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
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
Day 2: 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’;

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

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

**Heat Transfer**
- Heat transfer to the structural members due to fire action

**Thermo-mechanical**
- 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
Building OpenSees

Step 1: Compilation

project

object1.h
object1.cpp
object2.h
object2.cpp
...

Compiler

project

object1.obj
object2.obj
...

...
**Building OpenSees**

**Step 2: Assembling**
- project
  - object1.obj
  - object2.obj
  - ...

**Assembler**

**Step 3: Link**
- Project1.lib
  - ...
- Project.lib
  - Static library
  - ...
- Project\_n.lib

**Application**

OpenSees
Building OpenSees

OpenSees Source Code Package

- **DEVELOPER**
  - Packages
  - Scripts
  - Simplified developer test tool

- **EXAMPLES**
  - QuickMain
  - 3rd party Solver
  - Makes
    - Make tool definition

- **OTHER**
  - 3rd party Solver

- **WIN32**
  - Project files for VS
    - bin
    - lib
    - obj
    - proj
    - openses.sln

- **SRC**
  - Source code! .h & .cpp Makefile
  - Compiled objects
  - Assembled libraries
  - Debug/Release
  - Solution for Visual Studio
Building OpenSees

If you want to build it in Linux or MacOS?

Makefile.def + SRC + OTHER

GCC & GNU Make

make
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!
Give it a try to build your own OpenSees…
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

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

ElasticMaterialNewThermal

OpenSees/SRC/material/uniaxial
3. Make a copy of the header and source files in the same folder and rename them as: *ElasticMaterialNewThermal.cpp* and *ElasticMaterialNewThermal.h*. 
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*. 
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);
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 project `openSees`. Add a line or two using the new material. For example:

```tcl
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.
Day2: How to add a project
How to Add a Project

What in a new Project?

```markdown
project

object1.h
object1.cpp
object2.h
object2.cpp
...
```

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
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
Day2: How to 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.
• 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)

```c
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(
    ......
    ......
}
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]<<endl;
    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) ArrayOfTaggedObjects stores tags and corresponding 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]<<endl;
4) Argc, argv[ ]
   Node 1 1 0;