www.itascacg.com

ITASCA™

Consulting Group, Inc.

GEOMECHANICS, HYDROGEOLOGY, and MICROSEISMICS

CIVIL • ENVIRONMENTAL • MANUFACTURING • MINING • OIL & GAS • POWER GENERATION

Simulation of a Continuous Binary Powder Mixer using Tetrahedral Clumps

PFC3D 5.0 Example

Introduction

- In the pharmaceutical industry, the overall performance of a continuous, binary powder mixer is measured by:
  - how well components are mixed,
  - limitations of the recirculating regions,
  - how much shear is experienced by particles, and
  - how little product must be discarded at startup before a reliable steady-state mixture is achieved.
- *PFC*'s performance, ease of use and versatility, aided by its powerful scripting language that reaches deep into virtually every internal variable and model, enables process designers to sort out and understand every physical and geometrical factor that influences continuous mixing.
- Tetra clumps were used in an attempt to frustrate rotations. An alternative approach would be to use balls with a rolling resistance contact model, which is now available.

Modeling Procedure

Step 1 Define a modeling domain and initial conditions

Step 2 Define clump material properties

Step 3 Import DXF walls and assign a rotational spin to the rotor

Step 4 Create a template for the tetrahedral powder clumps

Step 5 Generate two groups of clumps using a *FISH* function

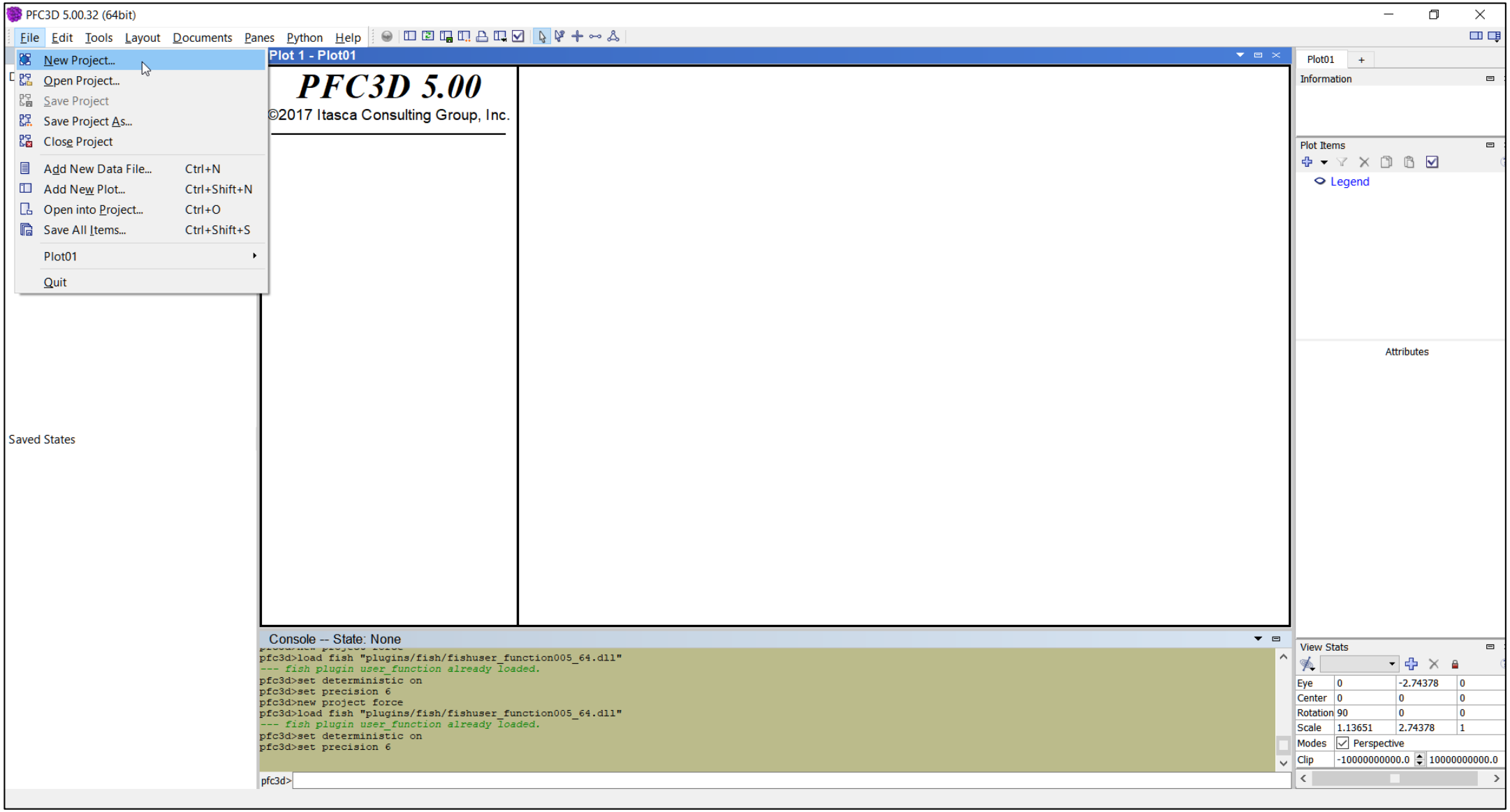
Step 6 Run the model for 100 seconds of simulated time

Example Limitations

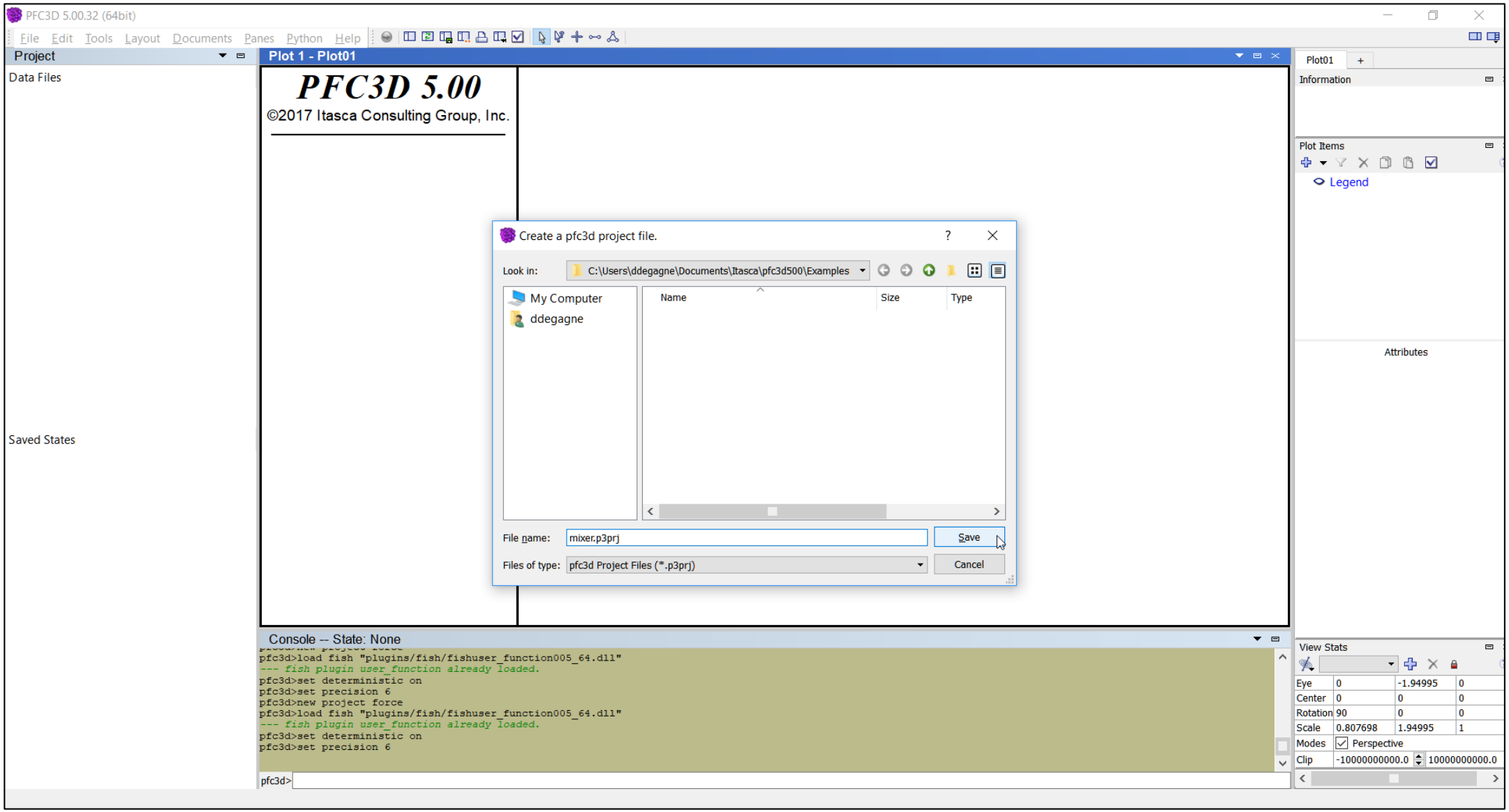
Some things to consider:

- This example is provided *as is* for demonstration purposes only.
- The stiffness are low for computational efficiency, but probably too low, for practical purposes (i.e., large overlaps with the walls).
- The feeding procedure could be improved.
- The tetrahedron approximation using clumps is very coarse; though perhaps it’s sufficient for the purpose of the simulation.

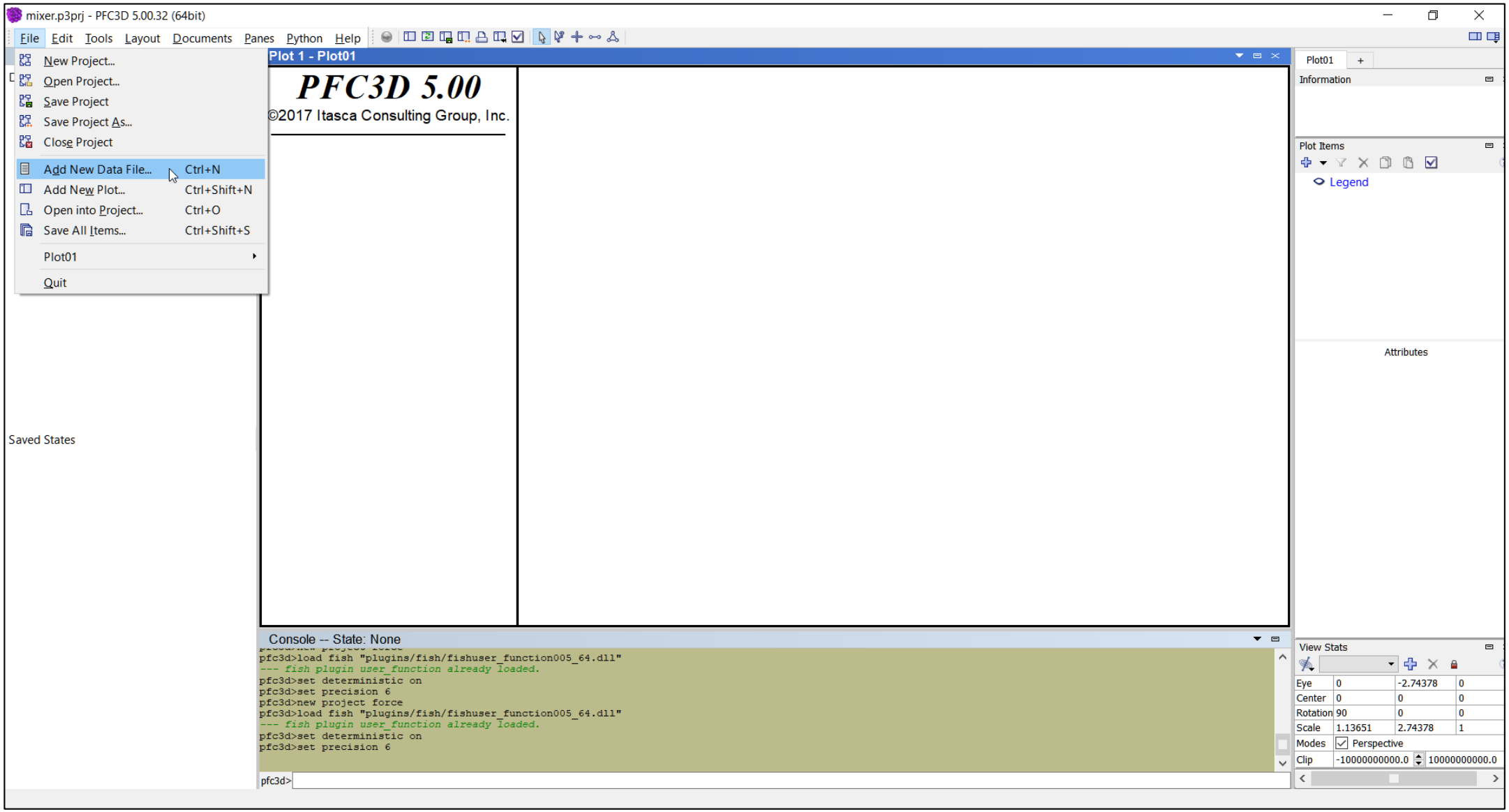
Step 1
Define a modeling domain and initial conditions



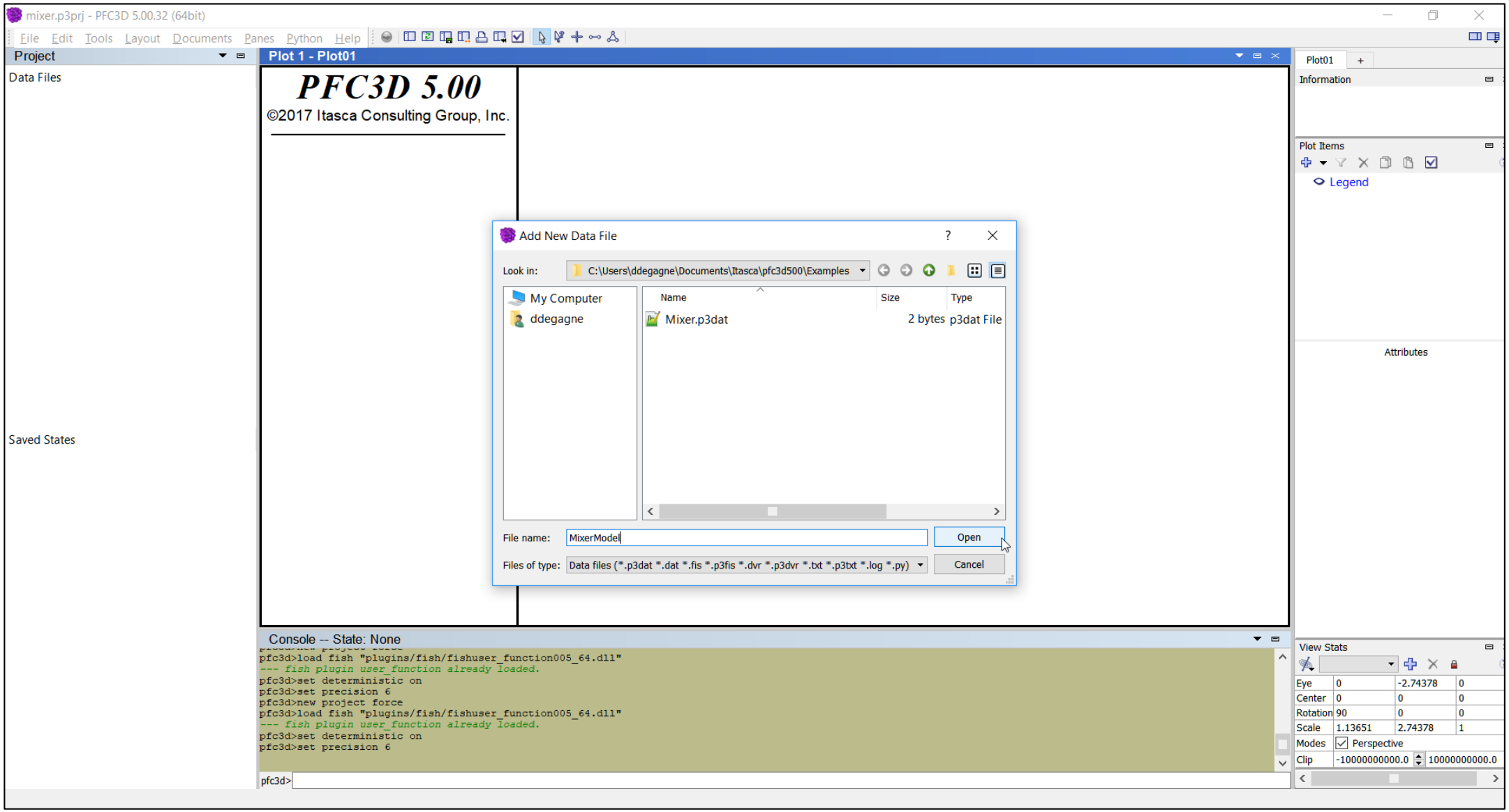
Create a new project called Mixer.



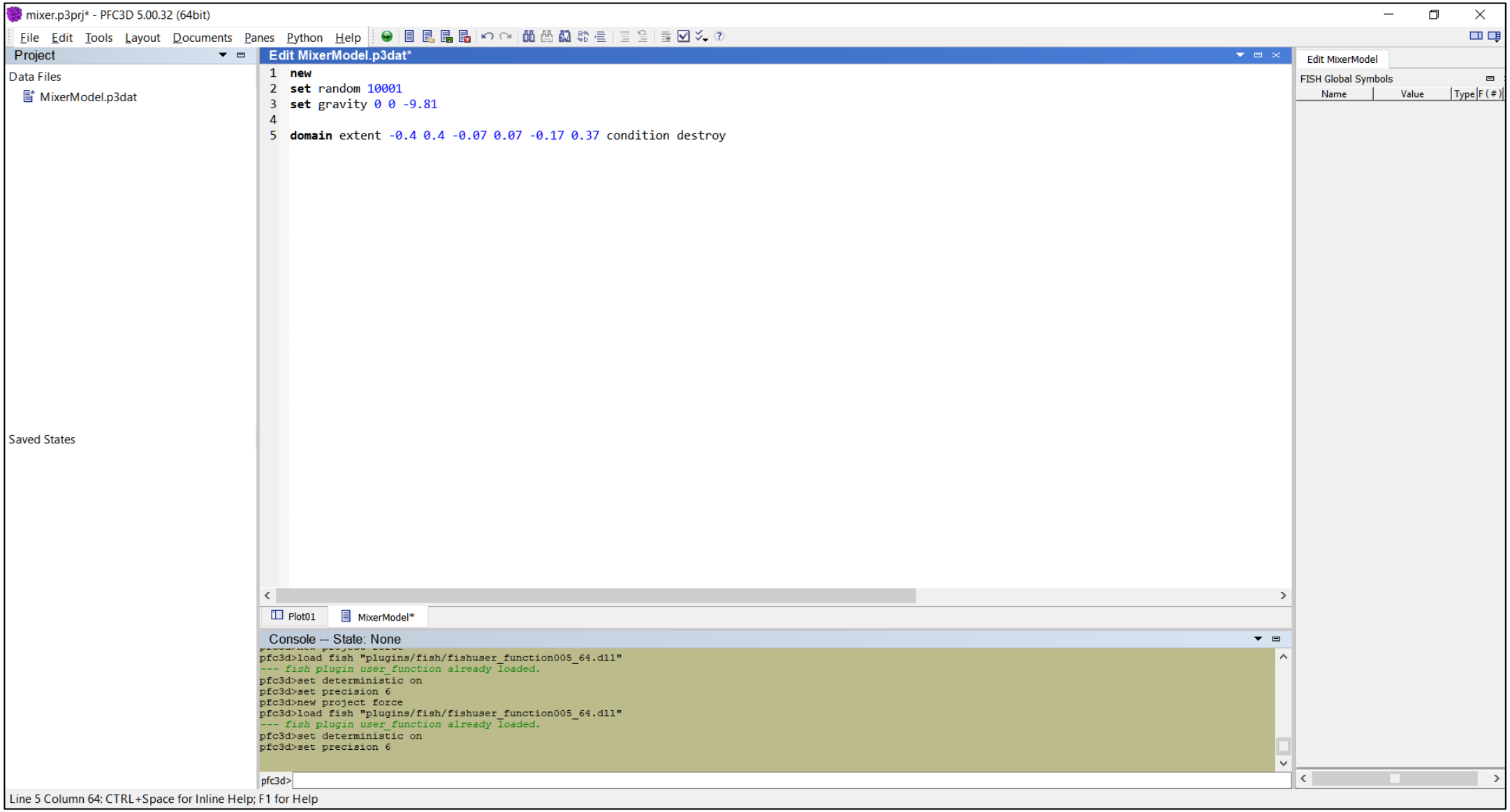
Create a new project called Mixer.



Add a new data file called MixerModel.

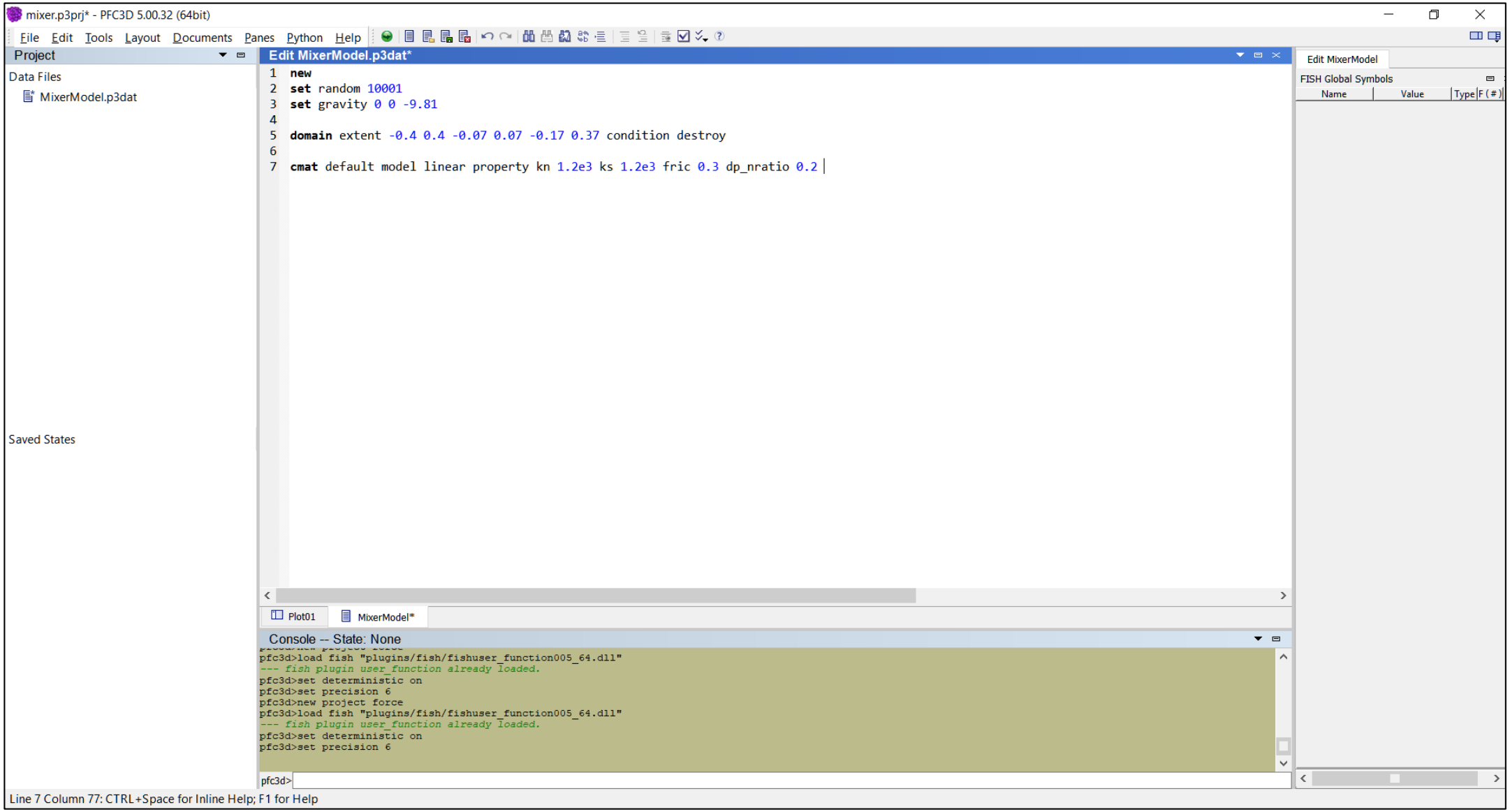


Add a new data file called MixerModel.



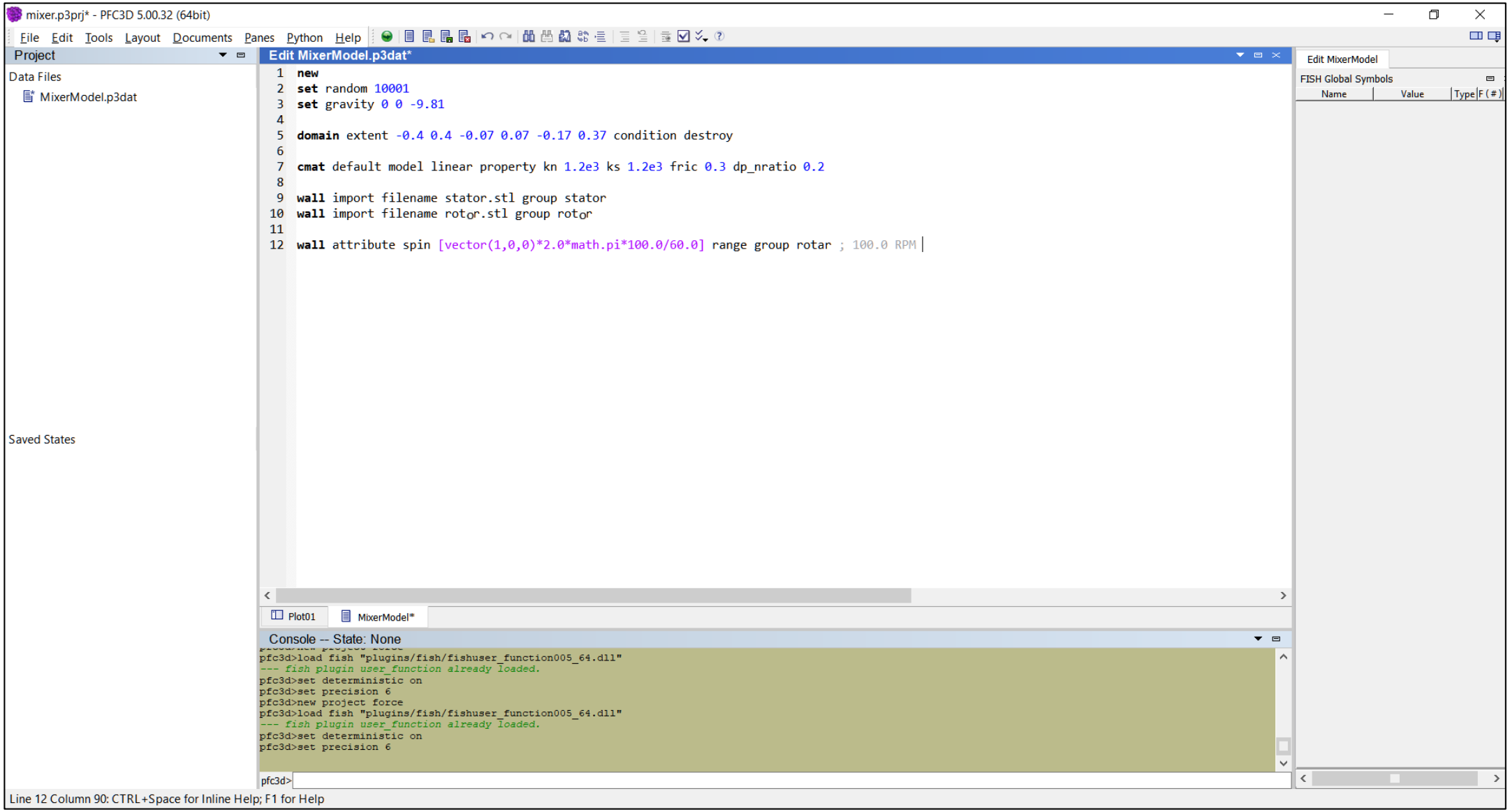
Declare a new model (clears any previous work), set a random variable seed, and activate gravity (x,y,z) to pull downwards. Define the extent of the model domain (like a bounding box) to range (in meters) from -0.4 to 0.4, -0.07 to 0.07, and -0.17 to 0.37 along the x, y, and z-axes, respectively. Set the domain boundary to destroy (i.e., delete) any particles that may pass outside the domain.

Step 2
Define clump material properties



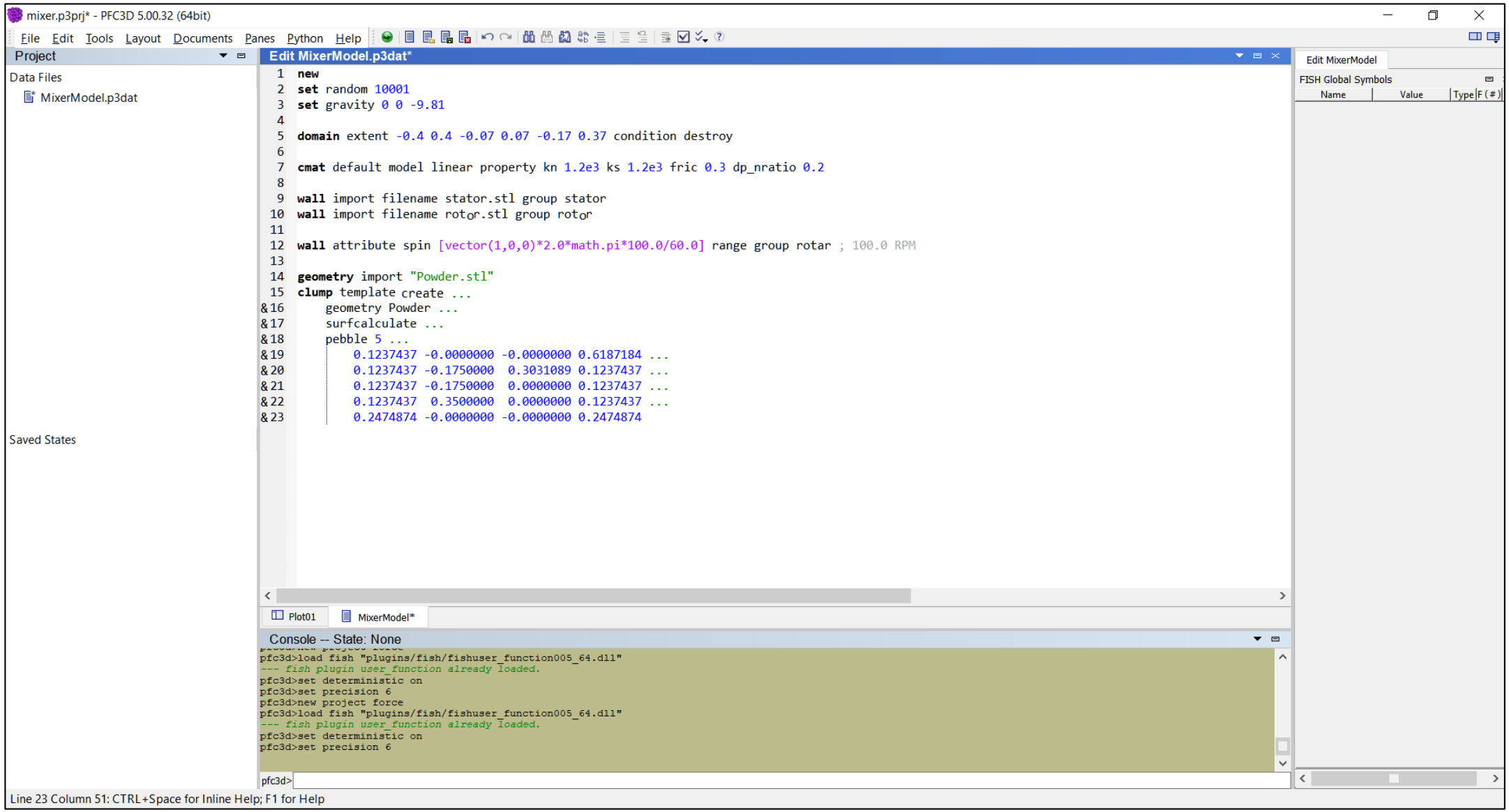
The CMAT (Contact Model Assignment Table) specifies all of the contact models between the model elements. The CMAT consists of an ordered set of (optional) slots along with a default slot for each contact type. In many situations where the constitutive behavior is homogeneous across the system, setting the default slots of the CMAT suffices. In this example, the linear contact model, with a normal (and shear) stiffness =1.2e3, a friction coefficient of 0.3, and a normal critical damping ratio of 0.2 (i.e., dashpot viscosity) is used.

Step 3
Import DXF walls and assign a rotational spin to the rotor



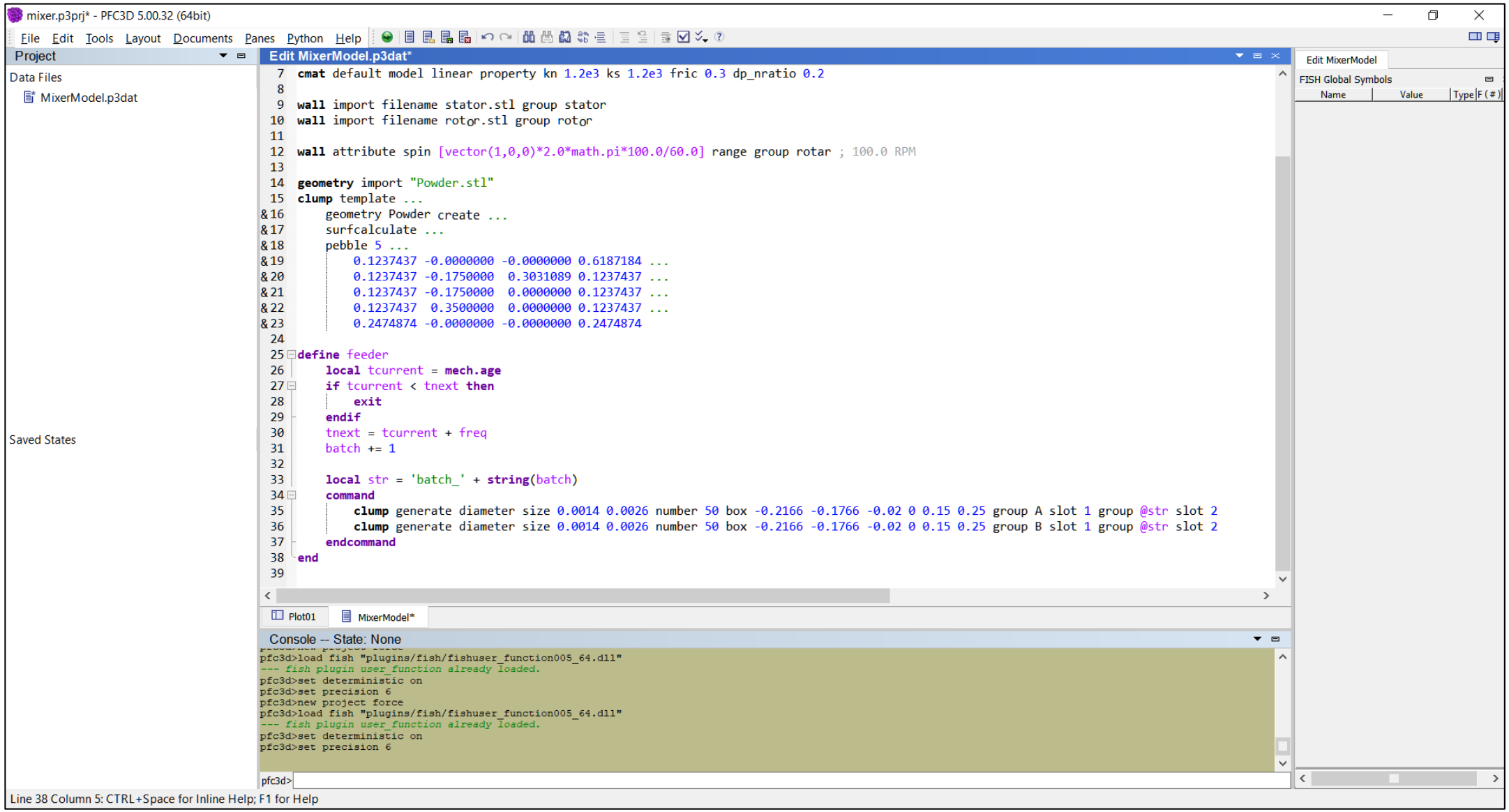
Walls can be imported directly from DXF or STL files providing the geometry is of good quality (facets don't interpenetrate or touch other facets and the surface is edge connected in 3D). One wall represents the stator while the other is the rotor, with each being assigned to a group of the same name so they can be distinguished from each other. Think of a group like it's a CAD layer. In order to spin the rotor around its axis (x=1,y=0,z=0) , a spin attribute is assigned. A *FISH* fragment within the command line is used to calculate the velocity vector to achieve a spin of 100 RPM (Revolutions Per Minute. The math.pi component is one of many built-in *FISH* intrinsics, in this case representing the numerical value of Pi. The attribute is assigned to the rotor group only.

Step 4
Create a template for the tetrahedral powder clumps



A reference geometry representing the tetrahedral power surface is imported. This is not a wall; but will be scaled and rotated as needed for visualization purposes only. A clump consists of two or more pebbles and a clump template is an easy way to define a clump baseline by defining the relative position of the its pebbles in a unit space. Multiple clump templates may be defined and then called upon when generating clumps. The geometry keyword links the template to the surface geometry description whereas the surfcalculate keyword causes the clump's inertial attributes to be calculated from the surface description (assuming uniform density) rather than the spherical pebbles.

Step 5
Generate two groups of clumps using a *FISH* function



The purpose of this *FISH* function is to generate two groups of clumps in the same space (as defined by a rectangular box region) above the hopper feeding the mixer. Clumps will be generated periodically in batches of 100 as defined by the *FISH* variable “freq”. To distinguish one set of clumps from the other they are assigned in equal number to different groups (“A” or “B’). In order to also view the clumps by the batch they were created, a second slot (i.e., group container) is specified for each group with a variable name defined by the string variable “str”. Note that a number of variables (like freq) are variables that are defined later in the data file. Refer to the next page for more details about the *FISH* function.

Local variables only exist inside the *FISH* function; **global** variables can be accessed anywhere in the model (*FISH* fragments in commands, other *FISH* functions, etc.).

Intrinsic *FISH* functions (e.g., **local**, **if**, **then**, **endif**, **string**, **mech.age**) are available to assist you in creating scripts. In this case, **mesh.age** returns the value of the accumulated mechanical model time.

A *FISH* function is declared by the command **define**, with a function name (e.g., feeder) and is terminated by the **end** command.

```
24
25 define feeder
26     local tcurrent = mech.age
27     if tcurrent < tnext then
28         exit
29     endif
30     tnext = tcurrent + freq
31     batch += 1
32
33     local str = 'batch_' + string(batch)
34     command
35         clump generate diameter size 0.0014 0.0026 number 50 box -0.2166 -0.1766 -0.02 0 0.15 0.25 group A slot 1 group @str slot 2
36         clump generate diameter size 0.0014 0.0026 number 50 box -0.2166 -0.1766 -0.02 0 0.15 0.25 group B slot 1 group @str slot 2
37     endcommand
38 end
39
```

In order to generate clumps at regular, periodic intervals (as defined by the user variable freq), this **if-endif** statement tests to see if sufficient model time has passed. If not true (the current time or tcurrent is less-than (<) the next time interval or tnext) then **exit** the *FISH* function; if true however, tnext will be incremented by the value of freq, the batch number will be increased by 1 (batch += 1 is a short form of batch = batch + 1), a new group name (for slot 2) will be created, and two sets of clumps will be generated. This function is only called when its name, feeder, is called. More on this later.

PFC commands may be embedded within *FISH* functions. These commands create the clumps. See the next page for more details.

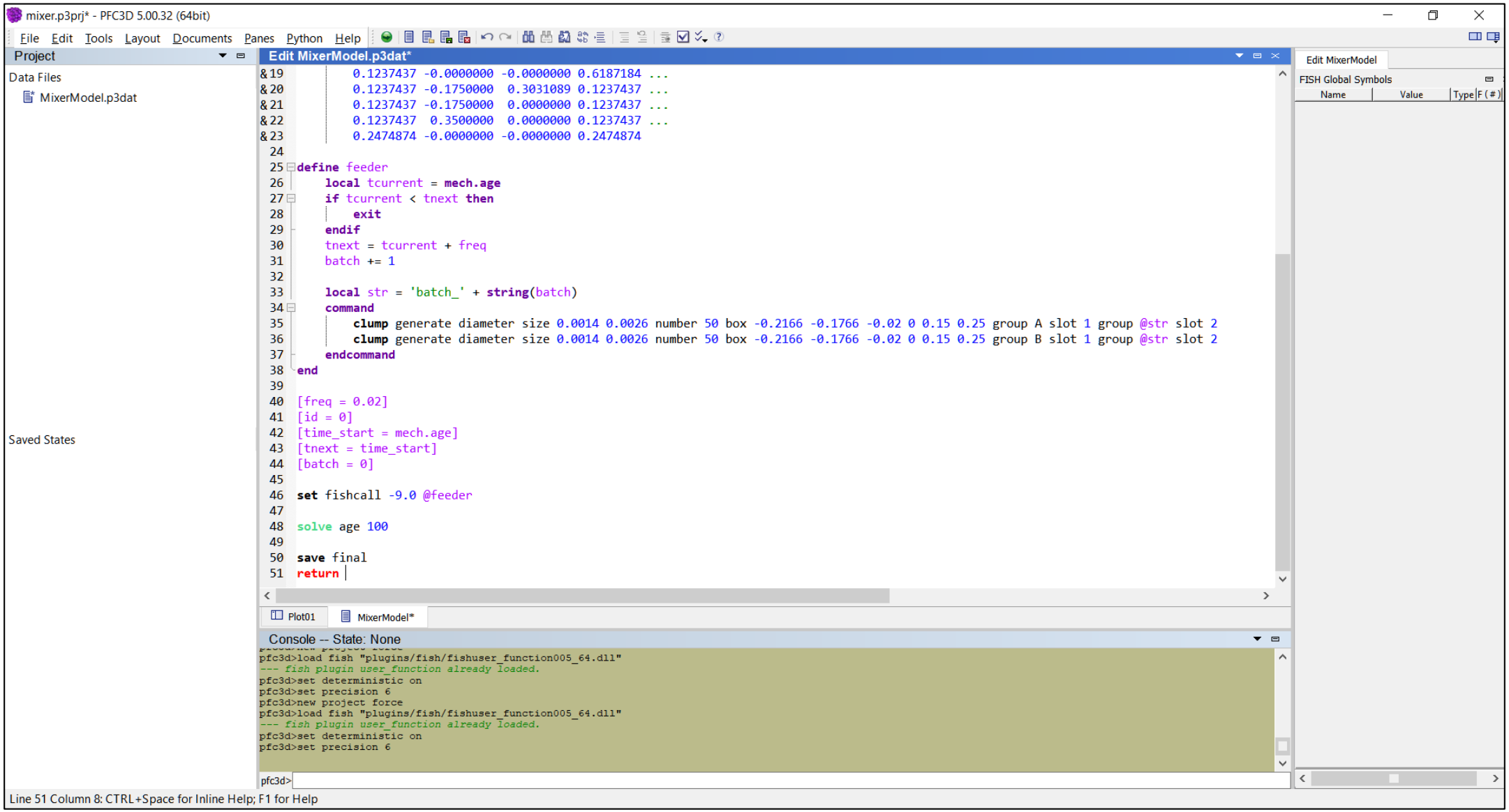
String (text) values are specified using single quotes.

Converts an integer value into a string (or text) value. String variables can be combined by adding them together. The value of str will change for each clump generation batch (**batch_1**, **batch_2**, **batch_3**, and so on).

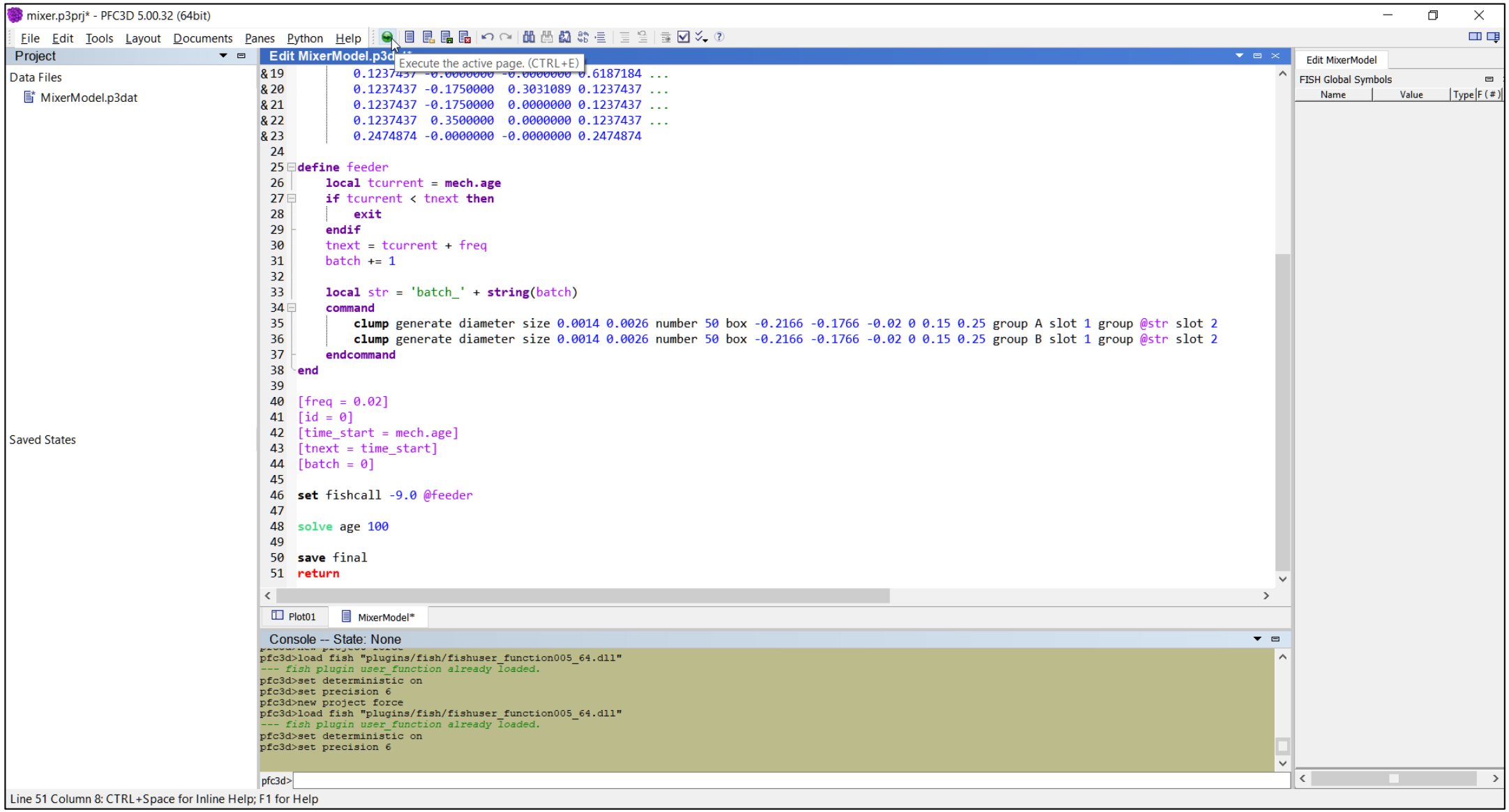
When the *safe conversion in command interpreter* is active (see Options under the Tools menu item), the @ symbol is required for *FISH* variables in a command. This special character unambiguously identifies these variables as *FISH* symbols. Using safe conversion is recommended.

Users may create their own *FISH* variables and functions (e.g., feeder, tcurrent, freq, etc.)

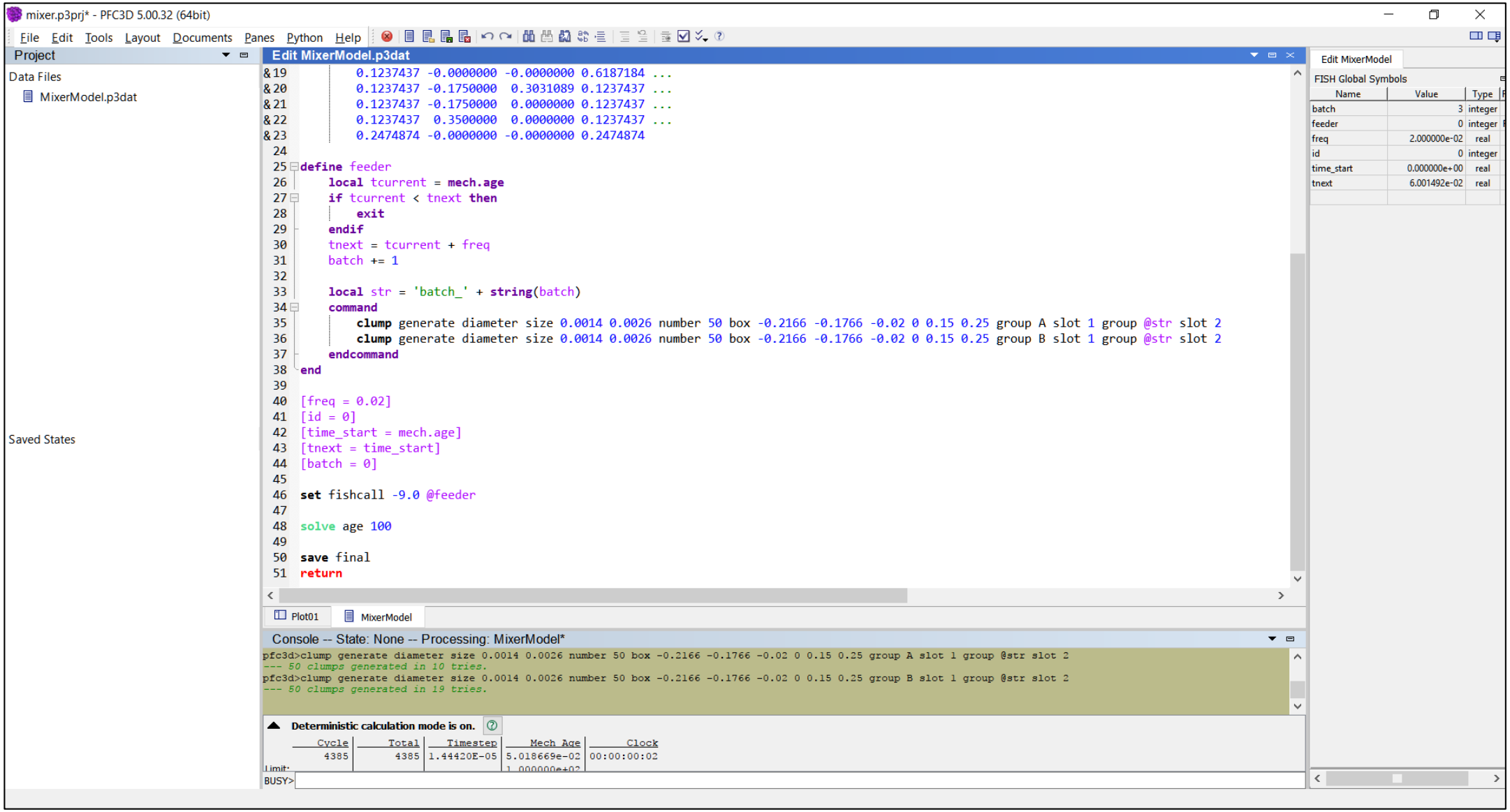
Step 6
Run the model for 100 seconds of simulated time



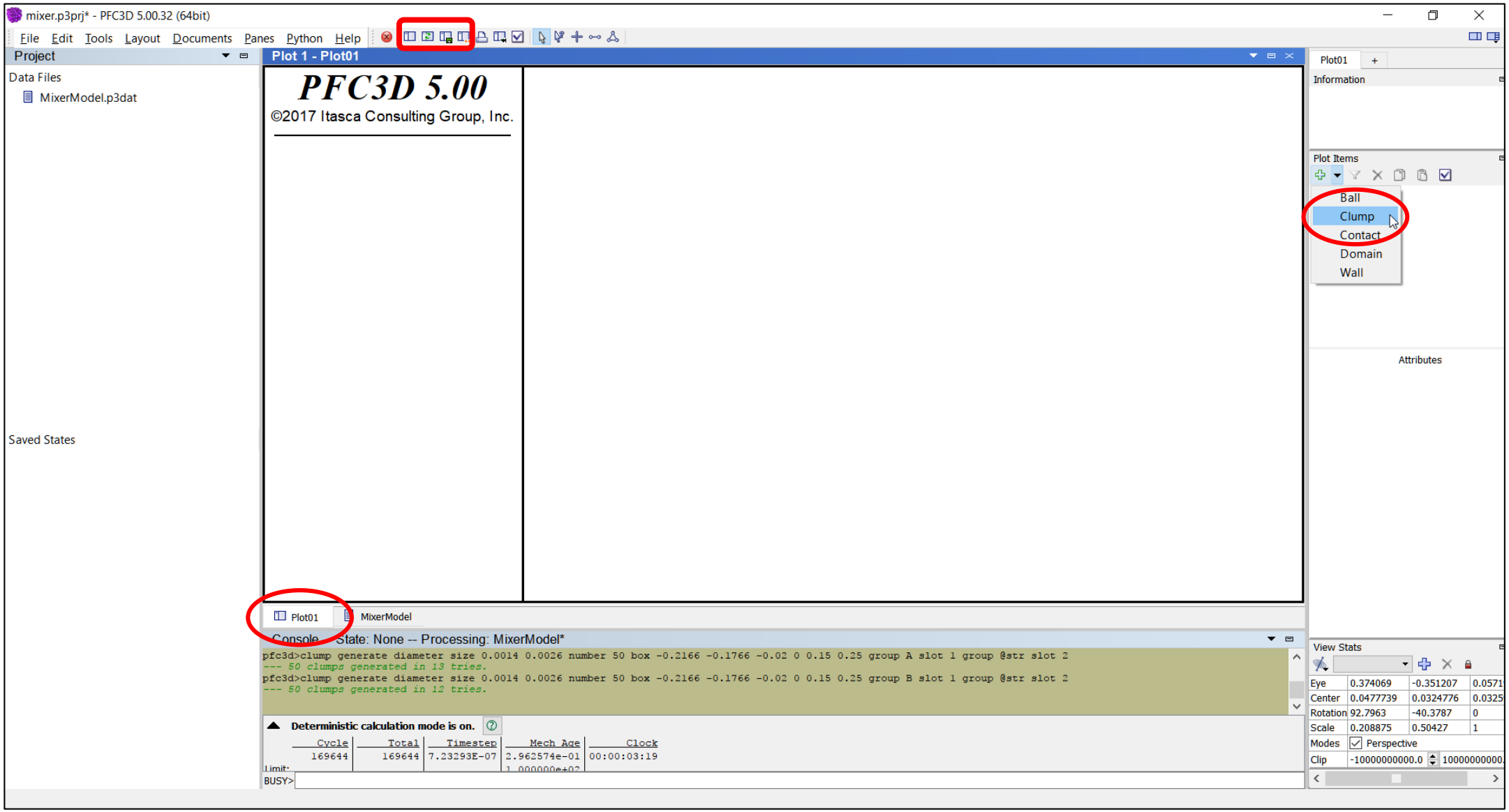
Before calling the *FISH* function **feeder**, *FISH* fragments are used to define the user variables **freq**, **id**, **time_start**, **tnext**, and **batch**. Attaching a *FISH* function to a callback event causes the *FISH* function to be executed by *PFC*, either at a fixed point in the cycle sequence or in response to a specific event. The float variable value **-9.0** specifies that the function **feeder** is called each model calculation cycle *after the data structures are validated and before the timestep is determined*. Refer to Table 1 (Cycle Operations and Associated Cycle Points) under the Cycling entry in the manual. Solve age 100 runs the model for 100 seconds of accumulated simulated time. After 100 seconds has passed the model state will be saved with the file name final[.p3sav]. The return command returns program control to the calling file if this data file had been called from another one.



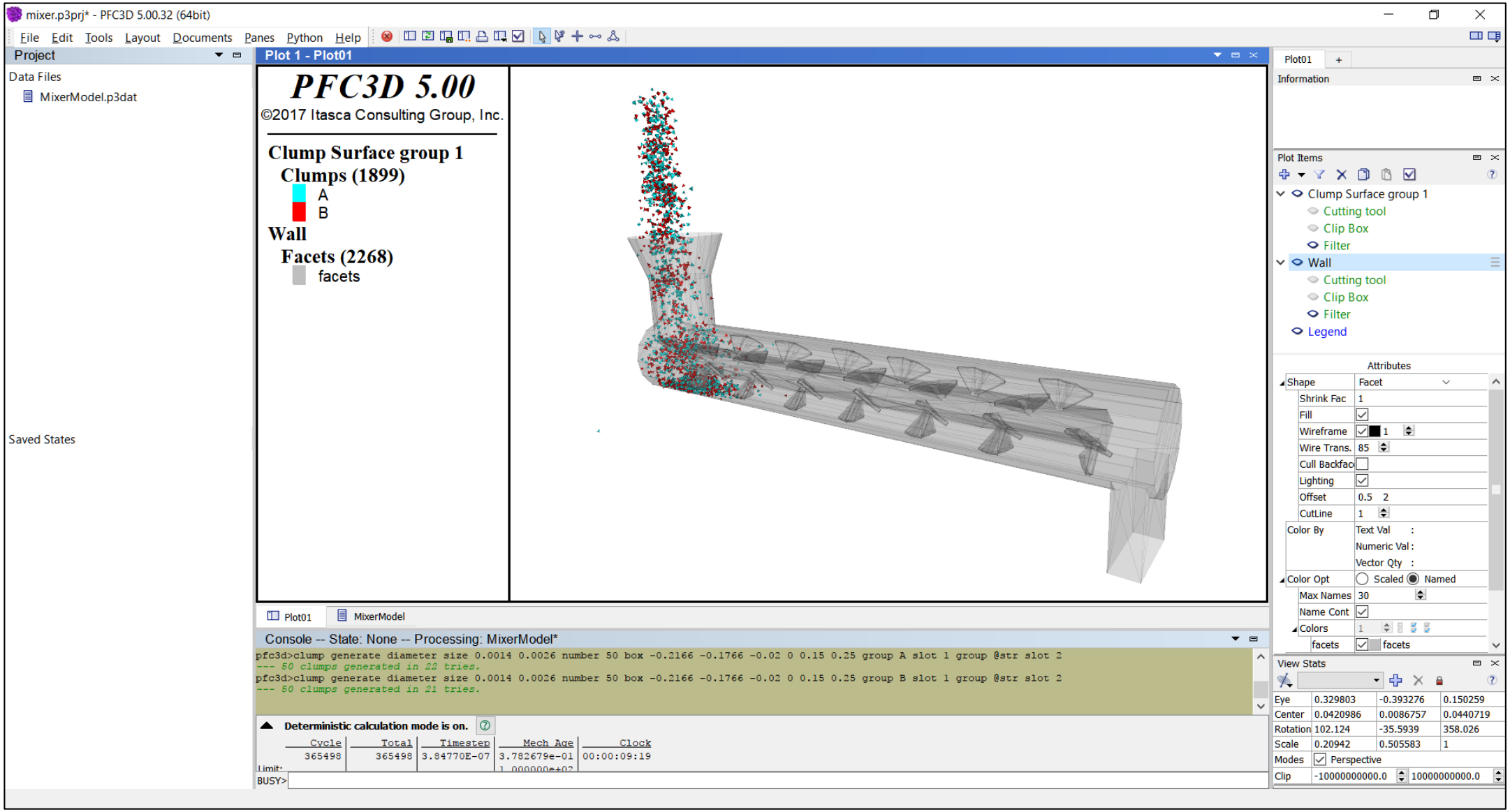
To execute the data file with all of its commands and functions, click on the green “Execute the current page icon.”



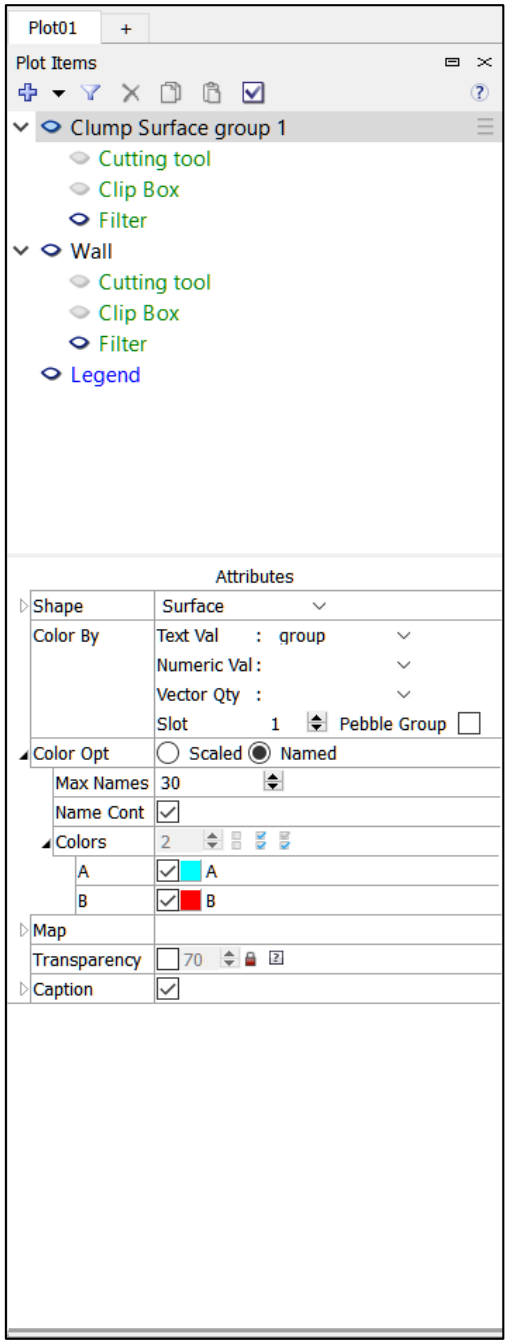
The model will start to cycle and the cycling information will display in the console showing the incremental and cumulative steps, current timestep, mechanical model age, and the actual run time (clock). To stop execution, click the red *Interrupt command processing at first safe opportunity* menu icon.



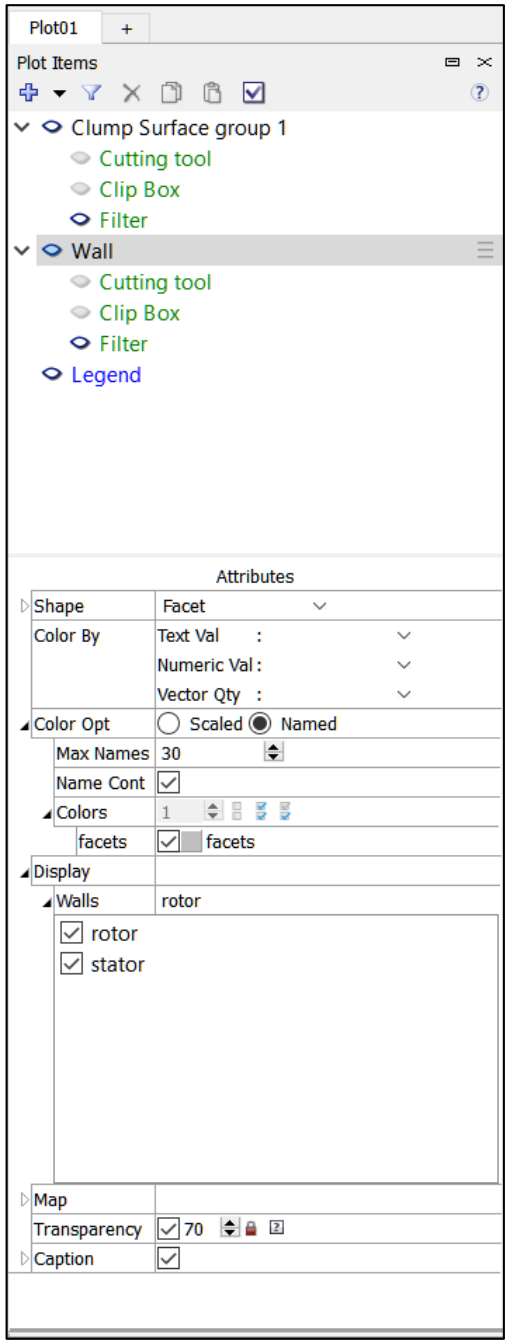
Plots can be created at any point by clicking on the Plot01 tab and adding Plot Items. Common plot-items are available in the drop-down list but many more are available by clicking on the + icon beside it. Additional plots can be added by creating them or cloning existing ones via the main menu icons.



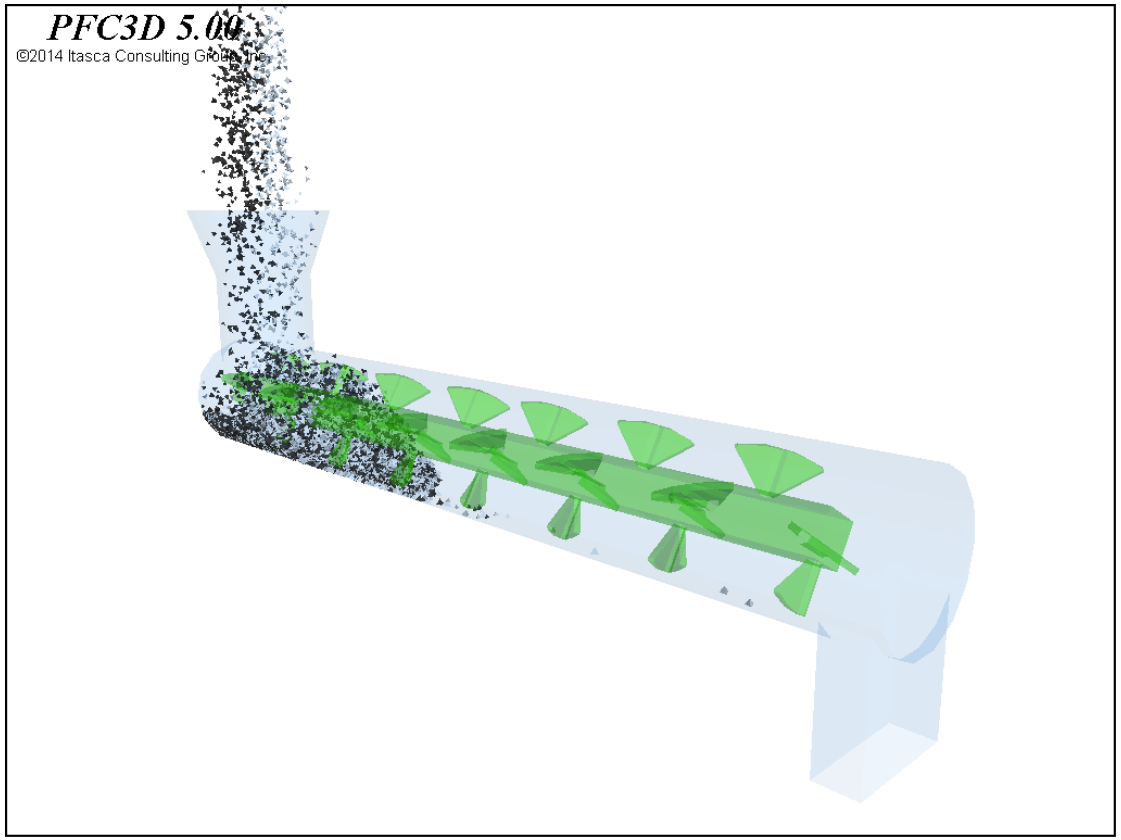
Here's a plot of the clumps, shown via the geometry surfaces, colored by A and B groups with transparent walls by changing the attributes for each plot-item. Refer to the next page for the attributes used to create this plot.

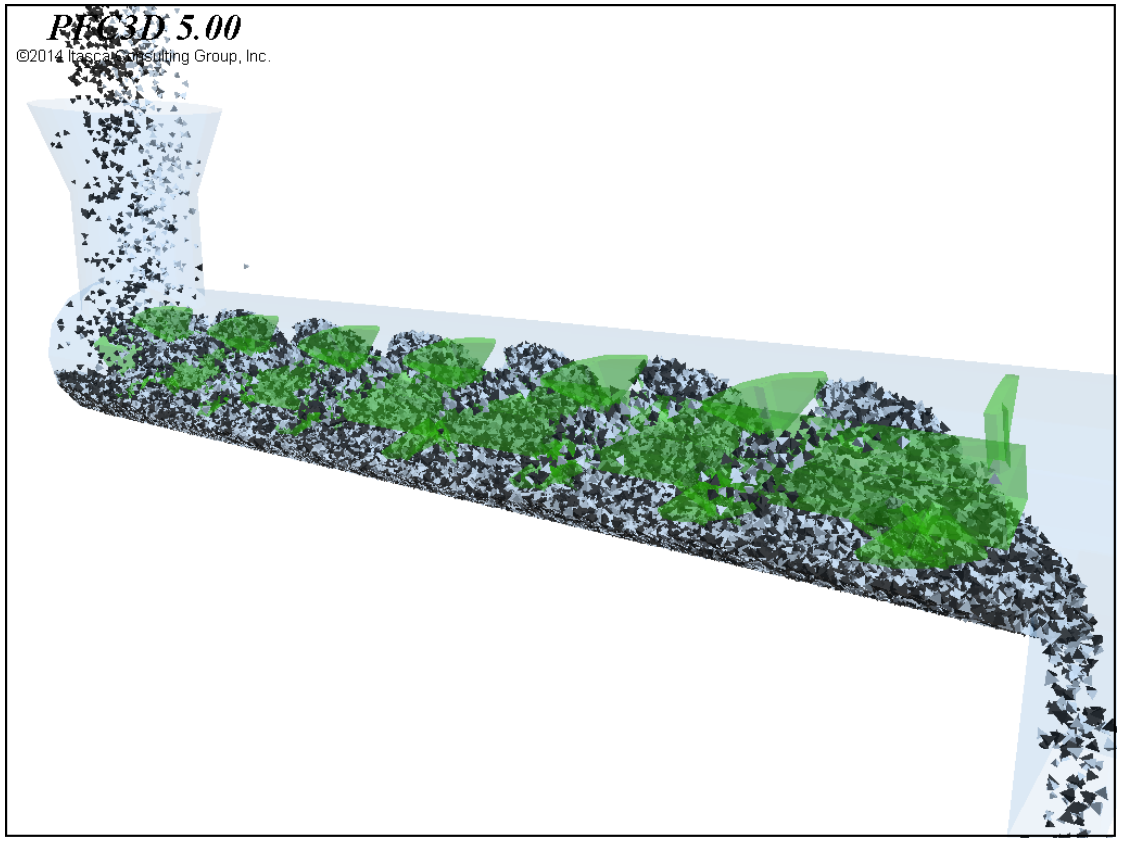


Clumps



Walls


PFC3D 5.00
©2014 Itasca Consulting Group, Inc.


PFC3D 5.00
©2014 Itasca Consulting Group, Inc.