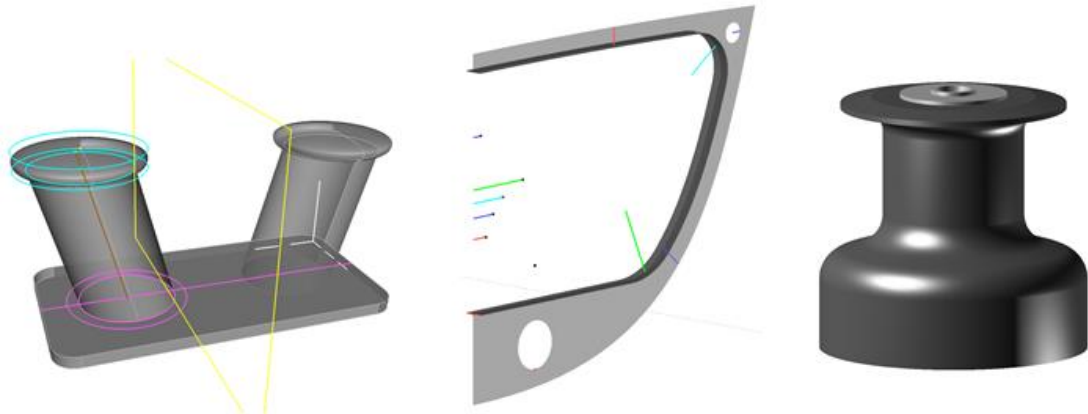


Komponenten

Teil 1 – Einführung und Beispiele

von Reinhard Siegel

Oktober 2021



Inhalt

- Einleitung
- Einführung – Komponente Winch
- Bearbeiten von Komponenten im Entities Manager
- Komponente Spiegel
- Komponente Kiel
- Komponente Längsspant
- Komponente Hauptabmessungen
- Schablonen für Plattenverformung
- Mögliche Komponenten-Probleme
- Modifizieren einer Komponente

Einleitung

Kein Entwurf sieht aus wie der andere. Aber Boote ähneln sich. Mit Komponenten kann man Zeit und Arbeit sparen. Eine Komponente ist ein Stück zusammenhängende MultiSurf-Geometrie, die sich in andere Modelle einfügen lässt. Anstatt jedesmal im Arbeitsmodell aus allen Einzelteilen seine bevorzugte Konstruktion für Kiel, Ruder, Deck, Spiegel etc. neu zu erzeugen, kann man sie mit wenig Aufwand über Komponenten einfügen.

Mit Komponenten lässt sich der Aufbau eines Modells strukturiert und übersichtlich gestalten. Mit einer Komponente kann man aus einem großen Modell eine Gruppe zusammenhängender Objekte herauslösen, als eigenständiges Modell bearbeiten und in andere Modelle einfügen.

In diesem Tutorial soll erklärt werden, wie man eine Komponente in ein Modell lädt, wie man sie erstellt, welche Hilfsmittel dabei zur Verfügung stehen. In Teil 1 dieser Tutorials wird eine Reihe Komponenten vorgestellt, die beim Aufbau eines Bootsmodells praktisch sind. In Teil 2 werden zahlreiche Komponenten gezeigt, mit denen Modellen ein realistisches Aussehen gegeben werden kann.

Verwendete Abkürzungen:

Cp: Kontrollpunkt, Stützpunkt (control point, support point); synonym verwendet.

Mc: Masterkurve, Stützkurve, Kontrollkurve (master curve, support curve, control curve); synonym verwendet.

cp1, cp2, ...: bezeichnet den 1., 2. ... Punkt in der Liste der Kontrollpunkte einer Kurve. Es ist kein Objektname.

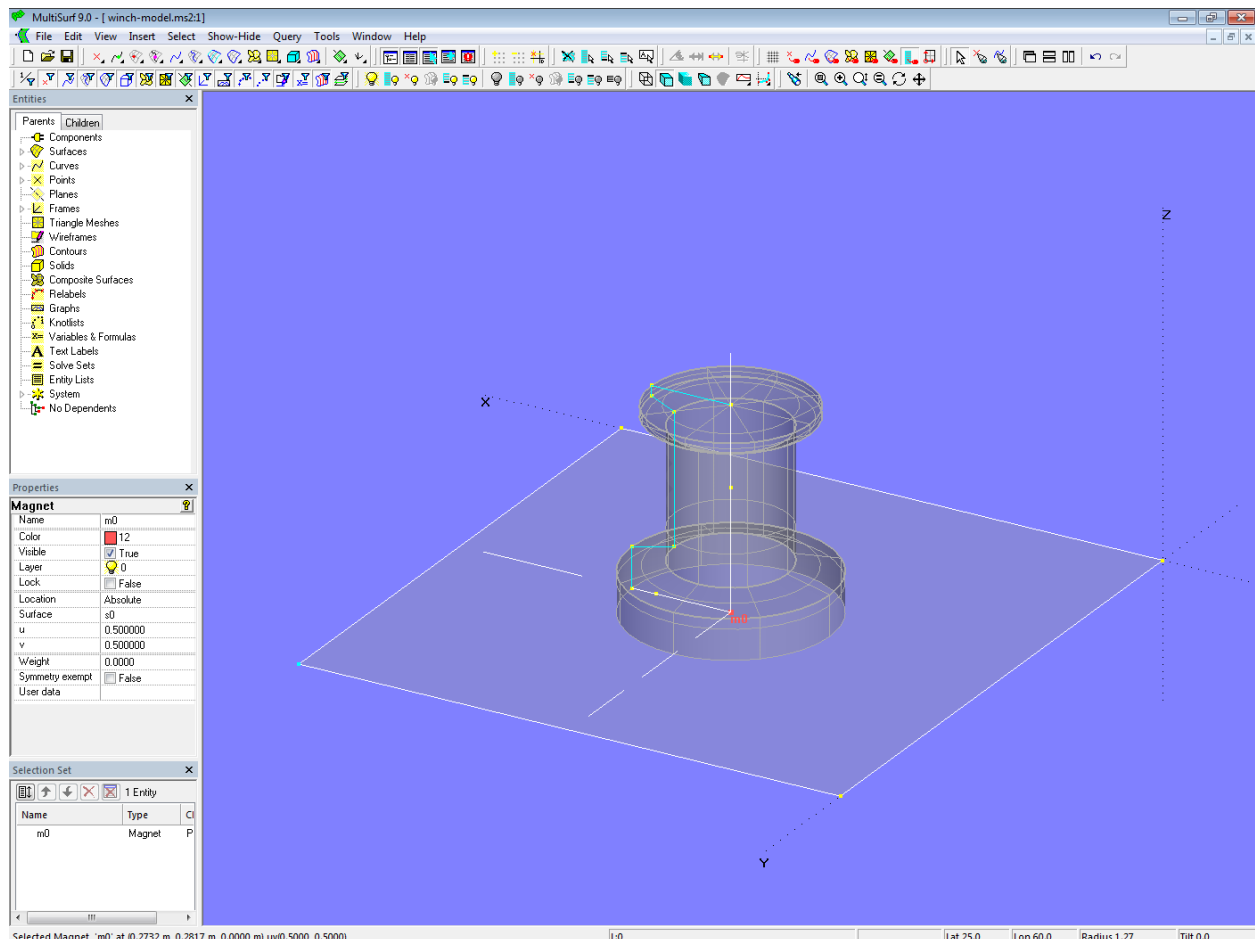
mc1, mc2, ...: bezeichnet die 1., 2. ... Kurve in der Liste der Stützkurven einer Fläche. Es ist kein Objektname.

Im Folgenden werden die MultiSurf-Namen für Punkt-, Kurven- und Flächenarten verwendet. Das ergibt zwar „denglische“ Sätze, soll aber dem Verständnis und der Nachvollziehbarkeit dienen.

Einführung – Komponente Winch

Quell-Modell der Komponente

Betrachten wir zur Einführung in das Thema Komponenten das Modell *winch-model.ms2*. Es enthält die Geometrie einer einfachen Winch. Auf der Basisfläche **s0** liegt Magnet **m0**; von ihm hängen Offset Point **p0** und Point **p1** ab. Mit **m0**, **p0** und **p1** ist der 3-point Frame **F** definiert. Auf dieses Benutzerkoordinatensystem beziehen sich die Punkte, die die Querschnittskurve **c0** bestimmen. Diese wiederum wird um die Line **l0** gedreht, was die Revolution Surface **s1** ergibt. Verschiebt man Magnet **m0**, wandert die Winch mit, ihre Drehachse ist immer senkrecht zur Basisfläche.

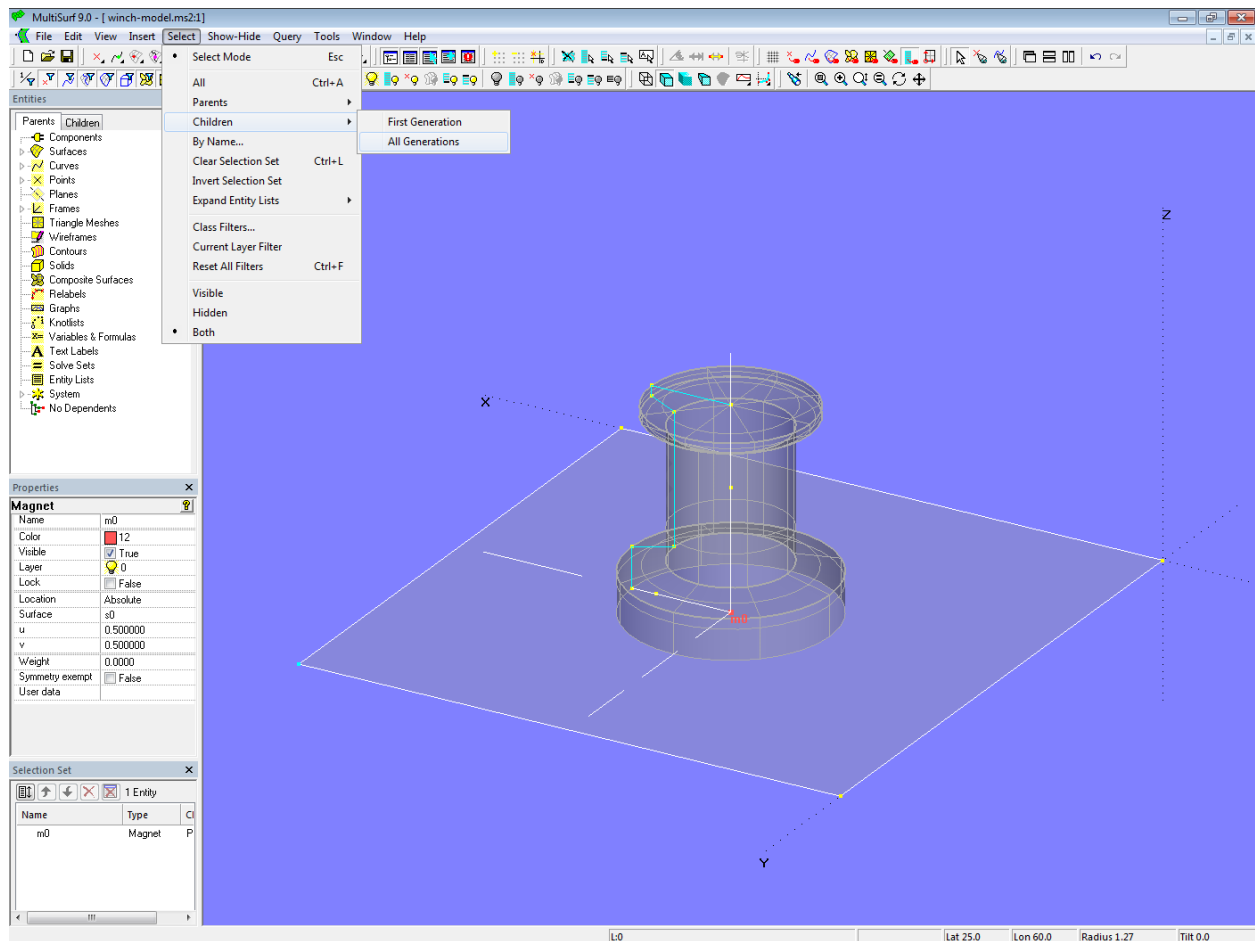


Modell winch-model.ms2 – Winch als Revolution Surface im Quell-Modell

Winch-model.ms2 ist keine Komponente, sondern eine normale MultiSurf-Modelldatei (Erweiterung des Dateinamens ist „.ms2“). Sie ist das Quell-Modell für die Winch-Komponente. Eine Komponentendatei hat die Erweiterung „.mc2“.

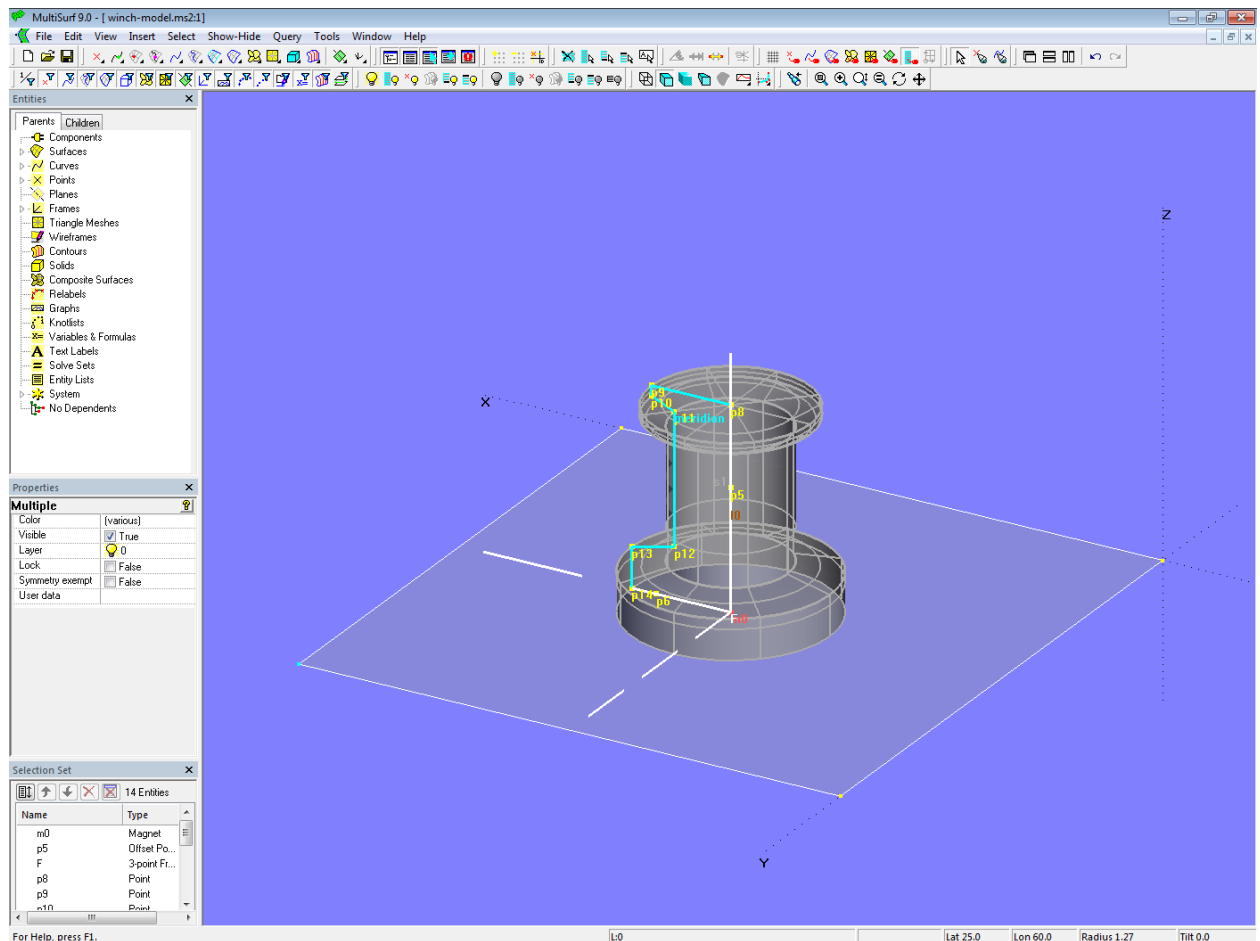
Speichern der Komponente

Im Allgemeinen braucht eine Komponente Eltern aus dem Host- oder Arbeitsmodell, das Modell, in das sie eingefügt wird. In unserem Beispiel soll die Fläche **s0**, auf der der Magnet **m0** liegt, beim Laden der Komponente durch eine Fläche im Arbeitsmodell ersetzt werden. Wählen wir also im ersten Schritt den Magnet **m0** aus.



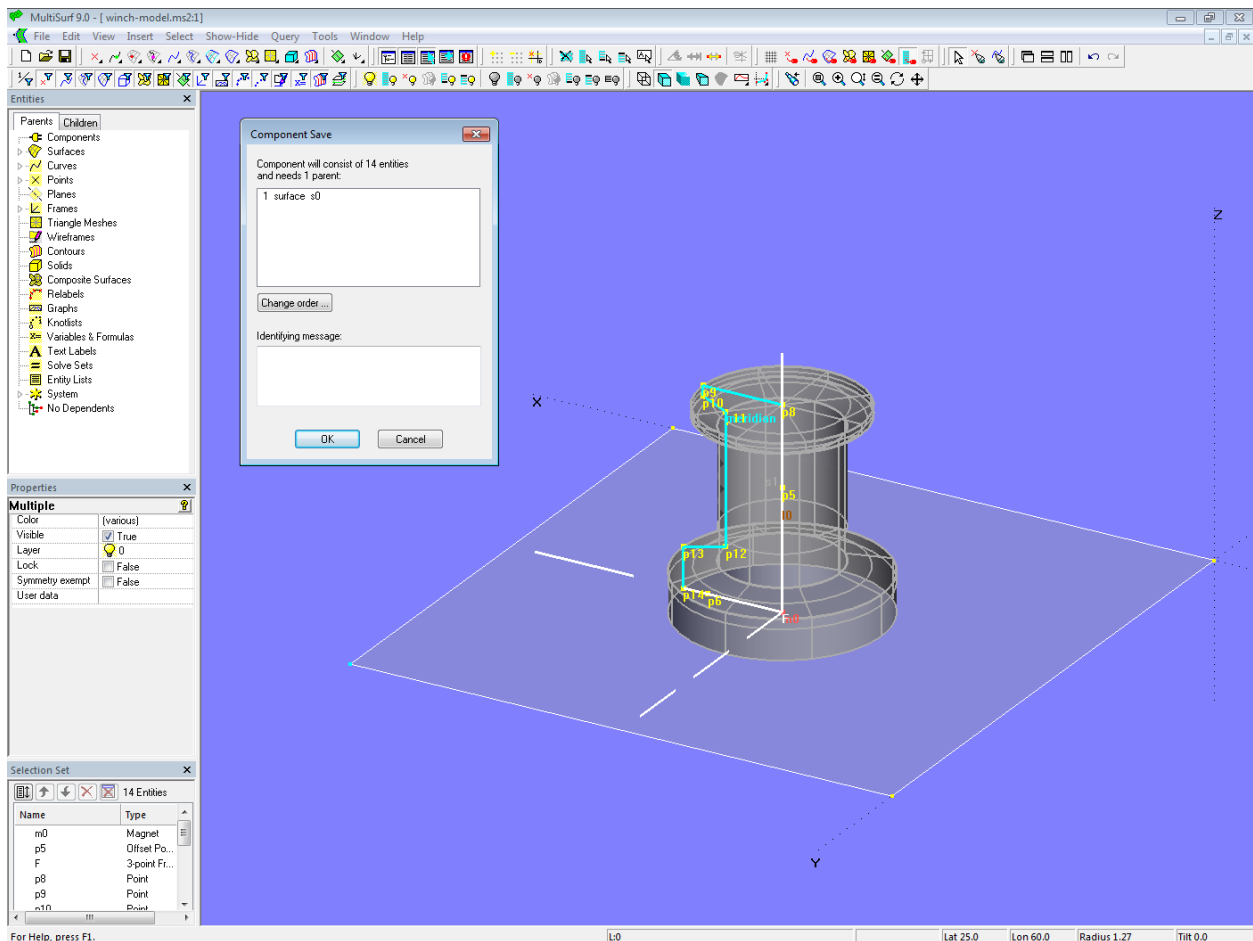
Modell winch-model.ms2 – Auswahl aller von Magnet **m0** abhängigen Objekte der Winchgeometrie

Und dann im zweiten Schritt, alles, was von ihm abhängt: **Select/ Children/ All Generations**. Im Selection Set stehen nun alle Objekte, die von **m0** abhängen, also die gesamte Winch-Konstruktion.



Modell winch-model.ms2 – von Magnet *m0* abhängige Objekte der Winchgeometrie im Quell-Modell

Nun wählen wir im Hauptmenü **File/ Component/ Save**, worauf sich folgendes Dialogfenster öffnet:

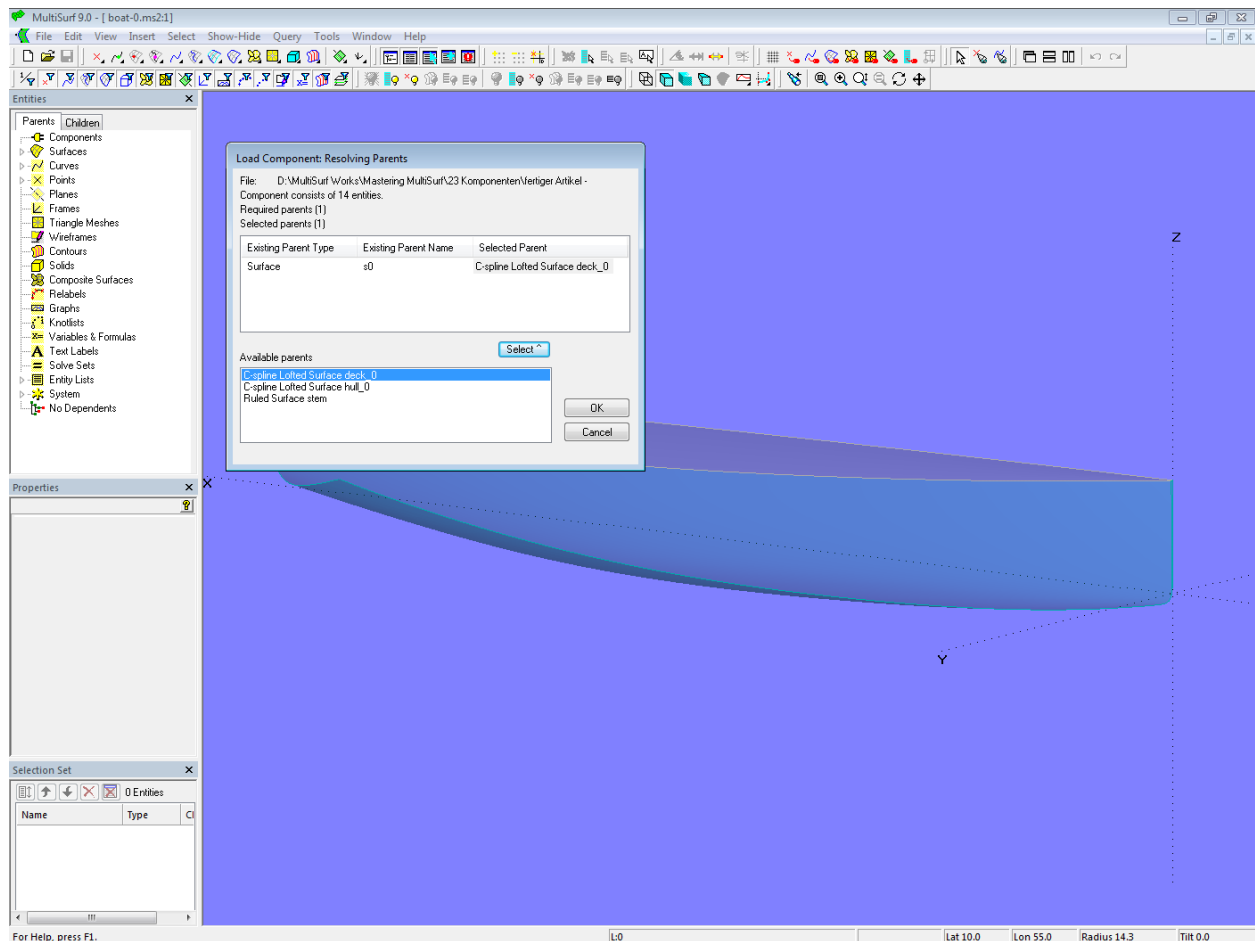


Modell winch-model.ms2 – Abspeichern der Winch-Komponente im Quell-Modell

Das Modell, in das die Winch-Komponente eingeladen werden soll (Host-Modell, Arbeitsmodell), muß also eine Fläche bereitstellen. Abschließend auf die **OK**-Schaltfläche klicken und die Komponente abspeichern unter dem Namen *winch-component.mc2*.

Laden der Komponente

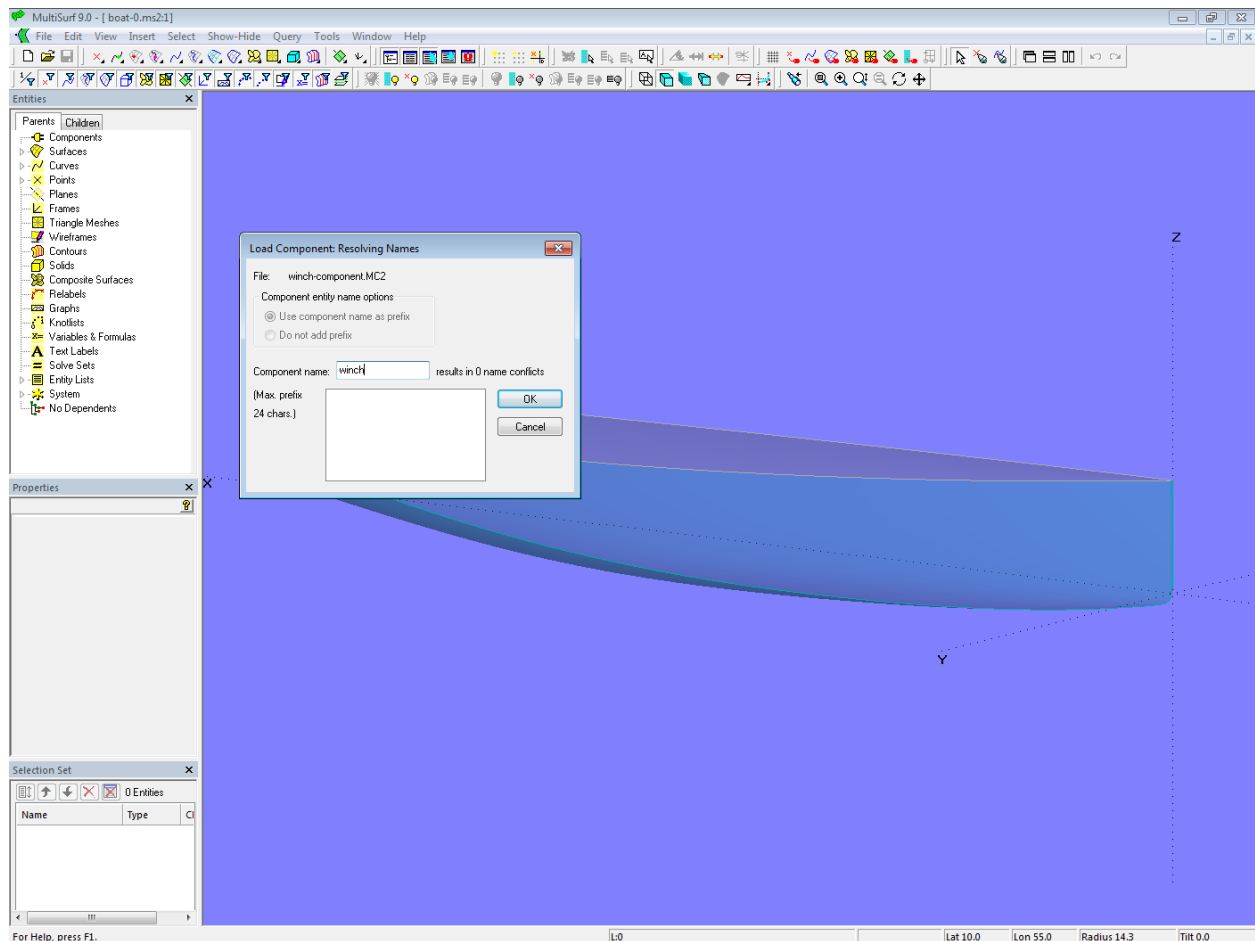
Probieren wir aus, wie man die Komponente in ein anderes Modell eingefügen kann. Dazu öffnen wir als Arbeitsmodell das Modell *boat-0.ms2*. Es enthält eine Rumpffläche (**hull_0**) und eine Decksfläche (**deck_0**). Dann **File/ Component/ Load** und im Dialogfenster die Komponentendatei *winch-component.mc2* auswählen. Daraufhin erscheint das Dialogfenster „Load Component: Resolving Parents“. Hier kann man die Eltern (parents) auswählen, an die die Komponente beim Einfügen in das Arbeitsmodell angeschlossen wird.



Modell boat-0.ms2 – Auswahl der Eltern beim Laden der Winch-Komponente in das Arbeitsmodell

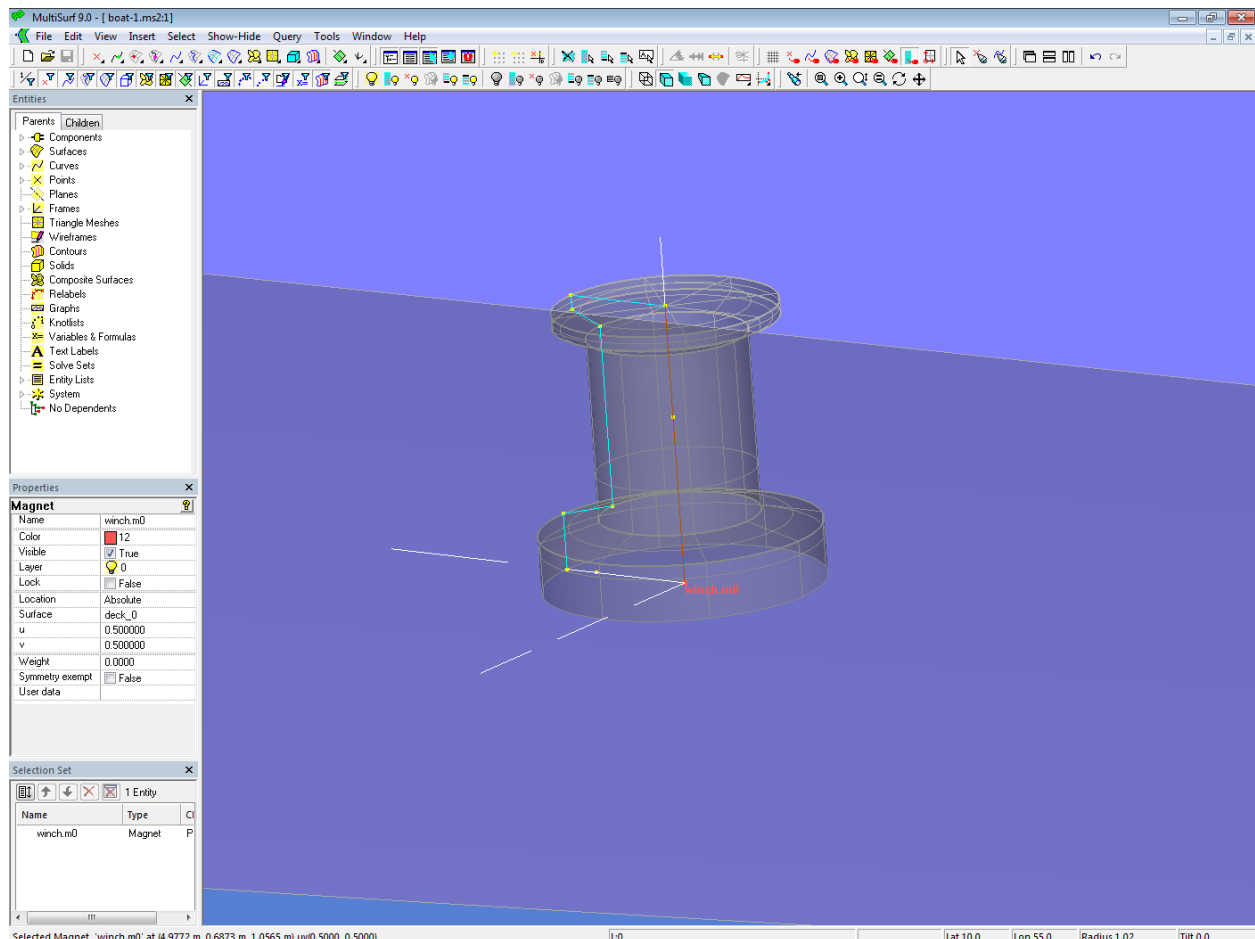
In diesem Beispiel ist es die Decksfläche **deck_0**, die die Fläche **s0** aus dem Quell-Modell ersetzen soll.

Nach Bestätigen mit **OK** wird das Dialogfenster „Load Component: Resolving Names“ angezeigt. Es dient dazu, der Komponente im Host-Modell einen Namen zu geben und Namenskonflikte aufzulösen. In diesem Beispiel wird der Komponente der Name „winch“ gegeben. Dieser Name wird den Namen aller Objekte, die über die Komponente eingeladen werden, vorangestellt. Damit gibt es keine Konflikte der Objektnamen.



Modell boat-0.ms2 – Eingabe des Namens der Winch-Komponente beim Laden in das Arbeitsmodell

Nach Bestätigen mit **OK** erscheint die Winch auf der Decksfläche. Mit dem Magneten **winch.m0** kann man sie an die gewünschte Position schieben. Nun das Modell abspeichern unter dem Dateinamen **boat-1.ms2**.



Modell boat-1.ms2 – Komponente Winch im Arbeitsmodell

Bearbeiten von Komponenten im Entities Manager

Im Entities Manager ist die oberste Kategorie „Components“. Hier werden Komponenten unter dem beim Laden vergebenen Namen aufgelistet.

Funktion Maus-Rechtsklick auf „Components“

Klickt man mit der rechten Maustaste auf „Components“, wird ein Kontext-Menü mit 2 Auswahlmöglichkeiten angezeigt:

Load – Laden einer Komponente (Abkürzung für **File/ Component/ Load**)

Change order – Ändern der Reihenfolge, in der Komponenten im Entities Manager angezeigt werden

Funktion Maus-Rechtsklick auf den Namen einer Komponente

Klickt man mit der rechten Maustaste auf den Namen einer Komponente, stehen im Kontext-Menü folgende Auswahlmöglichkeiten zur Verfügung:

Select – wählt alle Objekte der Komponente aus; praktisch für Abfragezwecke, Layer-Änderung

Delete – löscht alle Objekte der Komponente und entfernt sie aus dem Entity Manager

Show – setzt die Eigenschaft „Visible“ aller Objekte der Komponente auf „True“

Hide – setzt die Eigenschaft „Visible“ aller Objekte der Komponente auf „False“

Make internal – mitunter kann es zweckmäßig sein, eine Komponente in den Hauptteil des Host-Modells aufzulösen. Zum Beispiel, wenn Präfixe entfernt werden sollen oder wenn eine kleinere Konstruktionskomponente hinzugefügt wurde, die mit einer eigenen Überschrift aber nicht vom Modell getrennt werden

soll. **Make internal** entfernt alle Präfixe und löscht die Komponentenbezeichnung. Eventuelle Namenskonflikte können mit Hilfe eines speziellen Dialogfensters aufgelöst werden.

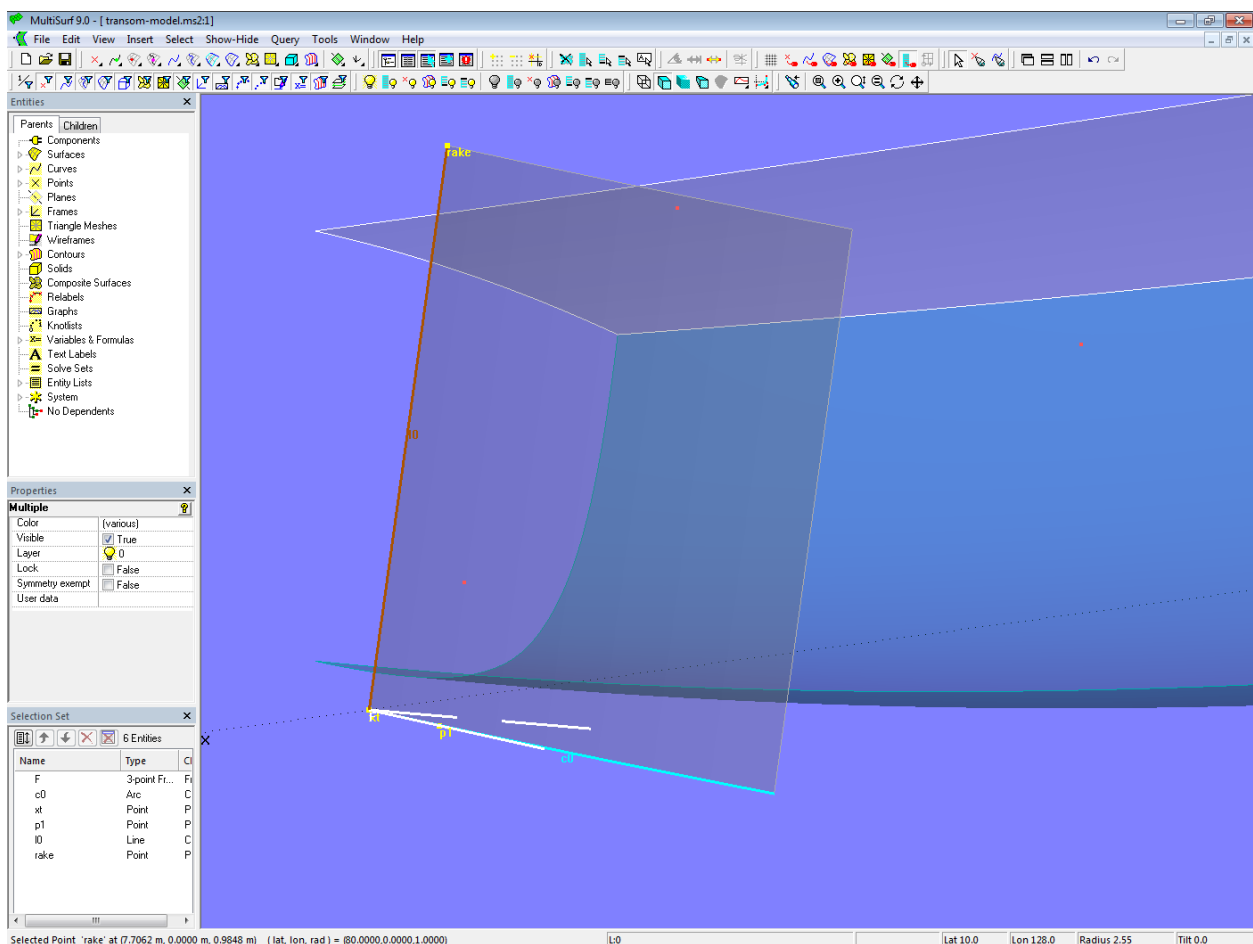
Current – eine aktuelle Komponente ähnelt einem aktuellen Layer. Alle nachfolgend eingefügten Objekte werden der aktuellen Komponente hinzugefügt.

Komponente Spiegel

Wir wollen nun die Geometrie für eine weitere Komponente erzeugen, dann als Komponente speichern und anschließend diese in das Modell *boat-1.ms2* laden. Die Komponente soll einen Kreiszylinder-Spiegel darstellen.

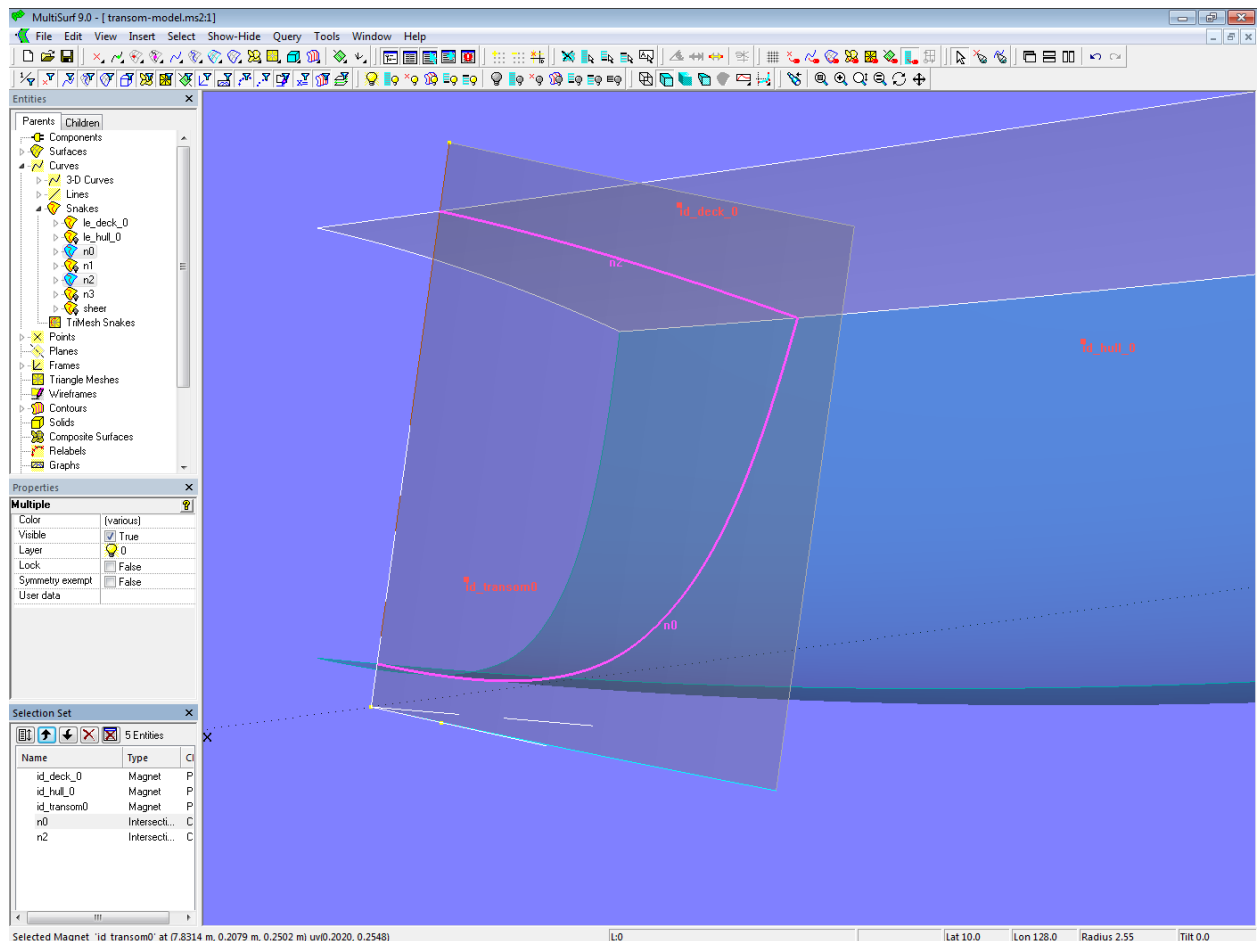
Quell-Modell Spiegel

Als Ausgangsmodell dient das Modell *transom-model.ms2*. Es enthält die Rumpffläche **hull_0** und die Decksfläche **deck_0**. Außerdem gibt es die Basisfläche des Spiegels, **transom_0**. Dies ist eine Translation Surface mit der Leitkurve **c0** (Kreisbogen) und der Erzeugenden **l0** (Line). Alle Punkte für **c0** und **l0** sind im 3-point Frame **F** definiert. Seinerseits wird **F** bestimmt durch Point **xt** (Ursprung des Frames auf der X-Achse), Point **rake** (Länge und Neigung der Mittellinie des Spiegelfläche) und Point **p1** (querab von **xt**).



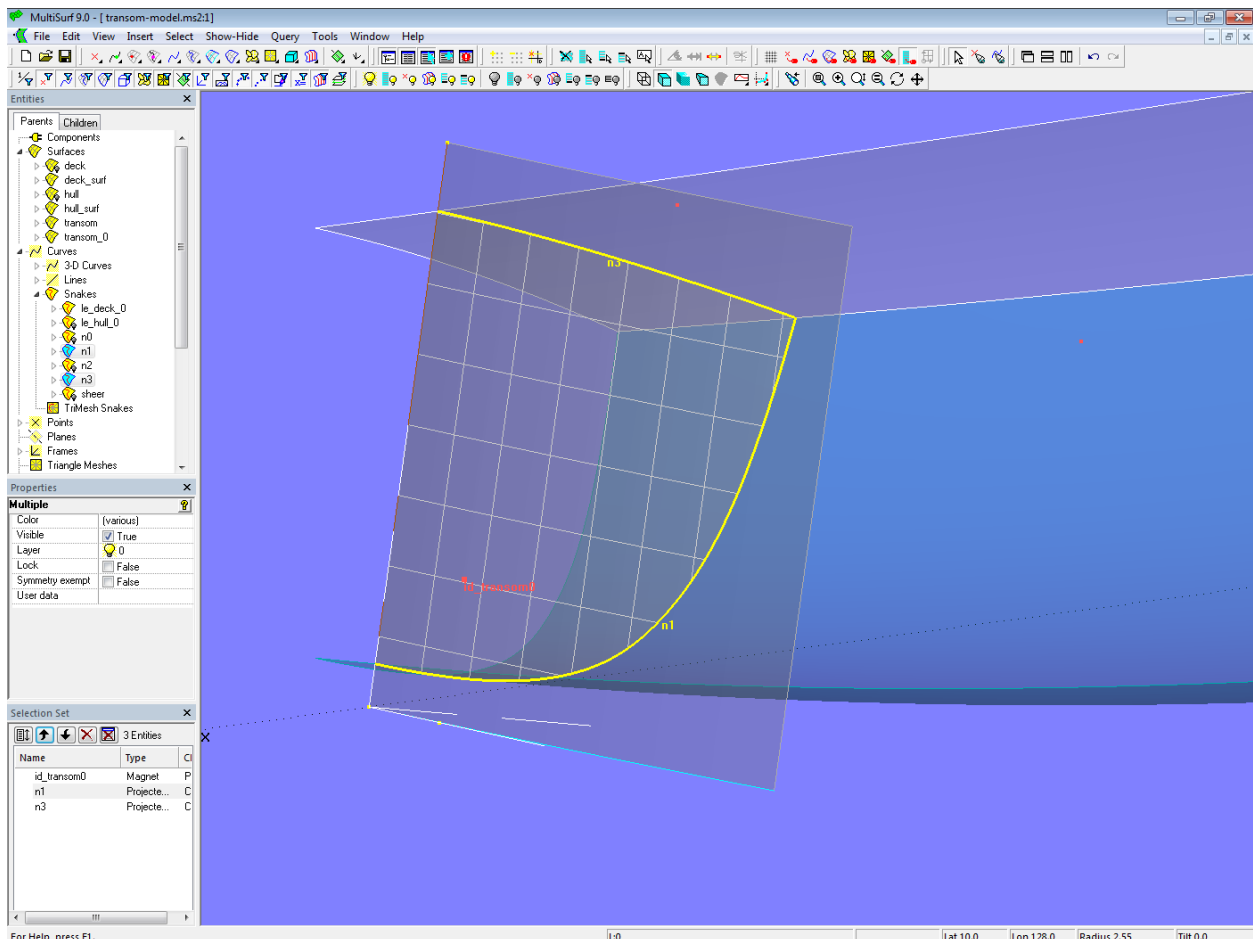
Modell *transom-model.ms2* – die Basisfläche **transom_0** des Kreiszylinder-Spiegels wird aufgespannt zwischen Line **l0** und Arc **c0**.

Die Spiegel-Basisfläche **transom_0** schneidet die Rumpffläche **hull_0** in der Intersection Snake **n0**. Ebenso die Deckfläche **deck_0** in der Intersection Snake **n2**.



Modell transom-model.ms2 – Spiegel-Basisfläche *transom_0* schneidet Rumpf und Deck in den Intersection Snakes *n0* und *n2*.

Beide Snakes werden als Projected Snakes *n1* und *n3* auf die Spiegel-Basisfläche projiziert. Dann wird mit diesen beiden Snakes die fertige Spiegelfläche erzeugt, die Trimmed Surface *transom*.



Modell transom-model.ms2 – fertige Spiegelfläche *transom* (Trimmed Surface)

Abschließend werden noch zwei Entity Lists erstellt. Die Entity List *parents* enthält die Flächen *hull_0* und *deck_0*. Die Entity List *products* enthält nur die Trimmed Surface *transom*. Der Zweck dieser Listen wird im folgenden Abschnitt erklärt.

Zum Schluß wird das Quell-Modell für die Spiegel-Komponente gespeichert als *transom-model.ms2*.

Speichern der Komponente Spiegel

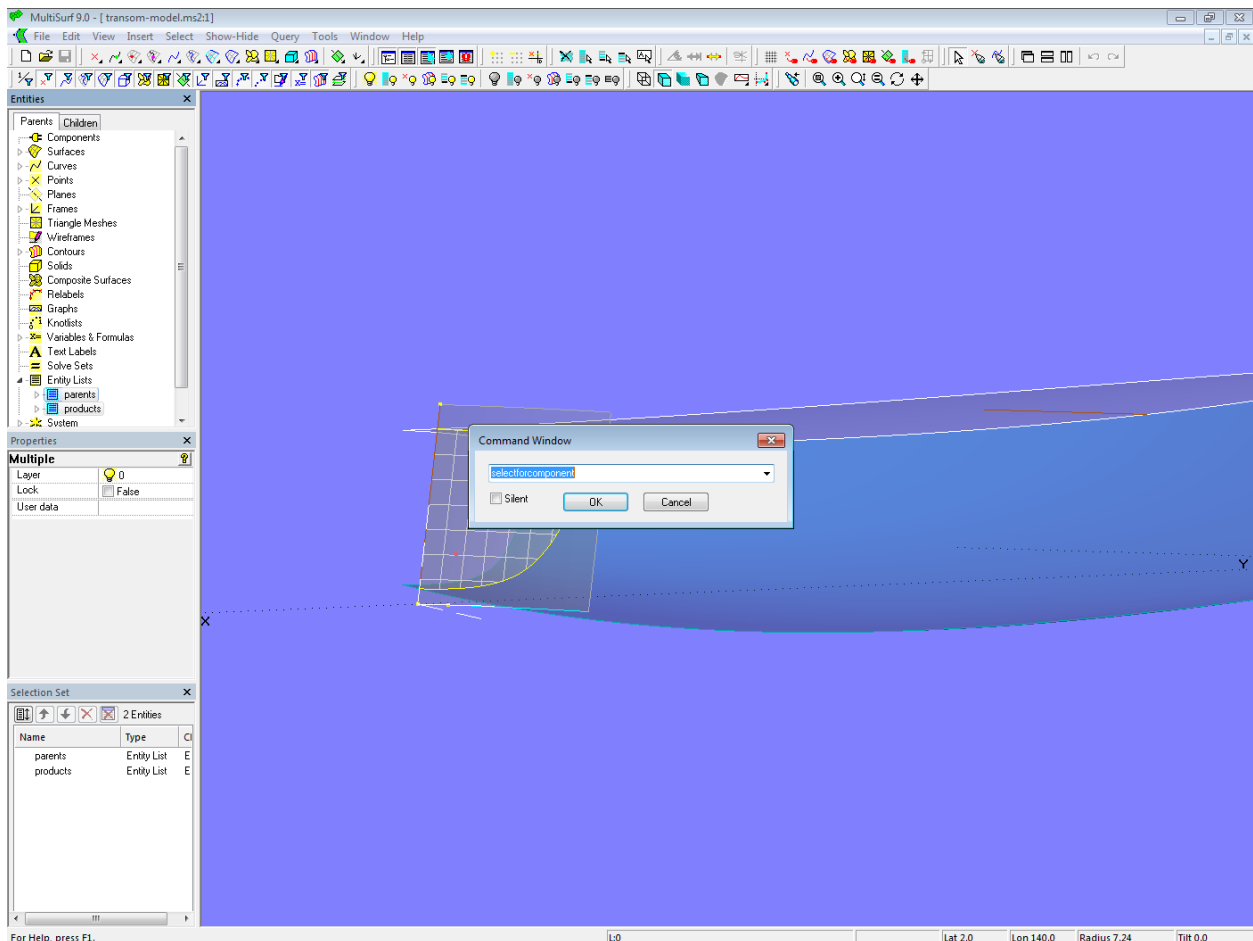
Command **SelectForComponent** – das perfekte Werkzeug für das Erstellen einer Komponente

Wir wollen nun das Spiegel-Modell als Komponente speichern. Das Host-oder Arbeitsmodell, also das Modell, in das die Komponente eingefügt werden soll, muß eine Rumpf- und eine Deckfläche zur Verfügung stellen (Eltern der Komponente). Wir müssen also alle Objekte auswählen, die den Spiegel erzeugen, aber nicht diese beiden Flächen. Würden wir Rumpf und Deck auswählen und dann wie beim Beispiel der Winch-Komponente wieder mit **Select/ Children/ All Generations** arbeiten, wären eventuell abhängige Objekte (Kinder) dieser Flächen enthalten, die aber mit dem Spiegel nichts zu tun haben, wie zum Beispiel die Contours *dwl_0*, *buttock_0* etc., die ebenfalls im Quell-Modell enthalten sind und von Deck und Rumpf abhängen. Sie sind momentan zwar ausgeblendet, würden aber mit ausgewählt.

Ist das Quell-Modell speziell für die Erstellung einer Komponente gemacht und wie bei der Winch-Komponente nur ein einziger Anschluß vom Host-Modell erforderlich, ist die Auswahl über **Select/ Children/ All Generations** am einfachsten. Will man aber aus einem umfangreichen Geometriemodell eine Komponente herauslösen oder Objekte für eine Komponente auswählen, die mehr als einen Anschluß vom Host-Modell benötigt, kann es rasch kompliziert werden, weil man immer alle Abhängigkeiten beachten muß, um nicht Eltern zu bekommen, die man nicht will.

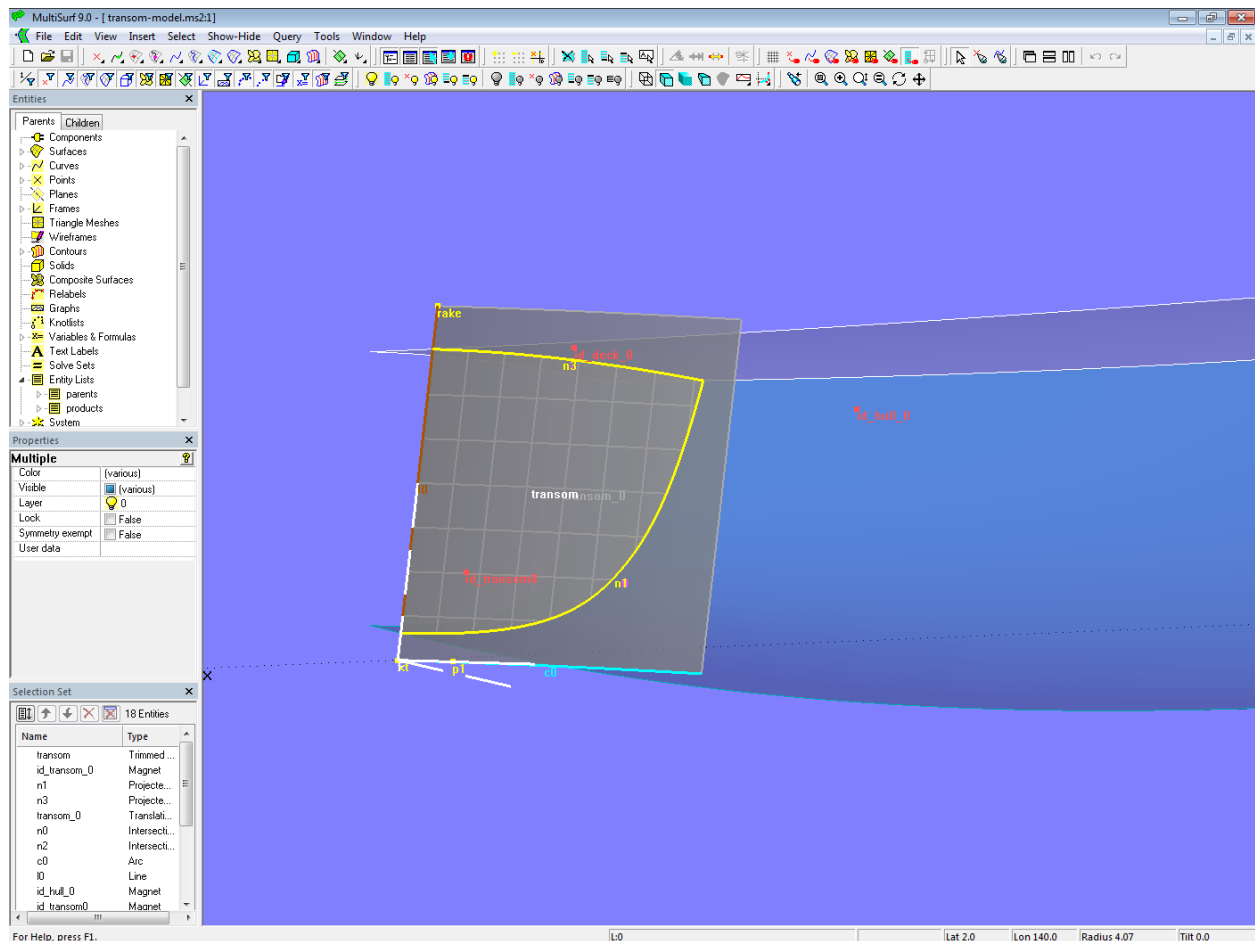
Um den Auswahlprozess zu vereinfachen, gibt es in MultiSurf den Befehl **SelectForComponent**. Er ist überaus hilfreich beim Erstellen von Komponenten. Denn er liefert den minimalen Auswahlset an Objekten, die erforderlich sind, um für die in der Entity List „parents“ angegebenen Eltern die in der Entity List „products“ aufgeführten Objekte zu generieren. Der Befehl **SelectForComponent** wählt schnell und effektiv alle nötigen Komponenten-Objekte aus.

Wählen wir darum nun die beiden Entity Lists **parents** und **products** aus und geben über **Tools/ Command Window** den Befehl **SelectForComponent** ein.



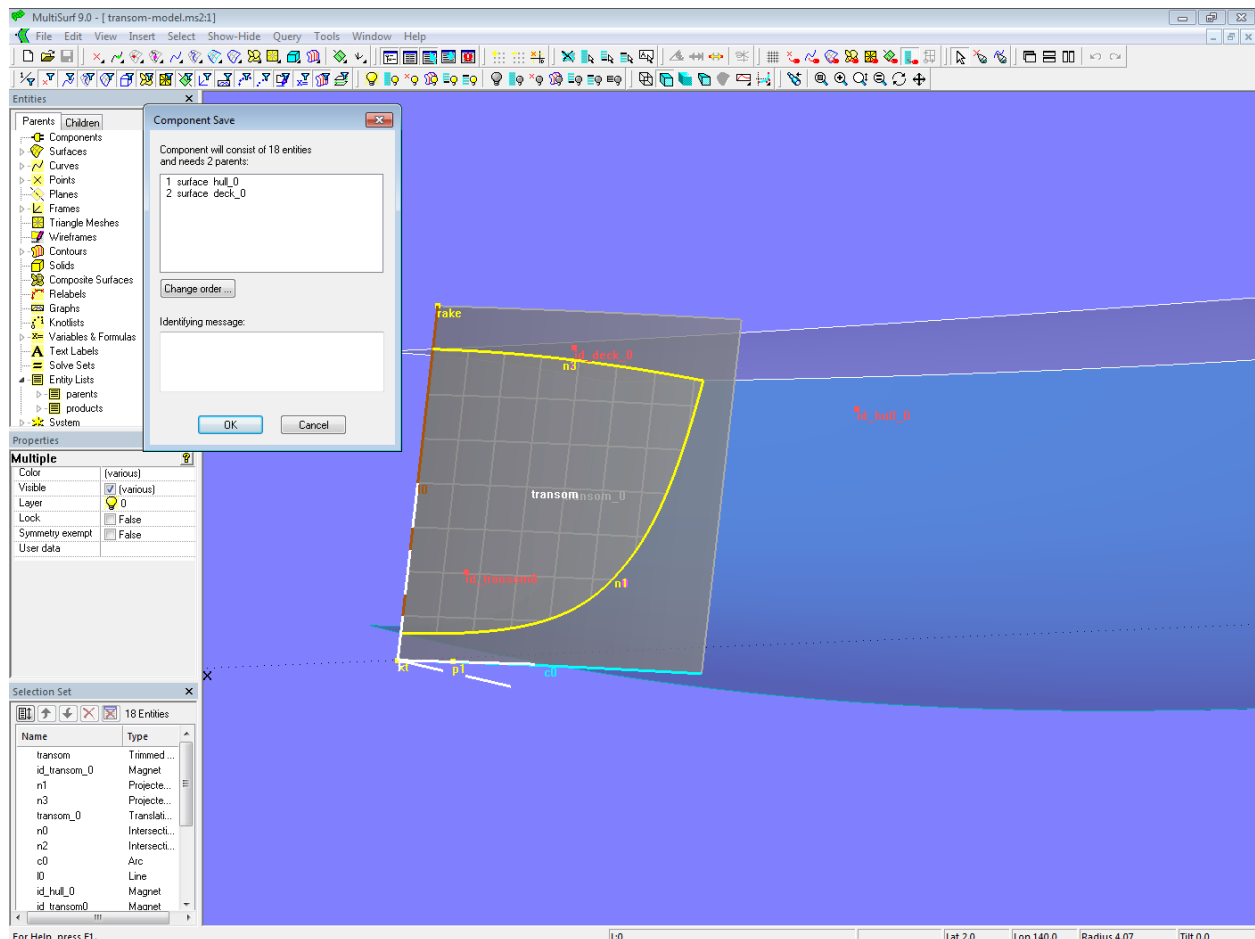
Modell transom-model.ms2 – Auswahl der Komponenten-Objekte mit dem Befehl "SelectForComponent"

Nach **OK** enthält dann der Selection Set-Manager alle notwendigen Objekte für die Spiegel-Komponente.



Modell transom-model.ms2 – mit dem Befehl “SelectForComponent“ ausgewählte Objekte für die Spiegel-Komponente

Weiter geht es mit **File/ Component/ Save**. Es erscheint das „Component Save“-Dialogfenster, das anzeigt, welche Eltern die Komponente vom Host-Modell erwartet.



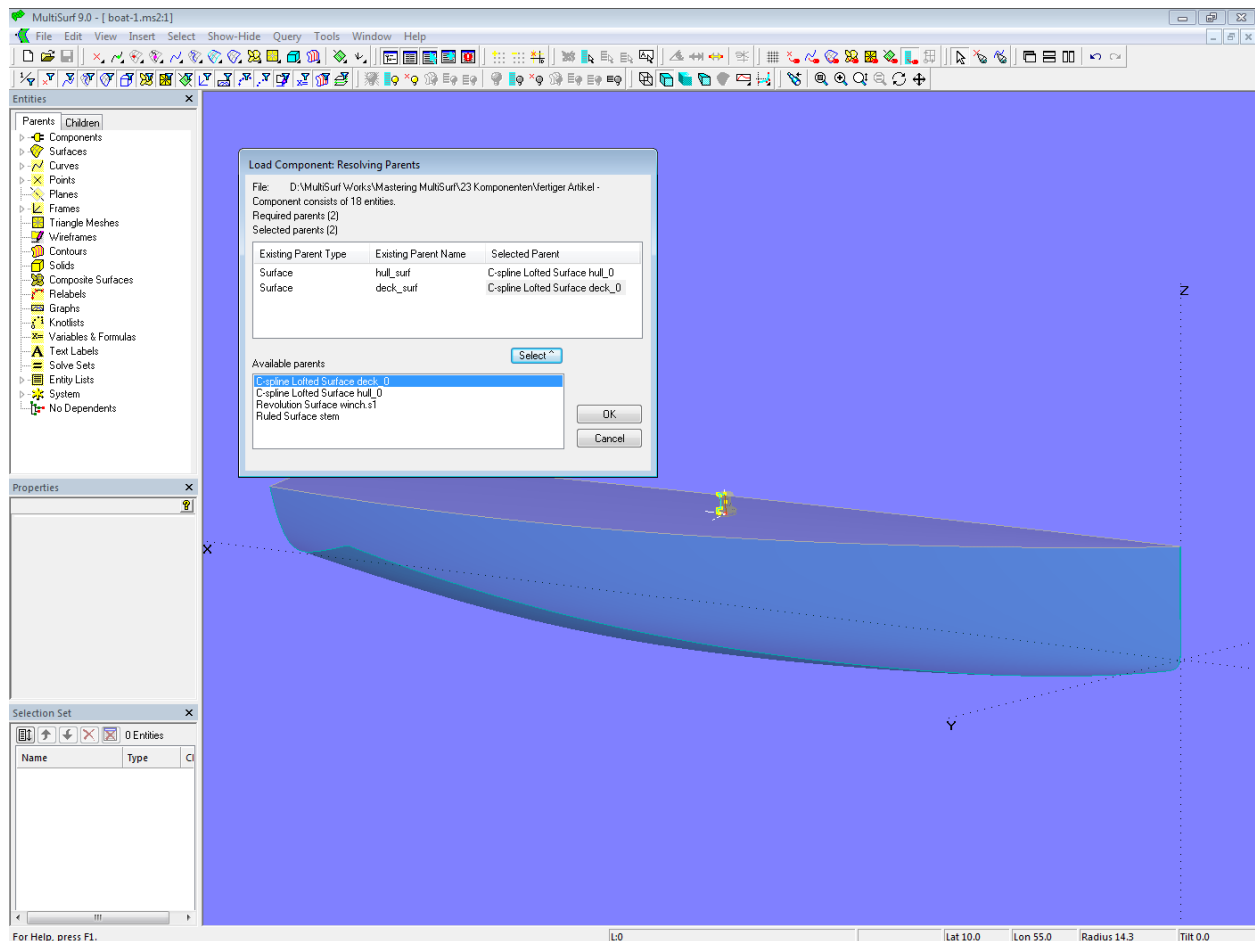
Modell *transom-model.ms2* – die Spiegel-Komponente benötigt 2 Eltern im Arbeitsmodell.

Dann **OK** und speichern der Spiegel-Komponente unter dem Namen *transom-component.mc2*.

Laden der Komponente Spiegel

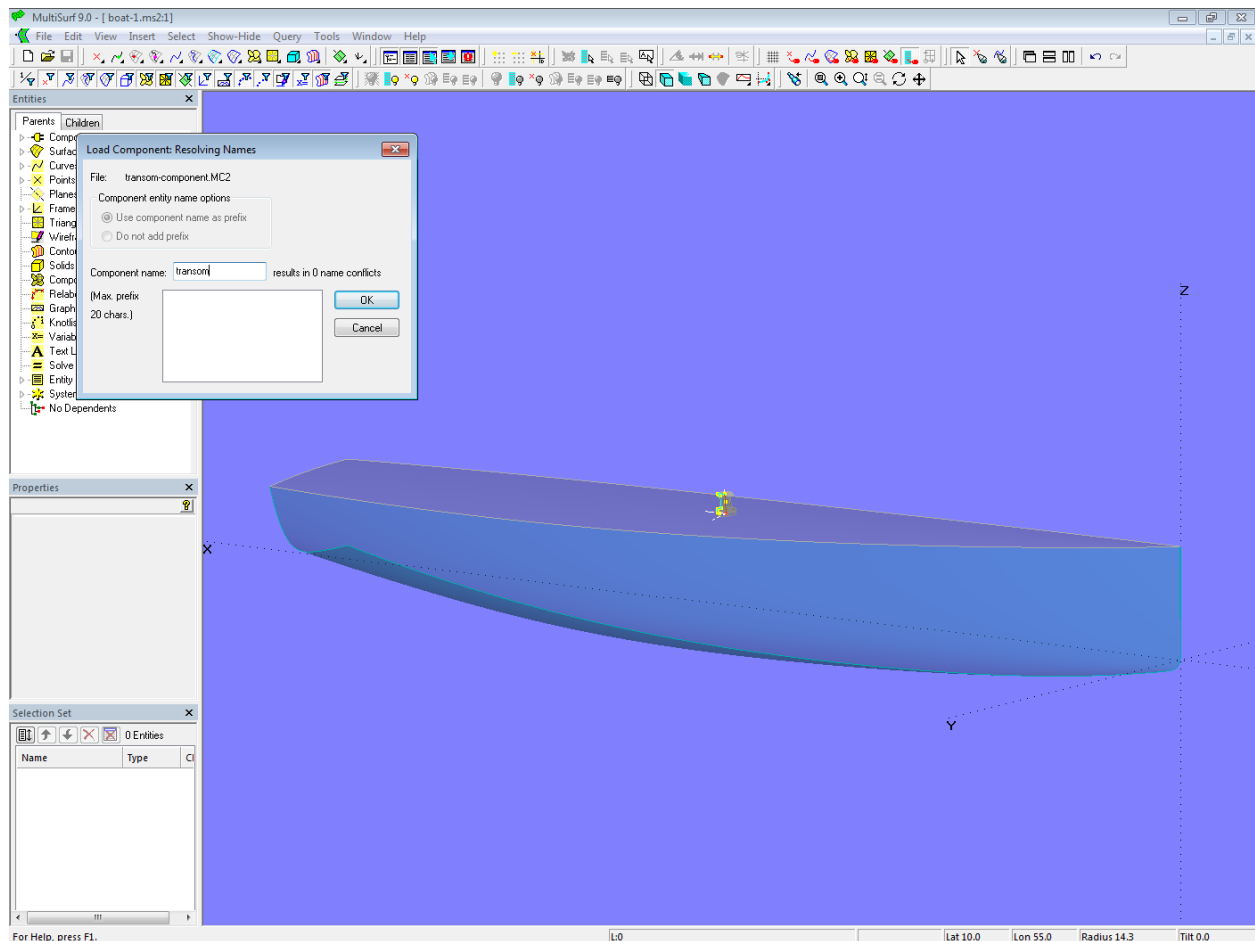
Nun wollen wir die Spiegel-Komponente in das Arbeitsmodell *boat-1.ms2* einfügen. Das Modell laden und dann über **File/ Component/ Load** die Komponentendatei *transom-component.mc2* auswählen und öffnen. Oder alternativ Rechts-Mausklick auf "Components" im Entities Manager und im Kontextmenü auswählen "Load".

Im Fenster „Load Component: Resolving Parents“ die entsprechenden Flächen auswählen, die von der Komponente verwendet werden sollen. Dann **OK**.



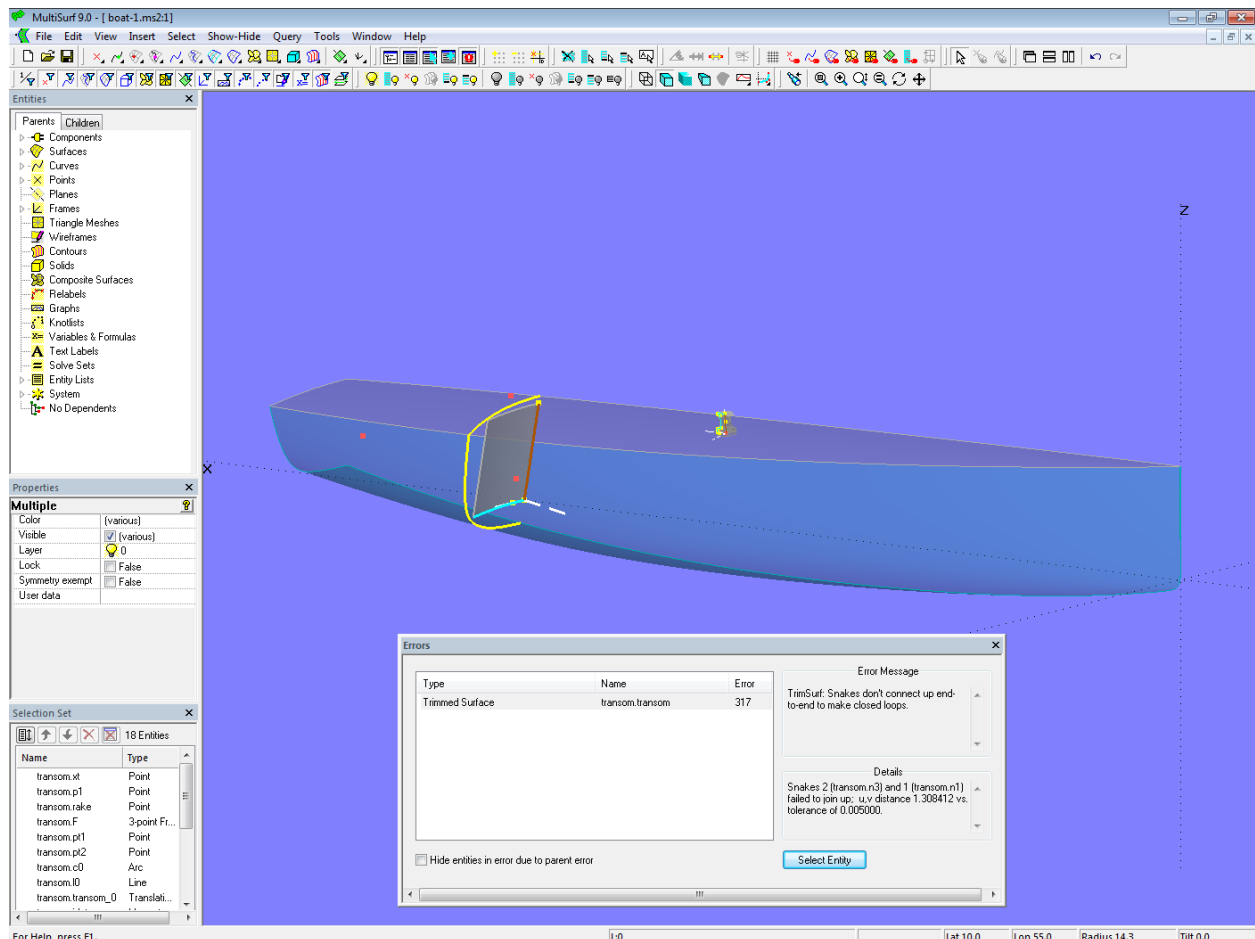
Modell boat-1.ms2 – Auswahl der Eltern beim Laden der Spiegel-Komponente in das Arbeitsmodell

Im Fenster „Load Component: Resolving Names“ als Name der Komponente eingeben: **transom**, dann **OK**.



Modell boat-1.ms2 – Eingabe des Namens beim Laden der Spiegel-Komponente in das Arbeitsmodell

Die Objekte der Komponente werden angezeigt, aber auch eine Fehlermeldung.

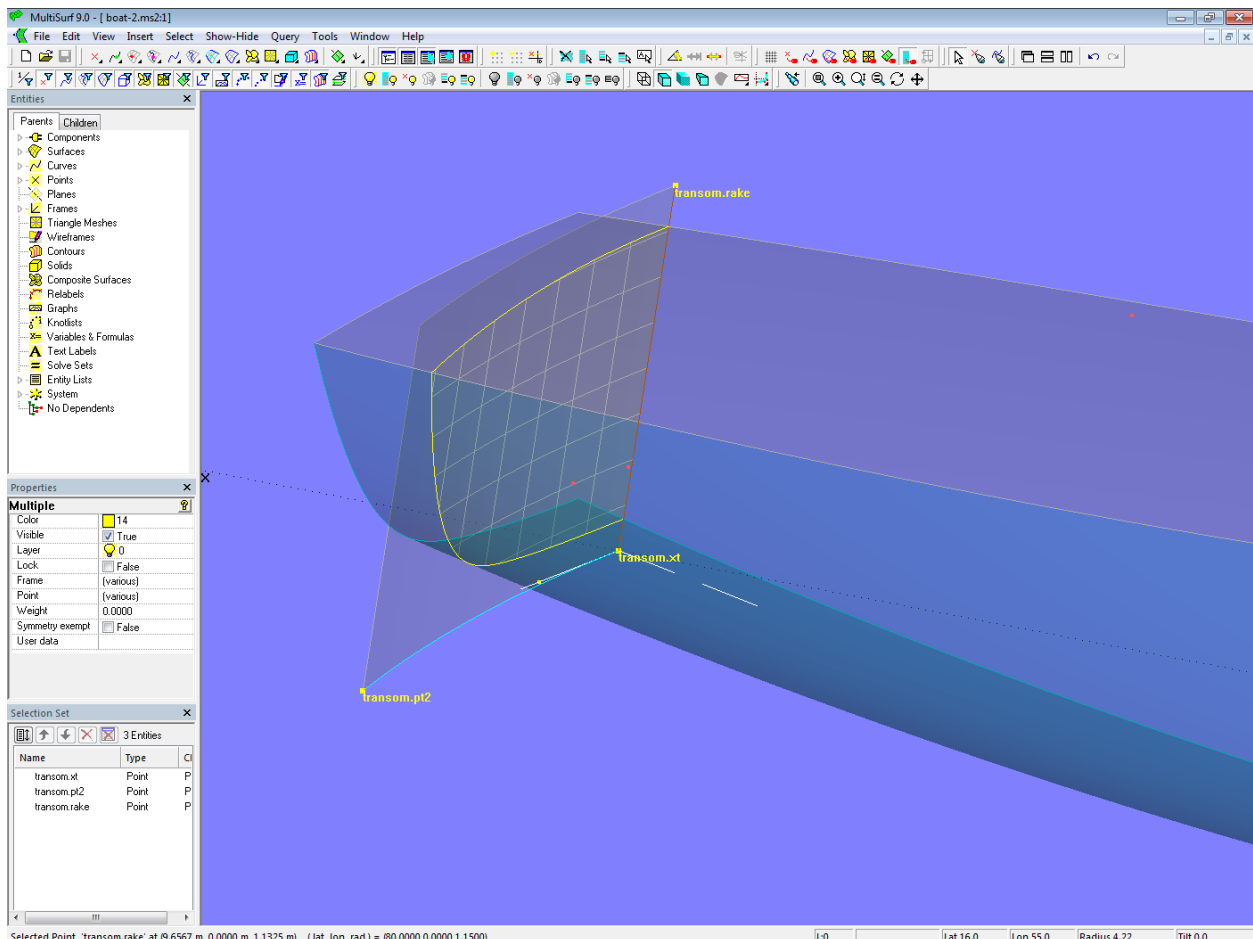


Modell boat-1.ms2 – die Spiegel-Komponente verursacht im Arbeitsmodell eine Fehlermeldung.

Der Grund für die Fehlermeldung ist unschwer zu erkennen: die Basisfläche **transom.transom_0** ist sowohl in der Breite wie in der Höhe zu klein, außerdem liegt sie ziemlich weit vorne. Denn im Quell-Modell, aus dem die Komponente gespeichert wurde, ist der Rumpf (LOA = 8 m) kleiner als der Rumpf im Arbeitsmodell, in das sie geladen wird (LOA = 10 m).

Was muß man tun? Lediglich Point **transom.xt** in den Bereich des achterlichen Überhangs schieben, und mit Point **transom.pt2** die Breite und mit Point **transom.rake** die Höhe die Basisfläche **transom.transom_0** an Rumpf und Deck anpassen.

Dann Abspeichern des Modell als **boat-2.ms2**.



Modell boat-2.ms2 – fertiger Kreiszylinder-Spiegel nach Anpassen der Basisfläche im Arbeitsmodell

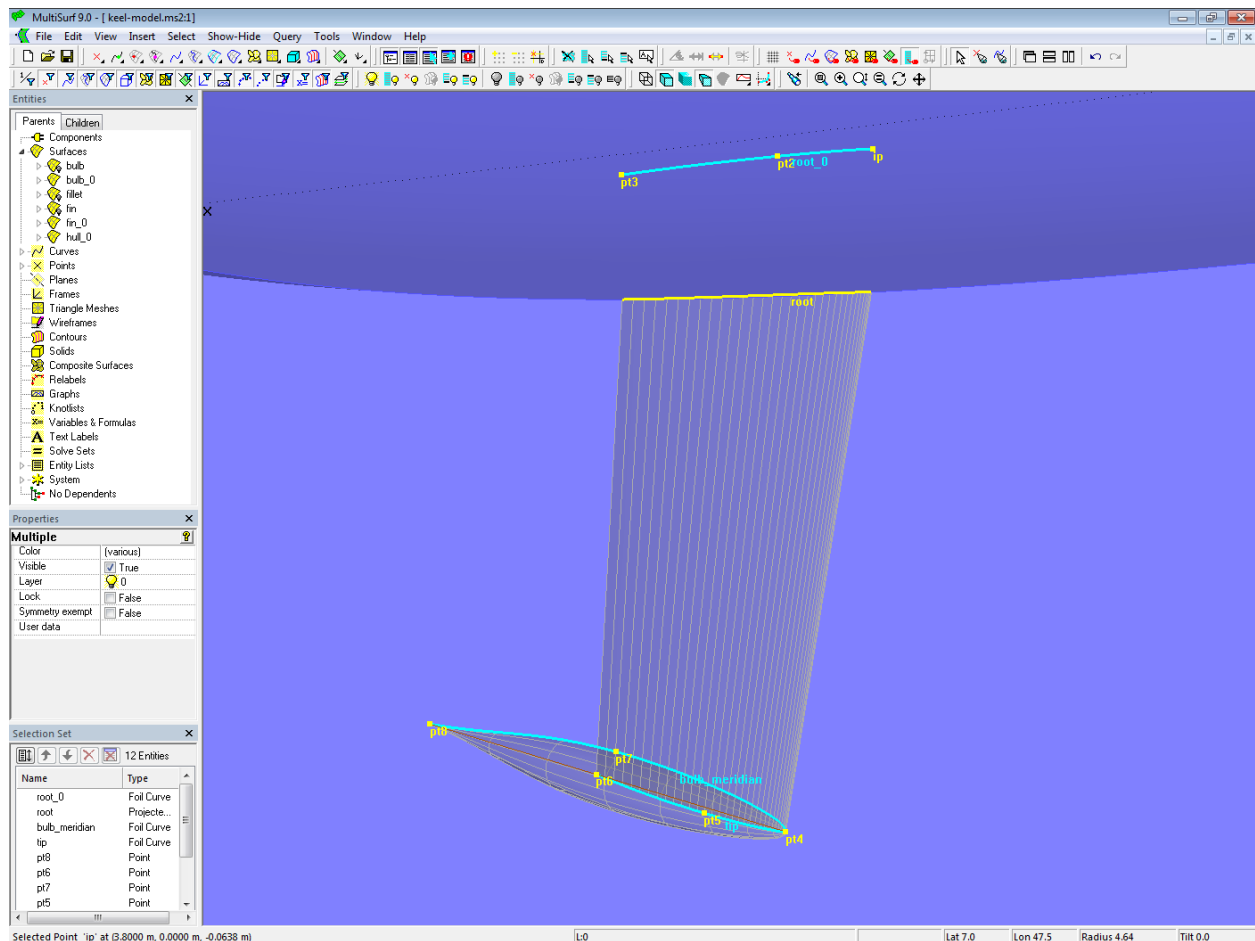
So viel zur Komponente Spiegel.

Komponente Kiel

Quell-Modell Kiel

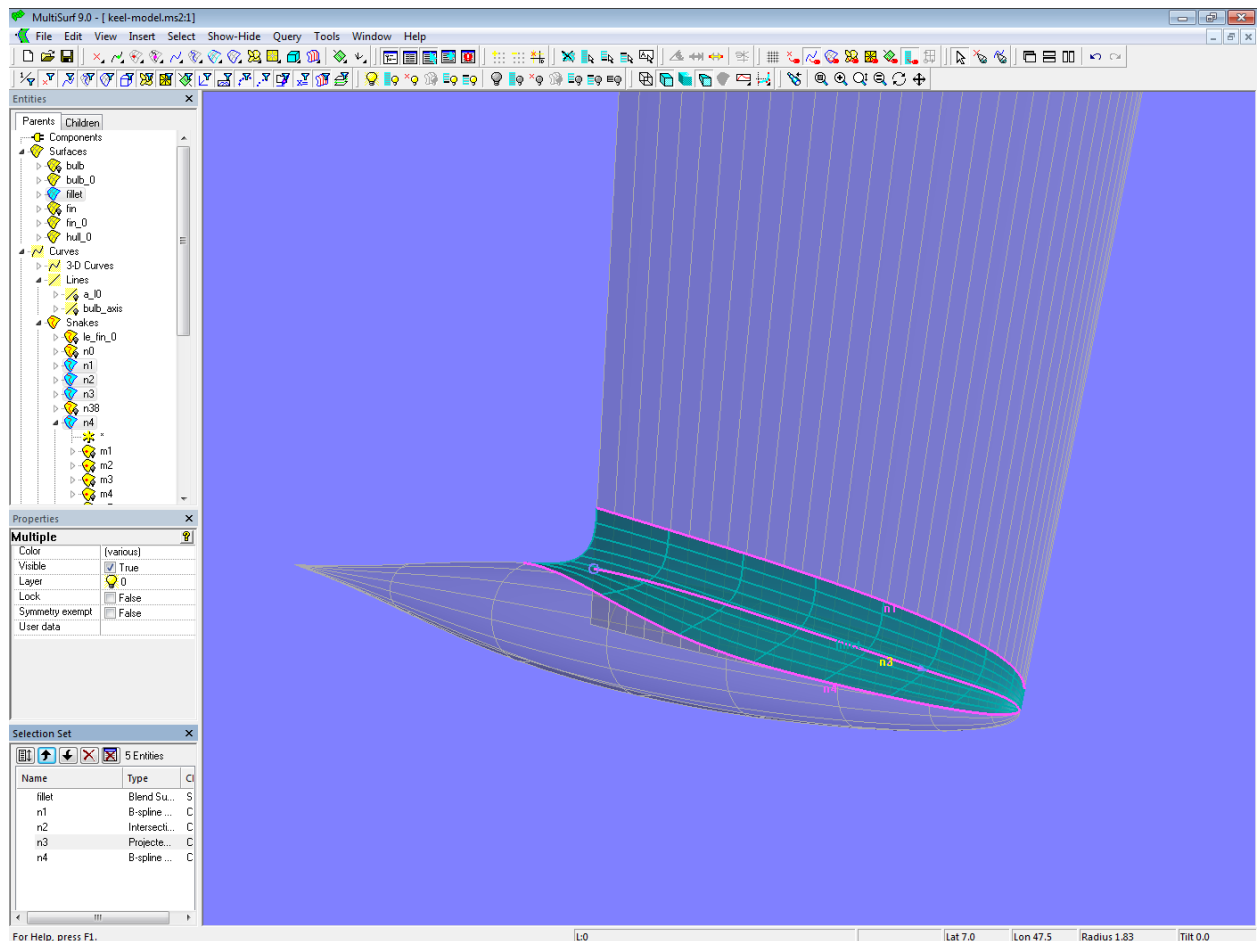
Ausgangspunkt ist das Modell *keel-model.ms2*. Es enthält einen Kiel, bestehend aus Flosse und Ballastbulb sowie einer Ausrundung zwischen beiden. Das Wurzelprofil der Flosse auf der X-Achse wird durch die Foil Curve *root_0* gebildet, definiert mit den Points *ip* (Vorderkante), *pt2* (Profildicke) und *pt3* (Hinterkante). Das Wurzelprofil auf der Rumpffläche entsteht durch die Projected Snake *root*. Die Foil Curve *tip* bestimmt das Spitzenprofil, definiert mit den Points *pt4* (Vorderkante), *pt5* (Profildicke) und *pt6* (Hinterkante). Zwischen den Kurven *root* und *tip* ist die Basisfläche der Flosse aufgespannt, die Ruled Surface *fin_0*.

Der Ballastbulb ist eine Rotationsfläche. Meridian ist die Foil Curve *bulb_meridian*, bestimmt mit den Points *pt4* (Vorderkante), *pt7* (Dicke) und *pt8* (Hinterkante). Drehachse ist Line *bulb_axis* zwischen *pt4* und *pt8*.



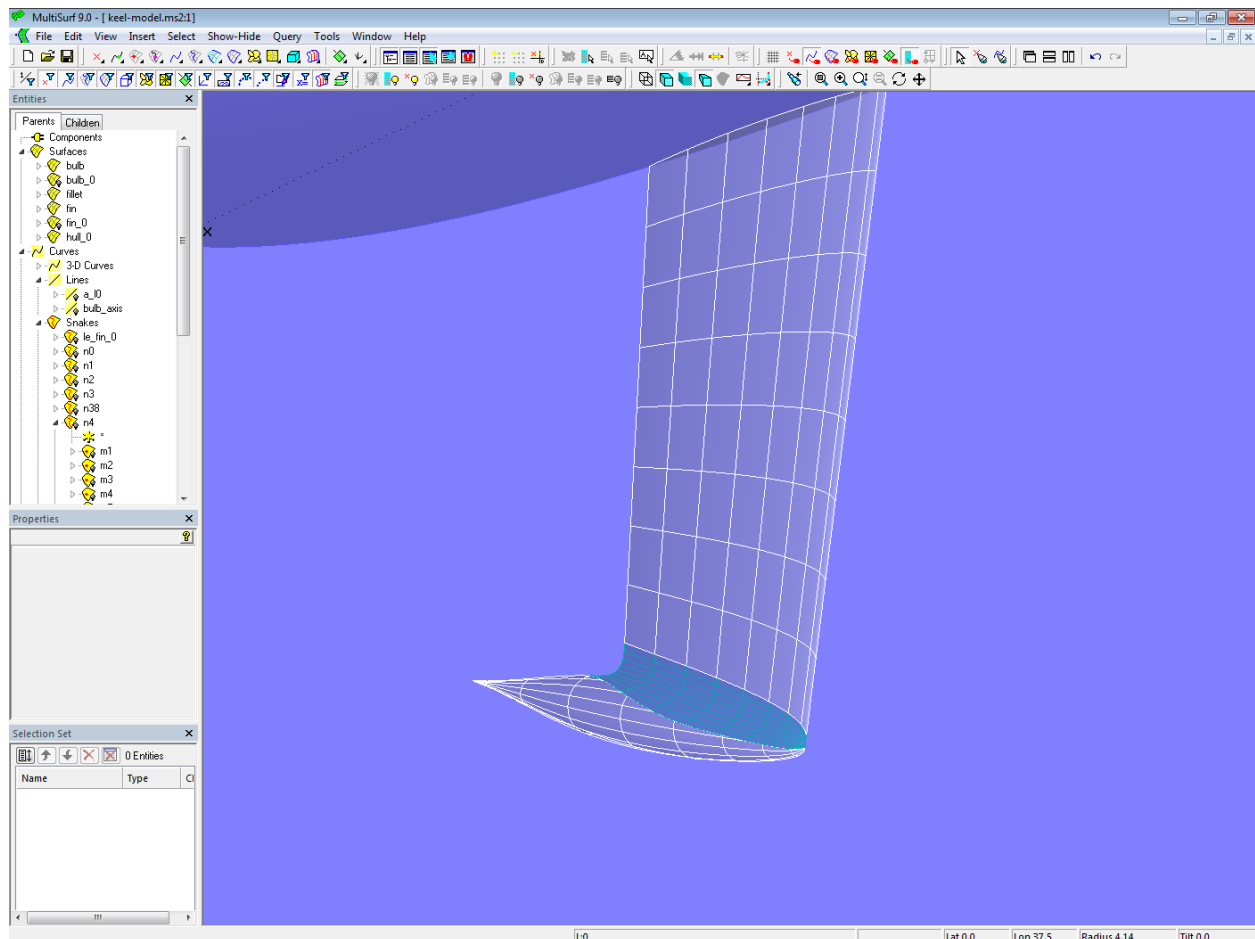
Modell keel-model.ms2 – Kielgeometrie im Quell-Modell

Die Ausrundung **fillet** zwischen Flosse und Bulb ist eine Blend Surface. Sie erfordert 4 Snakes, zwei auf der einen und zwei auf der anderen Fläche. B-spline Snake **n1** legt den Beginn der Ausrundung auf der Flosse fest. Intersection Snake **n2** ist die Durchdringung der Flosse durch den Bulb. Sie wird ihrerseits dann als Projected Snake **n3** auf die Bulbfläche projiziert. Das Ende der Ausrundung auf dem Bulb bestimmt B-spline Snake **n4**.



Modell keel-model.ms2 – Ausrundung zwischen Flosse und Bulb mit einer Blend Surface

Die Fläche von der Wurzel der Flosse bis zum Beginn der Ausrundung **fillet** ist SubSurface **fin**, Trimmed Surface **bulb** ist die Fläche des Bulbs abzüglich Ausrundung.

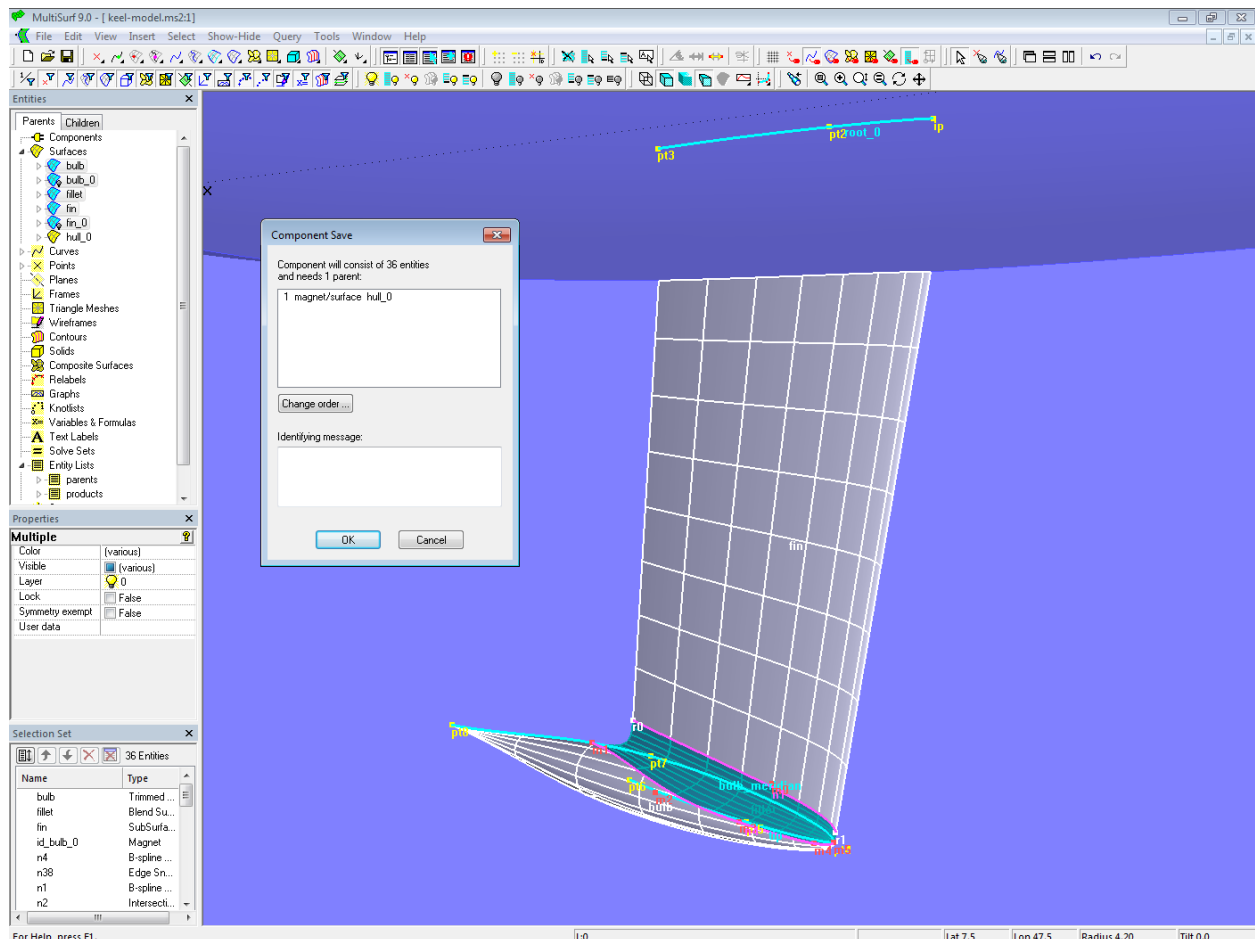


Modell keel-model.ms2 – fertige Flächen des Kiels im Quell-Modell

Speichern der Komponente Kiel

Es ist praktisch, wenn vor dem Speichern einer Komponente nur ihre wesentlichen Flächen, Ziehpunkte und Cps sichtbar sind, alles andere aber verborgen ist. Nach dem Einfügen der Komponente ist das Arbeitsmodell übersichtlicher und die Komponentengeometrie einfacher zu verändern, wenn nur die für die Form letztlich relevanten Objekte angezeigt werden.

Mit den Entity Lists **parents** und **products** und dem Befehl **SelectForComponent** nun alle Objekte für die Komponentengeometrie auswählen und das Ergebnis speichern über **File/ Component/ Save**. Im „Component Save“-Dialogfenster wird angezeigt, welche Eltern das Hostmodell bereitstellen muß.



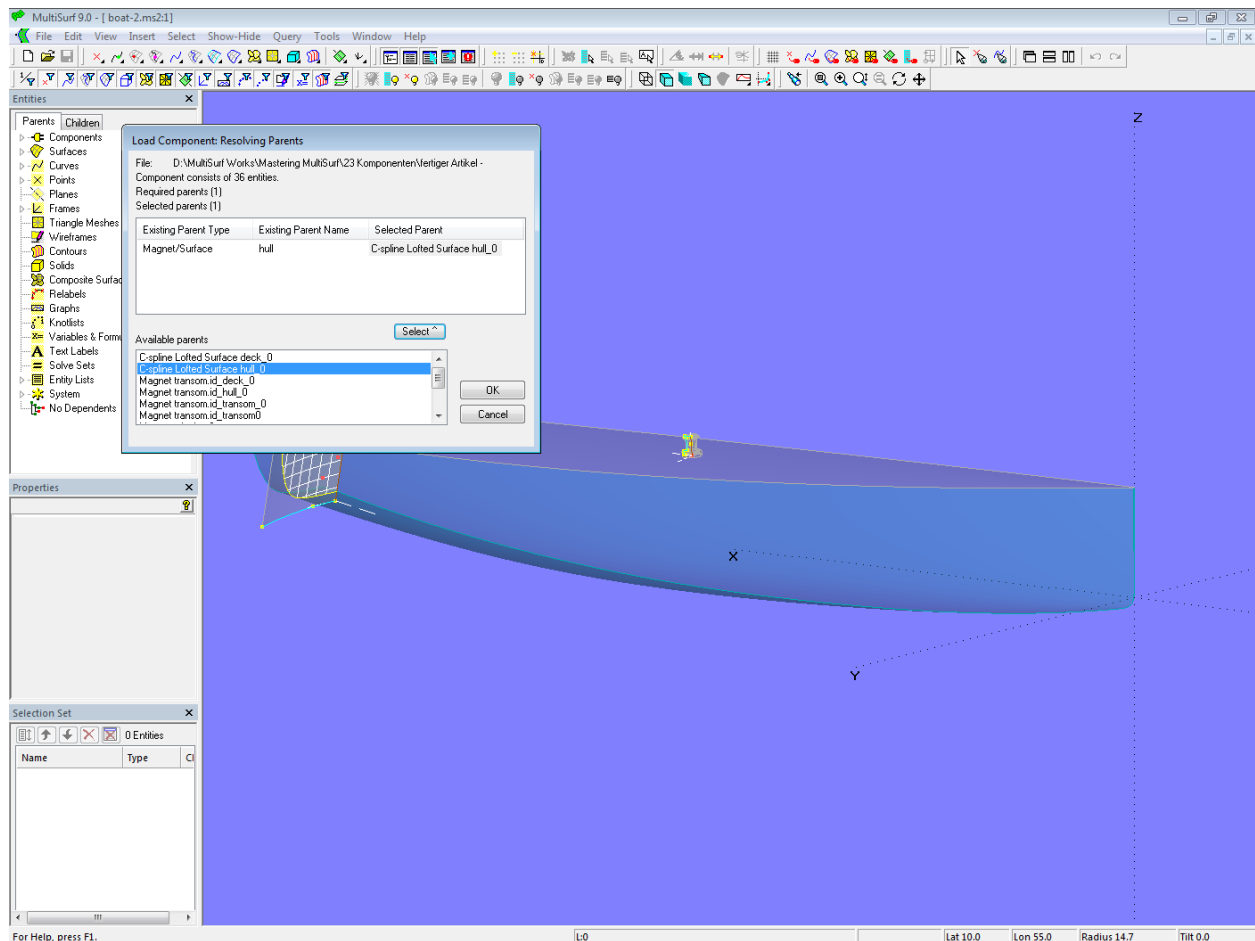
Modell keel-model.ms2 – ausgewählte Objekte für die Kiel-Komponente im Quell-Modell

Dann **OK** und speichern der Kiel-Komponente unter dem Namen *keel-component.mc2*.

Laden der Komponente Kiel

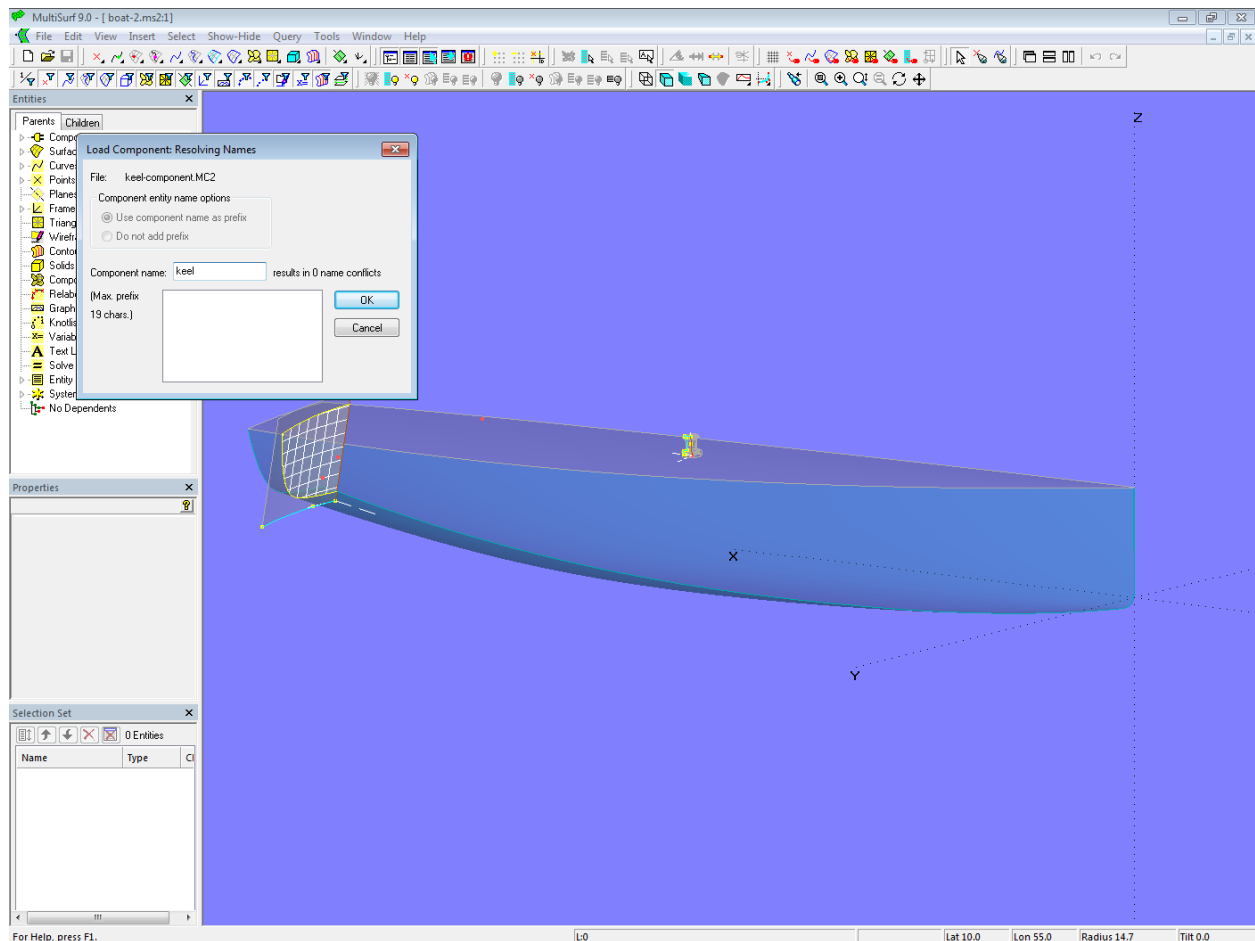
Nun soll die Kiel-Komponente in das Modell *boat-2.ms2* geladen werden. Das Modell öffnen und dann über **File/ Component/ Load** oder das „Components“-Kontextmenü im Entities Manager die Komponententdatei *keel-component.mc2* auswählen und öffnen.

Im Fenster „Load Component: Resolving Parents“ nun die entsprechende Fläche auswählen, an die die Komponente angeschlossen werden soll (**hull_0**). Dann **OK**.



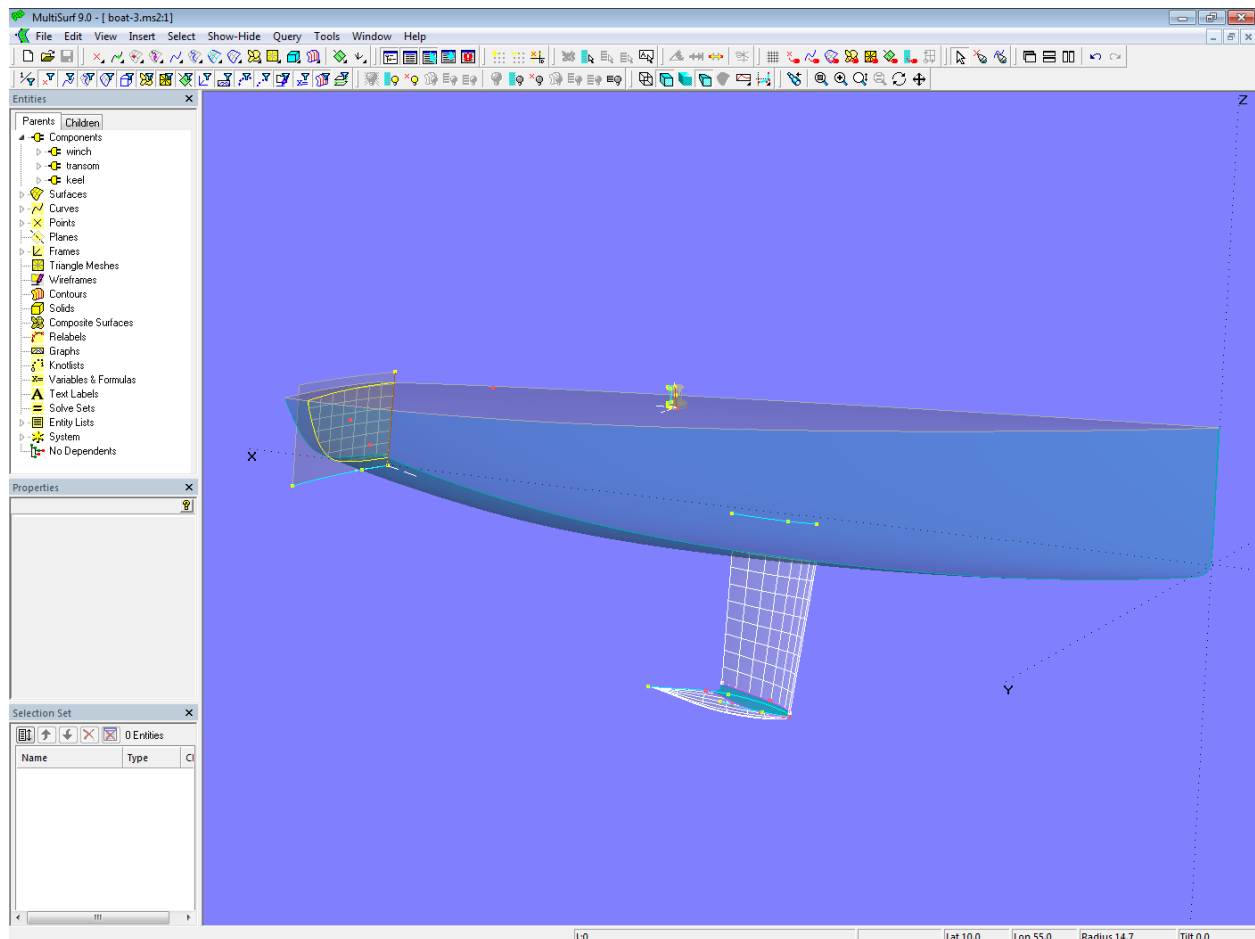
Modell boat-2.ms2 – Auswahl der Eltern der Kiel-Komponente beim Laden in das Host-Modell

Im Fenster „Load Component: Resolving Names“ als Name der Komponente eingeben: **keel**, dann **OK**.



Modell boat-2.ms2 – Eingabe des Namens der Kiel-Komponente beim Laden in das Host-Modell

Dann Abspeichern des Modell als *boat-3.ms2*.



Modell boat-3.ms2 – Kiel-Komponente eingeladen

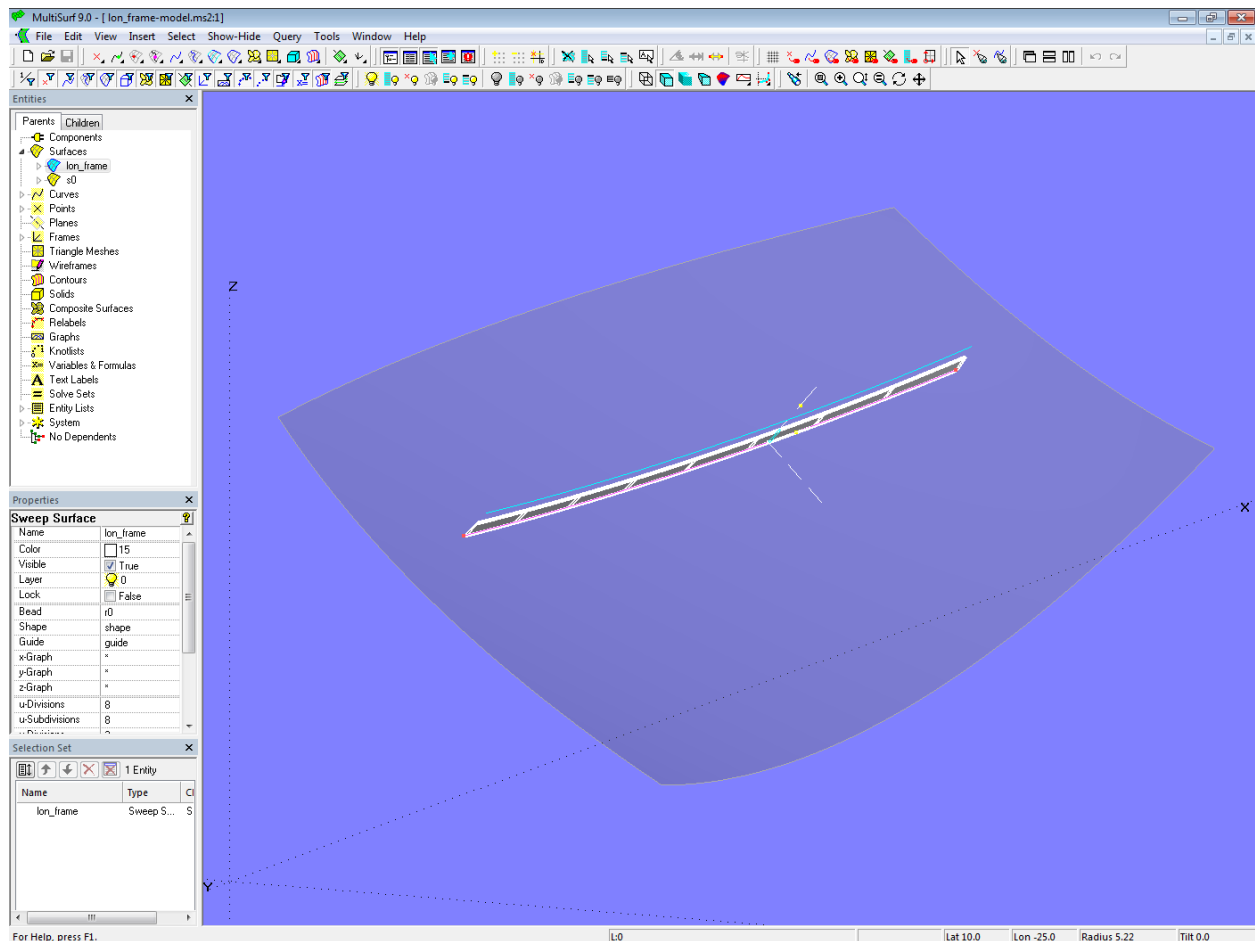
So weit zur Komponente Kiel.

Komponente Längsspant

Wird eine Rumpffläche mit Längsspanten ausgesteift, gibt es typischer Weise mehrere davon. Statt sie mehrmals auf die gleiche Weise Objekt für Objekt zu konstruieren, erspart eine Komponente Zeit und Arbeit.

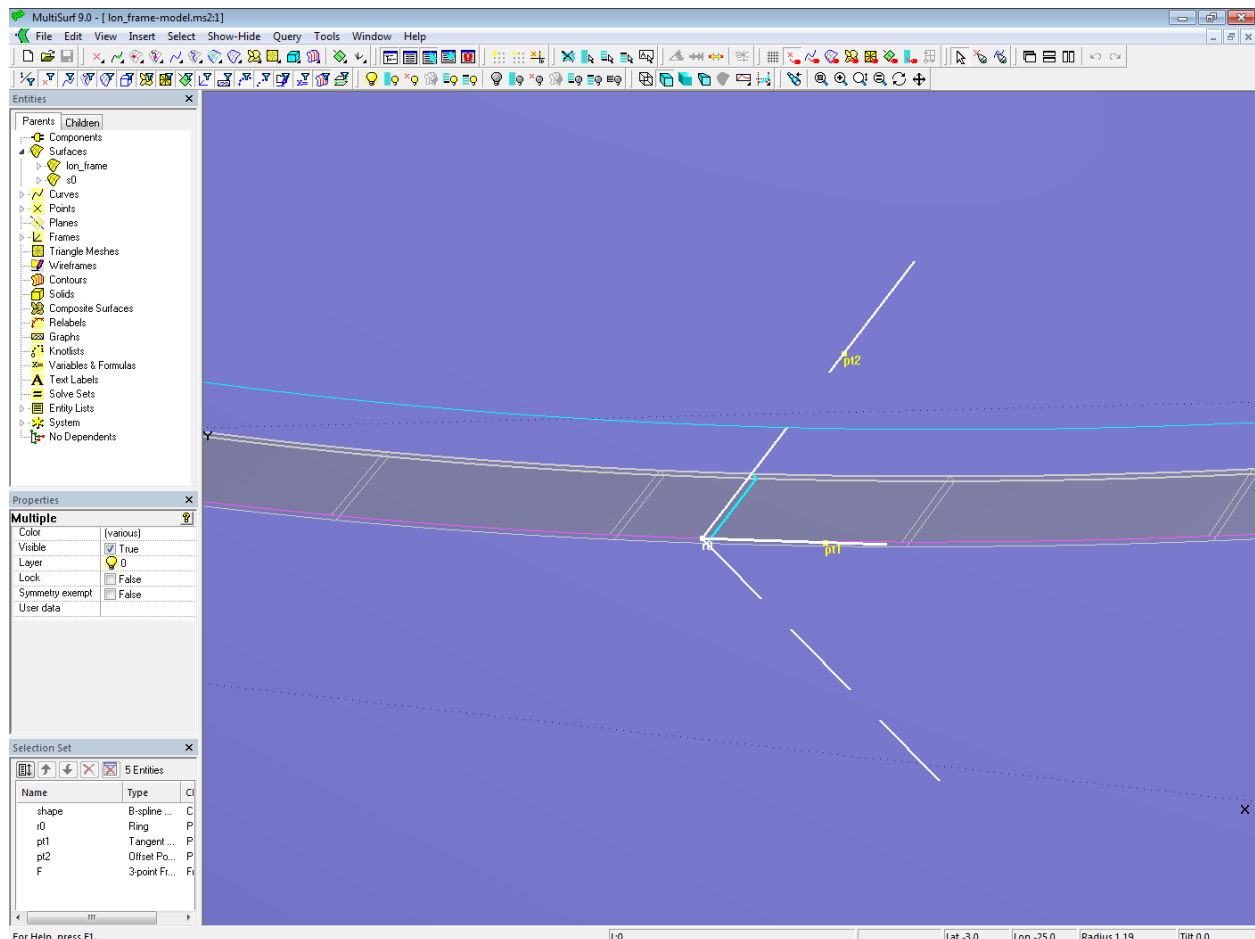
Quell-Modell Längsspant

Die Konstruktion eines Längsspants ist übersichtlich. Betrachten wir hierzu das Geometriemodell *lon_frame-model.ms2*. Die Fläche selbst ist eine Sweep Surface. Zu ihrer Definition braucht man einen Pfad, eine Querschnittsform des Spantprofils, und eine Kurve, die die Verwindung des Längsspants bestimmt.



Modell lon_frame-model.ms2 – Quell-Modell der Komponente für einen Längsspant

Der Pfad ist die Anschlußstelle der Komponente an das Host-Modell, wird also von diesem bereitgestellt. Darum reicht im Quell-Modell die einfache Line Snake [path](#) aus. Zunächst wird auf dem Pfad des Längsspants ([path](#)) der Ring [r0](#) erzeugt, zu diesem dann der Tangent Point [pt1](#) sowie der Offset Point [pt2](#). Mit diesen drei Punkten wird dann der 3-point Frame [F](#) bestimmt. In diesem Koordinatensystem sind die Eckpunkte des Spantprofils definiert, dass durch die B-spline Curve [shape](#) (Degree = 1) dargestellt ist. Hier im Beispiel ist die Querschnittsform ein einfaches Rechteck, andere Profilformen sind einfach zu modellieren.

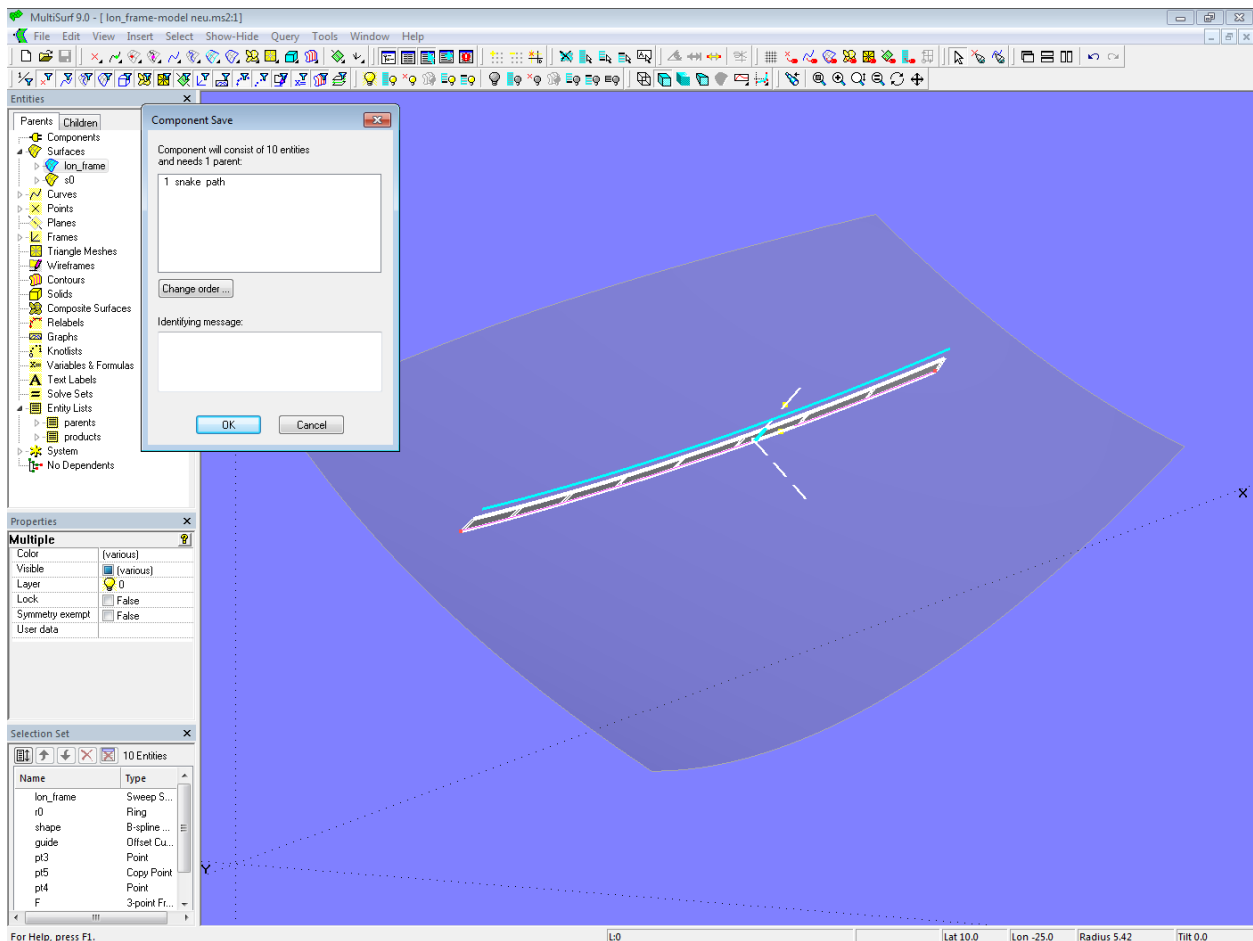


Modell *lon_frame-model.ms2* – Koordinatensystem für Spantprofil

Den eigentlichen Längsspannt bildet die Sweep Surface *lon_frame*. Damit das Profil überall senkrecht zur Rumpfoberfläche steht, verwendet die Sweep Surface als Guide eine Offset Curve (*guide*).

Speichern der Komponente Längsspannt

Mit den beiden Entity Lists *parents* und *products* und dem Befehl **SelectForComponent** nun alle Objekte für die Komponentengeometrie auswählen und das Ergebnis speichern über **File/ Component/ Save** (Dateiname: *lon_frame-component.mc2*). Das „Component Save“-Dialogfenster zeigt an, welche Eltern das Host-Modell bereitstellen muß.



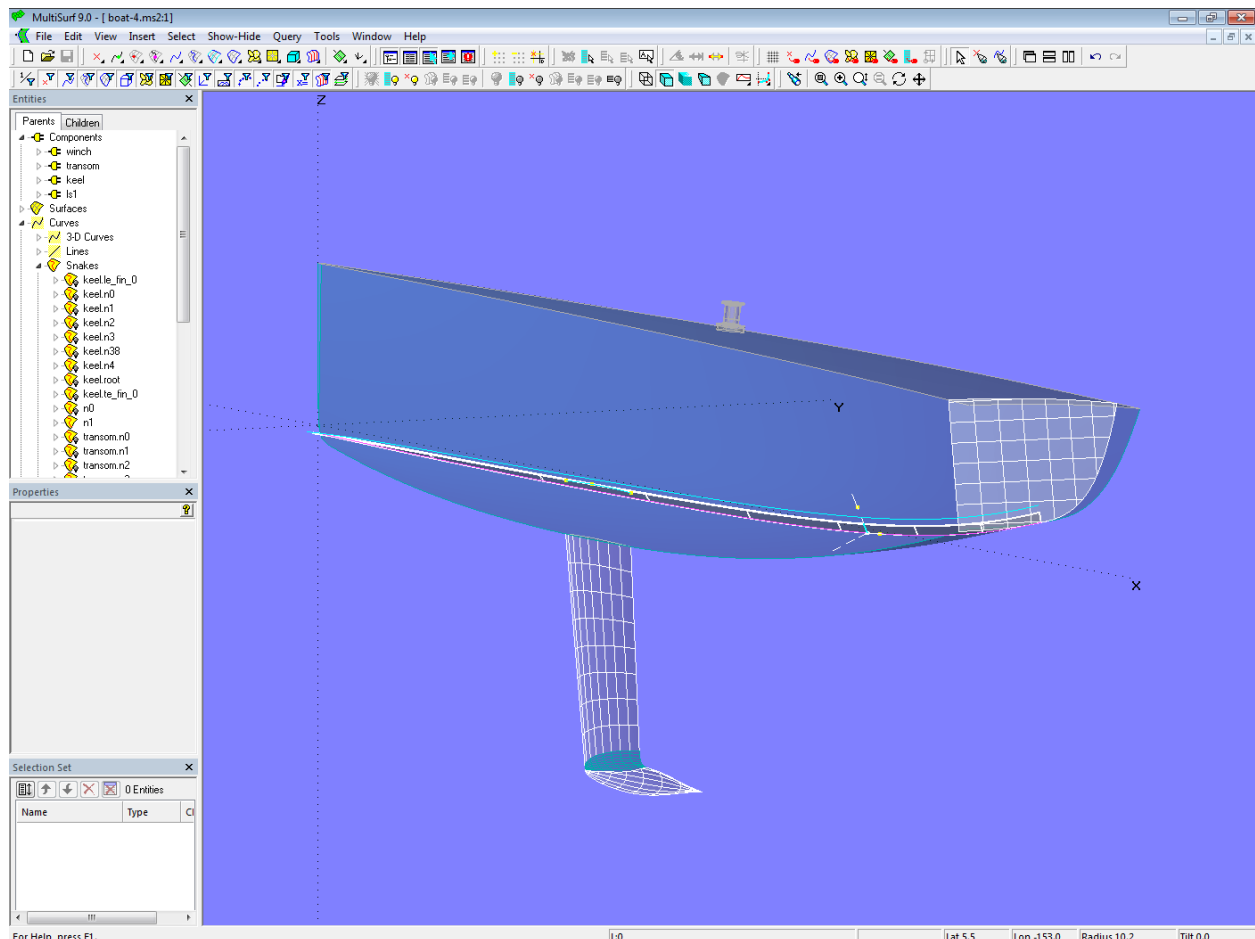
Modell *lon_frame-model.ms2* – Speichern der Objekte der Komponente

Laden der Komponente Längsspant

Wie in den vorangehenden Beispielen soll nun auch die Längsspant-Komponente in ein Arbeitsmodell eingefügt werden. Verwenden wir hierzu das Modell *boat-3.ms2*.

In diesem Modell gibt es für den Pfad der Komponente die in Längsrichtung laufende Snake **n1**. Ist **n1** ausgewählt, ist im Properties Manager ersichtlich, dass **n1** eine UVSnake ist, bestimmt mit Magnet **m0**. Natürlich ist dies eine einfache Konstruktion, mit wenig Einfluß auf den Pfadverlauf. Soll der Pfad in einer Sentebene liegen oder parallel zur Mittschiffsebene verlaufen, muß man die Rumpffläche mit einer entsprechenden Ebene schneiden (Intersection Snake). Um den Verlauf frei zu formen, kann man eine C-spline Snake oder B-spline Snake mit mehreren Cps (Magnete, Ringe) verwenden. Oder man projiziert eine mit mehreren Cps definierte C-spline Curve als Projected Snake auf die Rumpffläche, wodurch der Einfluß ihrer u/v-Parameterkurven auf den Snake-Verlauf aufgehoben wird.

Das Laden der Längsspant-Komponente erfolgt in gleicher Weise wie in den vorangehenden Beispielen (Komponentenname **ls1**). Anschließend das Arbeitsmodell speichern unter *boat-4.ms2*.



Modell boat-4.ms2 – Längsspann-Komponente *ls1* eingefügt

Die vorgestellte Komponente kann für die Konstruktion anderer Bauteile auf einfache Weise abgewandelt werden. Sollen zum Beispiel wie bei einem Längsspann für das Deck oder einem Querspann für den Rumpf die Querschnitte der Sweep Surface stets senkrecht sein, kann man die Offset Curve *guide* durch eine Copy Curve ersetzen.

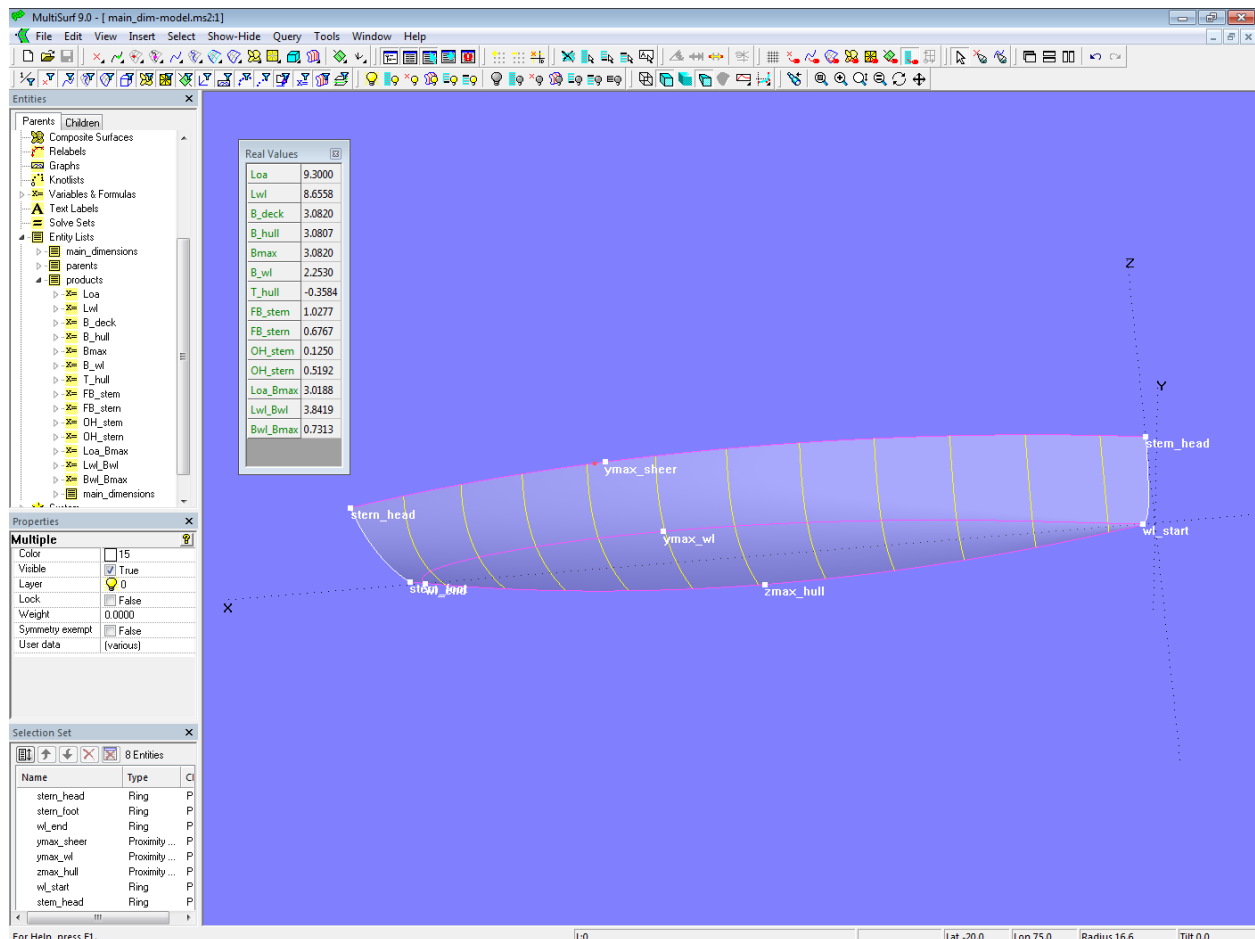
Komponente Hauptabmessungen

Im Tutorium 14, „*Rechnende Modelle in MultiSurf*“, wird über das Modell *main_dimensions.ms2* berichtet. Es ermittelt die Hauptabmessungen eines Bootsrumpfes. Mit Hilfe einer Komponente können die Berechnungsverfahren in andere Modelle eingefügt werden.

Quell-Modell Hauptabmessungen

Basismodell für die Komponente ist das Modell *main_dim-model.ms2*. Im Wesentlichen werden in den benötigten Formeln die Funktionen XPOS, YPOS und ZPOS verwendet, um die XYZ-Koordinaten von Punkten zu bestimmen. Mit diesen werden dann die Länge über alles, die Länge in der Wasserlinie, vorderer und hinterer Überhang etc. ermittelt.

Die maximale Decksbreite und die größte Breite in der Wasserlinie wird jeweils mit Hilfe eines Proximity Rings bestimmt. Hierbei wird automatisch derjenige Kurvenpunkt mit dem größten Abstand von der Mittschiffsebene gesucht.



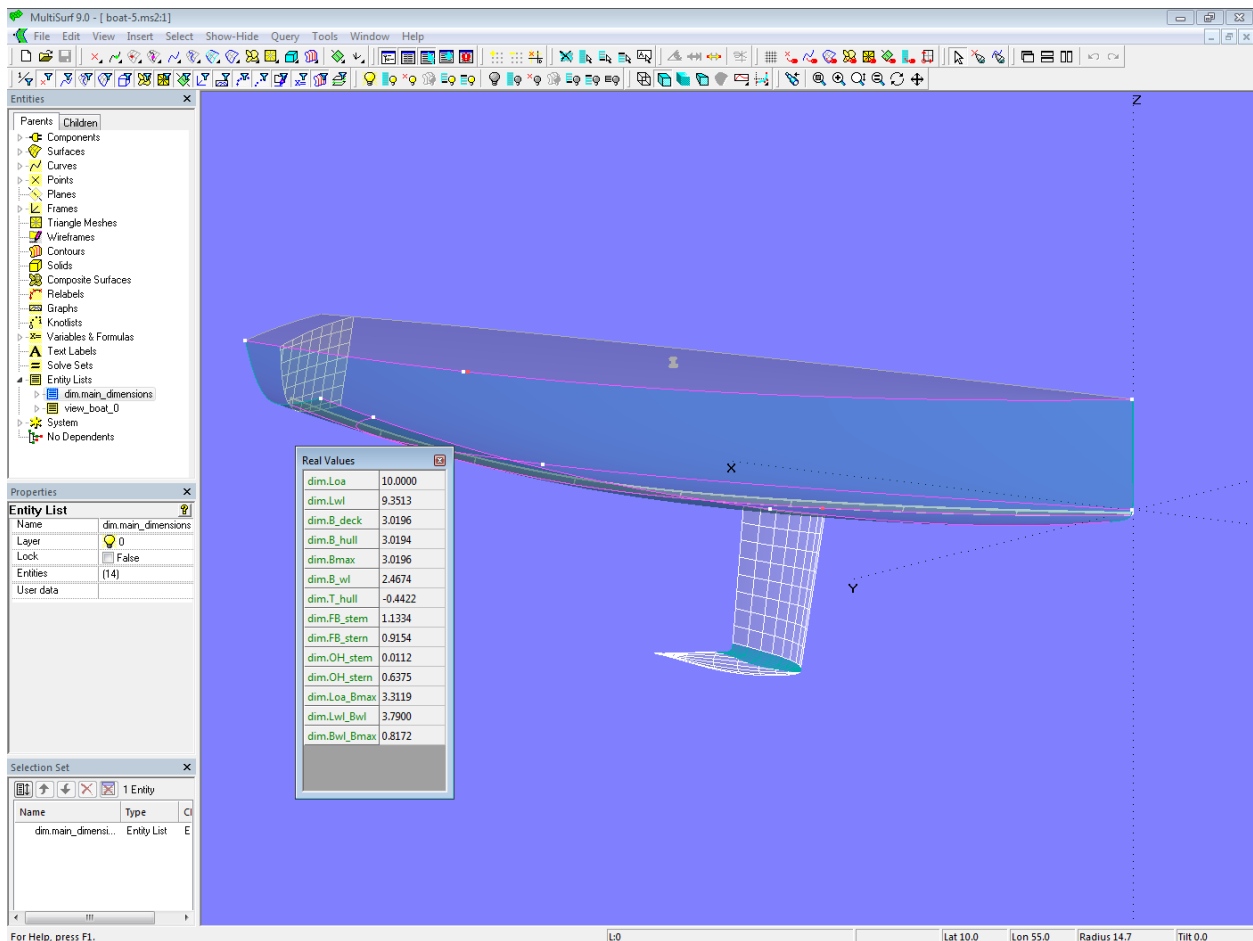
Modell *main_dimensions.ms2*

Speichern der Komponente Hauptabmessungen

In der Entity List *parents* ist nur die Rumpffläche *hull* enthalten, während in der Entity List *products* alle Formeln für die Berechnung der Hauptabmessungen stehen. Wie bei den bereits beschriebenen Komponenten können alle erforderlichen Objekte für die Komponentengeometrie mit diesen beiden Entity Lists und dem Befehl **SelectForComponent** auf einfache Weise ausgewählt werden. Anschließend das Ergebnis speichern über **File/ Component/ Save** (Dateiname *main_dim-component.mc2*).

Laden der Komponente Hauptabmessungen

Öffnen wir als Arbeitsmodell das Modell *boat-4.ms2* und fügen die Komponente *main_dim-component.mc2* über **File/ Component/ Load** oder das „Components“-Kontextmenü im Entities Manager ein. Die Komponente erfordert als Parent nur eine Fläche; dazu entsprechend *hull_0* auswählen. Komponentennamen: *dim*. Abschließend das Modell speichern als Datei *boat-5.ms2*.



Modell boat-5.ms2 –Komponente **dim** für Berechnung der Hauptabmessungen eingefügt

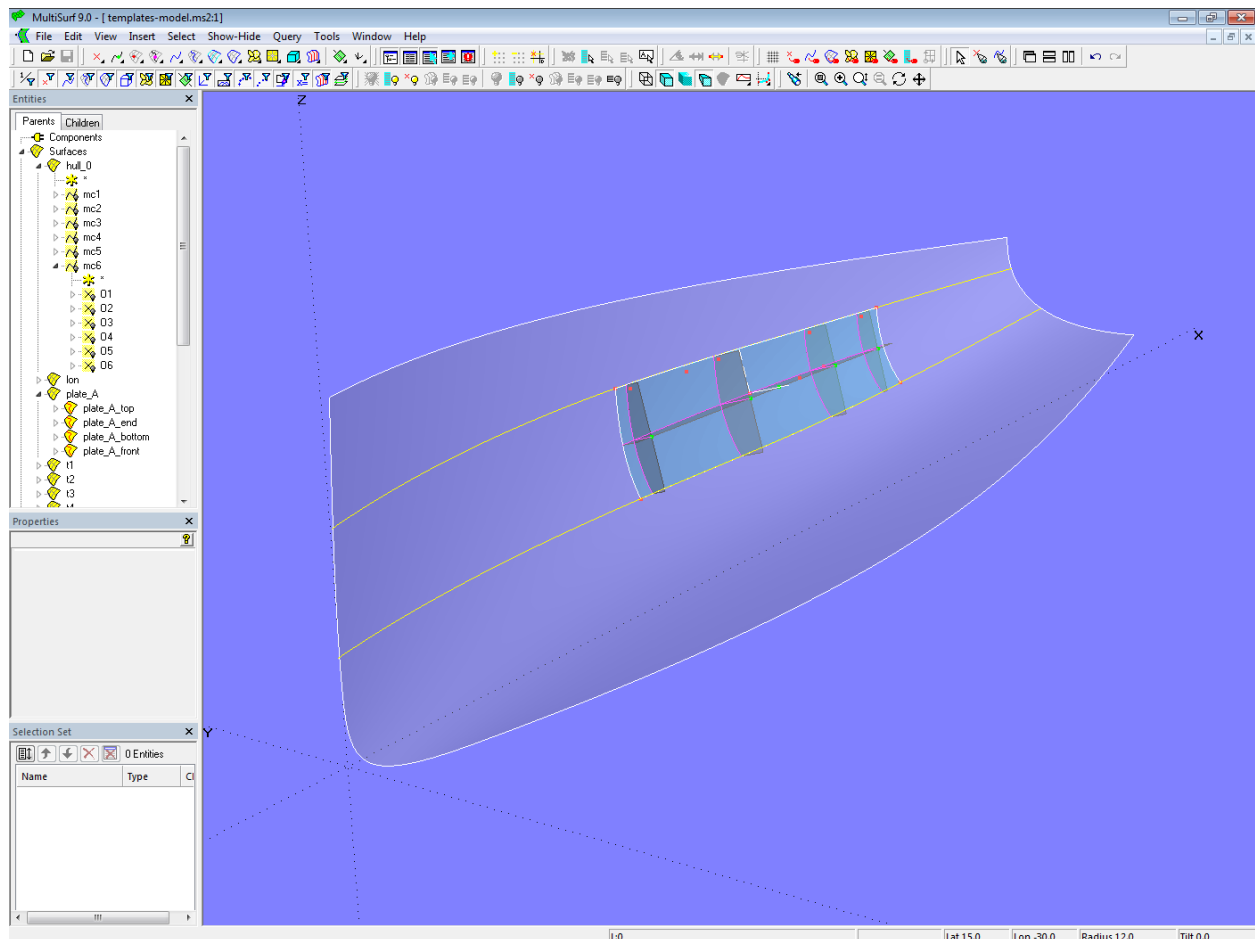
Die Zahlenwerte von Formeln- und Variablen lassen sich in MultiSurf über **Tools/ Real Values** (Hauptmenü) anzeigen, oder man drückt als Abkürzung einfach die Taste **V**. Damit nicht alle Formel- und Variablenwerte eines Modells angezeigt werden, ist es zweckmäßig, nur die gewünschten Werte in eine Entity List einzufügen. Diese im Entities Manager auswählen, dann **V**. Die eingefügte Komponente enthält dazu die Entity List **dim.main_dimensions**.

Schablonen für Plattenverformung

Beim Bau von Booten und Schiffen aus Metall besteht die Aussenhaut aus einer Reihe von Platten. Bei einem Rundspant-Rumpf sind sie in alle Richtungen gekrümmt und müssen durch bleibende Verformung in ihre dreidimensionale Gestalt gebracht werden. Um diesen Verformungsprozess zu kontrollieren, sind Schablonen für die Rumpfform im Bereich jeder Platte hilfreich. Mit einer Komponente lassen sich diese mit wenig Aufwand für jede einzelne Platte herstellen. Betrachten wir hierzu das Modell *templates-model.ms2*.

Quell-Modell Schablonen

Modell *templates-model.ms2* enthält die Rumpffläche **hull_0**, auf der mit 4 Randkurven (Snakes) die SubSurface **plate_A** bestimmt ist. Für diese Teilfläche werden 4 Schablonen in Platten-Querrichtung und eine Schablone in Platten-Längsrichtung erzeugt.

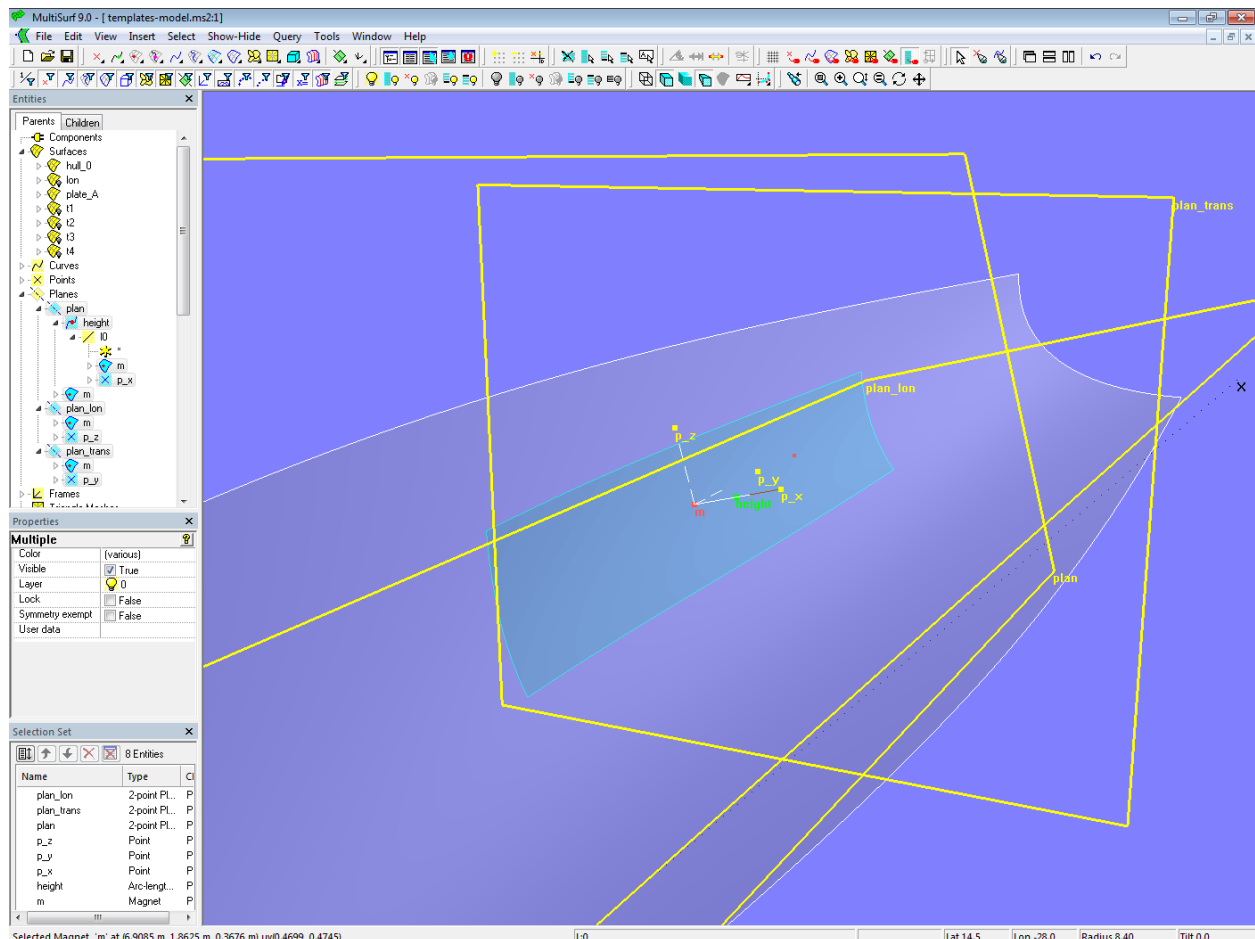


Modell templates-model.ms2 – Quell-Modell für Schablonen zur Kontrolle der Verformung von Aussenhautplatten

Die Querschablonen sind rechtwinklig zur Längsschablone, ihre Rückseiten liegen in einer gemeinsamen Ebene. Auf einen planen Untergrund gestellt, beschreiben die freien Schablonenkanten so die Form der SubSurface [plate_A](#).

Ausgangspunkt der Konstruktion ist Magnet [m](#) auf [plate_A](#). Von ihm hängt zum einen der Offset Point [p_off](#) ab, zum anderen der zu [m](#) relative Magnet [m_rot](#). Mit diesen 3 Punkten ist der 3-point Frame [F](#) definiert. Auf der X-Achse von [F](#) liegt Point [p_x](#), auf der Y-Achse Point [p_y](#), und auf der Z-Achse Point [p_z](#). Weiterhin gibt es Line [l0](#), die Verbindungslinie zwischen Magnet [m](#) und Point [p_x](#). Auf [l0](#) liegt der Arc-length Bead [height](#), mit dem die Höhe der Schablonen bestimmt wird.

Mit [p_x](#), [p_y](#), [p_z](#) und dem Magnet [m](#) werden 3 Ebenen als 2-point Planes erzeugt: [plan_lon](#) senkrecht zur Z-Achse von Frame [F](#), [plan_trans](#) senkrecht zu seiner Y-Achse, und [plan](#) senkrecht zu seiner X-Achse.

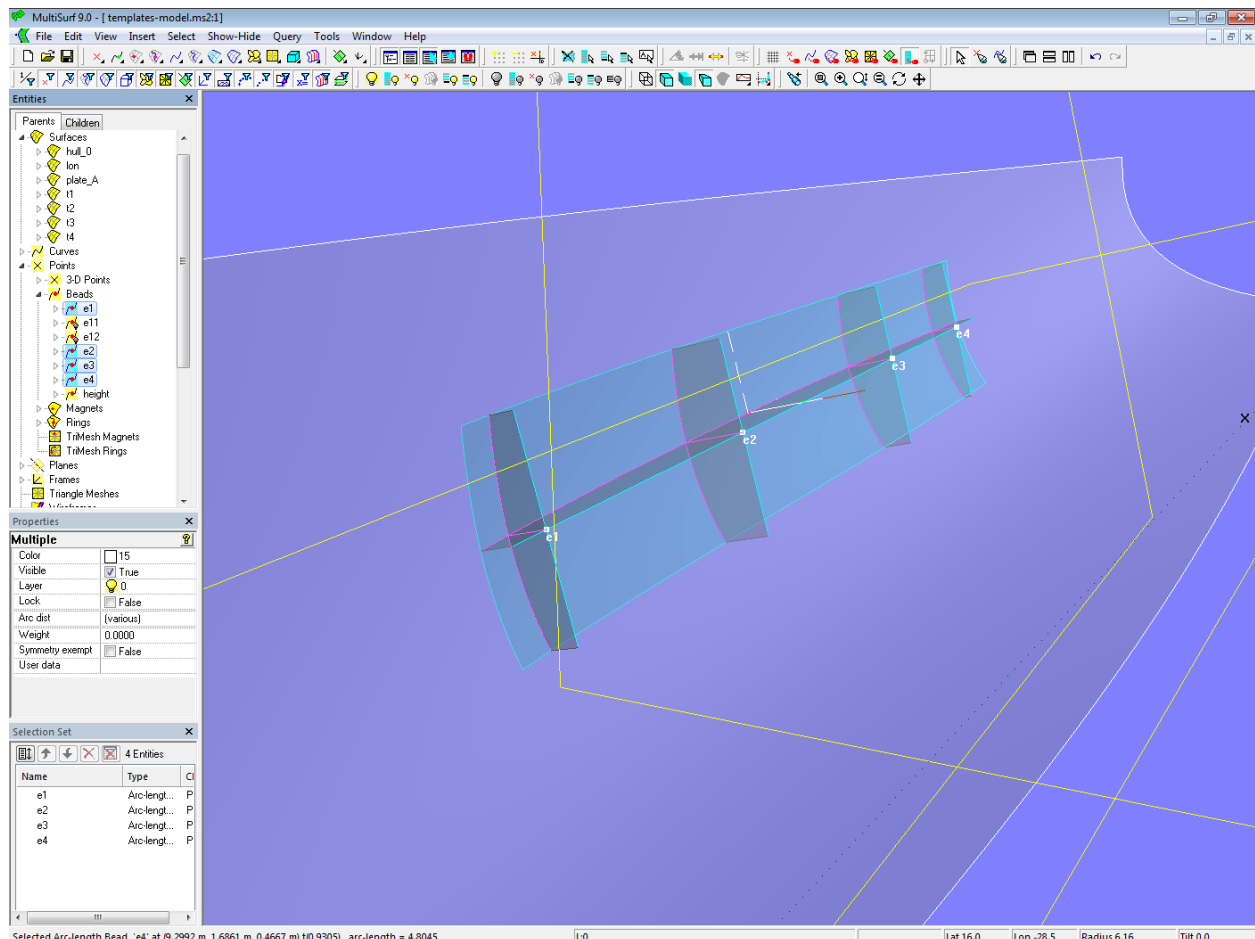


Modell templates-model.ms2 – Projektions- und Schnittebenen für Längs- und Querschablonen

Die 2-point Plane **plan_lon** schneidet nun **plate_A** in Längsrichtung in der Intersection Snake **n_lon**. Diese wird ihrerseits auf die 2-point Plane **plan** projiziert als Projected Curve **proj_lon**, auf der die 4 Arc-length Beads **e1**, **e2**, **e3** und **e4** liegen. An deren Position wird **plate_A** nun von der 2-point Plane **plan_trans** geschnitten in den Intersection Snakes **n1_trans**, **n2_trans**, **n3_trans** und **n4_trans**. Sie werden ebenfalls auf die Ebene **plan** projiziert.

Mit diesen Intersection Snakes und ihren Projektionen werden dann die Längsschablone und die 4 Querschablonen als Ruled Surfaces erzeugt (**t1**, **t2**, **t3**, **t4**, **lon**).

Die Schablonen lassen sich einfach an die jeweiligen Gegebenheiten der Aussenhautplatte, für die sie bestimmt sind, anpassen. Mit Magnet **m** und Magnet **m_rot** kann die Längsschablone verschoben und gedreht werden, so das sie etwa mittig in der Aussenhautplatte verläuft. Mit den 4 Arc-length Beads **e1**, **e2**, **e3** und **e4** können die Querschablonen positioniert werden, mit dem Arc-length Bead **height** läßt sich die Höhe der Schablonen einstellen.



Modell templates-model.ms2 – Quell-Modell für Schablonen zur Kontrolle der Verformung von Aussenhautplatten

So weit zum Aufbau des Quell-Modells für die Schablonen-Komponente.

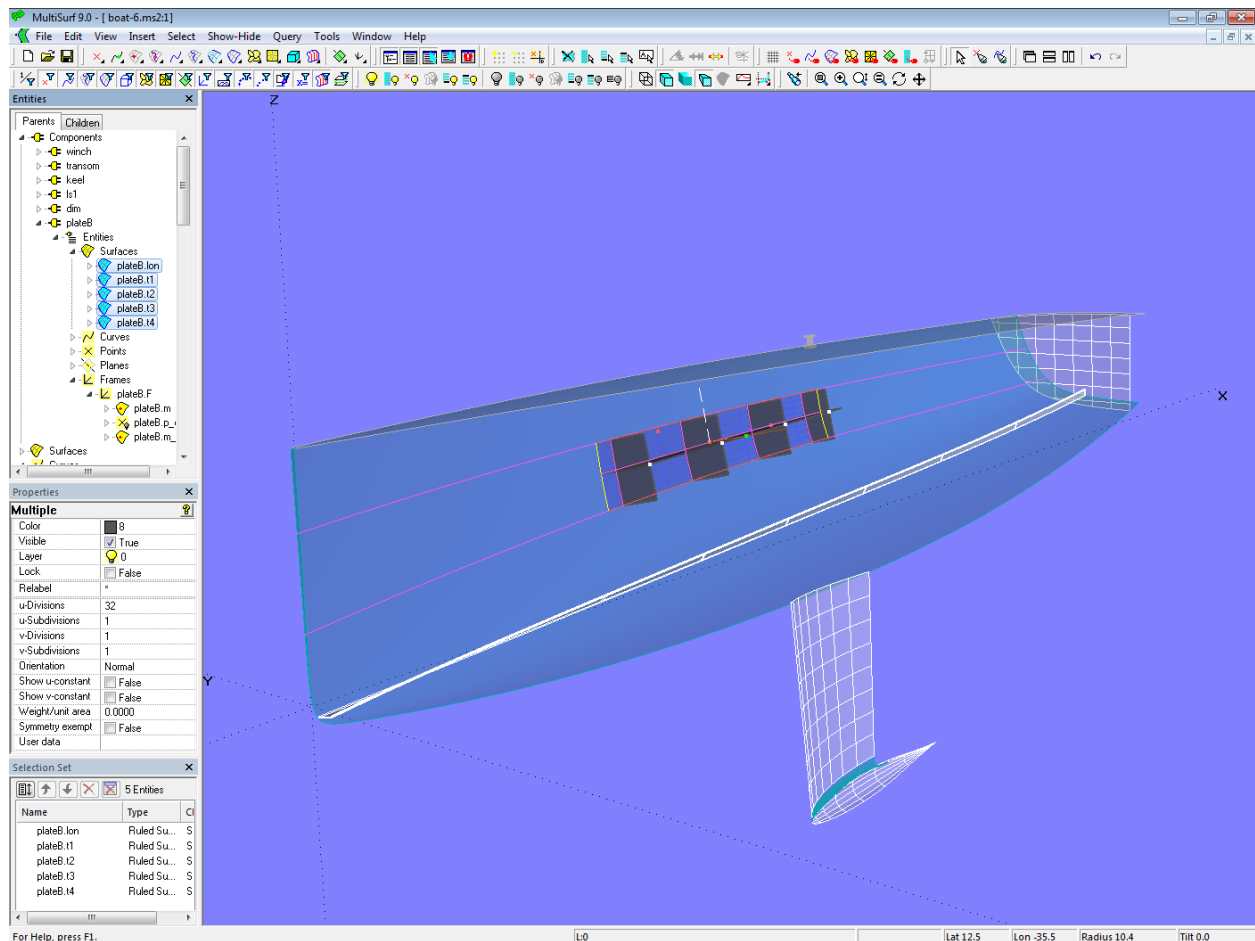
Speichern der Schablonen-Komponente

Es ist für die Übersichtlichkeit zweckmäßig, vor dem Abspeichern als Komponente alle Hilfsobjekte der Konstruktion zu verbergen. Lediglich die fertigen Flächen und Ziehpunkte, mit denen die Lage der Schablonen kontrolliert werden kann, sollten später im Hostmodell sichtbar sein.

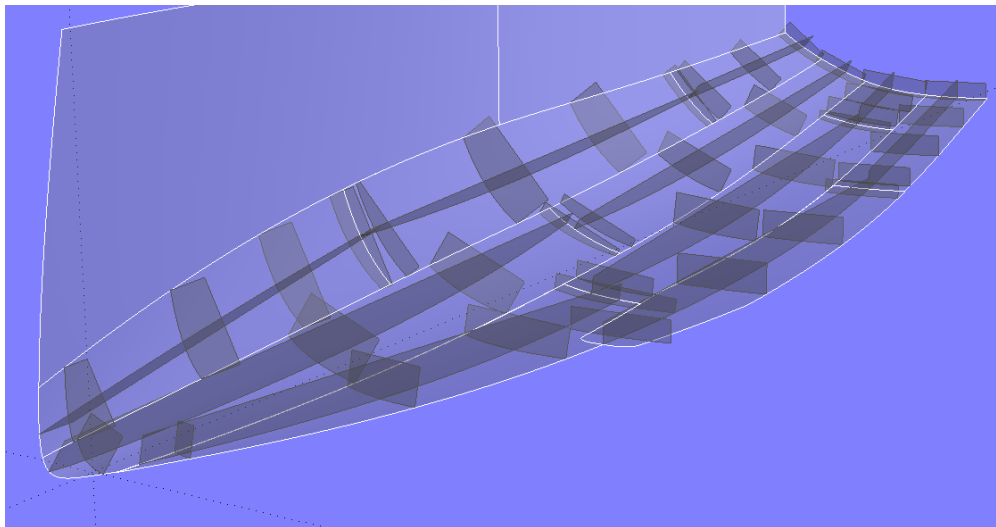
Entity List **parents** enthält lediglich die SubSurface **plate_A**, Entity List **products** die Schablonenflächen. Mit beiden Entity Lists und dem Befehl **SelectForComponent** alle Komponentenobjekte auswählen und das Ergebnis speichern über **File/ Component/ Save** (Dateiname: *templates-component.mc2*). Das Quell-Modell *templates-model.ms2* kann nun geschlossen werden.

Laden der Schablonen-Komponente

Fügen wir nun die Komponente in das Modell *boat-5.ms2* ein, das unser Host-Modell sein soll. Dazu Modell *boat-5.ms2* öffnen, die SubSurface **plate_B** sichtbar machen und auswählen. Dann **File/ Component/ Load** (oder Entities Manager und „Components“-Kontextmenü) und als Komponentennamen **plateB** wählen. Dann wird allen Namen der eingefügten Objekte dieser Textteil vorangestellt und man sieht gleich, zu welcher Aussenhautplatte die Schablonen gehören. Abschließend das Modell speichern unter dem Dateinamen *boat-6.ms2*.



Modell boat-6.ms2 – Komponente **plateB** für Plattenschablonen eingefügt



Aussenhautplatten mit Schablonen zur Kontrolle der Verformung

Mögliche Komponenten-Probleme

Eine Komponente ist ein Stück aus einem MultiSurf-Modell. Ihre Abhängigkeiten sind normalerweise unvollständig, sie braucht Eltern vom Host-Modell (Arbeitsmodell), in das sie eingefügt wird. Darum sollten Quell-Modell und Host-Modell in folgenden Merkmalen übereinstimmen:

- Ähnlichkeit der Eltern
- gleiche Normalen-Richtung bei Flächen
- gleiche u/v-Parameter-Richtung bei Flächen
- gleiche t-Parameter-Richtung bei Curves und Snakes

Gibt es Unterschiede, können beim Laden einer Komponente Fehler auftreten. Im Allgemeinen sind sie aber einfach zu beheben.

Ähnlichkeit der Eltern

Wenn zum Beispiel die Komponente eine C-Spline Lofted Surface ist, deren Eltern im Quell-Modell 3 Masterkurven sind, müssen die Eltern für die Komponente im Host-Modell drei sich als Kurven qualifizierende Objekte sein (Curves, Snakes, Points).

Gleiche Normalen-Richtung bei Flächen

Im Modell *boat-1.ms2* zeigt die Normalen-Orientierung der Fläche für das Deck ([deck_0](#)) nach oben. Die eingeladene Winch-Komponente steht auf dem Deck. Wäre [deck_0](#) nach innen orientiert, würde die Winch auf dem Kopf stehen. Das ist leicht zu korrigieren, in dem man dem Wert für „Offset“ von [winch.p5](#) ein entgegengesetztes Vorzeichen gibt. Zwar könnte man im Entities Manager auch die Eigenschaft „Orientation“ von [deck_0](#) von „Normal“ auf „Reverse“ setzen, aber davon sollte man absehen, denn andere Objekte könnten davon betroffen sein. Editiert man nur die Komponente, bleibt die Auswirkung auf deren Objekte beschränkt.

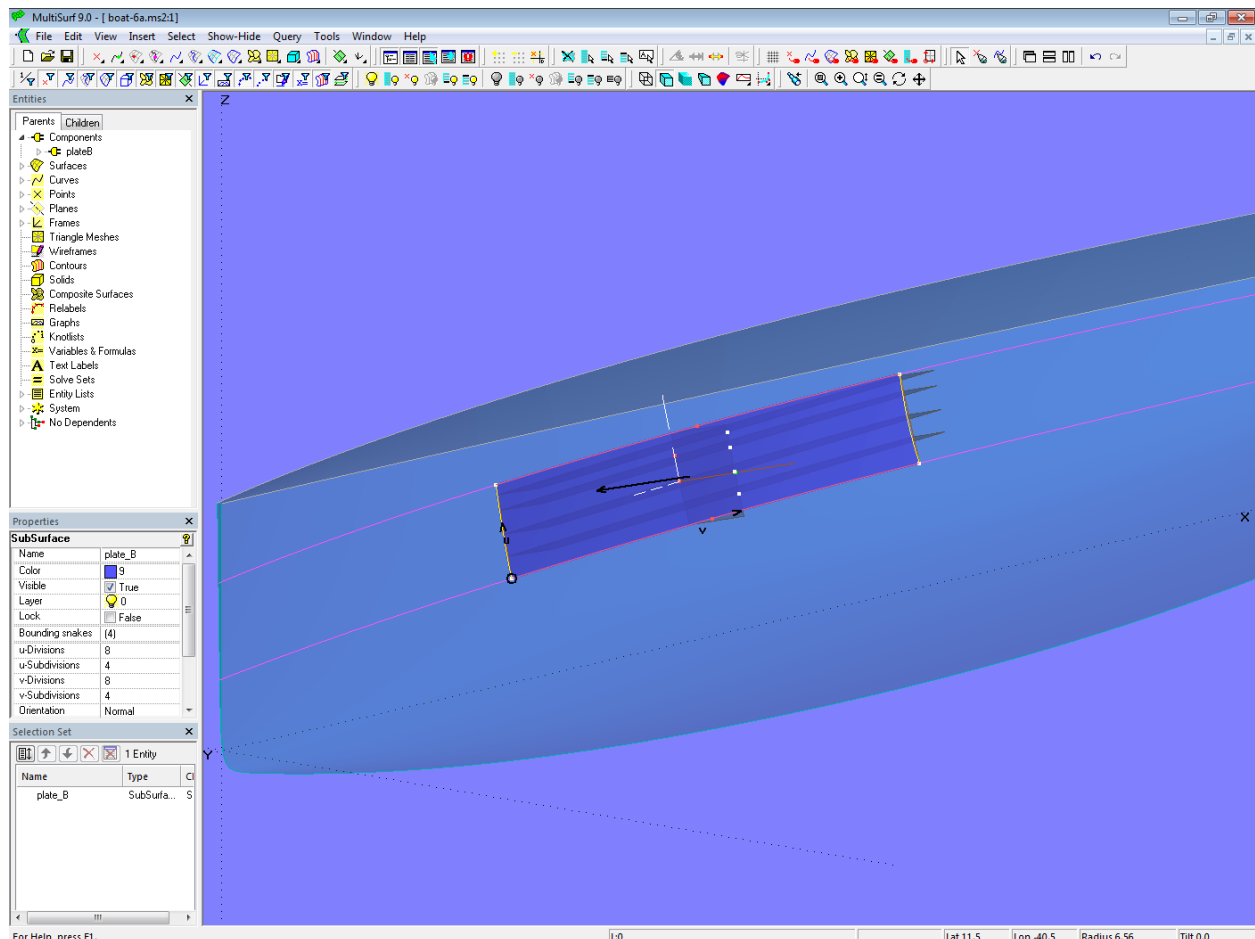
Gleiche u/v-Parameter-Richtung bei Flächen

Betrachten wir das Beispiel *boat-6a.ms2*, in das ebenfalls die Schablonen-Komponente geladen wird. Im Gegensatz zum Modell *boat-6.ms2* verläuft hier die u-Parameter-Richtung der SubSurface [plate_B](#) in Querrichtung. Entsprechend liegen die Schablonen verdreht auf der Fläche.

Mit dem Magneten [plateB.m_rot](#) können sie aber einfach in die gewünschte Richtung gedreht werden.

Größenunterschied zwischen Quell-Modell und Host-Modell

Dies ist zum Beispiel der Fall bei der oben beschriebenen Spiegel-Komponente. Sie wurde aus dem Modell für ein kleineres Boot gespeichert und dann in ein Modell für ein längeres Boot geladen.

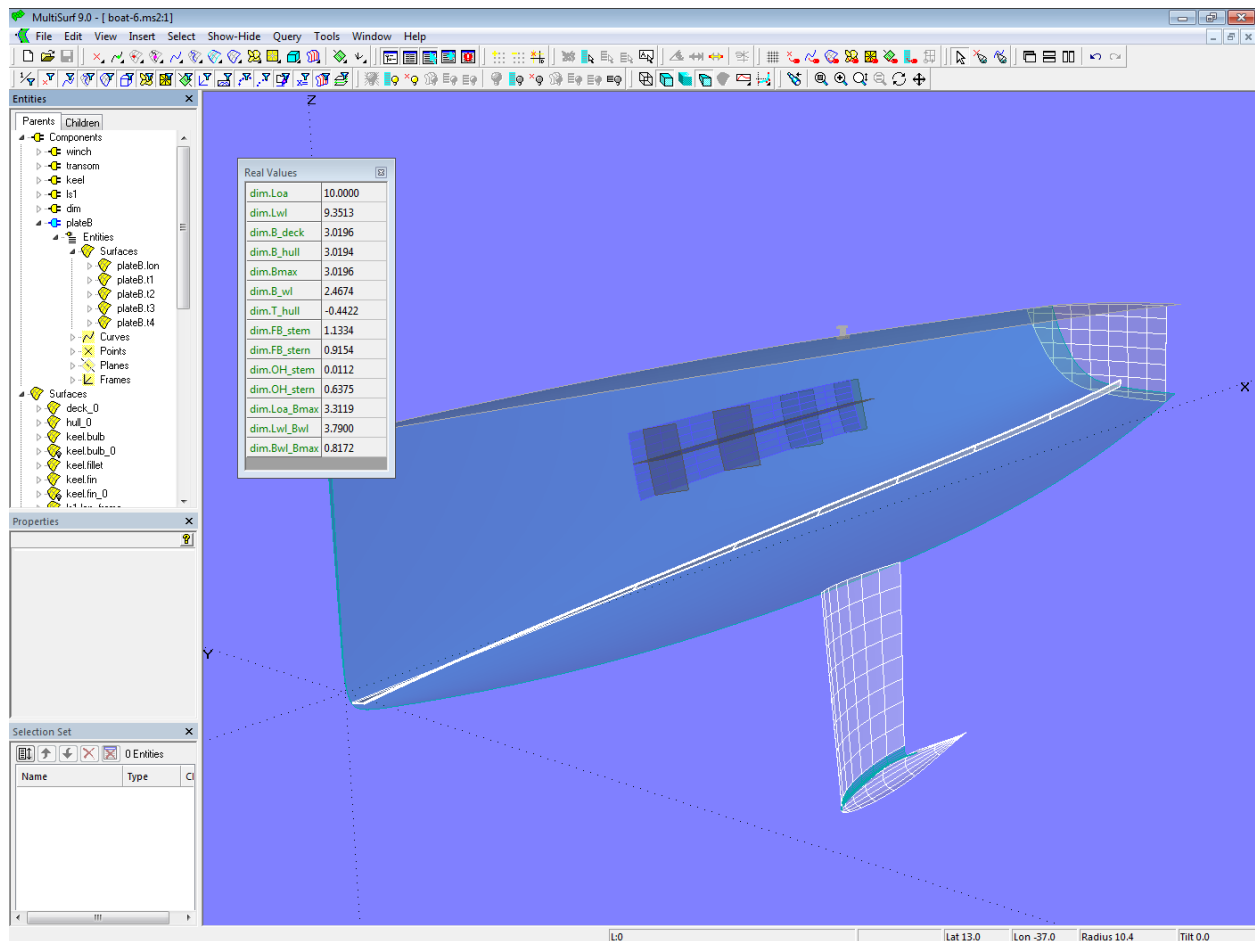


Modell boat-6a.ms2

Modifizieren einer Komponente

Wenn MultiSurf eine Komponentendatei speichert, sind in dieser alle Eltern der Komponente aus dem Quell-Modell als ausgeblendete Objekte enthalten. Darum kann eine Komponentendatei (*.mc2) auch eigenständig in MultiSurf geöffnet werden. Dazu nach **File/ Open** im Öffnen-Dialogfenster im Eingabefeld unten ganz rechts „Alle Dateien (*.*)“ auswählen. Im Auswahlfenster erscheinen dann nicht nur die normalen Modelldateien (*.ms2), sondern auch die Namen der Komponentendateien (*.mc2).

Auf diese Weise kann man eine Komponentendatei genauso bearbeiten wie ein normales Modell. Man kann sie anschließend als Modelldatei (.ms2) speichern, oder daraus eine modifizierte Komponente erstellen (.mc2).



Modell boat-6.ms2 – Beispielmodell mit 6 eingefügten Komponenten

So weit Teil 1 des Tutoriums über Modelle für Komponenten, Speichern von Komponenten und Laden von Komponenten in andere Modelle.

In Teil 2 wird eine Reihe von Komponenten für Ausrüstungsdetails vorgestellt, um ein Boot oder Schiff in Render View realistischer darzustellen.

MultiSurf - Relational 3D Surface Design Software



=====