OpenSees Workshop

Brunel, May 2016

Presented by **Dr Liming Jiang & Xu Dai**

With acknowledgements to:
Jian Zhang, Yaqiang Jiang, Jian Jiang, Panagiotis Kotsovinos,
Shaun Devaney, Ahmad Mejbas Al-Remal, & Praveen Kamath & the IIT Roorkee and Indian Institute of Science teams, and China Scholarship Council!

& special acknowledgement to:
Frank McKenna at University of California, Berkeley for OpenSees

OPENSEES WORKSHOP DAY 2

1. Framework of OpenSees and how to compile it
2. How to add your new class
3. How to add your new project
4. How to add Tcl commands for your project

OPENSEES WORKSHOP

Day2: Framework & building OpenSees

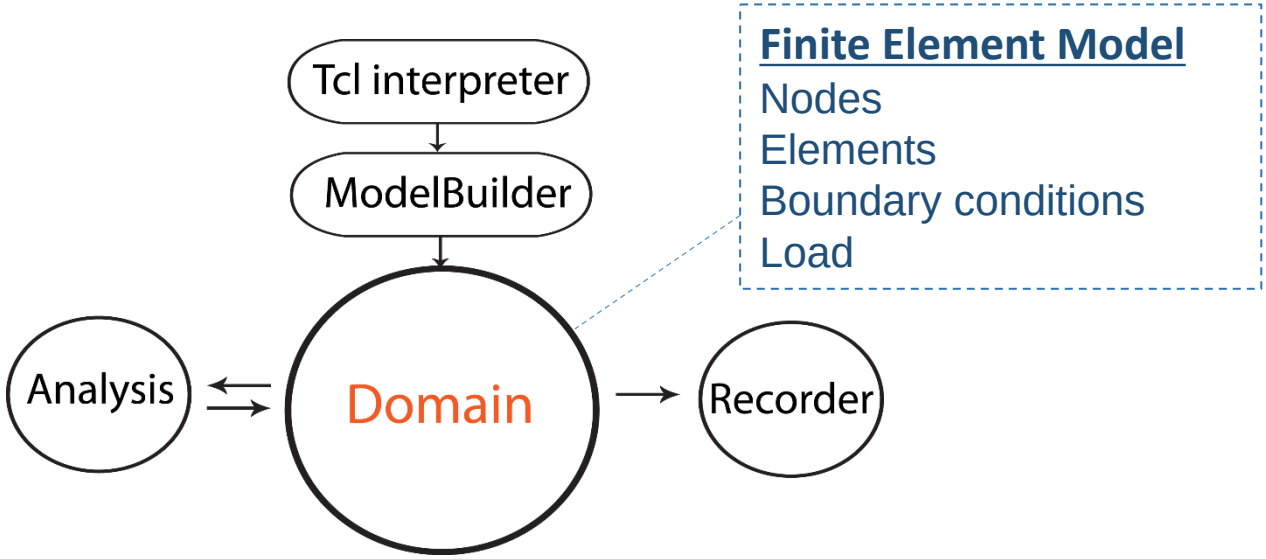
Temporary Source Code Package:

(link has been sent through email)

https://dl.dropboxusercontent.com/u/66579010/BrunelTest.zip

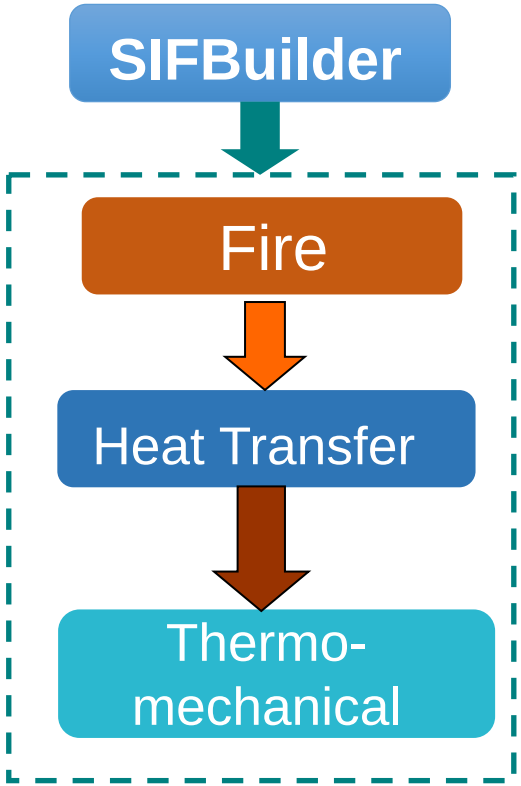
OpenSees Framework

- A framework is **NOT an executable**;
- It is a set of cooperating software components for building applications in a specific domain**;**
- It is a collection of **abstract and derived** classes;
- **Loose-coupling** of components within the framework is essential for extensibility and re-usability of the applications



OpenSees for Fire

- Started at Edinburgh University since 2009;
- Based on a group of PhD students' work**;**
- Developed for modelling '**Structures in Fire**';



User-friendly interface for creating (regular) structural models and enable consideration of realistic fire action

Models of fire action (only ***idealised*** fires), i.e., Standard fire, Parametric fire, EC1 Localised fire, Travelling fire

Heat transfer to the structural members due to fire action

Structural response to the elevated temperatures

OpenSees Framework

31 Projects in OpenSees

actor
cblas
convergence
cssparse
damage
database
graph
handler
matrix
modelbuilder
OpenSees
OpenSeesTk
OpenSeesTk
Optimization
quickMain
reliability
renderer
string
superLU
system
tagged
utility

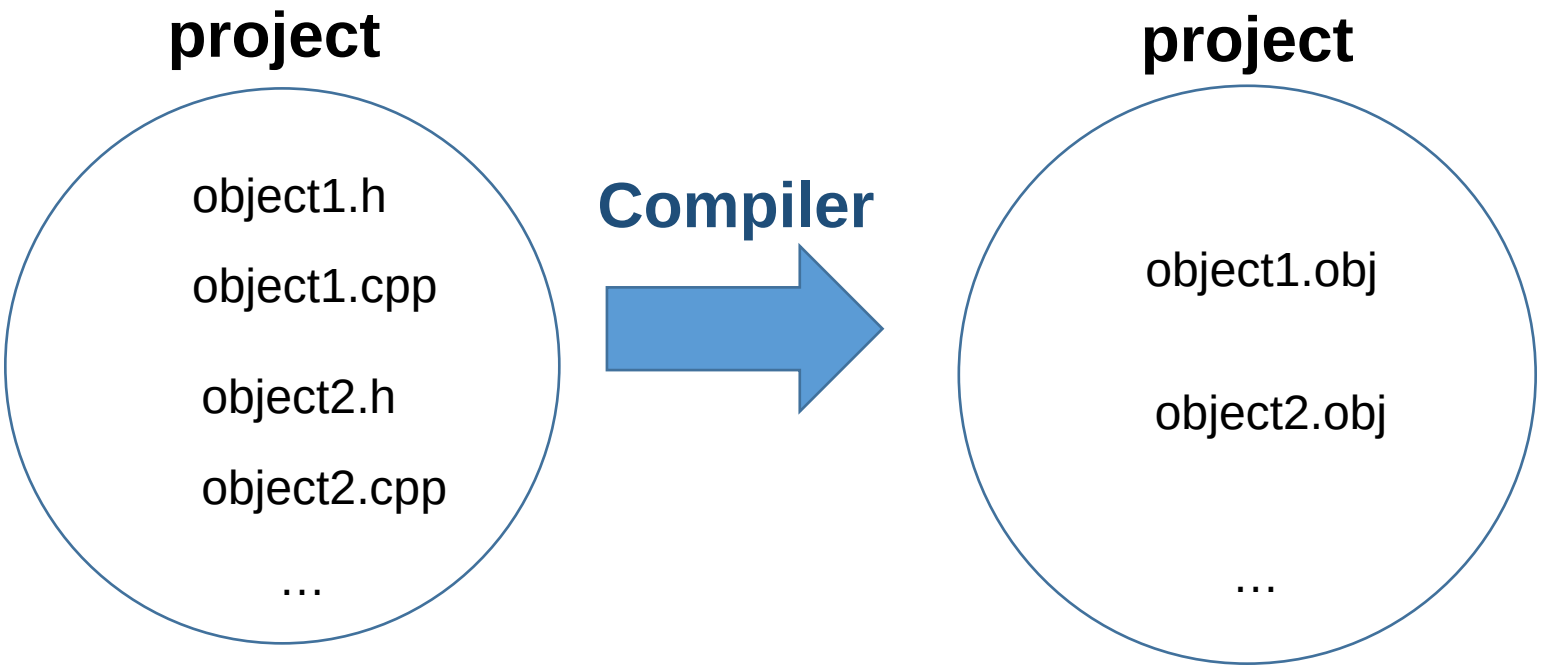
---

analysis
domain
element
fire
HeatTransfer
HTMain
material
recorder
SIFBuilder
tcl

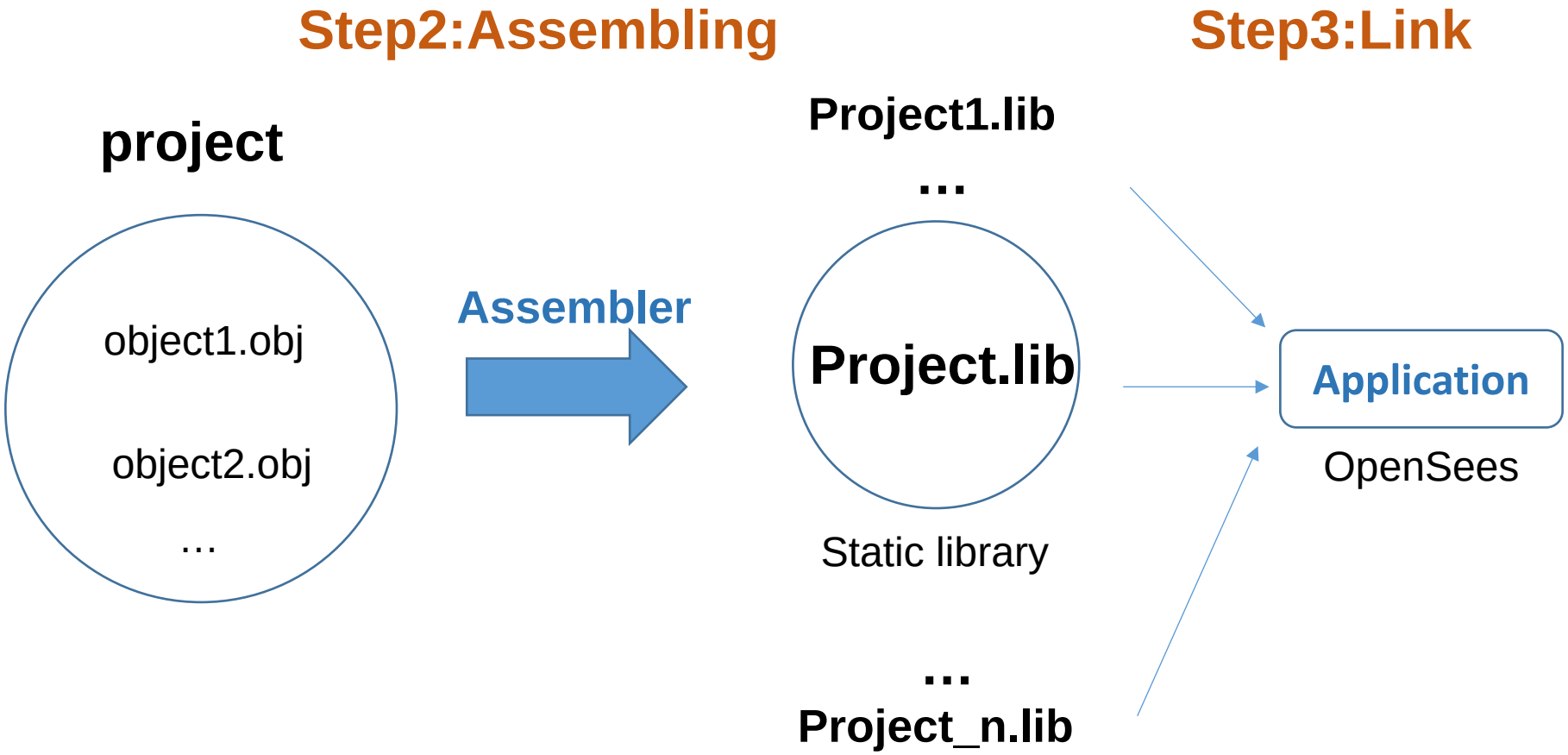
Modified/New Projects in OpenSees

- analysis
- domain
  - Node
  - Load
  - LoadPattern
- element
  - BeamColumn
  - Shell
- fire
  - FireModel
- HeatTransfer
- HTMain
  - Implementation of HT
- material
  - Uniaxial Mat
  - ND Mat
- recorder
- SIFBuilder
- tcl
  - Tcl entry

Step1:Compilation

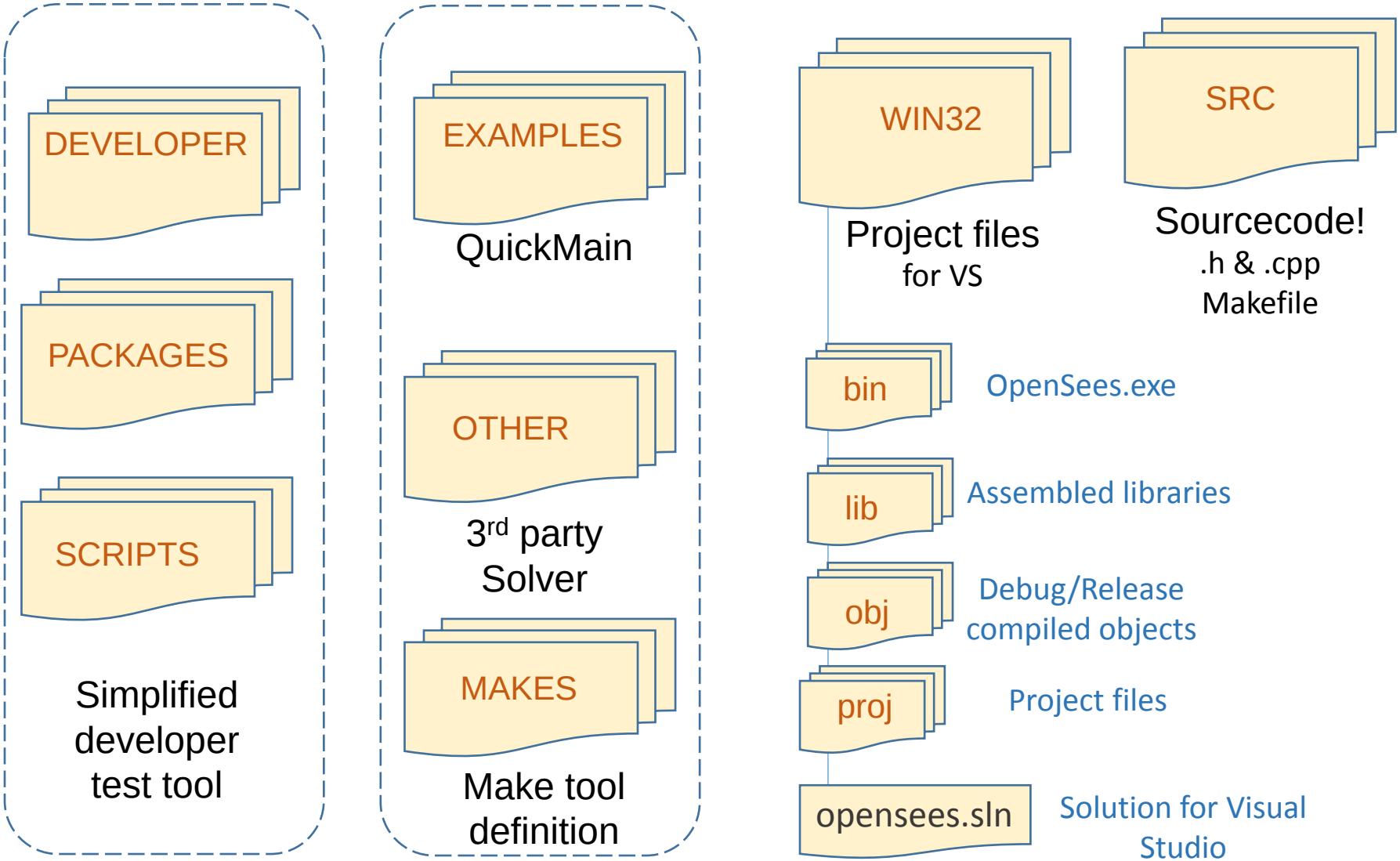


Building OpenSees



Building OpenSees

OpenSees Source Code Package

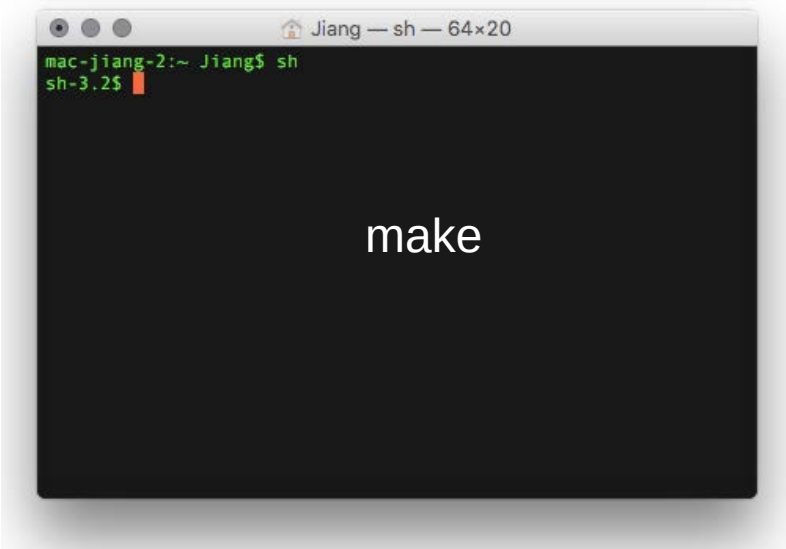


Building OpenSees

If you want to build it in Linux or MacOS?

GCC
&
GNU Make

Makefile.def + SRC + OTHER



Building OpenSees

GNU Make

GNU Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files.

Make gets its knowledge of how to build your program from a file called the *makefile*, which lists each of the non-source files and how to compute it from other files. When you write a program, you should write a makefile for it, so that it is possible to use Make to build and install the program.

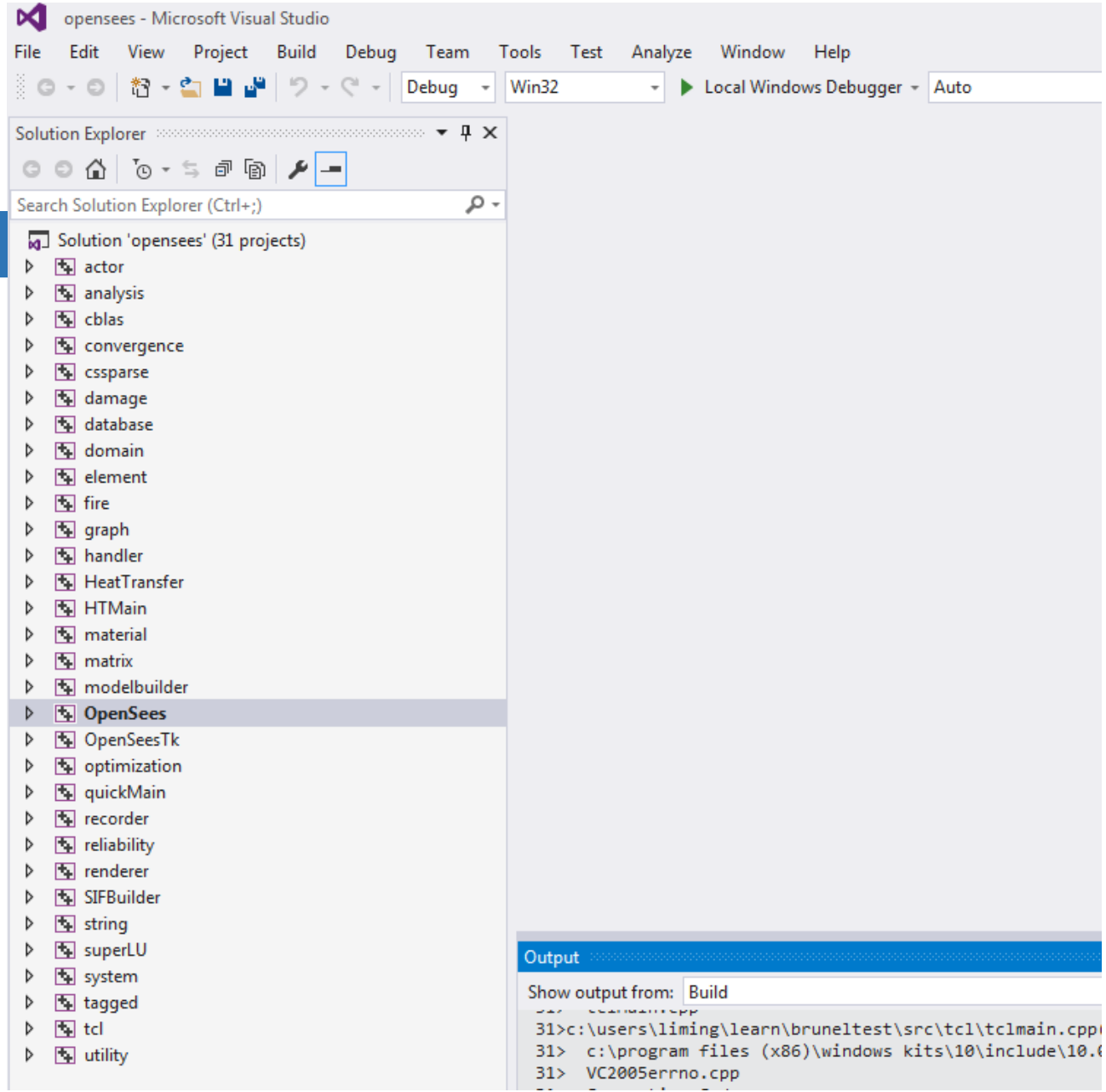
Makefile.def

- Program directory
- Paths (definition of SRC and OTHER directories)
- Libraries (definition of library location)
- Compilers (Compiler location & compiler and linker tags)
- Compilation behaviour
- Other supporting libraries
- Include files

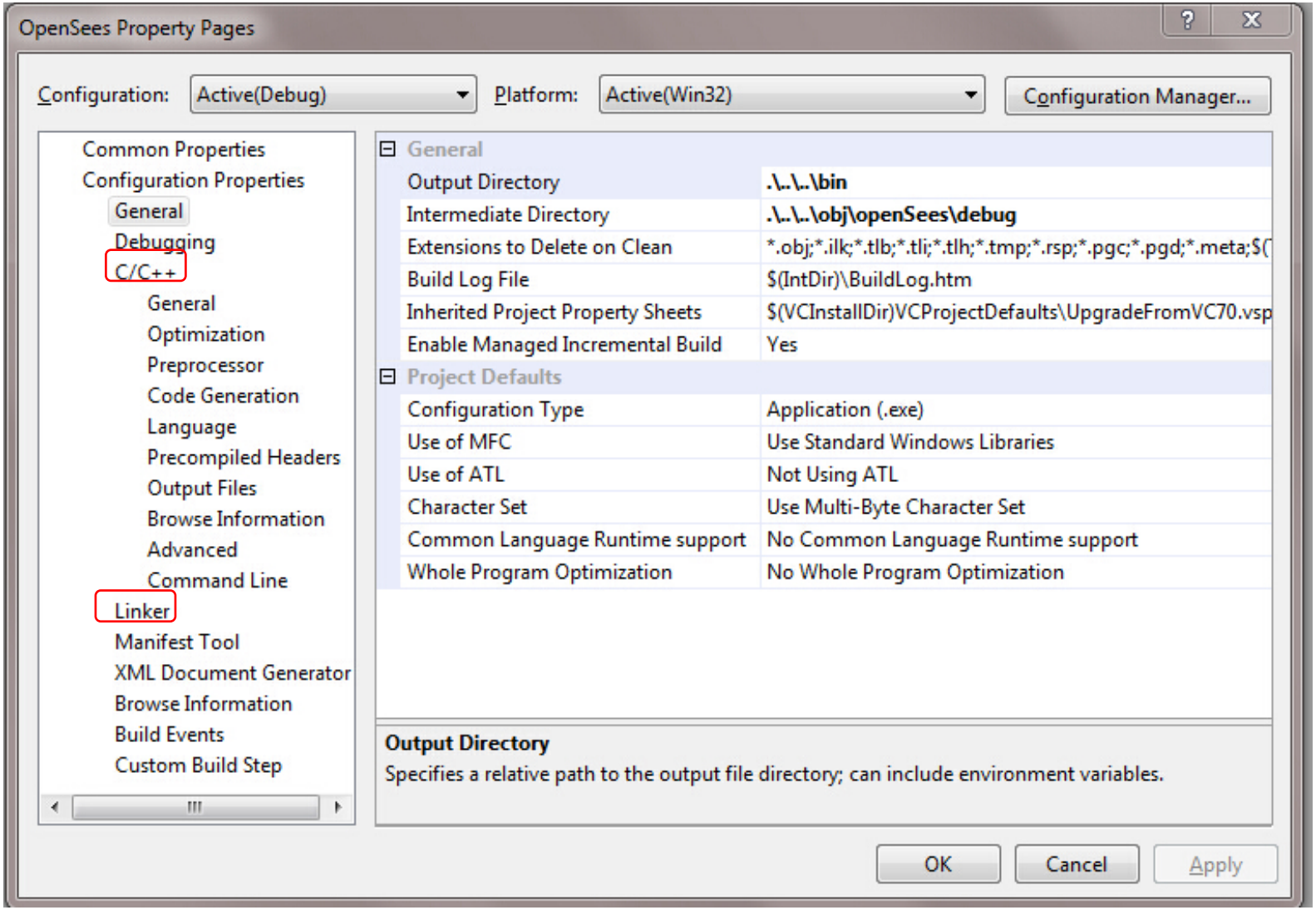
Building OpenSees

Using Windows PC

This is what **Visual Studio** looks like!



Building OpenSees



Give it a try to build your own OpenSees…

OPENSEES WORKSHOP

Day2: Add a new class to the framework

Add a new class to the framework: a material class example

Add a new material class

ElasticMaterialNewThermal

1. Find the material class which is most similar to the class you are trying to create

Add a new material class

ElasticMaterialNewThermal

2. Find the ‘similar’ material class file location:

OpenSees/SRC/material/uniaxial

Add a new material class

ElasticMaterialNewThermal

3. Make a copy of the header and source files in the same folder and rename them as: *ElasticMaterialNewThermal.cpp and ElasticMaterialNewThermal.h*.

Add a new material class

ElasticMaterialNewThermal

5. Add the two newly created files to the material project in the solution explorer:

Right click on *uniaxial* → add → Existing Item.

Select the source and header files for the new material from *OpenSees\SRC\material\uniaxial* and click *Add*.

Add a new material class

ElasticMaterialNewThermal

6. Open both the source and header files in a text editor (Notepad ++ or Microsoft Visual Studio editor) and make changes: replace the keyword *ElasticMaterialThermal* to *ElasticMaterialNewThermal*.

Add a new material class

ElasticMaterialNewThermal

7. Add a couple of lines for the newly created material in

TclModelBuilderUniaxialMaterialCommand.cpp.

declaration:

*extern void *OPS NewElasticMaterialNewThermal(void);*

Add a new material class

8. Add a couple of lines for the newly created material in

TclModelBuilderUniaxialMaterialCommand.cpp.

In function: TclModelBuilderUniaxialMaterialCommand()

```
else if (strcmp(argv[1],"ElasticNewThermal") == 0) {
        void *theMat = OPS NewElasticMaterialNewThermal();
        if (theMat != 0)
                theMaterial = (UniaxialMaterial *)theMat;
        else
                return TCL ERROR;}
```

Add a new material class

ElasticMaterialNewThermal

9. Rebuild only the material project:

Right click on material → Project Only → Rebuild Only material.

Add a new material class

ElasticMaterialNewThermal

Step 1: Add a tcl file *test.tcl* to the in the project *openSees*. Add a line or two using the new material. For example:

model BasicBuilder -ndm 2 -ndf 3;

uniaxialMaterial ElasticNewThermal 1 20000 0.01;

Step 2: Debug the successfully built version of OpenSees with the new material to bring up the OpenSees command window.

Step 3: Source the test.tcl file by typing *source test.tcl*. If the program outputs the desired lines (if added) or exits with no errors, you have SUCCESSFULLY added a new material.

OPENSEES WORKSHOP

Day2: How to add a project

How to Add a Project

What in a new Project?

project

object1.h

object1.cpp

object2.h

object2.cpp

…

Framework Hierarchy

In a header file (.h)

Inclusion of other header files
Declaration of variables
Declaration of functions

In a source file (.cpp)

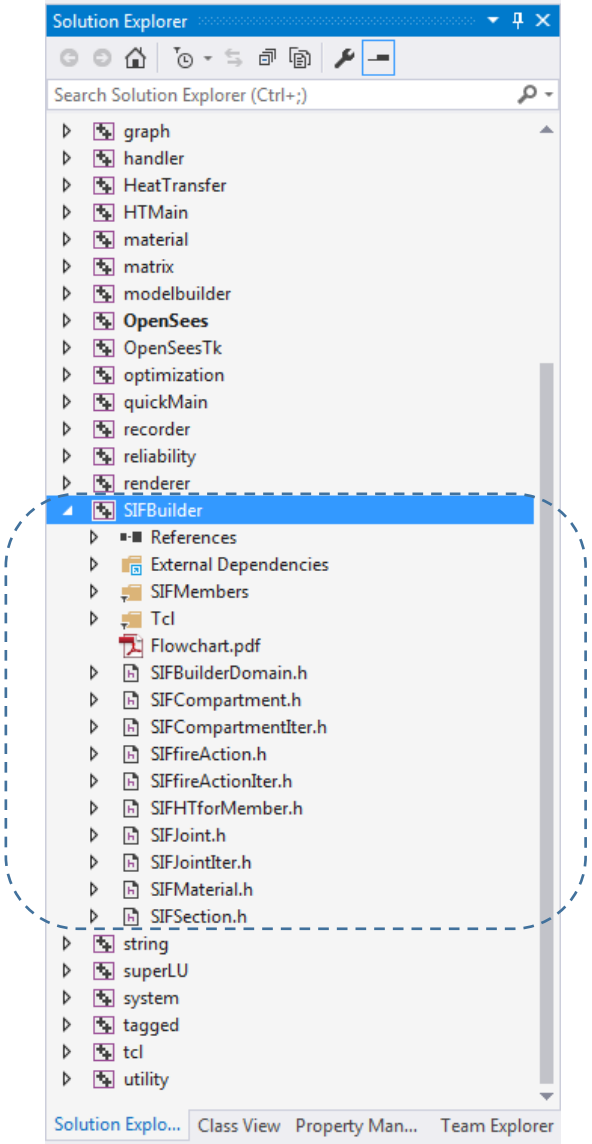
Inclusion of header files
Constructors of class
Destructor of class
Definition of functions

Prepare the files, and save them in the right folder

OpenSees/SRC/<your project>

How to Add a Project

- Add a new Project to OpenSees



❖ Create a project folder in

OpenSees/win32/proj/<your project>

❖ Add this new project

--if it is completely new, headers and sources have to be added;

--if it is not, files are imported automatically as the structure has been defined in the proj file

How to Add a Project

❖ **Project property** (right click at the project->configuration properties)

-Project properties are defined for **debug** and **release** separately

-Add the dependencies(additional included directories)

subfolders in SRC/<project name>

-Preprocessor tag(_SIFBUILDER, _HEATTRANSFER)

#ifdef could selectively activate code block

-output as multi-threaded debug for debugging build

-multi-thread for release

How to Add a Project

SIFBuilder Property Pages

Configuration: Debug Platform: Active(Win32) Configuration Manager...

- Configuration Properties
  - General
  - Debugging
  - VC++ Directories
  - C/C++
    - General
    - Optimization
    - Preprocessor
    - Code Generation
    - Language
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Advanced
    - All Options
    - Command Line
  - Librarian
  - XML Document Generator
  - Browse Information
  - Build Events
  - Custom Build Step
  - Code Analysis

Additional Include Directories	..\..\..\SRC\system_of_eqn\linearSOE\sparseSYM;..\..\..\SRC\syste
Additional #using Directories	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	
Warning Level	
Treat Warnings As Errors	
Warning Version	
SDL checks	
Multi-processor Compilation	

Additional Include Directories

Specifies one or more directories to add to th
(/I[path])

Additional Include Directories

..\..\..\SRC\system_of_eqn\linearSOE\sparseSYM
..\..\..\SRC\system_of_eqn\linearSOE\bandSPD
..\..\..\SRC\system_of_eqn\linearSOE
..\..\..\SRC\system_of_eqn\linearSOE\bandGEN
..\..\..\SRC\HeatTransfer\HeatTransferAnalysis\HeatTransferDOFNumber
..\..\..\OTHER\SuperLU_4.1\SRC

Evaluated value:

..\..\..\SRC\system_of_eqn\linearSOE\sparseSYM
..\..\..\SRC\system_of_eqn\linearSOE\bandSPD
..\..\..\SRC\system_of_eqn\linearSOE
..\..\..\SRC\system_of_eqn\linearSOE\bandGEN

Inherited values:

☑ Inherit from parent or project defaults Macros>>

OK Cancel

How to Add a Project

Possible Errors

Compiler

- Not including right headers
- Deleted variables (destructor)
- Mismatched returned value from a function
- Mismatched constructor and usage of a class
- Incorrect project properties

Linker

- Not including right libraries
- Referenced function can not be found because it's not correctly defined
- Library is not produced
- Linker property of OpenSees project

OPENSEES WORKSHOP

Day2: How to add Tcl commands

Add Tcl Commands

- The original Tcl offers a large collection of commands

  -File operation: eof, pwd, append,open, etc.

  -Control flow: if, for, switch, while, etc.

  -Those commands are well documented at the page: http://www.tcl.tk/man/tcl8.5/TclCmd/contents.htm

- The Tcl library

  -The library provides an interface to add extended Tcl commands

  -Tcl library was imported to the computer when the installation of Tcl happened.

Add Tcl Commands

- Tcl.h and Tcl.lib

  -Tcl is installed in C:\Program Files\Tcl in Windows

  -Tcl.h is a header file which has prototypes of the built-in functions.

  -The functions are enclosed in the Tcl library.

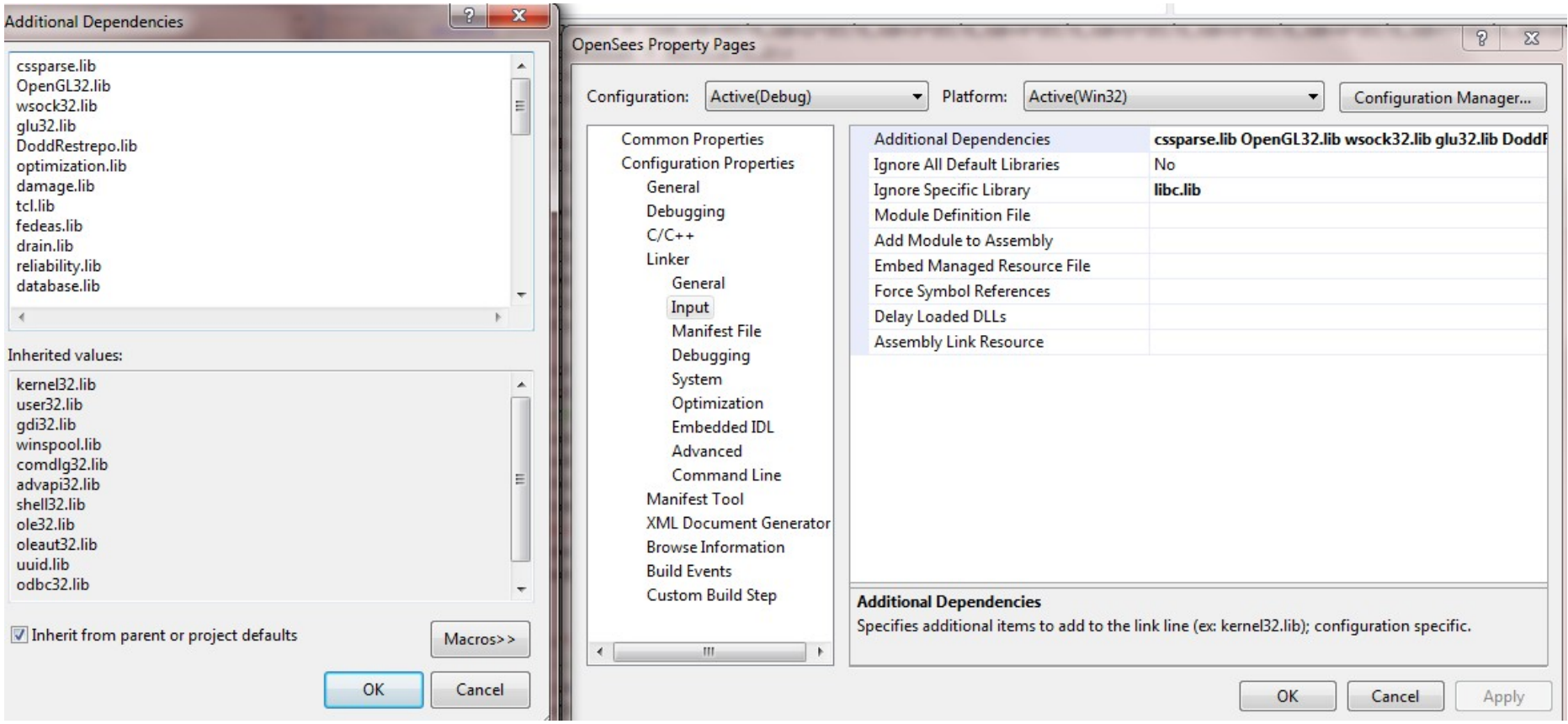
- Tcl in OpenSees

  -OpenSees inherits the original Tcl commands and extends the command library.

  -Most of commands are developed within the project tcl

  -The others are located in modelbuilder and sub-projects.

Add Tcl Commands



Extended Tcl command

- "Source" command

  - Global commands are located in commands.cpp

  in the function of OpenSeesAppInit(Tcl_Interp *interp)

```
Tcl_CreateObjCommand(interp, "source", &OPS_SourceCmd,
(ClientData)NULL, (Tcl_CmdDeleteProc*)NULL);
```

```
int  OPS_SourceCmd(ClientData clientData, Tcl_Interp *interp,
int  argc, Tcl_Obj * const *argv);
```

```
int  OPS_SourceCmd(
……
…..
}
```

Commands in Modelbuilder

Modelbuilder is called in commands.cpp to create a TclModelBuilder Class

TclModelBuilder has a huge constructor, which contains the creation of modelbuilder-related commands

The strategy of extending Tcl commands is creating a global command in commands.cpp, then putting the definition of new commands in the constructor of a TclModelBuilder type class.

“HeatTransfer” ->TclHeatTransferModelBuilder

Extended Tcl command

- Creating your own command

-i.e. in TclHeatTransferModule class

1) Using Tcl_CreateCommand to add a new command;

```
Tcl_CreateCommand(interp, "HTMaterial", (Tcl_CmdProc* )
TclHeatTransferCommand_addHTMaterial,(ClientData)NULL, NULL);
```

2)Specify the Procedure corresponding to this command;

```
Int TclHeatTransferCommand_addHTMaterial(ClientData clientData, Tcl_Interp *interp,
int argc, TCL_Char **argv)
```

```
if (theTclHTModule == 0) {
            opserr << "WARNING current HeatTransfer Module has been destroyed - HTMaterial\n";
             return TCL_ERROR;
}
if (theHTDomain == 0) {
   opserr << "WARNING no active HeatTransfer Domain - HTMaterial\n";
   return TCL_ERROR;
 }
HeatTransferMaterial* theHTMaterial=0;
int HTMaterialTag = 0;
if (Tcl_GetInt(interp, argv[2], &HTMaterialTag) != TCL_OK) {
opserr << "WARNING:: invalid material tag for defining HeatTransfer material: " << argv[1] << "\n"; return TCL_ERROR;
  }
 //Adding CarbonSteelEC3
if (strcmp(argv[1],"CarbonSteelEC3") == 0) {
     theHTMaterial = new CarbonSteelEC3(HTMaterialTag);
}
if(theHTMaterial!=0){
     theTclHTModule->addHTMaterial(*theHTMaterial);}
else{
opserr<<"WARNING: TclHTModule fail to add HeatTransfer Material: "<<argv[1]<<endln;}
return TCL_OK;
}
```

Extended Tcl command

- Creating your own command

-i.e. in TclHeatTransferModule class

1) Classes are mostly designed as tagged objects.

2) TclModelBuilder or domain classes holds the tags of materials, elements,etc.

```
theHTMaterials = new ArrayOfTaggedObjects(32);
```

3) ArraryOfTaggedObjects stores tags and correponding pointers to the objects.

```
theHTMaterial = new CarbonSteelEC3(HTMaterialTag);
```

4)Adding and returning the object pointer.

```
theTclHTModule->addHTMaterial(*theHTMaterial);
HeatTransferMaterial* TclHeatTransferModule->getHTMaterial(int tag)
```

Extended Tcl command

- Commonly used functions

1) Tcl_GetInt(interp, argv[2], &HTMaterialTag)

2) if (strcmp(argv[1],"CarbonSteelEC3") == 0)

3) opserr<<"WARNING: TclHTModule fail to add Simple Mesh: "<<argv[1]<<endln;

4) Argc, argv[ ]

Node 1 1 0;