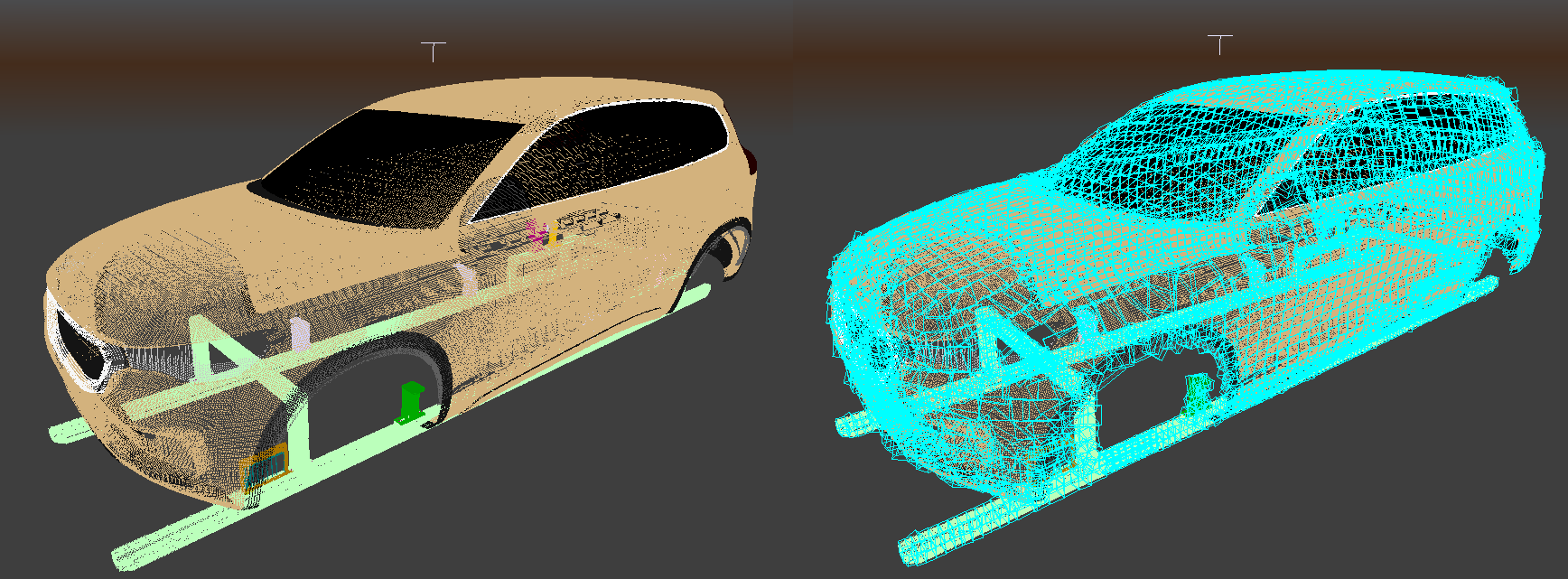
Pointools Vortex API Extension for Clash Data

# Introduction

Users of the Pointools Vortex API would like access to the clash data that is generated when performing the first stage of a clash operation. Clash data is a tree of bounding boxes that tightly encapsulates a point cloud, from the top level bounds, to tightly fitted non-axis aligned leaf nodes containing small amounts of points (in the order of 1000 points per leaf). This clash data can be used to segment and reduce the amount of primitives that need to be passed to D-Cubed by Descartes, allowing quick clash tests to be performed on bound boxes as a first stage in reducing the number of points that have to be processed.

To give an idea of what this clash data is the image below shows a point cloud on the left, and the same point cloud along with a visual representation of the clash data bounding boxes show in blue on the right.



# Requirements

Details of the clash data bounding boxes should be made available through the API, including the current transformation matrix that should be applied to the clash data boxes.

Clash data generation can take some time (in the order of minutes) so some sort of feedback mechanism should be provided to allow user expectations to be managed.

The function that starts the clash data generation should take be called from a separate thread because of the length of time it can take (in the order of minutes). This may be better managed by the caller of the API as part of the main application threading model.

# Proposed API

The proposed API will allow Vortex users to initiate the creation of clash data for a POD file, or to open a saved .clashdata file if one has already been created for a POD. Once the clash data has been created/opened an array of bounding boxes describing the clash data can be returned, as well as a the current transform matrix that should be applied to these boxes, and the filename of the file where the clash data has been saved to disk.

dEFINITION

typedef void (\***PTopenClashDataFeedback**)(int pcent, PTstr feedback);

Description

Function pointer to allow feedback to be returned on the progress of the clash data generation.

Parameters  
*pcent* Percentage of the clash data generation that has been completed

*feedback* Optional feedback string to be displayed to the user, may be NULL.

dEFINITION

PTres **ptOpenClashData**( const PTstr podFilename, PTclashDataHandle\* h, PTopenClashDataFeedback\* feedback )

Description

Open the stored clash data for the passed POD, if no clash data is found then generate new clash data and store this in the same folder as the passed podFilename.

Parameters  
*podFilename* The POD file for which clash data is to be opened or generated.

*h* A handle that represents the clash data in the vortex engine, this must be used when calling all other clash data functions, and discarded when the clash data is released.

*feedback* A callback function that Vortex will call periodically when generating the clash data to allow the caller of ptOpenClashData() to be updated on the current progress.

dEFINITION

void **ptReleaseClashData**( PTclashDataHandle h )

Description

Release any memory being held for the clash data refererred to by the passed clash data handle. After this function is called the handle will no longer be valid.

Parameters  
*h* A valid clash data handle that was returned when calling ptOpenClashData()

dEFINITION

PTres **ptGetClashDataTransform**( PTclashDataHandle h, PTdouble\* transform4x4,

PTbool row\_order )

Description

Get the transformation matrix for the passed clash data handle.

Parameters  
*h* A valid clash data handle that was returned when calling ptOpenClashData()

*transform4x4* An array of 16 double values representing a 4 by 4 transformation matrix

*row\_order* Boolean indicating the ordering of the values in the matrix. Use PT\_TRUE to indicate row major ordering and PT\_FALSE for column ordering.

dEFINITION

struct **PTbounds**  
{

PTfloat center[3];

PTfloat extents[3];

PTfloat xaxis[3];

PTfloat yaxis[3];

PTfloat zaxis[3];  
}

Description

Data structure to allow easy passing of non axis aligned bounding box data to Vortex users.

dEFINITION

PTres **ptGetClashDataBounds**( PTclashDataHandle h, PTbounds\*\* bounds,

PTuint numBounds )

Description

Return an array of bounds that represent the clash data structure for a POD file that was created by calling ptOpenClashData.

Parameters  
*h* A valid clash data handle that was returned when calling ptOpenClashData()

*bounds* Returns an array of PTbounds objects representing the tightly bound clash data bounding boxes for a POD file. Note that the caller of this function is responsible for deleting this array, Vortex allocates new memory for this array each time this function is called but does not manage the memory afterwards.

*numBounds* The number of PTbounds objects returned in the bounds array.

dEFINITION

PTres **ptGetClashDataFilename**( PTstr podFilename )

Description

Return the name of the clash data file that is used to store the clash data for the passed POD file.

Parameters  
*podFilename* The name of the clash data file is returned here.

# Example API Usage

void **openClashDataFeedback**(int pcent, PTstr feedback)  
{

std::cout << “clash data generation “ << pcent << “ complete\n”;

std::cout << feedback << “\n”;  
}

PTstr podFilename = “G:\\test\\test.pod”;

PTclashDataHandle h = 0;

PTbounds\* bounds = NULL;

PTuint numBounds = 0;

PTdouble transform[16] = {0};

PTbool rowOrder = PT\_TRUE;

// Note that it is recommended that ptOpenClashData is called from a separate thread

PTint res = **ptOpenClashData**(podFilename, h, openClashDataFeedback);

if (res == PTV\_SUCCESS)  
{

res = **ptGetClashDataBounds**(h, &bounds, &numBounds);

if (res == PTV\_SUCCESS)  
{

**ptGetClashDataTransform**(h, transform, rowOrder);

...  
}

delete [] bounds;

**ptReleaseClashData**(h);  
}