Arbitrary Shaped Room Mode Simulator (3D)

A Very Quick Review of the Theory

The room modes can be found by solving the *Helmholtz equation*, where $\hat{p}$ is the pressure amplitude, ∇^2 is the *Laplacian*, $k = \frac{2\pi f}{c}$ is the acoustic wavenumber, f is frequency in Hz, and c is speed of sound in m/s:

$$\nabla^2 \hat{p} + k^2 \hat{p} = 0$$

In the Cartesian coordinate system:

$$\nabla^2 \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

Rearrange the equation into the more recognizable form of the classic eigenvalue problem:

$$(-\nabla^2)\hat{p} = k^2 \hat{p}$$

After discretization, $(-\nabla^2)$ becomes the negative of the Laplacian matrix. k^2 is the eigenvalue and $\hat{p}$ is the corresponding eigenvector (discretized eigenfunction) of the Laplacian matrix.

This simulator assumes perfect reflection of the room walls, which is reasonable since we mostly concern with the bass frequencies. This simply means that the boundary conditions are naturally applied by the FEM formulations and we don't need to specify any boundary conditions.

Start

Clear all variable definitions.

Obtain the speed of sound at sea level using standard atmospheric data.

In [1]:
```
(* Clear all variable definitions *)
ClearAll["Global`*"];

(* Speed of sound at sea level standard atmosphere, in SI unit *)
c = QuantityMagnitude[StandardAtmosphereData[Quantity[0, "Meter"], "SoundSpeed"]]
```

Out[1]: 340.29

Construct our example room using basic 3D solid primatives

The interior volume of our example model — a house with a loft — is build using the 3D solid primatives cuboid, prism and hexahedron.

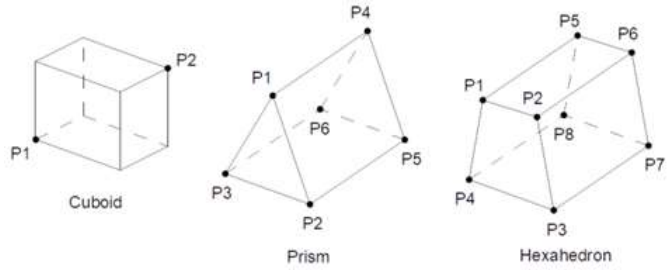
To define a cuboid: `Cuboid[p1, p2]` where `p1` is `{x1, y1, z1}` and `p2` is `{x2, y2, z2}`.

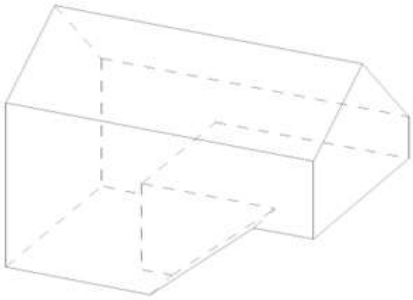
To define a prism: `Prism[p1, p2, p3, p4, p5, p6]` where `p1` is `{x1, y1, z1}`, etc.

To define a hexahedron: `Hexahedron[p1, p2, p3, p4, p5, p6, p7, p8]` where `p1` is `{x1, y1, z1}`, etc.

In [5]:
```
Import["3D_primatives.png"]
Import["Loft.png"]
```

Out[5]:



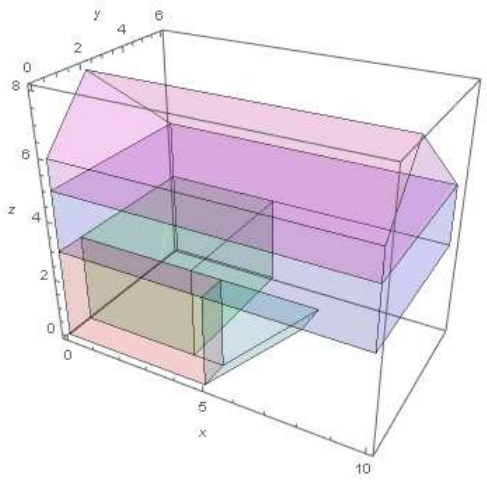


In [7]:
```
cb1 = Cuboid[{0, 0, 0}, {5, 1, 3}];
cb2 = Cuboid[{0, 1, 0}, {4, 6, 3}];
cb3 = Cuboid[{0, 0, 3}, {10, 6, 5}];
ps1 = Prism[{{5, 0, 0}, {8, 0, 3}, {5, 0, 3},
             {5, 1, 0}, {8, 1, 3}, {5, 1, 3}}];
hx1 = Hexahedron[{{0, 2, 8}, {0, 6, 5}, {0, 0, 5}, {0, 0, 6},
                  {10, 2, 8}, {10, 6, 5}, {10, 0, 5}, {10, 0, 6}}];
```

Display the solids

In [12]:
```
Graphics3D[{Opacity[0.1], {{Red, cb1}, {Green, cb2}, {Blue, cb3}, {Cyan, ps1}, {Magenta, hx1}}},
           ImageSize -> Medium, Axes -> True, AxesLabel -> {x, y, z}, ViewVector -> {15, -15, 15}]
```

Out[12]:



Merge the solids together and generate the mesh

Display the merged solid, the FEM mesh and report the number of elements in the mesh.

In [13]:
```
model = RegionUnion[cb1, cb2, cb3, ps1, hx1];

mesh = DiscretizeRegion[model, MeshQualityGoal -> "Maximal", AccuracyGoal -> 4,
                        MaxCellMeasure -> {"Volume" -> 0.01}, PerformanceGoal -> "Quality"];

Grid[{{Graphics3D[{Opacity[0.1], model}, ImageSize -> Medium, Axes -> True,
                  AxesLabel -> {x, y, z}, ViewVector -> {15, -15, 15}],
       Graphics3D[{Opacity[0.25], mesh}, ImageSize -> Medium, ViewVector -> {15, -15, 15}]}}]
MeshCellCount[mesh, 3]
```

Out[13]: