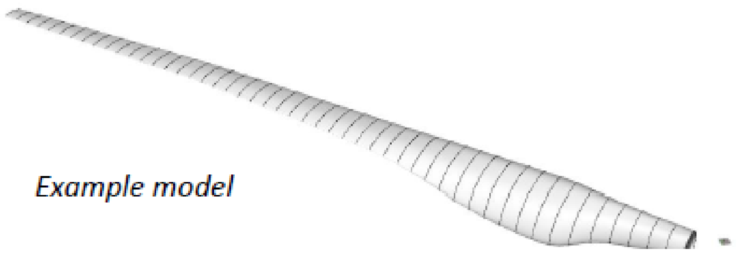
# Exercise: CE_cpp01 – *C++ Plug-Ins for Rhino*

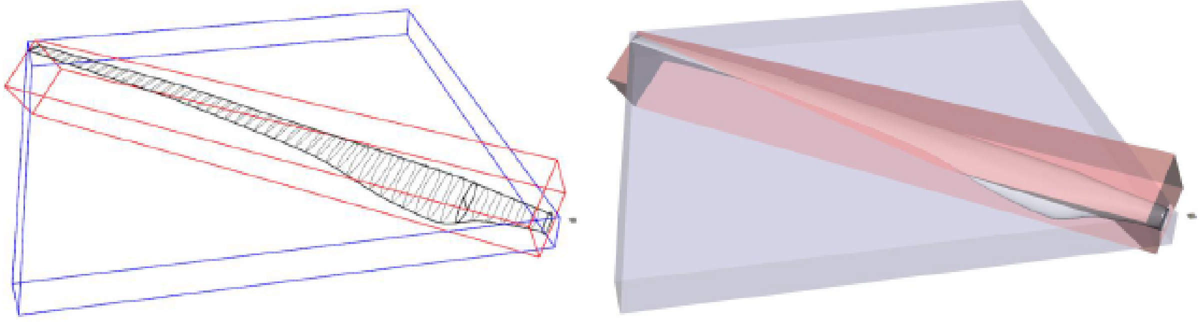
| | |
|---|---|
| **Topic** | C++ Plug-In Development for Rhino |
| **learning target** | - Setting up of a development environment in Windows using Visual Studio, environmental variables, batch files<br>- Introduction into Rhino C++ Plug-In projects: concepts, basic implementation, debugging, system architectural issues<br>- Creation of **oriented bounding boxes** |

Instructions

In this exercise you will learn to set up a ***plug-in project*** for Rhinoceros using a ***Visual Studio 2010*** development environment.
The purpose of this small example plug-in will be to create an oriented bounding box (OBBox) on a surface model. A OBBox has in general of smaller volume than a classical bounding box and approximates the real geometry better. Thus, helps to speed up various geometrical algorithms.
You will also learn how to perform bounding box intersection test on oriented BBoxes and in which algorithms these kind of bounding volumes are beneficial.



*Example model*



*Comparison of classical bounding box (blue) and oriented bounding box (red)*

**Pre-Requisites and Remarks**

All necessary software and third-party dependencies (libraries,...) used in this exercise are either provided or are already installed on the computer in the CIP-Pool.

**1] Project template**

Please download the project template to your local drive, in which you are supposed to do your code implementation. You will find it either on the P:\ drive, in case you are in the CIP-pool:

P:\LS_ComputationInEngineering\SS2016_ComputerAidedDesign\
CompExercise\Code\Rhino5_cPP_API_CIP

Or you can download the project files from the following link:
https://syncandshare.lrz.de/dl/fiRQSBaLG4EnMzb3onxEgELS/Rhino5_cPP_API_local.zip

Please make sure you downloaded the template project after this exercise sheet was handed out!

**2] Start the template project**

You are provided an entire solution directory by downloading the content from the link given above.
***Copy*** (and unpack) the folder to your ***private user drive***, where you have ***reading/writing rights***. In the CIP-pool this will be your user dive (e.g. Go to <Computer> <...TUMID...>).

The solution directory should contain a Visual Studio solution "set_RhinoPlugIN_vc10.sln" and a batch file called "set_RhinoPlugIN_vc10.bat".

Start the visual studio solution ONLY by using the batch file (double click)!
If this does not work, please contact your supervisor!
By double click on the file "set_RhinoPlugIN_vc10.bat" Visual Studio should start automatically.

## Pre-Requisites and Remarks

### 3] Working on your private computer

In general it is possible to work on your private computer, if the following pre-requisites are fulfilled and the software stated in the following is being installed:

- OS: Windows OS 64 bit
- Visual Studio 2010 (with C++ compiler 64 bit)
- Rhino 5.0 64bit SDK; downloadable here:

  http://www.rhino3d.com/download/rhino-sdk/5.0/commercial
- Rhino 5.0 64 bit; a suitable free evaluation version can be found here:

  http://www.rhino3d.com/download/rhino/5.0/evaluation

To make the provided software framework run on your local machine you have to set some paths by hand to fit your local installation configuration. You will find further instructions on the next page.

**Please contact the course lecturer in case of any technical difficulties!**

**.. Not matter if on your local machine or the CIP computers...**

**<u>Instructions</u>**

- Start the template project "set_RhinoPlugIN_vc10.sln" you are provided (see page before) in your solution folder by double clicking on the batch file "set_RhinoPlugIN_vc10.bat".
  In case you work in the CIP Pool, the project should open in Microsoft Visual Studio without errors or problems.

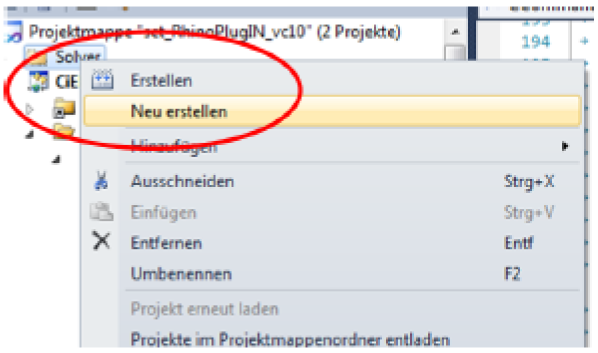
  <u>Remarks</u>:

  1) If you work in the CIP Pool, but the previous step fails, please contact your supervisor!

  2) If you work on your private computer the following steps are necessary:

     a) Go to the folder ".\ Env" inside the downloaded solution folder.
     b) <right click> on the file "Rhino_Env.bat" and choose <edit>
     You might have to adjust the paths pointing to your Rhino and Rhino SDK installation.
     c) Perform the same for the file "set_vs_2010_amd64env.bat".
     The paths to your Visual Studio installation might have to be adjusted.

- To test the solution environment, perform the following steps:
  <right click on "Solver"> in your project explorer and choose <rebuild>. This part of the provided solution should compile without errors.
  If this is not the case please contact your supervisor!



<u>Remark:</u>

***Initially*** the entire solution ***will not compile yet***!
You have to implement some parts before this will be possible.
After your job is done you can compile the entire solution by either pressing <F7> or <right click> on the Solution "set_RhinoPligIN_vc10" and choose <Build Solution>.

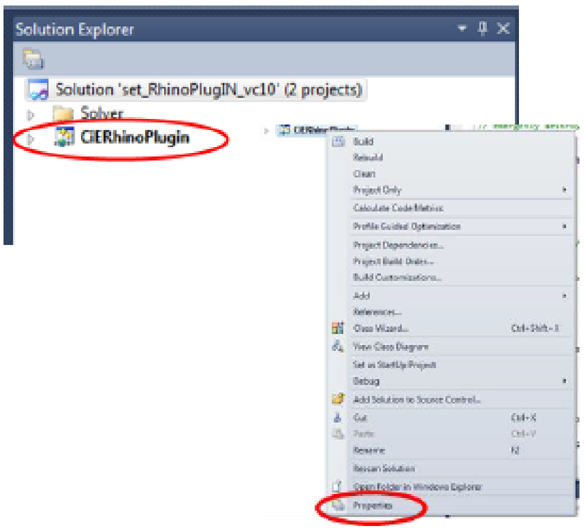
## Instructions

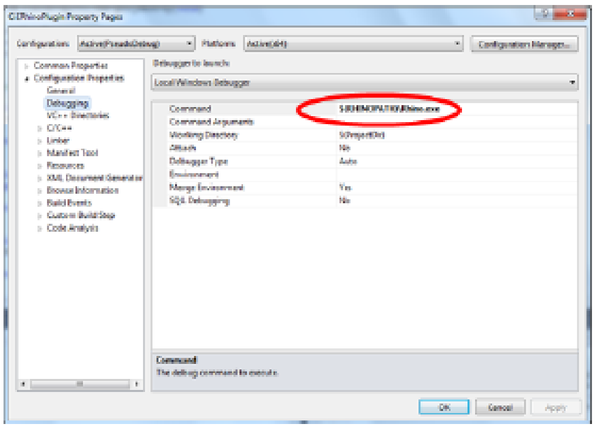
- To test your template plug-in project, after you coded and the solution compiles. You should start Rhinoceros out of your Visual Studio solution. This should happen automatically if you press <F5>.

  If Rhinoceros does not start, perform the following steps:

  a) <right click> on the project "CiERhinoPlugin" and choose <Properties>.
  b) Go to "Debugging" and check if in the field "command" the following is written. If this is not the case replace any entry by the following:

  *$(RHINOPATH)\Rhino.exe*




CiERhinoPlugin Property Pages

Configuration: Active(PseudoDebug) Platform: Active(x64)

Debugger to launch: Local Windows Debugger

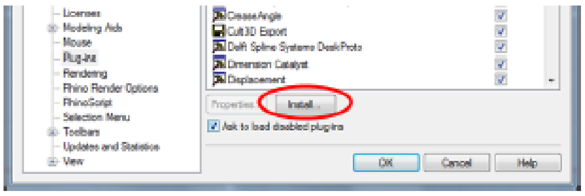
| | |
|---|---|
| Command | $(RHINOPATH)\Rhino.exe |
| Command Arguments | |
| Working Directory | $(ProjectDir) |
| Attach | No |
| Debugger Type | Auto |
| Environment | |
| Merge Environment | Yes |
| SQL Debugging | No |

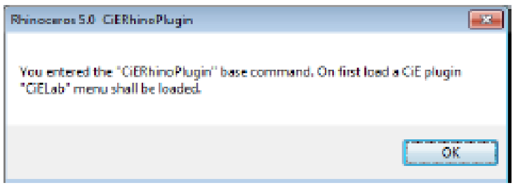
Command
The debug command to execute.

- Then it is time to make your plug-in known to Rhinoceros by installing it.

  Type in _PlugInManager in the command line of Rhinoceros.
  The plug-in manager should start. Choose <Install> and select the file "CiE_RhinoPlugIn.rhp" located in your solution folder under ".\x64\plugin\".

  If everything succeeded can be tested by typing the command _CiERhinoPlugin. A message box should appear.
  If this step fails, please contact your supervisor!




Rhinoceros 5.0 CiERhinoPlugin

You entered the "CiERhinoPlugin" base command. On first load a CiE plugin "CiELab" menu shall be loaded.

## Instructions and Workflow

- The steps performed before need to be done only once.
  However, you can run the plug-in only after it compiles, which requires some programming tasks performed by YOU!

- For those who are interested in the ***theoretical background***, the general outline of the solution workflow to create a oriented bounding box is the following.

  1] Select surfaces in your model to be fitted by an oriented bounding box.

  2] Create a temporary mesh on the surfaces to be fitted by the oriented bounding box.

  We will use the Rhino meshing features to perform this step.

  3] We will use the mesh vertices to analyze a 3x3 so-called covariance matrix known from statistical methods:

  COV(i,j) = E[(xi - M_i)*(xj – M_j)]

  M_i = E[xi] denotes the barycenter of the point set
  E[*] denotes the expectation operator

  We are interested in the eigenvectors of this matrix to define a plane fitting the given point set (points of mesh vertices).

  We will make use of a external linear algebra library to perform this steps.

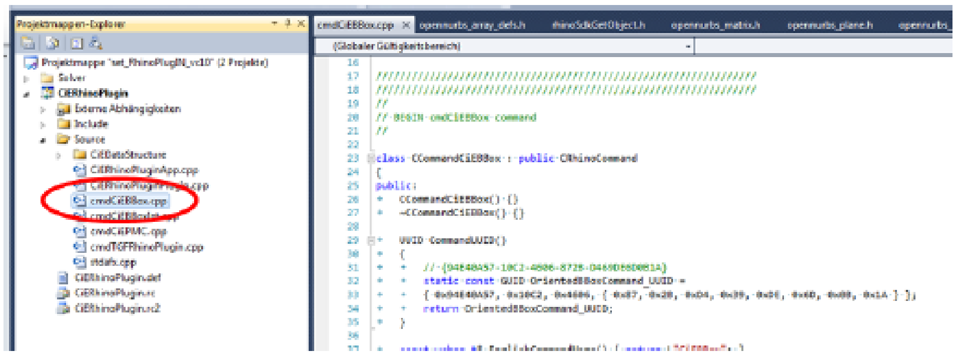
  4] Given this plane we can set this plane as a construction plane (CPlane) and transform all coordinates of the mesh points with respect to the Cplane's local coordinate system.
  Thus, we can find the min/max values in local coordinates.

  A back transformation gives us finally the oriented BBox w.r.t. the global coordinate system.

Furthermore we will have to implement an intersection algorithm for the oriented bounding boxes.
You will be guided step by step through the necessary steps.

## Instructions

- Now we can start implementing the missing code pieces to create an oriented bounding box.
- Open the file "cmdCiEBBox.cpp" in visual Studio (VS) from the solution browser (by <double click>).



```
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
//
// BEGIN cmdCiEBBox command
//

class CCommandCiEBBox : public CRhinoCommand
{
public:
    CCommandCiEBBox() {}
    ~CCommandCiEBBox() {}

    UUID CommandUUID()
    {
        // {94E4BA57-10C2-4606-8728-0469DE6D0B1A}
        static const GUID OrientedBBoxCommand_UUID =
        { 0x94E4BA57, 0x10C2, 0x4606, { 0x87, 0x28, 0x04, 0x39, 0x0E, 0x6D, 0x0B, 0x1A } };
        return OrientedBBoxCommand_UUID;
    }

    const wchar_t* EnglishCommandName() { return L"CiEBBox"; }
```

- This file holds the code for the command to generate our bounding boxes if the user will type the command _CiEBBox in Rhino.
  The user will be asked to give several options used for the generation of a BBox, in particular the type of BBox: a classically axis aligned or oriented box.
- To understand how the interaction with the GUI of Rhino works, have a look at the ***lines 55 to 146***.
  And to understand how the user can select objects, check the ***lines 153-181***.
  You do not have to program here, but for later you should understand the procedure in principal.
- **Uncomment** the ***lines 196-197*** and ***210-213***.
  Here two important routines are called to compute and visualize the bounding boxes.
  Jump to the declaration of the function "doVisualBBox()" (<right click> on "doVisualBBox()" and <go to definition>).
  If this does not work, simply open the file "TGFRhinoPlugin_BBox.cxx " within VS and go to ***line 83***.
- This is our **first task**:
  You shall visualize a box in Rhino by given corner points. See further instruction in the code. Use also this help documentation:

  http://wiki.mcneel.com/developer/sdksamples/addbrepbox

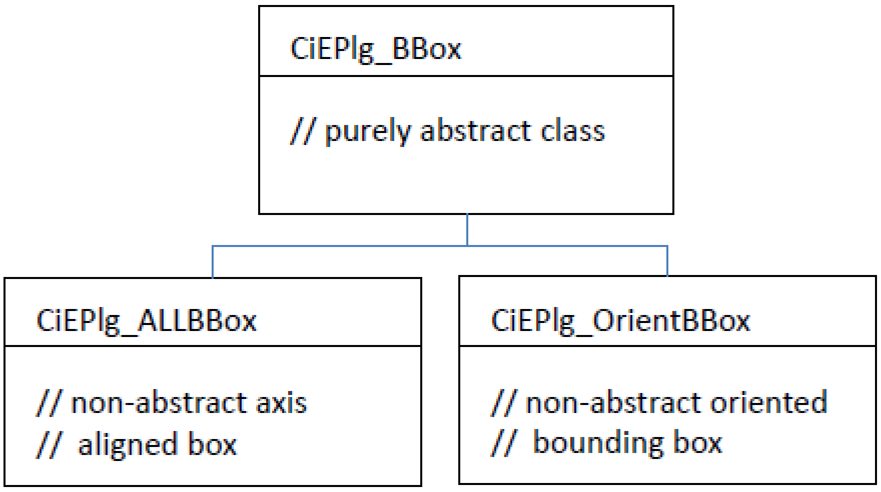
## Instructions

- **Second Task**:

  After the visualization of a bounding box is done, we have to implement the creation of the bounding boxes, ***both classical axis aligned and oriented***.

  First get familiar with the class structure of our data model. Therefore have a look at the files:

  TGFRhinoPlugin_BBox.h
  TGFRhinoPlugin_ALLBBox.h
  TGFRhinoPlugin_OrientBBox.h



  We have a purely abstract base class, and two non-abstract classes for axis aligned and oriented BBoxes.
  In the file “TGFRhinoPlugin_BBox.h” in ***lines 85 to 88*** you will find those functions which have to be implemented by all derived classes to become non-abstract. Unless implemented the code will not compile!

  So go to the file “TGFRhinoPlugin_ALLBBox.h” and uncomment the ***line 56*** and ***61***.
  Go to the file “TGFRhinoPlugin_ALLBBox.cxx” to ***lines 55 to 102***. Uncomment the member routine computeBBox(..) and implement the missing pieces as described in the code, to create a classical BBox.

  Then go to ***lines 110 to 151***. Uncomment the member routine BBoxIntersection (..) and implement the missing pieces as described in the code.

## Instructions

- **Third Task**:

  Now we have to implement the algorithms for the generation of oriented bounding boxes. The theoretical background is given in the course material.

  At first, we have to generate meshes of selected object. We will use the Rhino meshing tools. Go to the file "TGFRhinoPlugin_Meshing.cxx". In ***line 18 to 21*** you have to make use of the Rhino meshing tools.
  You will find a helpful link within the code.

- **Fourth Task:**

  Now we are ready to finish the oriented bounding box algorithm. Open the file "TGFRhinoPlugin_OrientBBox.cxx" and go to ***lines 81 to 361***.
  Some pieces of the pipeline to generate oriented BBoxes are missing. Implement them as stated in the code. In particular check the lines:

  ***line 147***
  ***line 157***
  ***line 275***
  ***line 303***

  Also try to follow the overall pipeline to understand the procedure.

- **Fifth Task:**

  Open the file "cmdCiEPMC.cpp".
  You will have to prepare this file for an lecture exercise (not being part of the midterm!).
  Here we want to make use of oriented bounding boxes within a PMC := point membership classification algorithm, which determines if a point is inside or outside of a volume.
  See the code at ***lines 51*** following for further instructions. In principal the following tasks are to be solved:

  - let the user pick a point
  - let the user pick a volume
  - let the user set some options (e.g. whether to use a Oriented BBox)
  - pass everything to another routine.