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 1

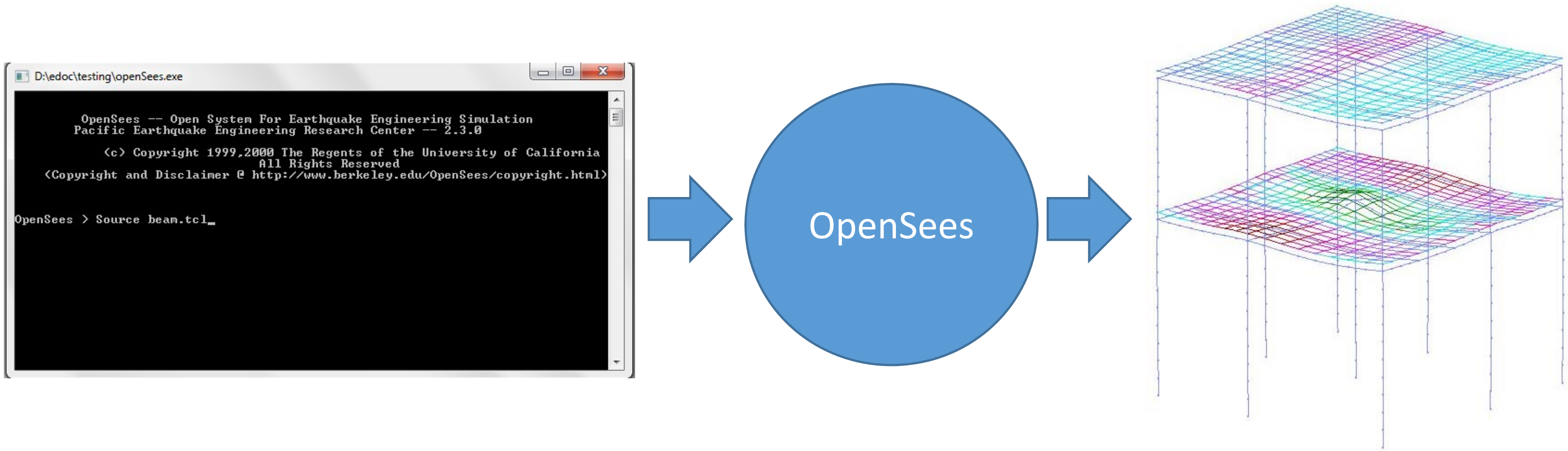
1. A Brief Intro & Preparation
2. Basic knowledge about Tcl
3. Getting Started with Standard OpenSees
4. Getting Started with OpenSees for Fire

OPENSEES WORKSHOP

Day1: A Brief Intro & Preparation

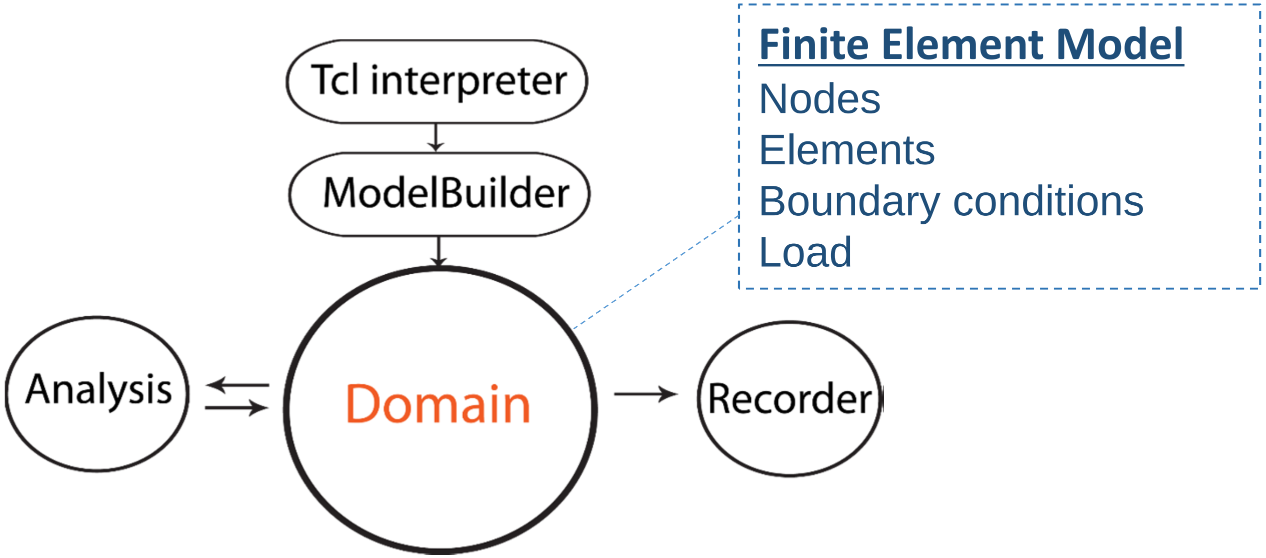
What is OpenSees?

- The Open System for **Earthquake Engineering** Simulation;
- Developed at **Berkeley** and based on Frank McKenna's PhD work;
- A software framework based on **finite element method;**
- It is written primarily in the **object-oriented** programming language C++;
- It uses **Tcl Interpreter** to interpret a user-input script.



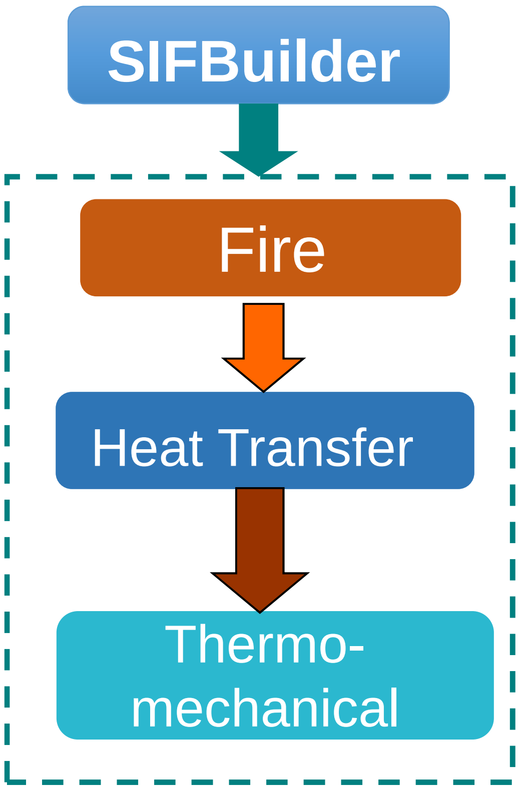
A Software Framework

- 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

Where to get OpenSees?

- Berkeley Main Site: http://opensees.berkeley.edu/

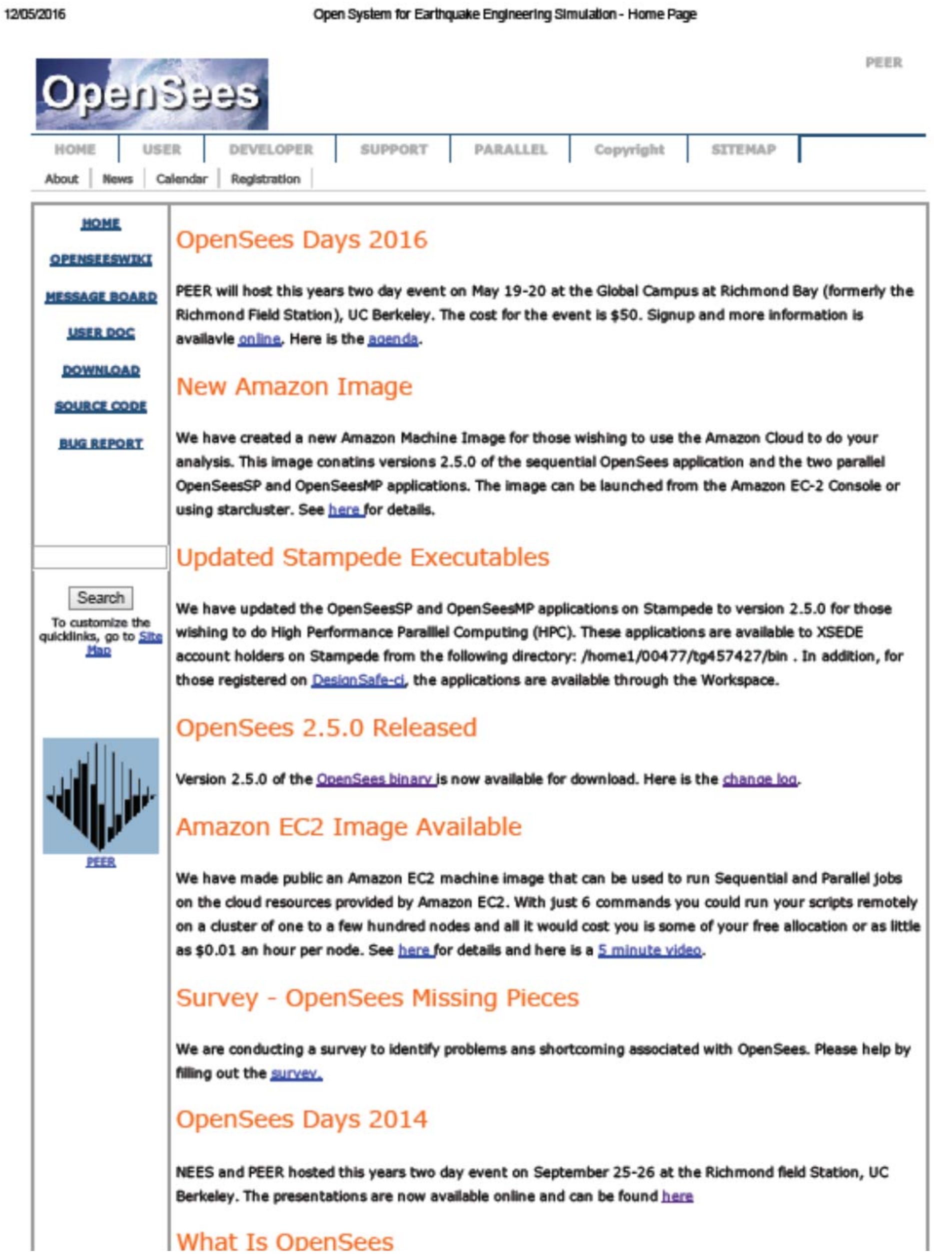
✓ Download

✓ User manual

✓ Source code

✓ Examples

Standard release


12/05/2016 Open System for Earthquake Engineering Simulation - Home Page

OpenSees PEER

HOME | USER | DEVELOPER | SUPPORT | PARALLEL | Copyright | SITEMAP

About | News | Calendar | Registration

HOME
OPENSEESWIKI
MESSAGE BOARD
USER DOC
DOWNLOAD
SOURCE CODE
BUG REPORT

Search

To customize the quicklinks, go to Site Map

PEER

OpenSees Days 2016

PEER will host this years two day event on May 19-20 at the Global Campus at Richmond Bay (formerly the Richmond Field Station), UC Berkeley. The cost for the event is $50. Signup and more information is availavle online. Here is the agenda.

New Amazon Image

We have created a new Amazon Machine Image for those wishing to use the Amazon Cloud to do your analysis. This image conatins versions 2.5.0 of the sequential OpenSees application and the two parallel OpenSeesSP and OpenSeesMP applications. The image can be launched from the Amazon EC-2 Console or using starcluster. See here for details.

Updated Stampede Executables

We have updated the OpenSeesSP and OpenSeesMP applications on Stampede to version 2.5.0 for those wishing to do High Performance Parallel Computing (HPC). These applications are available to XSEDE account holders on Stampede from the following directory: /home1/00477/tg457427/bin . In addition, for those registered on DesignSafe-ci, the applications are available through the Workspace.

OpenSees 2.5.0 Released

Version 2.5.0 of the OpenSees binary is now available for download. Here is the change log.

Amazon EC2 Image Available

We have made public an Amazon EC2 machine image that can be used to run Sequential and Parallel jobs on the cloud resources provided by Amazon EC2. With just 6 commands you could run your scripts remotely on a cluster of one to a few hundred nodes and all it would cost you is some of your free allocation or as little as $0.01 an hour per node. See here for details and here is a 5 minute video.

Survey - OpenSees Missing Pieces

We are conducting a survey to identify problems ans shortcoming associated with OpenSees. Please help by filling out the survey.

OpenSees Days 2014

NEES and PEER hosted this years two day event on September 25-26 at the Richmond field Station, UC Berkeley. The presentations are now available online and can be found here

What Is OpenSees

Where to get OpenSees?

- Edinburgh Wiki Site:

https://www.wiki.ed.ac.uk/display/opensees

SIF
Release

Pages

UoE OpenSees

3 Added by Andrew McFarlane, last edited by Liming Jiang on Apr 29, 2014 (view change)

University of Edinburgh
OpenSees
Developers Group

Home | About | Users | Developers | Publications | Download

OpenSees

The Open System for Earthquake Engineering Simulation, featured as an object-oriented and open source framework.

About OpenSees at UoE

The OpenSees developers group based in the School of Engineering, University of Edinburgh first started in 2009. The aim of this work is to add a "structures in fire" modelling capability in OpenSees.

Users

A number of wiki pages are provided to help users to carry out thermomechanical analyses with OpenSees using simple examples.

Developers

A detailed description of all the new or modified classes developed for enabling thermomehcanical analyses in OpenSees.

Publications

Links to publications by the group are provided here.

Download

An executable version of OpenSees compiled for use in Windows can be downloaded and source codes developed can be browsed or downloaded. We'll update all the bug-fixing issues on that page.

Where to get OpenSees?

- GitHub Site: http://openseesforfire.github.io/

SIF Release

OPENSEES FOR FIRE

View the Project on GitHub
OpenSees@GitHub

View People
OpenSees for Fire Group

Go To **Berkeley**	Download **OpenSees**	View On **GitHub**

OpenSees

About

The OpenSees development for modelling 'structures in fire' was first started at University of Edinburgh in 2009. A couples of students and researchers worked on this long-term project with their own contributions which enable OpenSees to perform heat transfer and thermo-mechanical analyses.

Users

A number of web pages are constructed to offer the users a detailed guidance to the recently added capabilities within OpenSees

Developers

A detailed description of all the new or modified classes developed for enabling thermomehcanical analyses in OpenSees.

Publications

Relevent publications to OpenSees for fire development can be found from the above section

Download

An executable version of OpenSees compiled for use in Windows can be downloaded and source codes developed can be browsed or downloaded. We'll update all the bug-fixing issues on that page.

This project is maintained by
LimingXLiming

Hosted on GitHub Pages — Theme by orderedlist

Before we start…

Download

Computer

Tcl

- **Installation** file can be downloaded from **ActivateState** site;

ActiveState®

http://www.activestate.com/

- **Tcl 8.5** should be installed into a proper directory;

C:\Program files\Tcl

Download

Standard release

http://opensees.berkeley.edu/

Download page

32bit / 64bit

OpenSees

Mac Version

SIF Release

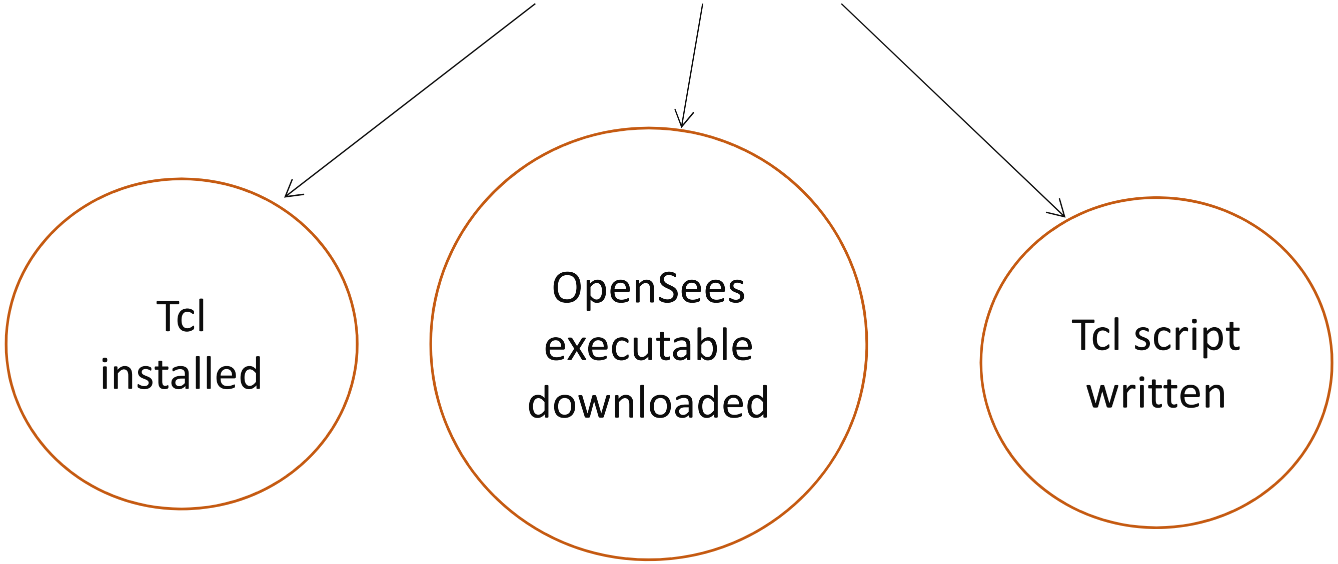
http://openseesforfire.github.io/

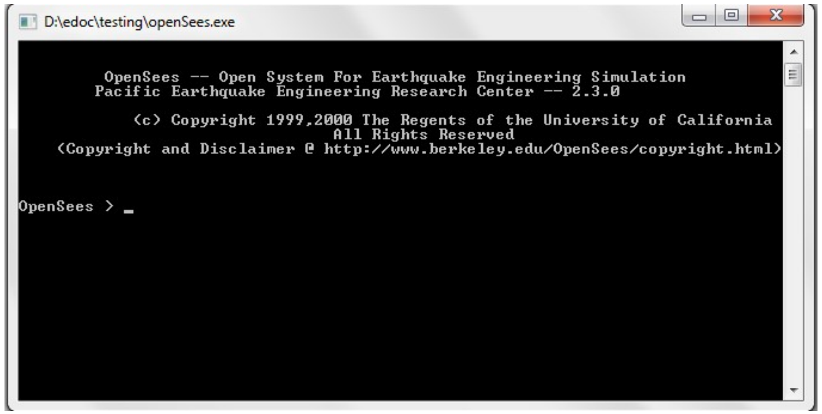
Download page

32bit

Preparation

To Use OpenSees





- ✓ OpenSees.exe placed in the same folder with Tcl script (recommended)
- ✓ Run the exe and import the script using source command

Dev Tool

Microsoft Visual Studio

2008 2010 2010 2012

etc

To Develop OpenSees

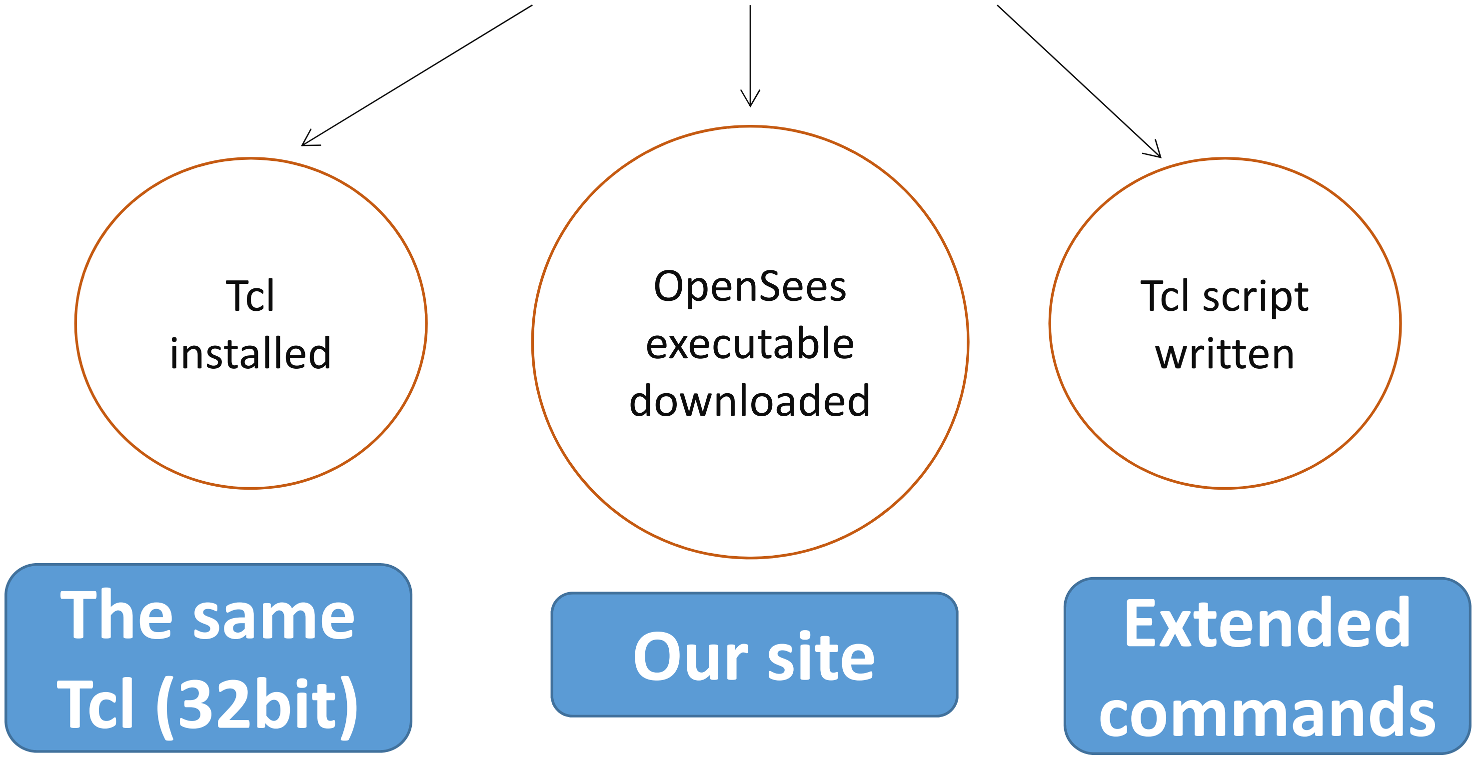
IDE
Integrated
Development
Environment

GCC compiler + Textmate

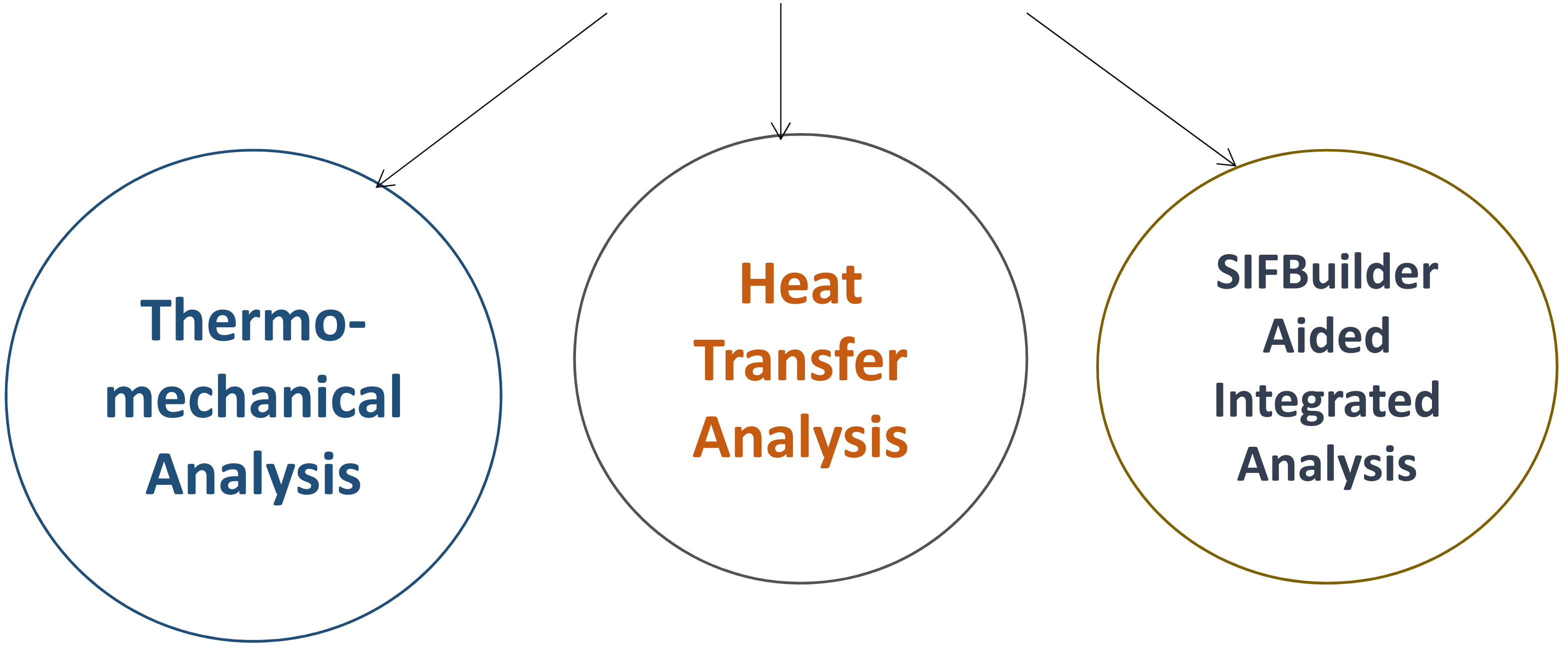
OPENSEES WORKSHOP

Day1: How to use OpenSees

To Use OpenSees modelling Structures in fire



To Use OpenSees modelling Structures in fire



Part 1

Part 2

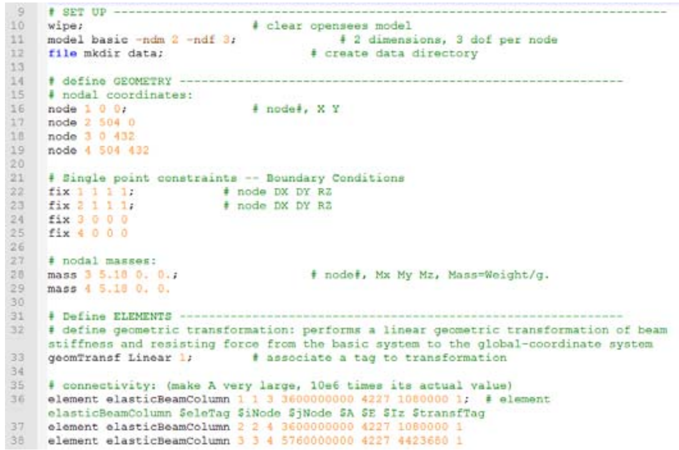
Part 3

To get started with OpenSees

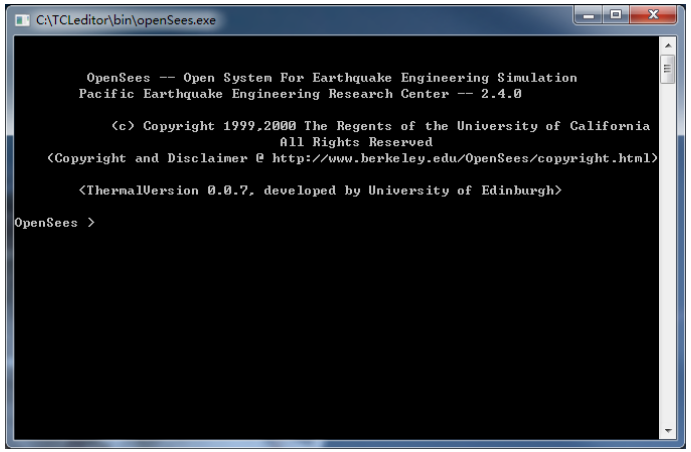
(Berkeley standard version)

How does it work?

1. Write your own **tcl script** to build up your model



2. Then it uses **OpenSees interpreters** to read Tcl commands (Tcl version 8.5) for finite element analysis



What is Tcl ?

Tcl is a dynamic programming language

- It is a **string** based command language;
- Variables and variable **substitution**;
- Expression evaluation**;**
- Basic **control structures** (if , while, for, foreach…);
- Procedures, file manipulation, sourcing other files.

Tcl resources

- Tcl documentation link: http://www.tcl.tk/doc/

- Tcl commands manual link:

  http://www.tcl.tk/man/tcl8.5/TclCmd/contents.htm

Tcl Syntax Rules

commandName $arg1 $arg2 $arg3 ..

- The first word is the command name
- The remaining words are the command arguments
- A Tcl Script is **a sequence of** Tcl Commands
- Commands in script are separated by **newlines** or **;**
- The words of a command are separated by **white spaces**

Tcl Syntax Rules

- **#** code that is **skipped** by the computer, but allows you/someone else to understand what is happening in the code

- **set** a variable is set with a symbolic name used to refer to some location in memory that has a value, such as *set a 2.0*

- **$** to use the value of the variable, such as *set b $a*

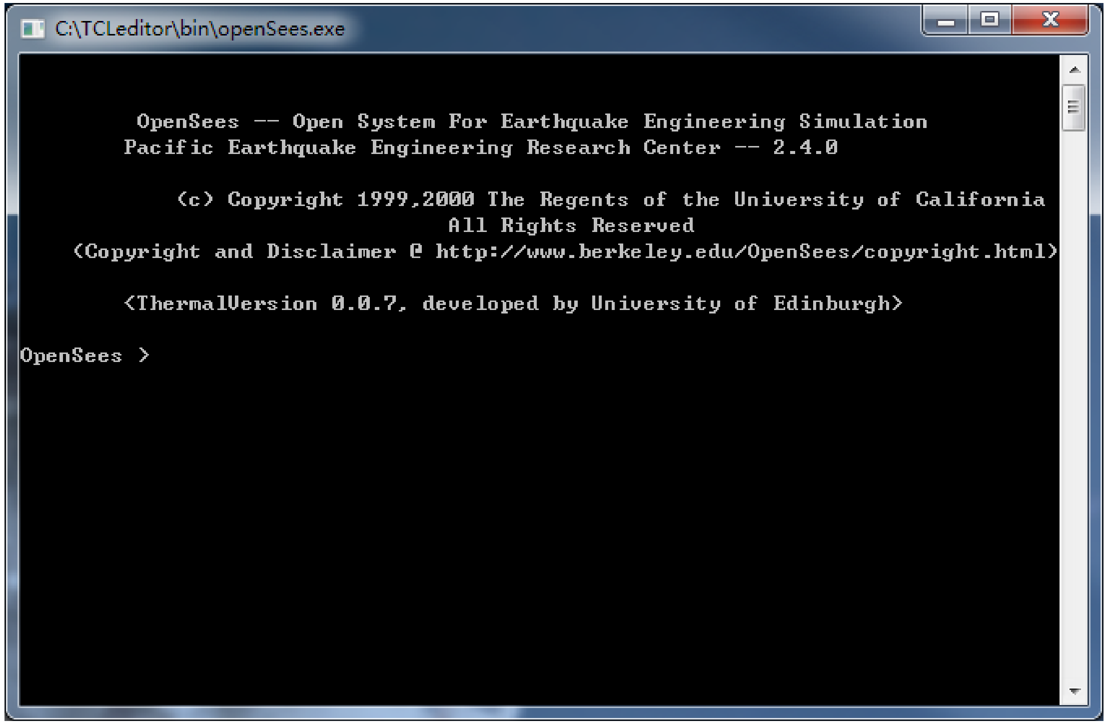
Tcl Syntax Rules

- **puts** result sent to screen, usually used for tcl script debugging
- **expr** command is used to calculate mathematical expressions, such as *expr sqrt(($x*$x)+($y*$y))*
- **proc** command is used to create procedures, first arg is your own procedure name, such as:

  proc sum {a b} {return [expr $a + $b]}

OpenSees Interpretors

The OpenSees interpreters are **tcl interpreters** which have been extended to include commands for **finite element analysis**



OpenSees Interpretors

```
# -----------------------------------------------------------
# Example 1. portal frame in 2D
# static pushover analysis of Portal Frame, with gravity.
# all units are in kip, inch, second
# elasticBeamColumn ELEMENT
#       Silvia Mazzoni & Frank McKenna, 2006
# -----------------------------------------------------------

# SET UP ----------------------------------------------------------------------------
wipe;                            # clear opensees model
model basic -ndm 2 -ndf 3;                   # 2 dimensions, 3 dof per node
file mkdir data;                         # create data directory

# define GEOMETRY -------------------------------------------------------------
# nodal coordinates:
node 1 0 0;                      # node#, X Y
node 2 504 0
node 3 0 432
node 4 504 432

# Single point constraints -- Boundary Conditions
fix 1 1 1 1;                 # node DX DY RZ
fix 2 1 1 1;                 # node DX DY RZ
fix 3 0 0 0
fix 4 0 0 0

# nodal masses:
mass 3 5.18 0. 0.;                       # node#, Mx My Mz, Mass=Weight/g.
mass 4 5.18 0. 0.

# Define ELEMENTS -------------------------------------------------------------
# define geometric transformation: performs a linear geometric transformation of beam stiffness and resisting
force from the basic system to the global-coordinate system
geomTransf Linear 1;             # associate a tag to transformation

# connectivity: (make A very large, 10e6 times its actual value)
element elasticBeamColumn 1 1 3 3600000000 4227 1080000 1;  # element elasticBeamColumn $eleTag $iNode $jNode $A
$E $Iz $transfTag
element elasticBeamColumn 2 2 4 3600000000 4227 1080000 1
element elasticBeamColumn 3 3 4 5760000000 4227 4423680 1

# Define RECORDERS -------------------------------------------------------------
recorder Node -file Data/DFree.out -time -node 3 4 -dof 1 2 3 disp;         # displacements of free nodes
recorder Node -file Data/DBase.out -time -node 1 2 -dof 1 2 3 disp;         # displacements of support nodes
recorder Node -file Data/RBase.out -time -node 1 2 -dof 1 2 3 reaction;     # support reaction
recorder Drift -file Data/Drift.out -time -iNode 1 2 -jNode 3 4 -dof 1  -perpDirn 2 ;   # lateral drift
recorder Element -file Data/FCol.out -time -ele 1 2 globalForce;            # element forces -- column
recorder Element -file Data/FBeam.out -time -ele 3 globalForce;         # element forces -- beam
```

Modeling

OpenSees Interpretors

```
# Define ELEMENTS -------------------------------------------------------------
# define geometric transformation: performs a linear geometric transformation of beam stiffness and resisting force from the basic system to the global-coordinate system
geomTransf Linear 1;         # associate a tag to transformation

# connectivity: (make A very large, 10e6 times its actual value)
element elasticBeamColumn 1 1 3 3600000000 4227 1080000 1;  # element elasticBeamColumn $eleTag $iNode $jNode $A $E $Iz $transfTag
element elasticBeamColumn 2 2 4 3600000000 4227 1080000 1
element elasticBeamColumn 3 3 4 5760000000 4227 4423680 1

# Define RECORDERS -------------------------------------------------------------
recorder Node -file Data/DFree.out -time -node 3 4 -dof 1 2 3 disp;         # displacements of free nodes
recorder Node -file Data/DBase.out -time -node 1 2 -dof 1 2 3 disp;         # displacements of support nodes
recorder Node -file Data/RBase.out -time -node 1 2 -dof 1 2 3 reaction;     # support reaction
recorder Drift -file Data/Drift.out -time -iNode 1 2 -jNode 3 4 -dof 1  -perpDirn 2 ;   # lateral drift
recorder Element -file Data/FCol.out -time -ele 1 2 globalForce;            # element forces -- column
recorder Element -file Data/FBeam.out -time -ele 3 globalForce;         # element forces -- beam

# define GRAVITY -------------------------------------------------------------
pattern Plain 1 Linear {
   eleLoad -ele 3 -type -beamUniform -7.94 ; # distributed superstructure-weight on beam
}
constraints Plain;                   # how it handles boundary conditions
numberer Plain;                  # renumber dof's to minimize band-width (optimization), if you want to
system BandGeneral;              # how to store and solve the system of equations in the analysis
test NormDispIncr 1.0e-8 6 ;                 # determine if convergence has been achieved at the end of an iteration step
algorithm Newton;                    # use Newton's solution algorithm: updates tangent stiffness at every iteration
integrator LoadControl 0.1;              # determine the next time step for an analysis, # apply gravity in 10 steps
analysis Static                  # define type of analysis static or transient
analyze 10;                  # perform gravity analysis
loadConst -time 0.0;                 # hold gravity constant and restart time

# define LATERAL load -------------------------------------------------------------
# Lateral load pattern
pattern Plain 2 Linear {
    load 3 2000. 0.0 0.0;            # node#, FX FY MZ -- representative lateral load at top nodes
    load 4 2000. 0.0 0.0;            # place 1/2 of the weight for each node to get shear coefficient
}

# pushover: diplacement controlled static analysis
integrator DisplacementControl 3 1 0.1;      # switch to displacement control, for node 11, dof 1, 0.1 increment
analyze 100;                     # apply 100 steps of pushover analysis to a displacement of 10

puts "Done!"
```

Output

Analysis

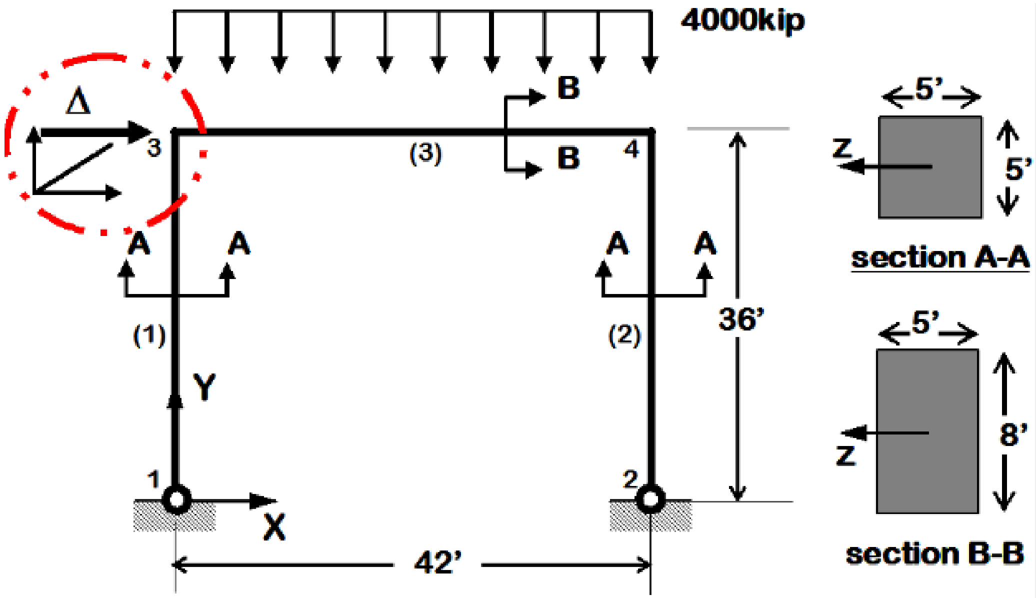
OpenSees Interpretors

The OpenSees interpreters are **tcl interpreters** which have been extended to include commands for **finite element analysis**

- **Modeling** – create nodes, elements and constraints
- **Analysis** – specify the analysis procedure.
- **Output** – specify what it is you want to monitor during the analysis.

2D Elastic Portal Frame

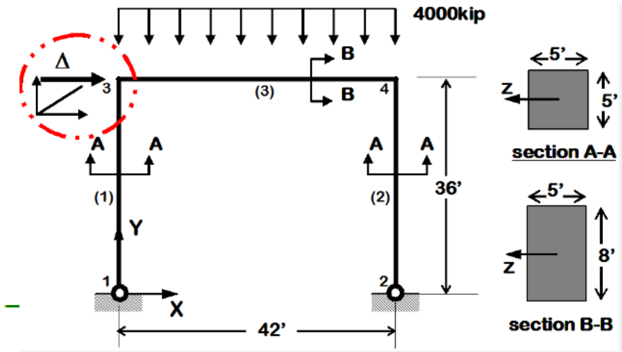
A simple model of **an elastic portal frame**. The objective of this example is to give an overview of input format.



An 2D portal frame example from Berkeley website

2D Elastic Portal Frame

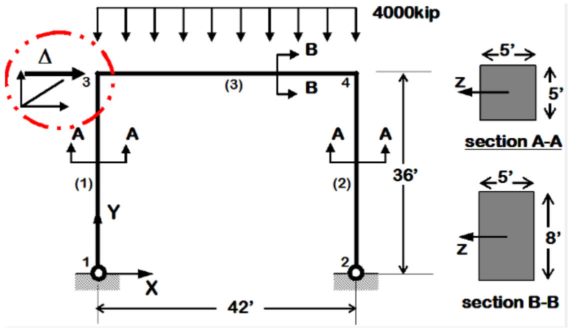
Comments & Model preparation




```
# ---------------------------------------------------
# Example 1. portal frame in 2D
# static pushover analysis of Portal Frame, with gravity.
# all units are in kip, inch, second
# elasticBeamColumn ELEMENT
#       Silvia Mazzoni & Frank McKenna, 2006
# ------------------------------------------------------------

# SET UP ------------------------------------------------------------------------
wipe;                        # clear opensees model
model basic -ndm 2 -ndf 3;               # 2 dimensions, 3 dof per node
file mkdir data;                     # create data directory
source DisplayPlane.tcl;        # procedure for displaying a plane in model
source DisplayModel2D.tcl;      # procedure for displaying 2D perspective of model
```

2D Elastic Portal Frame

Building up the model

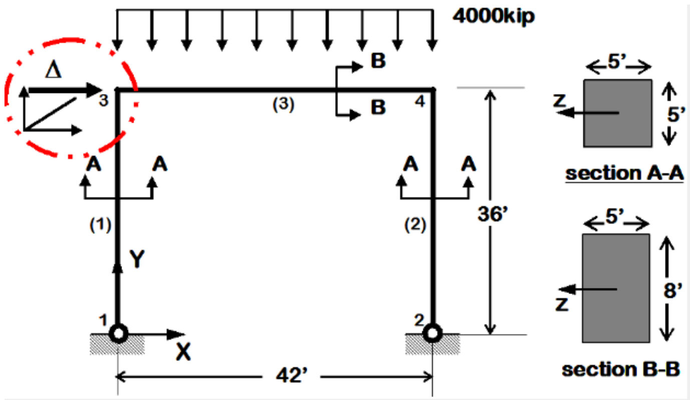



```
# define GEOMETRY -------------------------------------------------------------
# nodal coordinates:
node 1 0 0;                 # node#, X Y
node 2 504 0
node 3 0 432
node 4 504 432

# Single point constraints -- Boundary Conditions
fix 1 1 1 1;            # node DX DY RZ
fix 2 1 1 1;            # node DX DY RZ
fix 3 0 0 0
fix 4 0 0 0

# nodal masses:
mass 3 5.18 0. 0.;                   # node#, Mx My Mz, Mass=Weight/g.
mass 4 5.18 0. 0.

# Define ELEMENTS -------------------------------------------------------------
# define geometric transformation: performs a linear geometric transformation of beam stiffness and
force from the basic system to the global-coordinate system
geomTransf Linear 1;         # associate a tag to transformation

# connectivity: (make A very large, 10e6 times its actual value)
element elasticBeamColumn 1 1 3 3600000000 4227 1080000 1;  # element elasticBeamColumn $eleTag $iN
$A $E $Iz $transfTag
element elasticBeamColumn 2 2 4 3600000000 4227 1080000 1
element elasticBeamColumn 3 3 4 5760000000 4227 4423680 1
```

2D Elastic Portal Frame

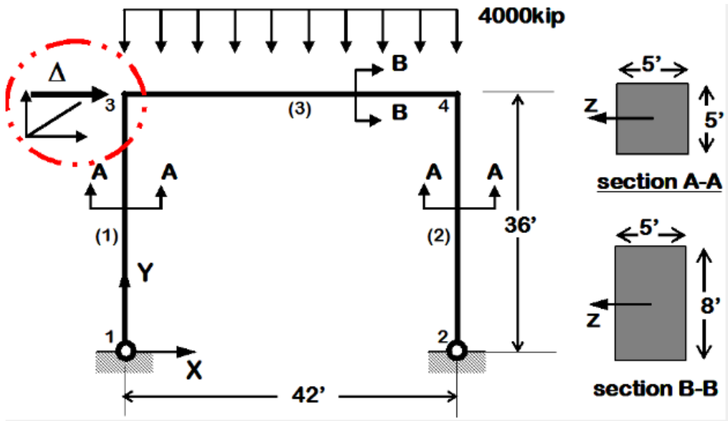
Define recorders

```
# Define RECORDERS -------------------------------------------------------------
recorder Node -file Data/DFree.out -time -node 3 4 -dof 1 2 3 disp;         # displacements of free nod
recorder Node -file Data/DBase.out -time -node 1 2 -dof 1 2 3 disp;         # displacements of support
recorder Node -file Data/RBase.out -time -node 1 2 -dof 1 2 3 reaction;     # support reaction
recorder Drift -file Data/Drift.out -time -iNode 1 2 -jNode 3 4 -dof 1  -perpDirn 2 ;   # lateral drift
recorder Element -file Data/FCol.out -time -ele 1 2 globalForce;                # element forces -- column
recorder Element -file Data/FBeam.out -time -ele 3 globalForce;             # element forces -- beam

# view the deformed shape
set ViewScale 5;
DisplayModel2D DeformedShape $ViewScale ;   # display deformed shape, the scaling factor needs to be ad
for each model

```



2D Elastic Portal Frame

Define gravity & analysis

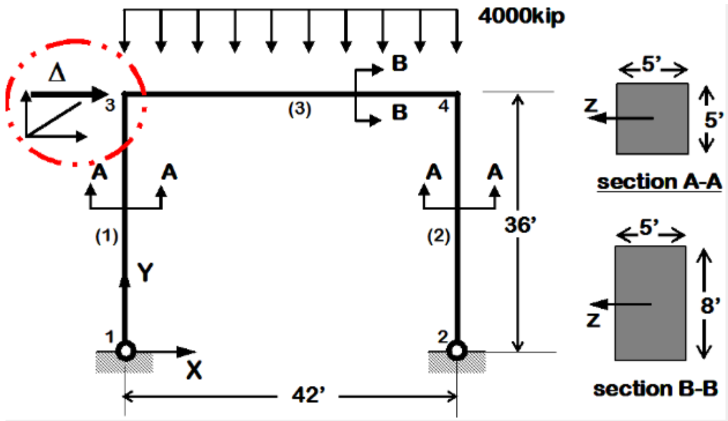



```
# define GRAVITY -------------------------------------------------------------
pattern Plain 1 Linear {
   eleLoad -ele 3 -type -beamUniform -7.94 ; # distributed superstructure-weight on beam
}
constraints Plain;                  # how it handles boundary conditions
numberer Plain;                 # renumber dof's to minimize band-width (optimization), if you
system BandGeneral;             # how to store and solve the system of equations in the analysi
test NormDispIncr 1.0e-8 6 ;              # determine if convergence has been achieved at the iteration step
algorithm Newton;                   # use Newton's solution algorithm: updates tangent stiffnes iteration
integrator LoadControl 0.1;             # determine the next time step for an analysis, # apply steps
analysis Static                 # define type of analysis static or transient
analyze 10;                 # perform gravity analysis
loadConst -time 0.0;                # hold gravity constant and restart time
```

2D Elastic Portal Frame

Define lateral load

& analysis




```
# define LATERAL load ------------------------------------------------------------
# Lateral load pattern
pattern Plain 2 Linear {
    load 3 2000. 0.0 0.0;          # node#, FX FY MZ -- representative lateral load at
    load 4 2000. 0.0 0.0;          # place 1/2 of the weight for each node to get shea
}

# pushover: diplacement controlled static analysis
integrator DisplacementControl 3 1 0.1;      # switch to displacement control, for node
analyze 100;                    # apply 100 steps of pushover analysis to a displacemen

puts "Done!"
```

2D Elastic Portal Frame

Analysis results can

be found in:

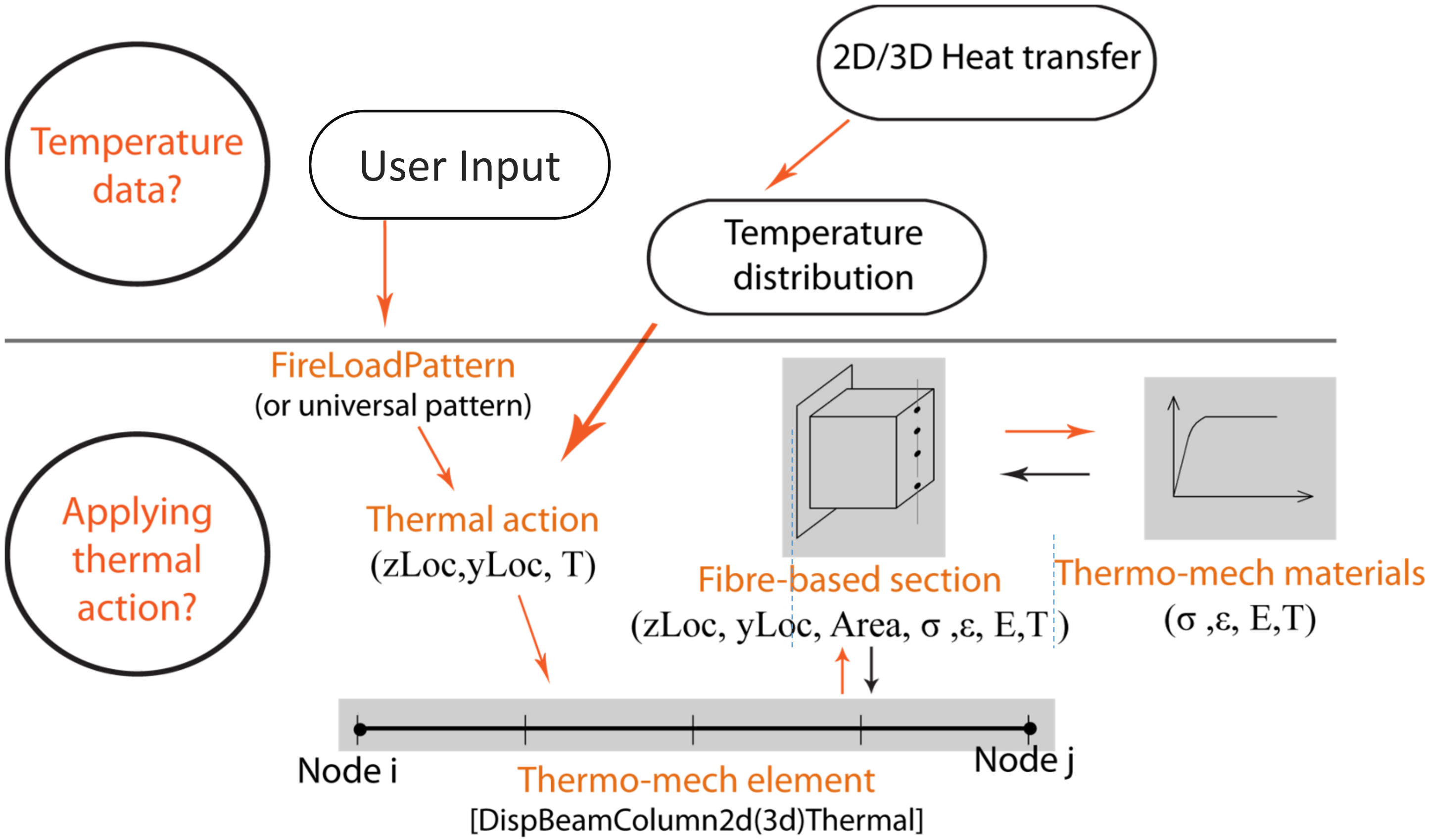
.../bin/data

```
0.1 21.1796 200.088 6099.74 -21.1796 200.088 -6099.74
0.2 42.3593 400.176 12199.5 -42.3593 400.176 -12199.5
0.3 63.5389 600.264 18299.2 -63.5389 600.264 -18299.2
0.4 84.7186 800.352 24399 -84.7186 800.352 -24399
0.5 105.898 1000.44 30498.7 -105.898 1000.44 -30498.7
0.6 127.078 1200.53 36598.4 -127.078 1200.53 -36598.4
0.7 148.258 1400.62 42698.2 -148.258 1400.62 -42698.2
0.8 169.437 1600.7 48797.9 -169.437 1600.7 -48797.9
0.9 190.617 1800.79 54897.6 -190.617 1800.79 -54897.6
1 211.796 2000.88 60997.4 -211.796 2000.88 -60997.4
0.0299084 211.796 1951.93 48662.5 -211.796 2049.83 -73332.2
0.0598167 211.796 1902.98 36327.7 -211.796 2098.78 -85667.1
0.0897251 211.796 1854.04 23992.8 -211.796 2147.72 -98001.9
0.119633 211.796 1805.09 11658 -211.796 2196.67 -110337
0.149542 211.796 1756.14 -676.886 -211.796 2245.62 -122672
0.17945 211.796 1707.19 -13011.7 -211.796 2294.57 -135006
0.209359 211.796 1658.25 -25346.6 -211.796 2343.51 -147341
0.239267 211.796 1609.3 -37681.4 -211.796 2392.46 -159676
0.269175 211.796 1560.35 -50016.3 -211.796 2441.41 -172011
0.299084 211.796 1511.4 -62351.2 -211.796 2490.36 -184346
0.328992 211.796 1462.45 -74686 -211.796 2539.31 -196681
0.3589 211.796 1413.51 -87020.9 -211.796 2588.25 -209016
0.388809 211.796 1364.56 -99355.7 -211.796 2637.2 -221350
0.418717 211.796 1315.61 -111691 -211.796 2686.15 -233685
0.448625 211.796 1266.66 -124025 -211.796 2735.1 -246020
0.478534 211.796 1217.71 -136360 -211.796 2784.05 -258355
0.508442 211.796 1168.77 -148695 -211.796 2832.99 -270690
0.53835 211.796 1119.82 -161030 -211.796 2881.94 -283025
0.568259 211.796 1070.87 -173365 -211.796 2930.89 -295360
0.598167 211.796 1021.92 -185700 -211.796 2979.84 -307694
0.628076 211.796 972.976 -198035 -211.796 3028.78 -320029
0.657984 211.796 924.028 -210369 -211.796 3077.73 -332364
0.687892 211.796 875.08 -222704 -211.796 3126.68 -344699
0.717801 211.796 826.132 -235039 -211.796 3175.63 -357034
0.747709 211.796 777.184 -247374 -211.796 3224.58 -369369
0.777617 211.796 728.236 -259709 -211.796 3273.52 -381704
```

First column is the pseudo-time

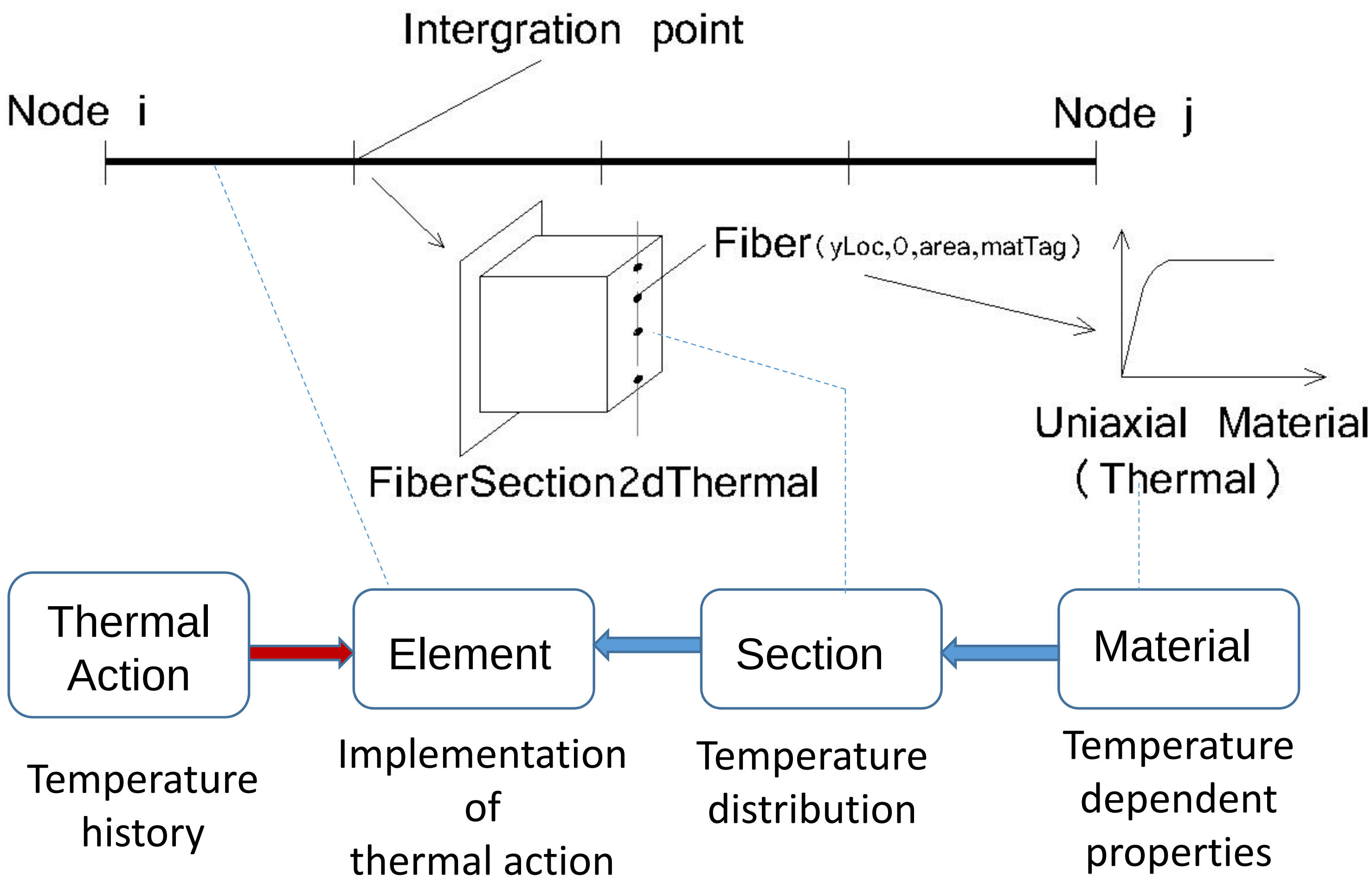
To run a thermo-mechanical analysis in OpenSees…

Modules extended for Thermo-mechanical analyses



BeamColumn element

Modules extended for Thermo-mechanical analyses



Getting Started!

Thermo-mechanical analyses

A beam example

Files in the folder

DisplayModel2D.tcl

DisplayModel3D.tcl

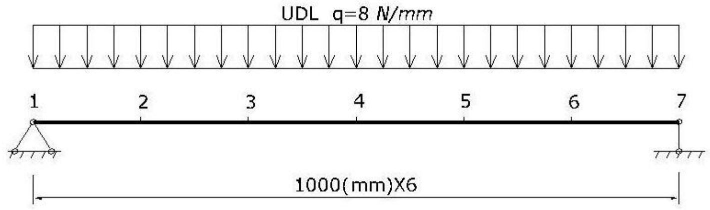
DisplayPlane.tcl

example-SteelBeam-uniform.tcl

Wsection.tcl

- Download from website
- Place the Tcl script with OpenSees

- A steel I-section beam
- Simply supported
- UDL q=8N/mm
- Uniform temperature



example-SteelBeam-uniform.tcl

```
wipe;
model BasicBuilder -ndm 2 -ndf 3;
source Wsection.tcl;
source DisplayPlane.tcl
source DisplayModel2D.tcl

node 1 0 0;
node 2 1000 0;
node 3 2000 0;
node 4 3000 0;
node 5 4000 0;
node 6 5000 0;
node 7 6000 0;

fix 1 1 1 0;
fix 7 0 1 0;
```

Clear the domain

2D model

External script for I-section

Display facilities

Nodal definition

Boundary condition

Using OpenSees

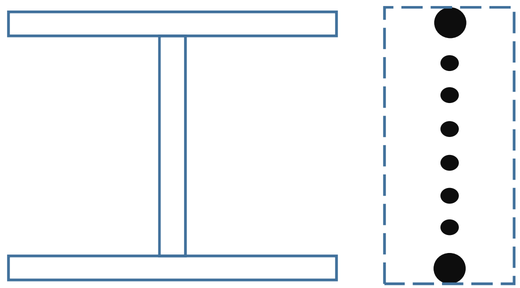
example-SteelBeam-uniform.tcl

```
uniaxialMaterial  SteelECThermal  1 308 2.1e5;

set d 355;     set bf 171.5;
set tf 11.5;   set tw 7.4;
set nfdw 8;    set nftw 1;
set nfbf 1;    set nftf 4;

Wsection  1 1 $d $bf $tf $tw $nfdw $nftw $nfbf $nftf
```

Thermo-mechanical material

I-section geometry

Call the pre-defined I section script



```
section fiberSecThermal  $secID  {
    fiber $yLoc $zLoc $A $matTag
    patch  quad $matTag $nIJ $nJK $yI  $zI $yJ $Zj $yK $zK $yL $zL
}
```

Fibre based I-section

example-SteelBeam-uniform.tcl

```
recorder Node -file 1.out -time -node 1 -dof 1 2 3 disp;
pattern Plain 1 Linear {

  eleLoad -ele 1 -type -beamUniform $UDL 0
   ….
  eleLoad -ele 6 -type -beamUniform $UDL 0
}
constraints Plain;
..
integrator LoadControl 0.1;
analysis Static;
analyze 10;

puts "Fire";
loadConst -time 0.0
```

Recorder for data request

UDL applied

Analysis control for UDL

Print command

Reset Pseudo time

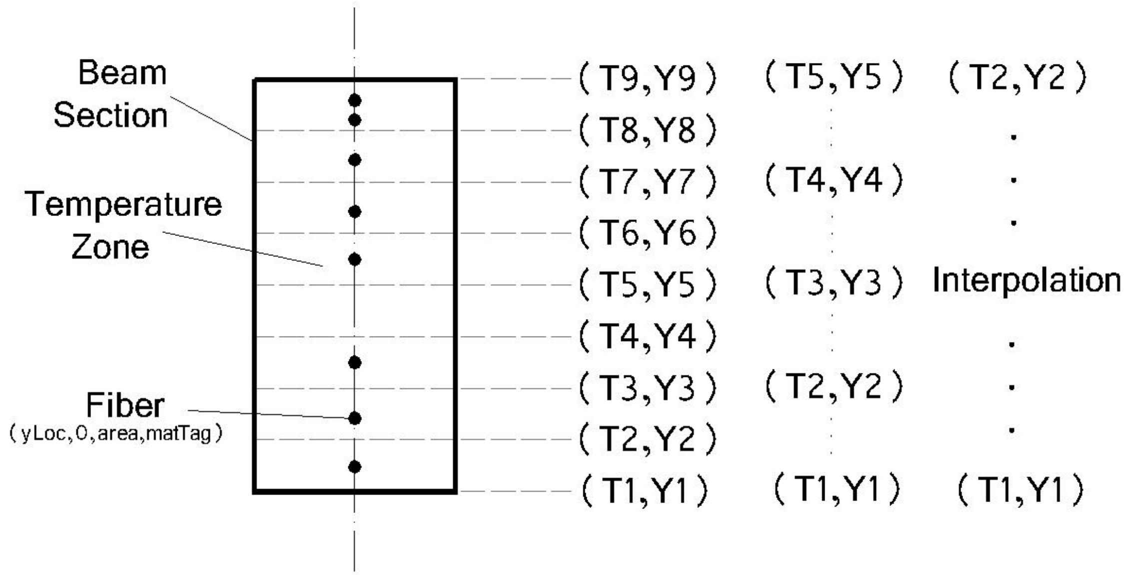
ts=0~1;

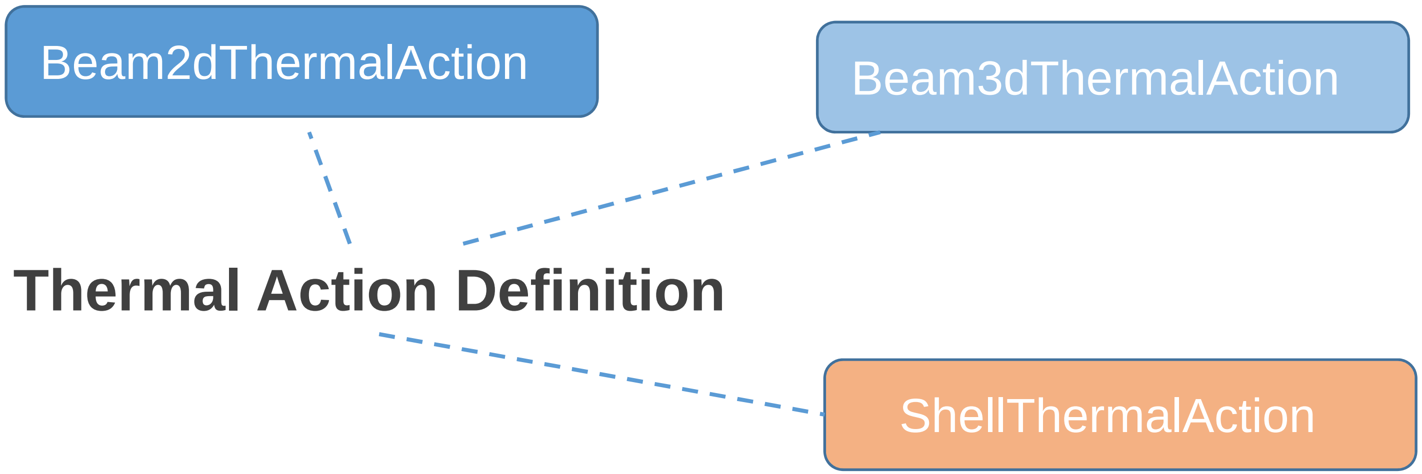
example-SteelBeam-uniform.tcl

```
set HalfD [expr $d/2];

pattern Plain 1 Linear {
  eleLoad -range 1 6 -type -beamThermal 1000 -$HalfD 1000 $HalfD;
}

constraints Plain;
..
integrator LoadControl 0.01;
analysis Static;
analyze 100;
```

BeamThermalAction

Analysis control for fire load

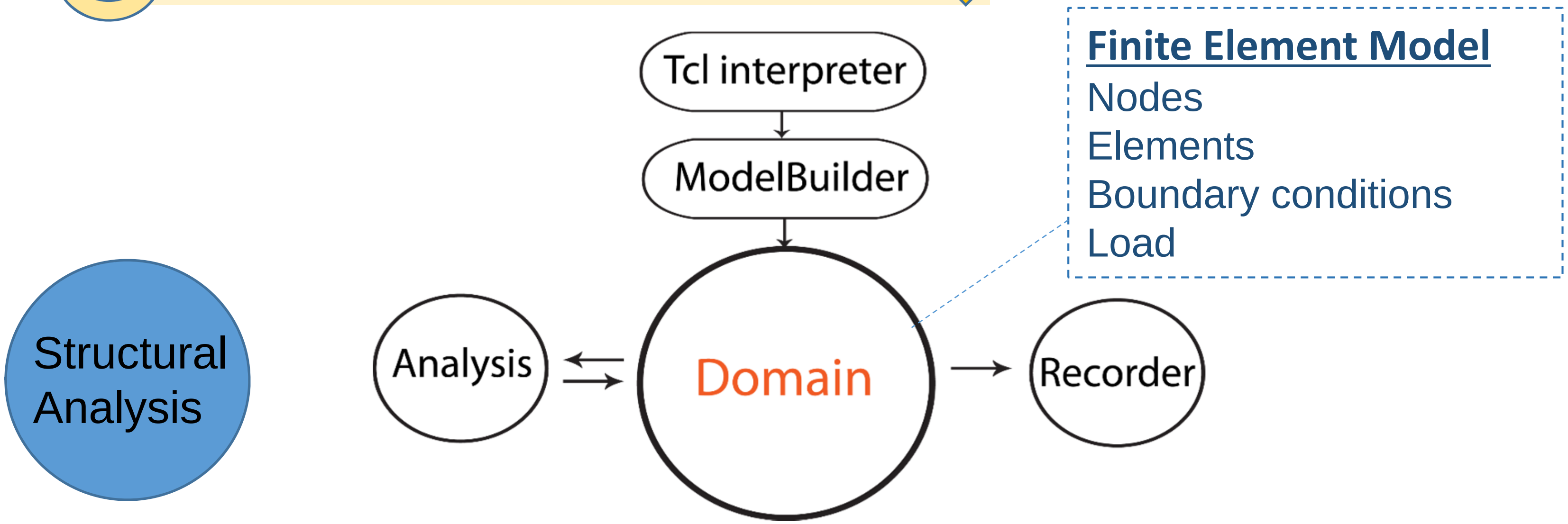

Beam
Section
Temperature
Zone
Fiber
(yLoc,0,area,matTag)
(T9,Y9)
(T8,Y8)
(T7,Y7)
(T6,Y6)
(T5,Y5)
(T4,Y4)
(T3,Y3)
(T2,Y2)
(T1,Y1)
(T5,Y5)
(T4,Y4)
(T3,Y3)
(T2,Y2)
(T1,Y1)
(T2,Y2)
Interpolation
(T1,Y1)

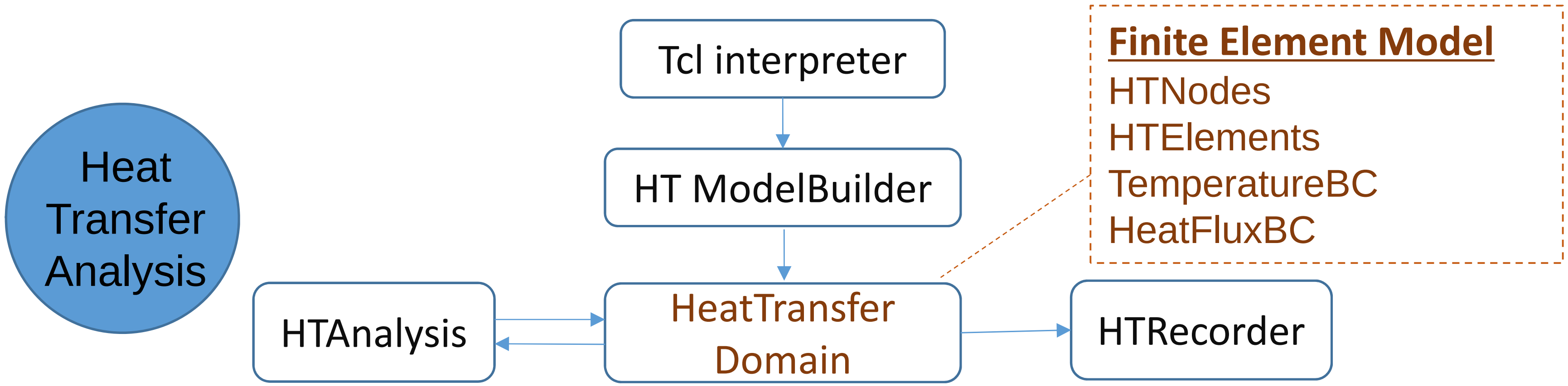

Beam2dThermalAction
Beam3dThermalAction
Thermal Action Definition
ShellThermalAction

To run a Heat Transfer analysis in OpenSees…

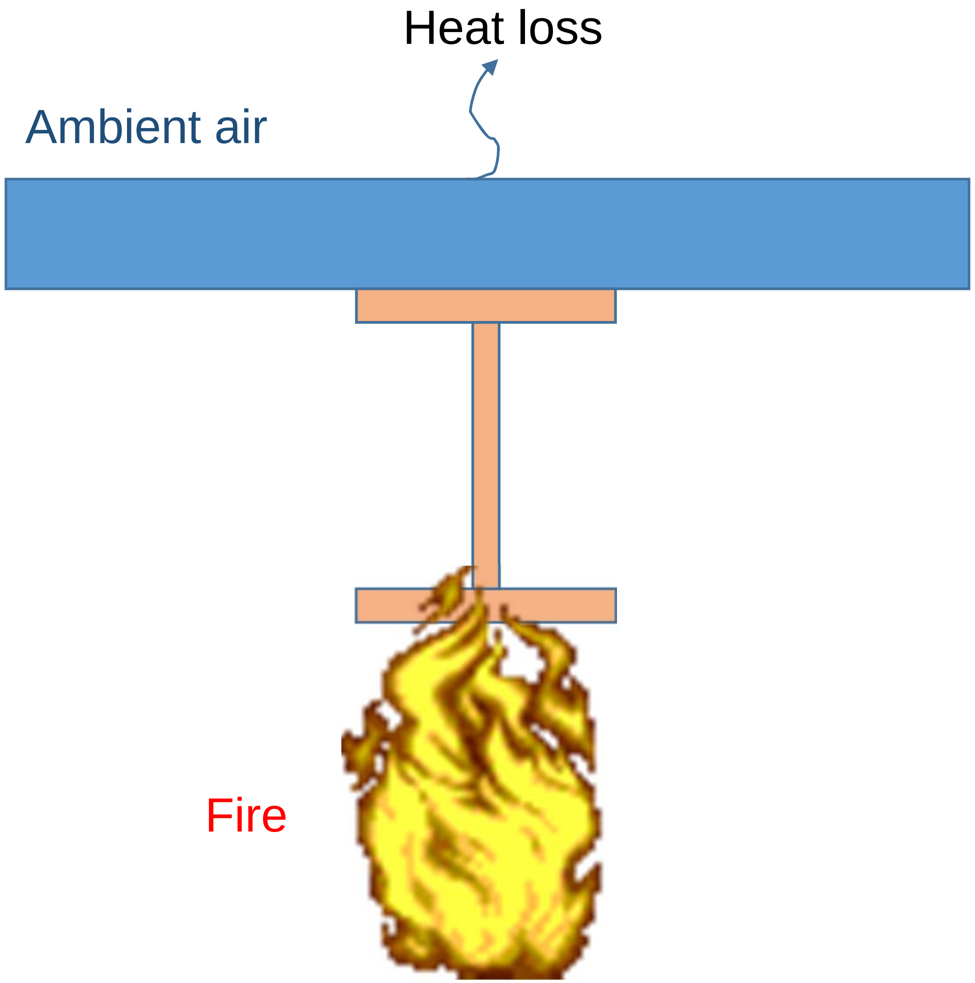
Using OpenSees

Let's recall the OpenSees framework

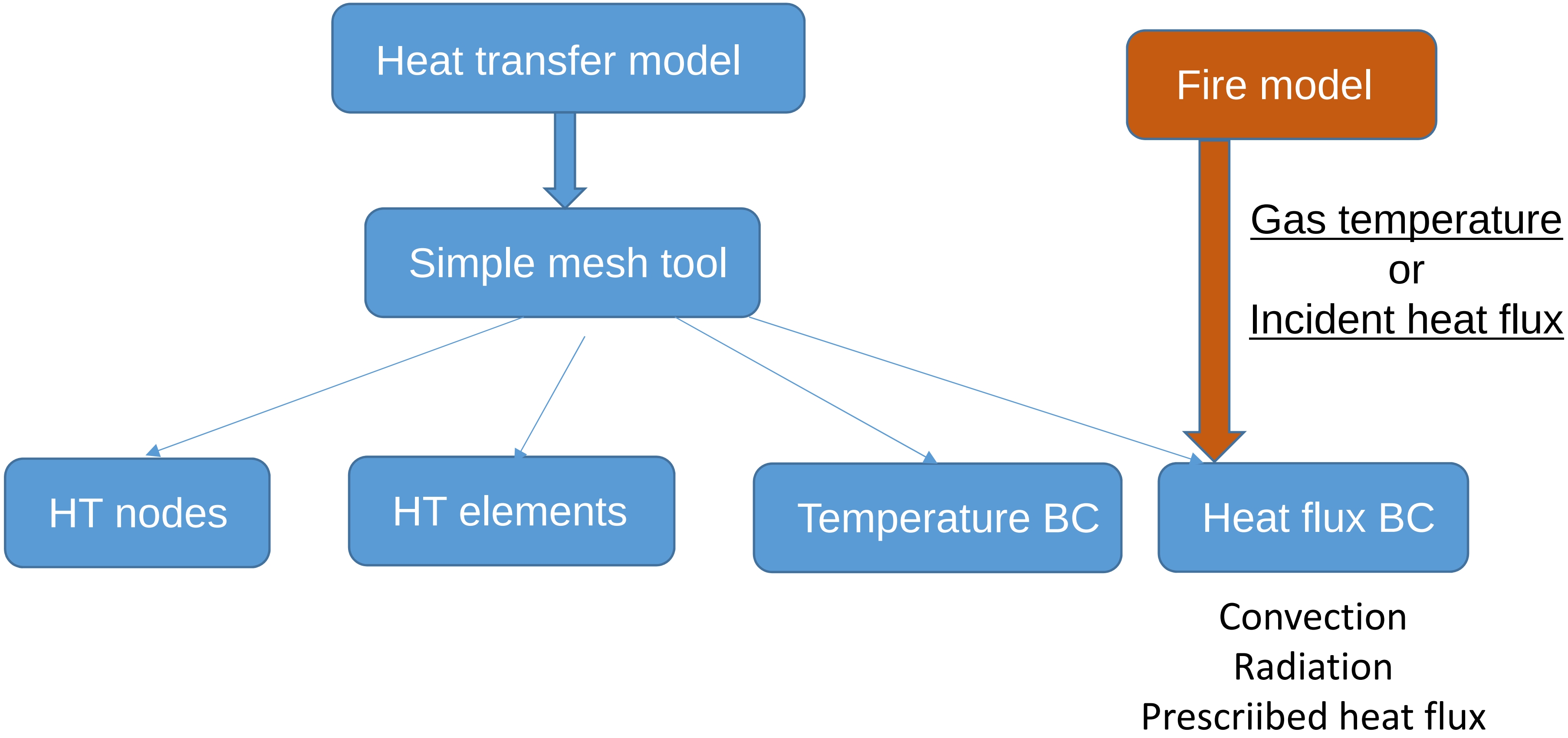




A typical heat transfer problem in structure

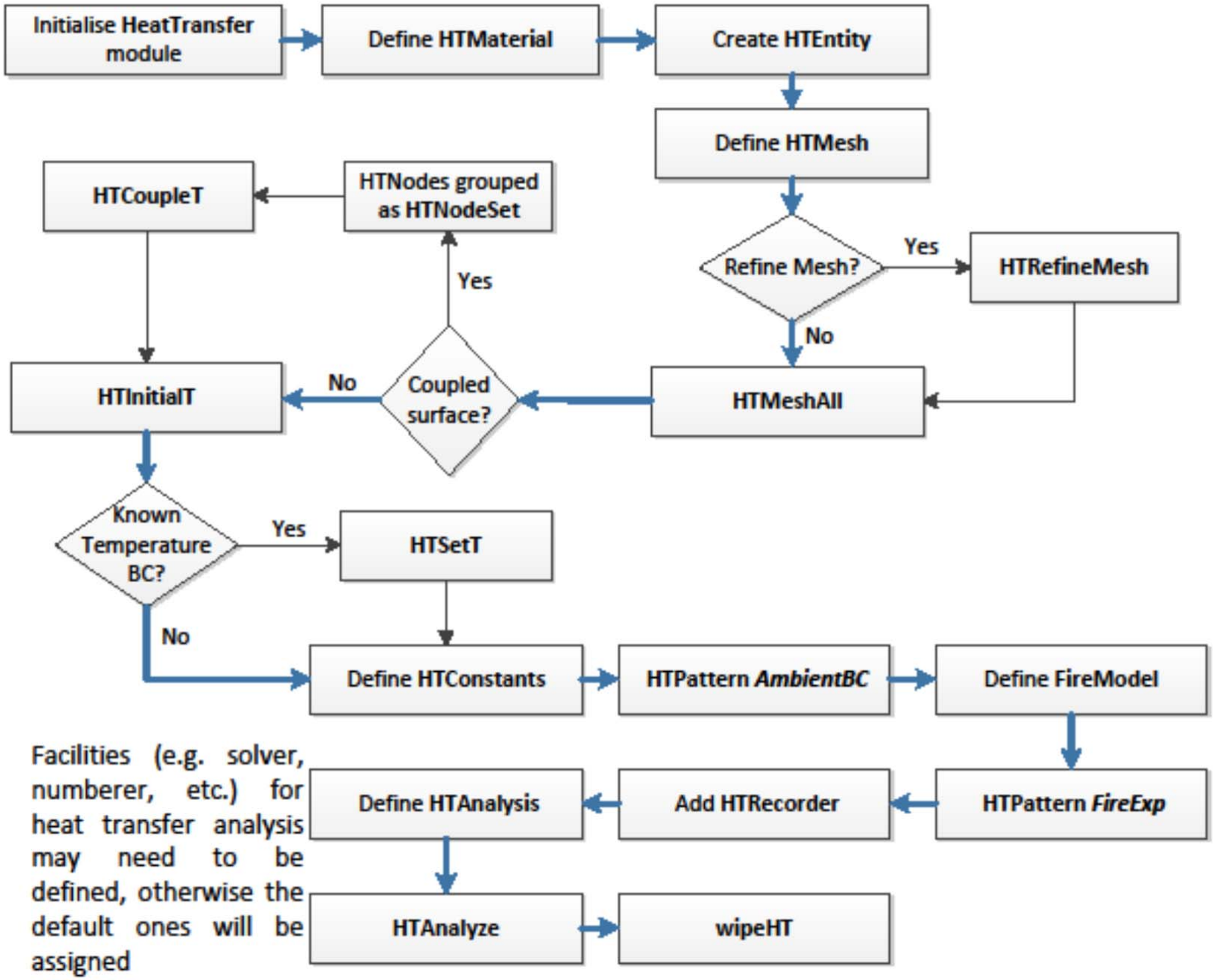


Development for Heat Transfer Analysis



Using OpenSees

Heat transfer analysis flowchart



Heat transfer analyses

Getting Started!

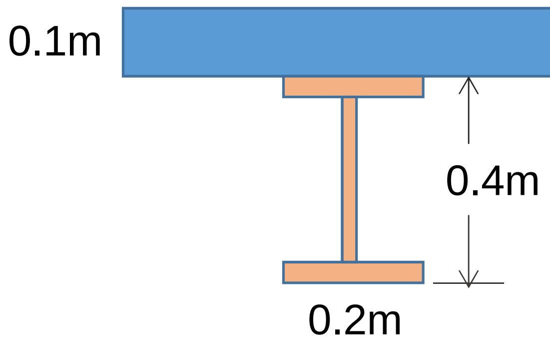
Composite section

Files in the folder

HT_demo.tcl

- Download from website
- Place the Tcl script with OpenSees

- A steel I-section connected to a concrete slab section
- Heat loss at the slab top
- Fire beneath the composite section



HT_demo.tcl

```
wipe;
HeatTransfer 2D;

HTMaterial CarbonSteelEC3 1;
HTMaterial ConcreteEC2 2 0.0;
```

Clear the domain

2D section analysis

Heat transfer material

HTMaterial $materialType $materialTag <$par_1...$par_n>

```
HTEntity  Isection   1    0.0      0.2     0.2 0.40 0.02 0.02;
HTEntity  Block   2    0.0      0.45     0.6    0.1;
```

HTEntity defintion

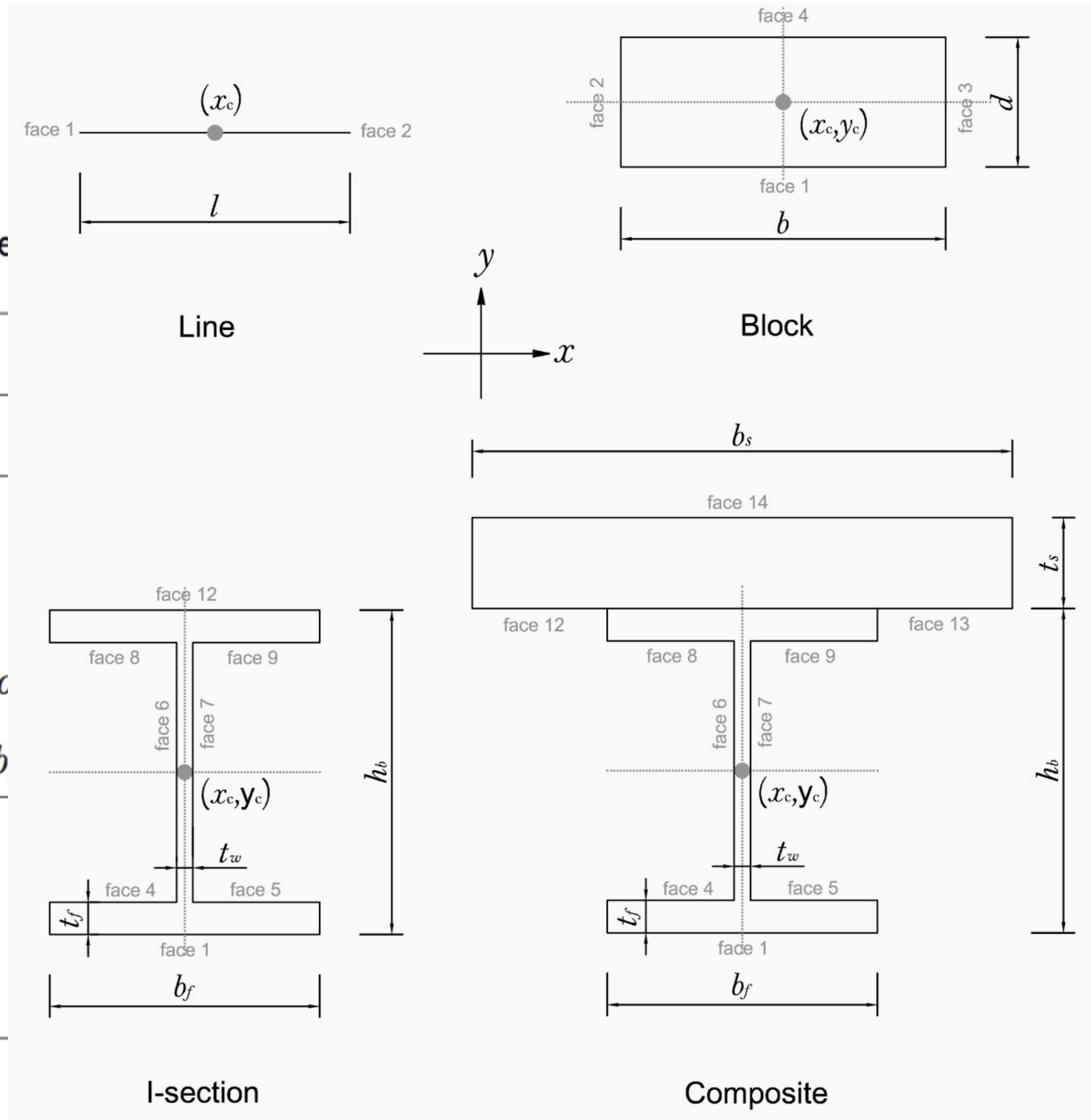
HTEntity $EntityType $EntityTag $centre_x $centre_y $dim_1 <$dim_2..$dim_n>

Using OpenSees

HTEntity Definition

Table A.1: Commands for creating a heat transfer e

Type	HTEntity	Centroid	Dimension
1D	Line	x_c	l
2D	Block	x_c, y_c	b, d
	Isection	x_c, y_c	b_f, h_b, t_w, t_f
	IsectionPro	x_c, y_c	b_f, h_b, t_w, t_f,cc
	Composite	x_c, y_c	b_f, h_b, t_w, t_f, b
3D	Brick	x_c, y_c	b,d ,
	Isection3D	x_c, y_c, z_c	b_f, h_b, t_w, t_f
	Composite3D	x_c, y_c, z_c	b_f, h_b, t_w, t_f



Using OpenSees

HT_demo.tcl

```
HTMesh 1 1 1 -phaseChange 0 -MeshCtrls 0.01 0.005 0.005 0.014
HTMesh 2 2 1 -phaseChange 1 -MeshCtrls 0.02 0.02
```

Mesh definition

```
HTRefineMesh -Entity 2 -SeedTag 1 4 -space 0.02 10 0.01 9 0.005 4 0.01 9 0.02 10;
```

Refine mesh to match the interface nodes

```
HTMeshAll;
```

Do the mesh now

```
SetInitialT 293.15;
```

Initialise temperature

```
HTNodeSet 1 -Entity 1 -face 12;
HTNodeSet 2 -Entity 2 -face 1 -locx -0.1 0.1;
```

HTNodeSet

```
HTCoupleT -NodeSet 1 2;
```

Coupling nodal temperature

HT_demo.tcl

```
HTConstants 1 4.0 293.15 0.7 5.67e-8 0.7 ;
HTConstants 2 25.0 293.15 0.7 5.67e-8 0.7;

HTPattern AmbientBC 1 {
        HeatFluxBC -HTEntity 2 -faceTag 4 -type ConvecAndRad -HTConstants 1;
}

HTRecorder -file temp0.out -NodeSet 1;
HTRecorder -file temp1.out -NodeSet 2;

HTAnalysis HeatTransfer
HTAnalyze 20 30;

wipeHT;
```

Heat transfer coefficients

Heat flux BC for the unexposed surface

Heat transfer recorder

Heat transfer analysis control

HT_demo.tcl

```
FireModel standard 1;

HTNodeSet 3 -Entity 2 -Locx -0.3 -0.1;
HTEleSet 1 -Entity 2 -NodeSet 3 -face 1;
HTNodeSet 4 -Entity 2 -Locx 0.1 0.3;
HTEleSet 2 -Entity 2 -NodeSet 4 -face 1;

HTPattern fire 2 model 1 {
        HeatFluxBC -HTEntity 1 -face 1 4 5 6 7 8 9 -type ConvecAndRad -HTConstants 2;
        HeatFluxBC -HTEleSet 1 -face 1 -type ConvecAndRad -HTConstants 2;
        HeatFluxBC -HTEleSet 2 -face 1 -type ConvecAndRad -HTConstants 2;
}
```

Heat transfer coefficients

Nodal selection
Element selection

Heat flux BC for exposed surfaces

HT_demo.tcl

```
HTRecorder -file temp0.out -NodeSet 1;
HTRecorder -file temp1.out -NodeSet 2;

HTAnalysis HeatTransfer
HTAnalyze 20 30;

wipeHT;
```

Heat transfer recorder

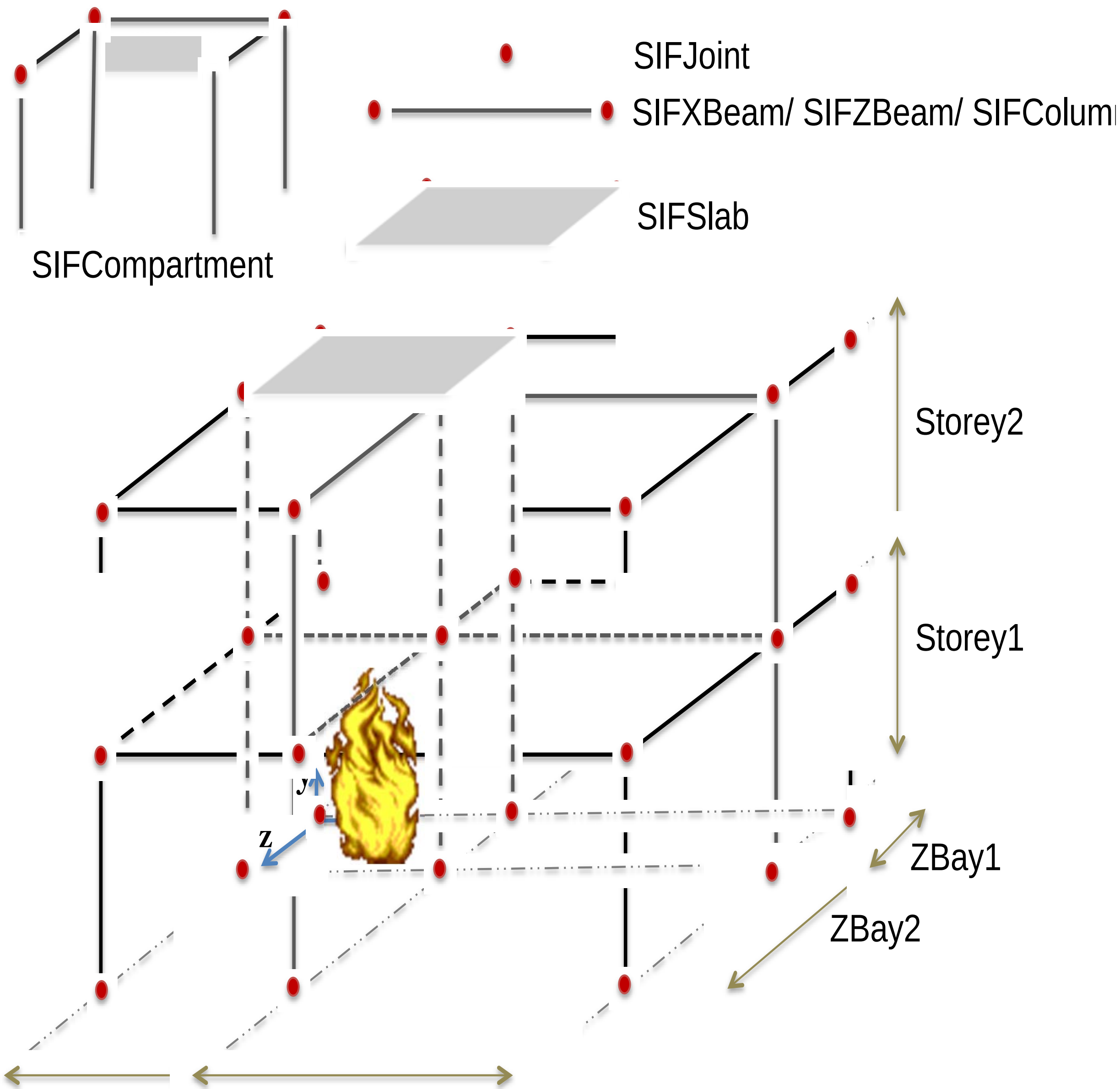
Heat transfer analysis control (20 steps, 30s each step)

To run a SIFBuilder aided analysis in OpenSees…

What is SIFBuilder ?

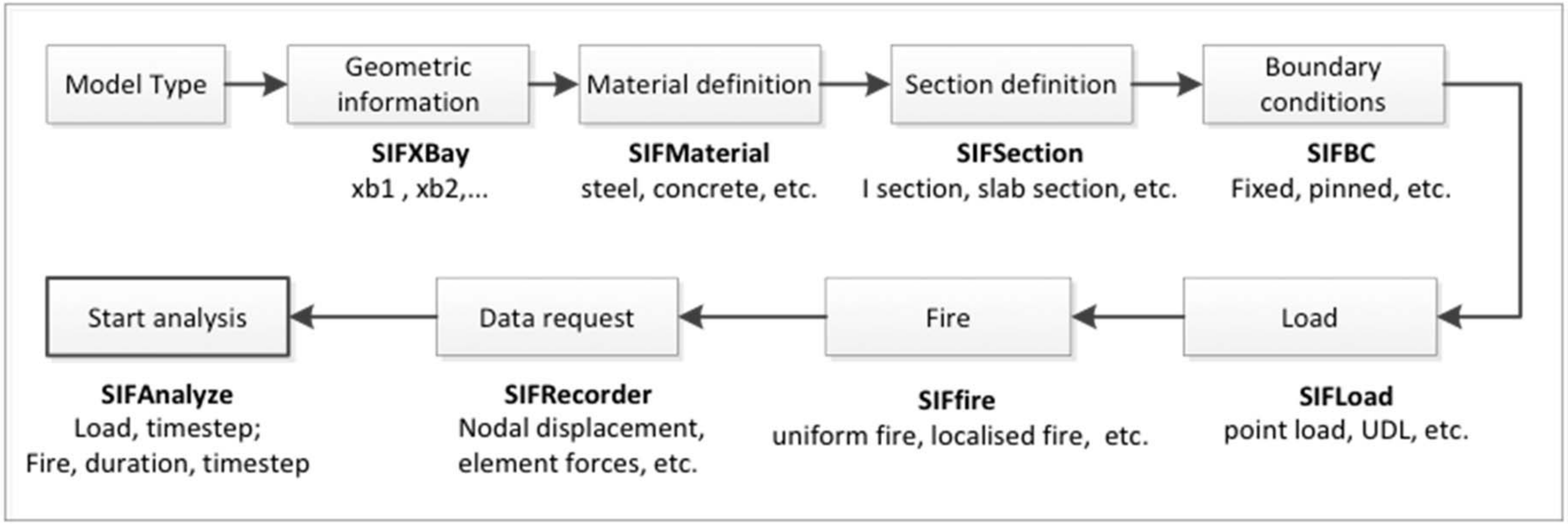
SIFBuilder is an **unified** tool for performing **automated**

structural fire analysis for **large** structures under **realistic** fire



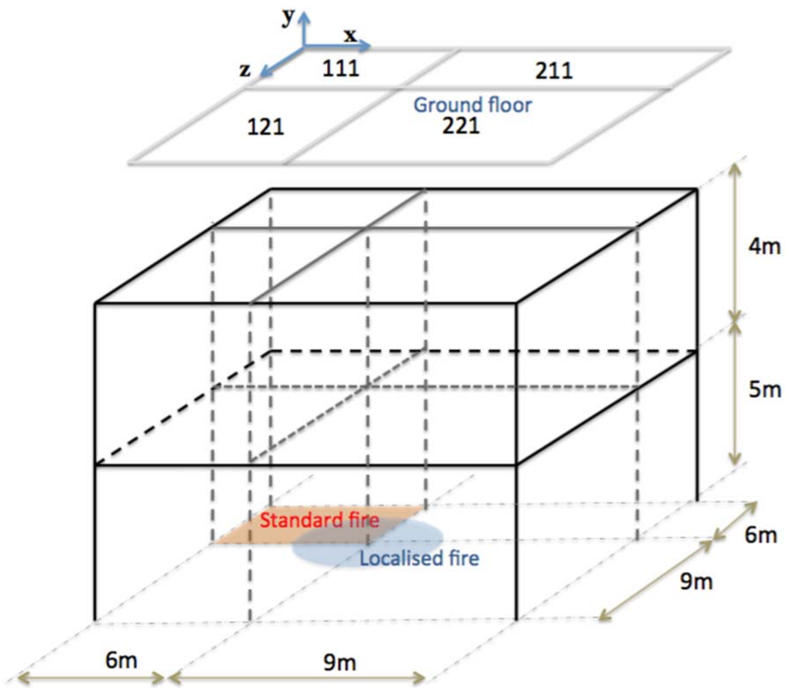
SIFJoint
SIFXBeam/ SIFZBeam/ SIFColum
SIFSlab
SIFCompartment
Storey2
Storey1
ZBay1
ZBay2
y
z

SIFBuilder workflow



2x2x2 Frame with slab

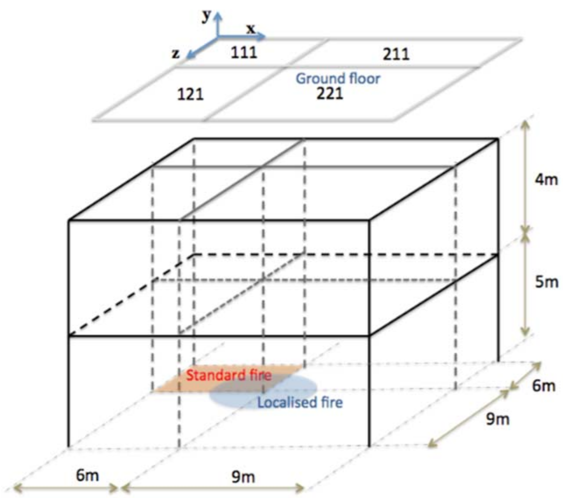
A large model of **2x2x2 frame with slab**, under the **localised fire / compartment fire**.



An 2x2x2 frame example from Edinburgh wiki

2x2x2 Frame with slab

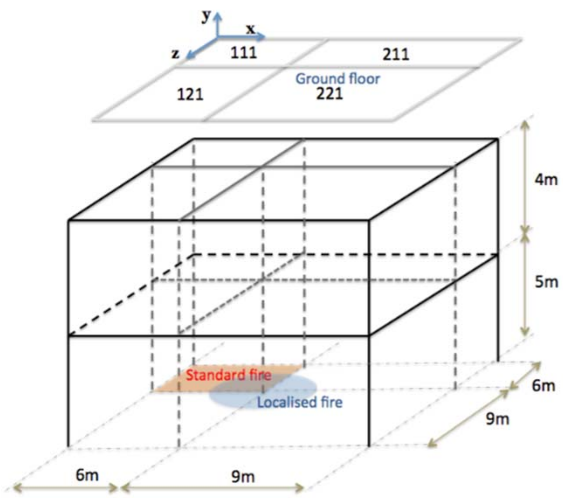
Geometry of the structure (bay lengths in each direction and storey heights in a Cartesian coordinate system);




```
source DisplayPlane.tcl;        # procedure for displaying a plane in model
source DisplayModel2D.tcl;      # procedure for displaying 2D perspective of model
source DisplayModel3D.tcl

file mkdir HTData;      #define the directory for storing data
SIFBuilder;  #initialise SIFBuilder, (SIFBuilder frame) is accepted for defining frame only wi

#[BUILDING INFO]
SIFXBay 6 9 ;   #XBAY SPAN|<----6m------->|<----------9m------->| along global x direction
SIFZBay 6 9;    #ZBAY SPAN|<----6m------->|<----------9m------->| along global Z direction
SIFStorey 5 4; #Storey Height|<----5m---->|<---4m--->| along global y direction

```

2x2x2 Frame with slab

Material type and **cross section** type for the structural members;

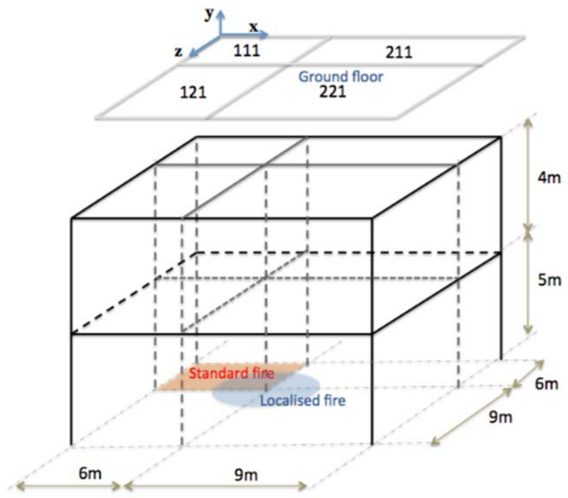



```
#[DEFINE MATERIAL AND SECTION]
AddMaterial steel 1 -type EC3 3e8  2e11;   #E0 : 3e8 , fy: 2e11, EN-1993-1-2 Steel Mat
AddMaterial concrete 2 -type EC2 0 30;  #moisture ratio:0 , fc :30, EN-1992-1-2 Concre

AddSection ISection 1 1 0.203 0.102 0.0054 0.009;  # $d $bf $tw $tf  UB203x102x23
AddSection ISection 2 1 0.203 0.203 0.007 0.011;  # $d $bf $tw $tf   UC203*203*46
AddSection  SlabSection 3 2 0.1;  #

#[ASSIGN SECTION]
AssignSection beams 1;
AssignSection columns 2;
AssignSection slabs 3;
```

2x2x2 Frame with slab

Boundary conditions for the structural model,

and define the **structural loading & Fire**.

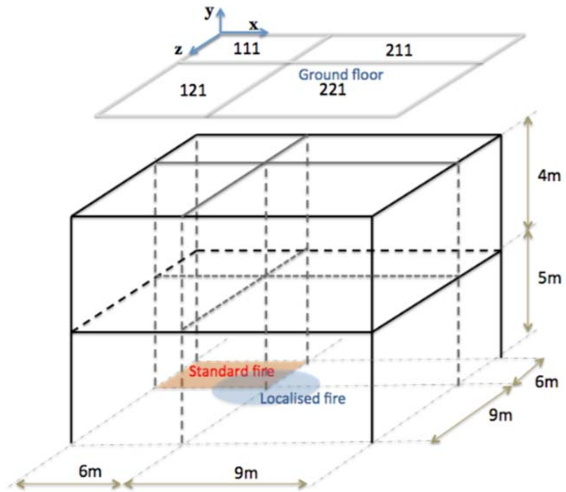



```
#[DEFINE BC AND LOAD]
SetBC fixedJoint -locy 0;     #set boundary condition
AddLoad -member allslabs -load 0 -1000 0;
AddFire -compartment 111 -type standard;

```

2x2x2 Frame with slab

Mesh control for the thermo-mechanical analysis,

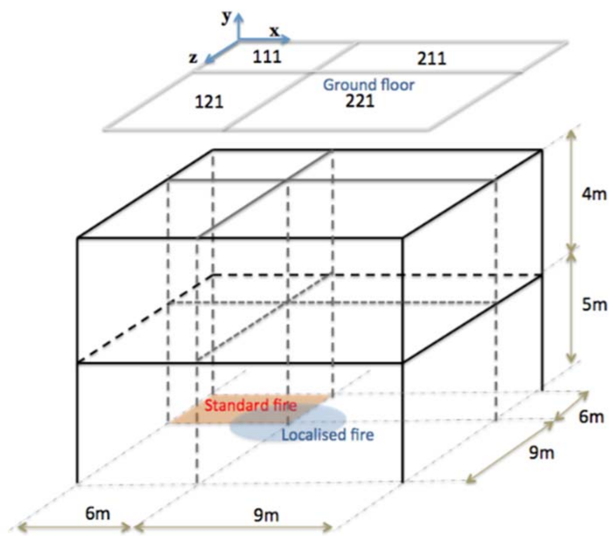
and define the **display**.




```
#[BUILD MODEL]
BuildModel -MeshCtrl 6 6 6;         #Number of Eles meshed for each member (along gl

#[Define DISPLAY]
    set  xPixels 800;   # height of graphical window in pixels
    set  yPixels 800;   # height of graphical window in pixels
    set  xLoc1 100;     # horizontal location of graphical window (0=upper left-mos
    set  yLoc1 60;      # vertical location of graphical window (0=upper left-most 
    set ViewScale 1;    # scaling factor for viewing deformed shape, it depends on 
DisplayModel3D DeformedShape $ViewScale $xLoc1 $yLoc1 $xPixels $yPixels 0
```

2x2x2 Frame with slab

Define recorder and **analysis**




```
#[Define SIFRECORDER]
SIFRecorder Joint -file Joint111.out -joint 111 disp;                    #Def
SIFRecorder Member -file XBeam111.out -xBeam 111 Mideflect;              #Rec
SIFRecorder Member -file Slab111.out -slab 111 Mideflect;                #Rec

#[RUN ANALYSES]
SIFAnalyze Load -dt 0.2 Fire -dt 30 -duration 1800 -output HTData;        #F
print domain.out
```

SIFBuilder exercises

- Change bay dimensions, material parameters, section types, etc
- Change standard fire durations
- Add standard fire to several compartments
- Apply localized fire at the centre column