **Shift + Up Arrow** : Highlights from the current cursor position to the left-most character in the string (0 or prefix)
- **Shift + Down Arrow** : Highlights from the current cursor position to the right-most character (length of the string)
- **Ctrl + Left Arrow** : Jumps the cursor from the current cursor position to the left-most character of the word
- **Ctrl + Right Arrow** : Jumps the cursor from the current cursor position to the right-most character of the word
- **Ctrl + Shift + Left Arrow** : Highlight the rest of the word to the left of the cursor position up to the breaking character
- **Ctrl + Shift + Right Arrow** : Highlight the rest of the word to the right of the cursor position up to the breaking character
- **Double Click Left Mouse Button** : Highlights the whole word
- **Triple Click Left Mouse Button or Ctrl + A** : Highlights the whole line

Contact Information

ARUP

www.arup.com/dyna

For more information please contact us:

UK

T: +44 121 213 3399
dyna.support@arup.com

China

T: +86 21 3118 8875
china.support@arup.com

India

T: +91 40 69019797 / 98
india.support@arup.com

USA West

T: +1 415 940 0959
us.support@arup.com

or your local Oasys distributor