Oasys
REPORTER

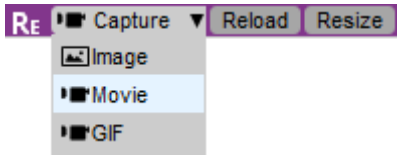
REPORTER 18.0

REPORTER 18.0 – Contents

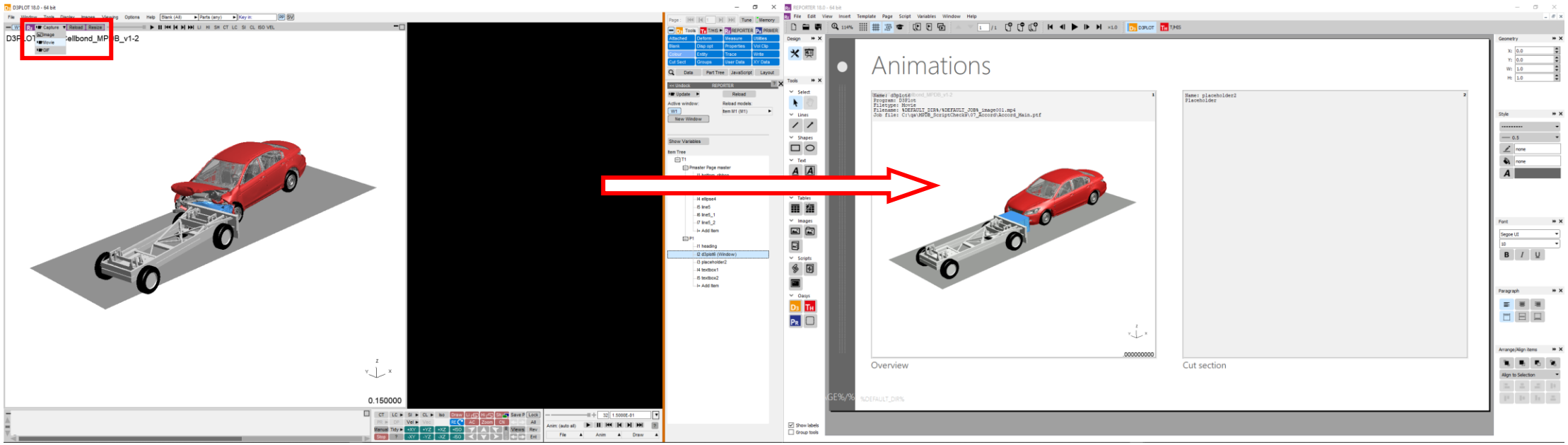
Animations in REPORTER

Animations

With REPORTER 18.0, bring your reports to life with animations of your LS-DYNA simulations:



You can now capture Images, Movies (MP4) or GIFs from D3PLOT into your REPORTER templates



Animations

Once generated, you can play the animations in REPORTER using the playback controls in the new Animation toolbar or by hovering over Movie and GIF items while in Presentation mode.

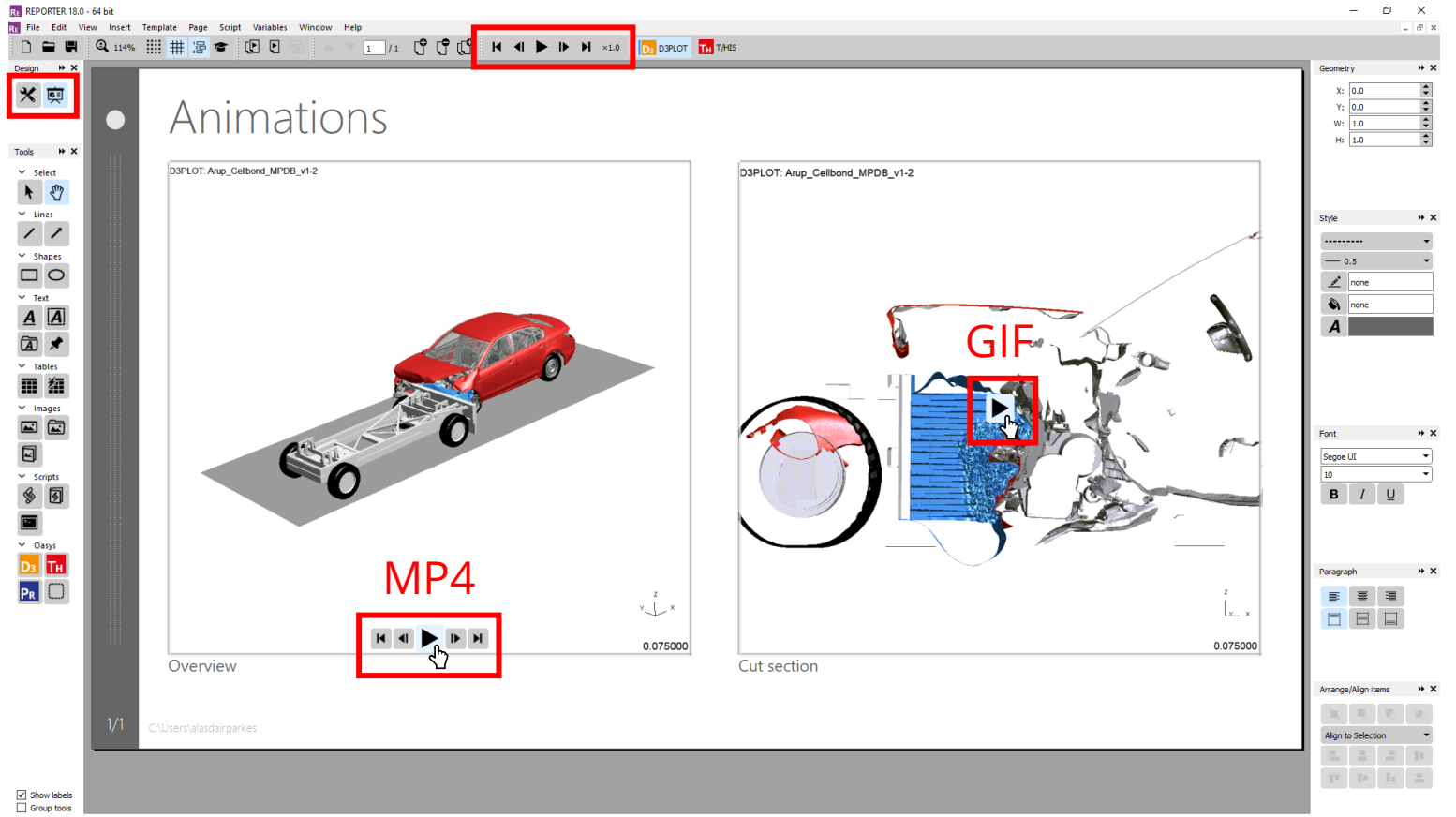
Animation toolbar

Presentation mode

Hover controls

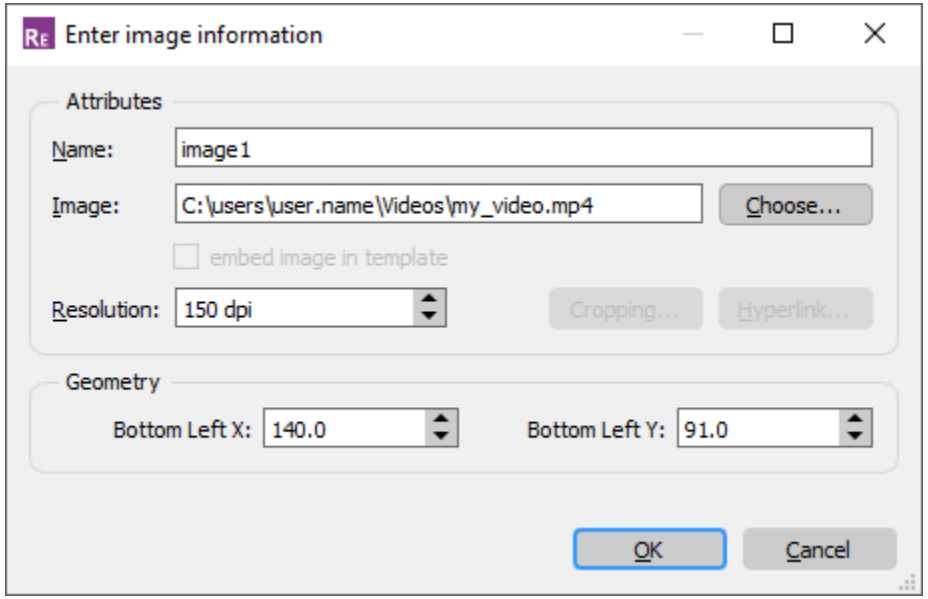
MP4

GIF



Animations

- Image items and Image File items now also support Movies and GIFs. Simply select a **.mp4** or **.gif** file when choosing your image.



- You can export Movies and GIFs to PowerPoint to share your reports with your wider team.

Supported features

Animation feature	GIF	MP4
Playback controls in REPORTER	✓	✓
Step through frame-by-frame in REPORTER	×	✓
Embed in Templates (.ort) and Reports (.orr)*	✓	×
Export to PowerPoint	✓	✓
Export to PDF or HTML**	×	×

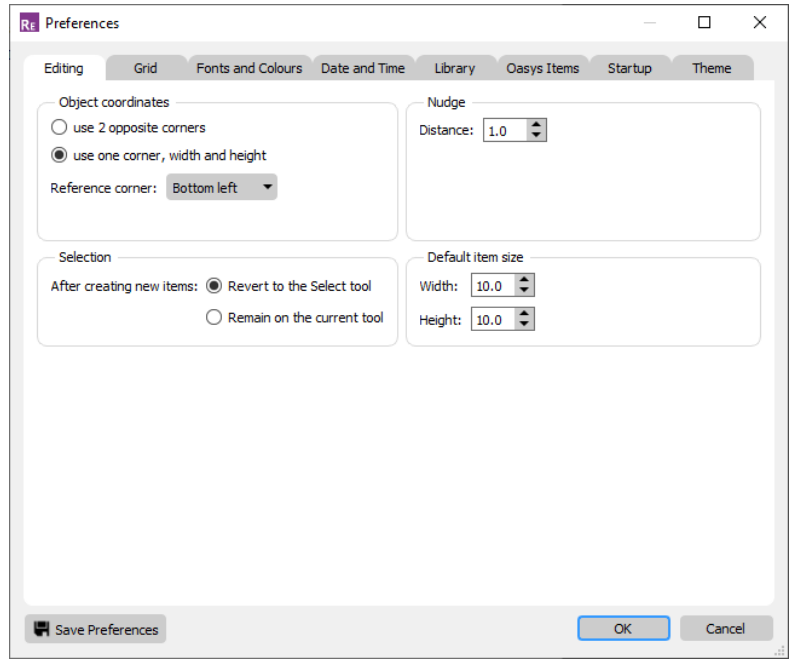
* Although MP4 cannot be embedded, an MP4 Image Item in a Template or Report will still load the .mp4 file if saved with a valid file path.

** Export to PDF and HTML produce static images in place of animations.

Preferences

Preferences

REPORTER preferences can now be edited and saved from directly within REPORTER using the Preferences window, located at *File → Preferences*.



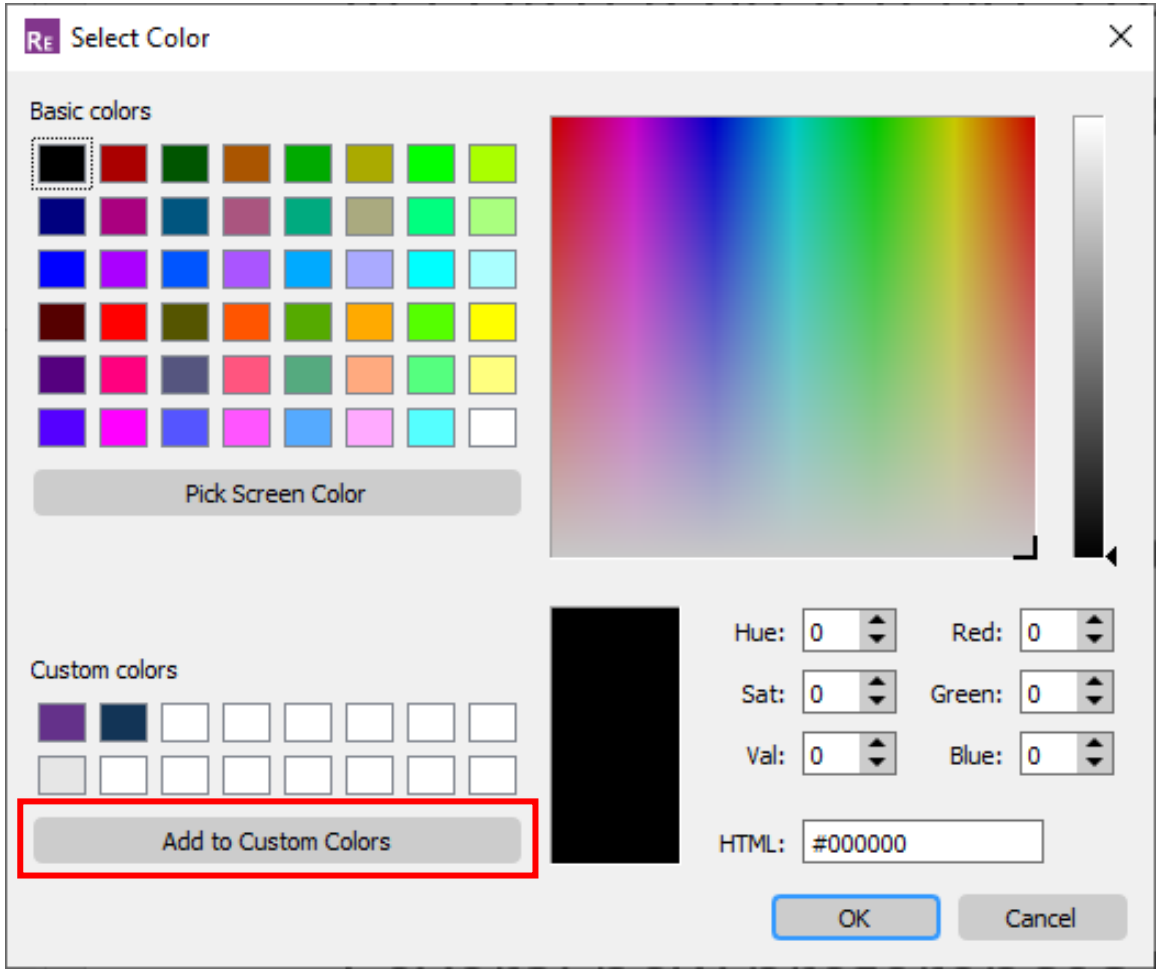
Pressing **Save Preferences** will save to the *oa_pref* file in your HOME directory.

Several new preferences have also been added to improve customisation, and the *File → Program locations* options are now configurable from Preferences.

User Colours

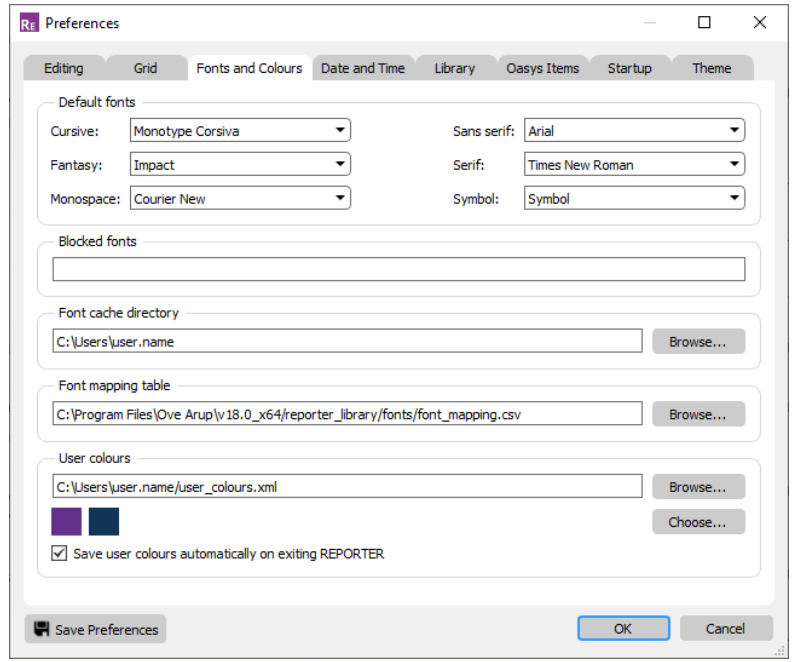
User Colours

Any user colours added in REPORTER via 'Add to Custom Colors' are now saved for future sessions. User colours are now synchronised across REPORTER, D3PLOT and T/HIS (the first sixteen colours saved in D3PLOT or T/HIS are accessible in the panel).



User Colours

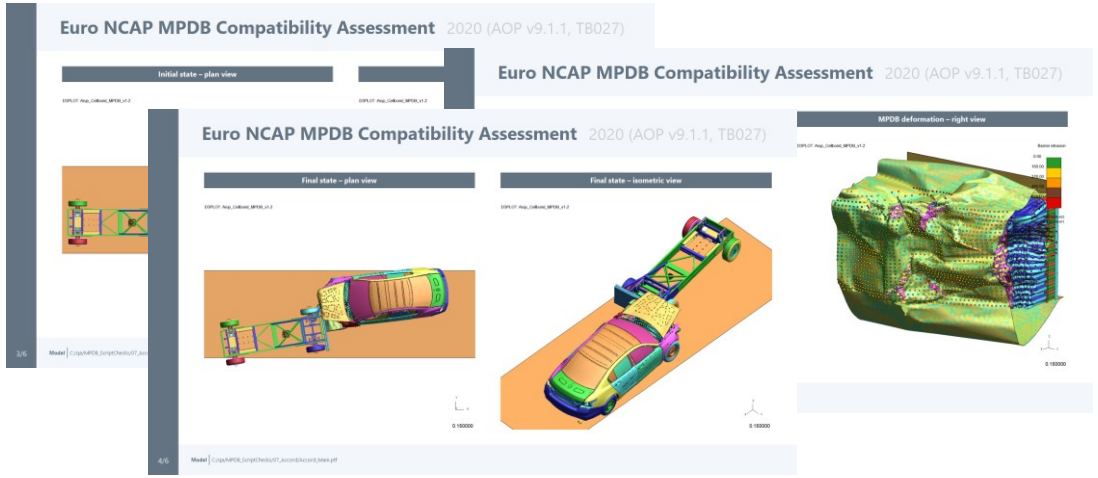
User colours are automatically saved to the *user_colours.xml* file, which is shared by D3PLOT and T/HIS. You can control which XML file is used via the new 'Fonts and Colours' tab in the Preferences dialog. You can also control whether user colours are automatically saved when exiting REPORTER (on by default).



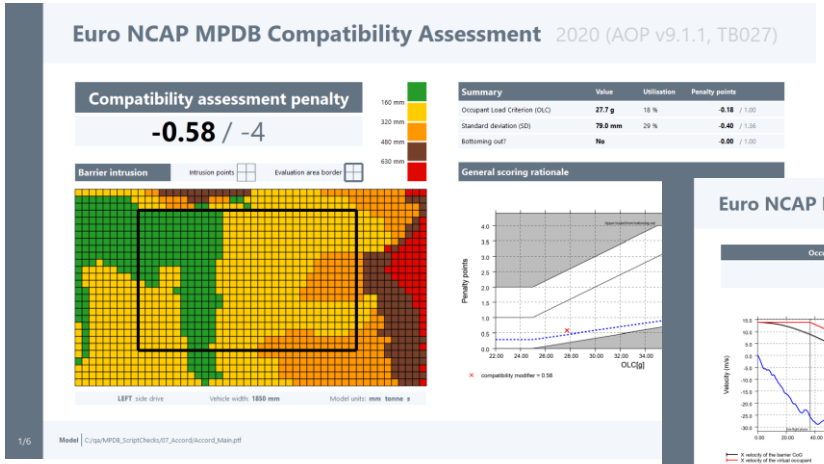
Automotive Library Templates

MPDB Compatibility Assessment templates

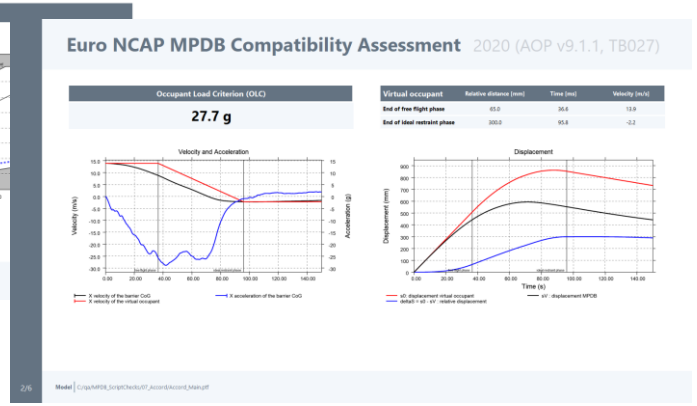
Automated postprocessing for the MPDB Compatibility Assessment with REPORTER library templates:



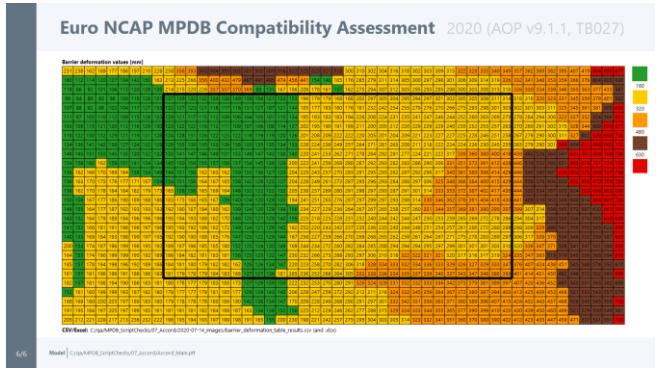
pp3-5 Animations/views of barrier deformation



p1 Overall scoring rationale and barrier deformation plot



p2 OLC calculation details



p6 Detailed barrier deformation plot

MPDB Compatibility Assessment templates

REPORTER 17.1 saw the introduction of Euro NCAP MPDB Compatibility Assessment templates. In REPORTER 18.0, we have added variants for C-NCAP:

- Euro NCAP MPDB Compatibility Assessment **2020**
- Euro NCAP MPDB Compatibility Assessment **2023**
- **C-NCAP** MPDB Compatibility Assessment **2022**
- **C-NCAP** MPDB Compatibility Assessment **2023**

Each has two year variants to accommodate changes in the scoring described in the Euro NCAP Adult Occupant Protection Assessment Protocol v9.1.1 and Technical Bulletin (TB 027) v1.1.1, and in the C-NCAP Management Protocol (2021 edition).

The templates are configured for use with the with the Arup Cellbond MPDB Shell Model, but can be adapted for use with other barrier models.

MPDB Compatibility Assessment templates

- The templates now feature animations of the simulation.
- OLC and barrier deformation results are automatically exported to CSV and Excel formats to aid further analysis.
- To learn how to use the templates in REPORTER, select:
  Help → Additional user guides
- The C-NCAP template is similar to the Euro NCAP version, with the addition of the **barrier intrusion height** check. Please read our interpretation of the C-NCAP Management Protocol in REPORTER by selecting:
  Help → Additional user guides → C-NCAP_MPDB_Barrier_Intrusion_Height.pdf
  We will continue to update the templates in future releases of REPORTER.

JavaScript Engine Upgrade

JavaScript engine upgrade

- For REPORTER 18.0 the JavaScript engine used in REPORTER has been significantly upgraded.
- In REPORTER 17.0 and earlier the engine only supported ECMAScript 5 features.
- In REPORTER 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 REPORTER 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 REPORTER did not work in REPORTER 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

```
// draw a rectangle on an image
var pt1 = [0,     0];
var pt2 = [100, 100];
image.lineColour = "red";
image.Rectangle(pt1[0], pt1[1],
                pt2[0], pt2[1]);
```

ES 6

```
// draw a rectangle on an image
var pt1 = [0,     0];
var pt2 = [100, 100];
image.lineColour = "red";
image.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 REPORTER 18.0 is approximately 2/3 of the size in REPORTER 17.0.

    With the default memory size of 25Mb
    - REPORTER 17.0 could create ~350,000 objects.
    - REPORTER 18.0 can now create ~500,000 objects

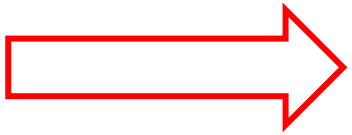
JavaScript engine upgrade – other benefits

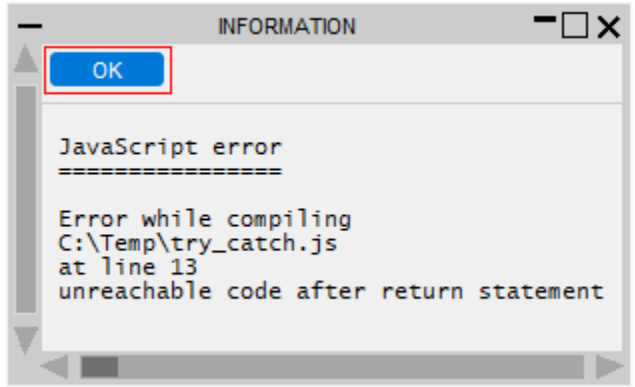
- Speed
  - Scripts which do a lot of mathematical operations will be faster (~ x3.5 speed increase in our tests).
  - String manipulation in scripts is faster (~ x3 speed increase in our tests).
  - Regular expressions in scripts are faster (~ x2.5 speed increase in our tests).
  - Several other features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

JavaScript engine upgrade – other benefits

- 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 REPORTER 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 REPORTER 17.0 and earlier
  - The JavaScript engine from REPORTER 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 REPORTER 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

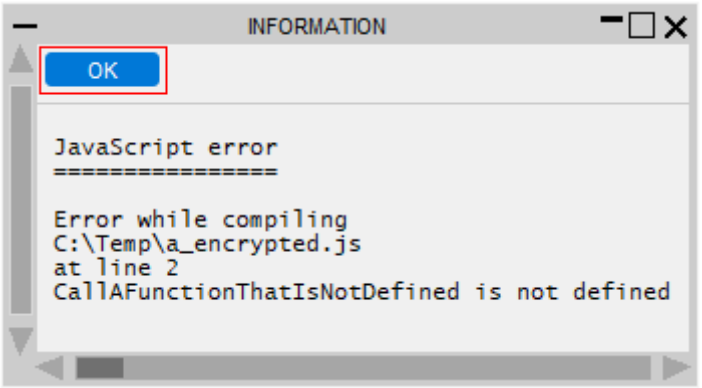
- Extra checking *may* occasionally mean old scripts that ran in REPORTER 17.0 no longer compile in REPORTER 18.0.
  - As the updated engine has better checking (such as the check for unreachable code mentioned earlier) in some rare cases it may mean that a script which worked in REPORTER 17.0 will fail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

JavaScript engine upgrade – important changes

- Error messages have been enabled for encrypted scripts
  - In REPORTER 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 REPORTER 18.0 so error messages will now be given for encrypted scripts.



JavaScript API

JavaScript API Improvements

- The JavaScript function **Template.EditVariables** now accepts an optional boolean argument to determine whether selected Variables should be displayed alphabetically (true) or in the list order in which they were passed to the function (false).

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 44369797 / 98
india.support@arup.com

USA West

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

or your local Oasys distributor