FC TD Pack

by Fabricio Chamon
fabricio.chamon@gmail.com

This pack consists of many scripts and compounds built over years of production needs, they have helped in many ways to increase productivity and efficiency on daily tasks. Here's a comprehensive list of what's inside the pack:

ICE COMPOUNDS

Over 100 compounds, organised in the following structure:

- ARRAY
  - get cumulative distance
- COLORS
  - COLOR GOAL
    - Init Color Goal
    - Get Color Goal Position
    - Set Color Goal
    - Set Color Goal Extended
  - Change Particle Alpha
  - Copy Color from Instances
  - Linear Interpolate Color
- CONDITIONALS
  - Test Current Frame Between
  - Test Particle State ID
- CROWDFX
  - Append Materials
  - In Place Motion
  - Pose Constraint to Crowd Character
  - Randomize Materials By Weights
- DEFORMATION
  - Push
- ENVELOPE
  - Normalize Envelope Weights
- FRACTALS
  - Ridged Multifractal
- MATH
  - Multiply Vector by Self Inverted Matrix
  - Normalize Scalar Set
  - Quantize Value

- MELENA *(Studio Nest's Melena extensions)*
    - Constraint Strands to Offset Curves
    - Create Hair Cage Modeling
    - Follow Curve Force
    - Follow curve post sim
    - Get Source Color
    - Store Curve Offset 2
    - Store curve offset
- MODELING
    - Blend 3 Curves
    - deform by wave
    - FC Rock
    - Loft
    - Merge Objects on Group
- MOMENTUM *(Exocortex Momentum extensions)*
    - MOM Break Cluster on Impact
    - MOM Constraint to RBD closest surface
    - MOM Delete Constraint at Impact Threshold
    - MOM Network Constraint
- PARTICLES
    - Add Tendril from Null
    - calc particle velocity
    - Delete by Camera
    - Delete Particles by Alpha 0
    - Delete Particles by Null
    - Delete Particles by Position
    - Delete Random Particles
    - Edge Detection
    - gradient by velocity
    - Incidence Emitter
    - Modify by Sinus
    - Multiply Particle Size
    - Offset Particles from Surface
    - Offset Particles
    - Offset Rotation
    - Random Spin No Simulation
    - Random Spin
    - Redistribute Shape IDs
    - Relax Particles
    - Set Axis
- RIGID BODY
    - Simulate Bullet RigidBodies from Group
- STRANDS
    - BASIC
        - Get First Strand
        - Get Last Strand
        - Get Strand Count
        - Get Strand Length

            - Get Strand Orientation
            - Get Strand Position
            - Get Strand Size
            - Remove First Strand
            - Remove Last Strand
            - Reverse Strands
            - Set Strand Color
            - Set Strand Count
            - Set Strand Deform
            - Set Strand Orientation
            - Set Strand Position
            - Set Strand Shape Resolution
            - Set Strand Shape Size
            - Set Strand Size
            - Set StrandSize Root to Tip
        - Aging Cobwebs
        - Connect Strands
        - Offset StrandPosition by Fcurve
        - Resample Strands by Bezier
        - Trim Strands by Volume
- STRANDTREE *(Procedural Trees and Plants nodes)*
    - DEBUG
        - Debug Iteration
        - Volume Preview
    - EMITTERS
        - Emit Branches
        - Emit Leaves
    - GROW CONTROL
        - Even Distribution Grow
        - Random Distribution Grow
    - MODIFIERS
        - Modify Branch Color by Iteration
        - Modify Branch Width by Iteration
        - Modify Particle Size by Iteration
        - Modify Value by Iteration
        - Modify Value by Normalized Strand Segment
    - PRESETS
        - Basic Tree
        - Pine Tree
    - RENDERING
        - Optimize Tree
        - Tree Visibility Options
        - Use Geometry Branches
    - STRAND EFFECTS
        - Deform Tree by Curve
        - Surface Grow
    - UTILS
        - Clone Leaves from Strand Tree

- STRING
    - Change File Name
- UTILS
    - blend 3 values
    - Debug Distance To Camera
    - Force Attribute Eval
    - Force Per Point Data
    - Get Group Data
    - Get Point Position from Location
    - Get Self
    - Gradient by Distance to Camera
    - Ruler
- WEIGHTMAP
    - Blur Weightmap
    - Init Grow Weightmap
    - Grow Weightmap
    - Set Weightmap Value
    - Smooth Weightmap

SCRIPT COMMANDS AND IMPORTERS

In the “Softimage/Scripts/Applications/Plugins” subfolder, you’ll find some useful commands. To install, just copy them to your Application/Plugins user folder. It is recommended to bind each command to a key shortcut to make best use of them.

Command_SelectChildren.js : select parent and run the command to add entire hierarchy. Useful for selecting whole chains without having to pick bones one by one.

CustomizedOpenView*_[VIEWNAME]*.js : this set of commands are very helpful for dual monitors. They launch the desired view fitted onto the second monitor. You may need to change the default resolution, which is HD 1920x1080.

Bvh_Importer.js : Drag’n’drop BVH files directly into softimage viewport. Option to time-scale the motion file when importing.

Obj_Importer.js : Drag’n’drop OBJ files directly into softimage viewport.

SCRIPT TOOLS AND UTILITIES

General purpose scripts, divided into categories.

Animation:

Follow_Object_Camera.js : Creates a camera that follows selected object. Useful to track local animation while the object is moving (example: tweak facial expressions of a running character)

Offset_Keyframes_on_Objects.js : Offsets all animation keys on selected objects by N frames (variable “offsetFrames”).

Recursive_Models_Anim_Save.js : When dealing with characters that have models inside models, this is script is gold. It loops through all selected rig controllers, groups them by model and save actions under the respective models. Usage: Select all the animation controls you want to save > run. Type animation name and pick the folder where they will be saved.

Recursive_Models_Anim_Read.js : Reads saved animations made with “Recursive_Models_Anim_Save.js”. Usage: Select the top character model > run. Select just one of the actions previously exported (the script will load all matching sources).

ICE:

Bake_ICE_UV.js : creates a copy of the selected mesh, and bakes ice created UVs on it.

Create_Melena_Guide_Pointcloud_From_Curves.js : This aimed at hair setups made with StudioNest’s Melena Hair System. Given a set of nurbs curves, it creates and connects a Melena Guide Pointcloud.

Layers_Groups_and_Partitions:

Create_Groups_From_Models.js : For each selected model, creates a group with its contents inside (flattened hierarchy). (It’s not the same as branch-selection > add to group!). Usage: select one or more models > run.

Create_Layers_From_Materials.js : Useful to isolate objects by material type (example: I have a car and want to render only the chrome metal parts). Usage: run > pick source matlib in the explorer. It will create layers with name “MAT_[materialname]” for each material in the matlib and put relevant objects inside.

Create_Partitions_From_Groups.js : Create partitions with same name and contents as the groups selected. Usage: select one or more groups > run.

Modeling:

Break_Polygon_Islands.js : Extracts all islands from selected object into separate meshes. Useful to split shattered parts from a single object.

Create_Tubes_From_Curves.js : Useful for quick creation of ropes and cables. Select one or more curves > run.

Explode_Subcurves.js : Extracts all subcurves from selected nurbs curve.

ICE_Cage_Deformer.js : Sets up a cage deformer. Usage: Select hi-poly geometry > run > pick cage.

Merge_Selected_Curves.js : Merge a set of nurbs curve into one single object.

Select_Random_Polygon_Islands.js : it can serve many purposes, one of them is to add variation to shattered pieces of a single object (select object > run script, then move/rotate pieces).

Select_Recursive_Child_Polymeshes.js.js : sometimes you have a model with many herarchy levels and want to isolate only the polygon meshes. It can be difficult to select on the viewport, because parts can be hidden, or have selectability overrides. Whenever it is difficult to reach all geometry under any object, just use this script.

Old:

Ambient_LightRig_xsi7.js : This is an old (XSI 7) script that generates an “environment” light rig using lights attached to a geometry. Although it may be old, some nifty tricks are used inside this script to pass texture color on to lights. Maybe you can detach some parts and use into your own script.

Particles_and_Instances:

Convert_Instances_to_Models.js : revert back selected instances to models. It is very useful to tweak some objects when you already have a hand scattered scene. Usage: select instances > run > pick source model (or a different model if you want to change).

Freeze_PointCloud_Instances.js : create a snapshot of current ice pointcloud instances into meshes. It matches size/translation/rotation/scale of particles. Select pointcloud > run. (* If you are instancing models, “createInstances” variable will tell whether the script creates instances or duplicates of the original model).

Instantiate_on_Objects.js : Useful for scene assembling and layout. Example: you have a city scene with dozens of proxy cubes (constructions) placed by hand, then you want to populate randomly with highpoly buildings and houses. Usage: have 2 distinct groups: one for all the proxies, another for the hi-poly models. With nothing selected run the script > pick hi-poly models group first > then pick proxy group. It will scatter random hi-poly models onto the proxies and pose-constraint them.

Rigging_and_Technical:

Create_Curve_At_Objects_And_Constraint_Curve.js : creates a smooth curve that is pinned to selected controllers (respects order of selection)..

Create_Curve_At_Objects_And_Constraint_Objects.js : inverse of the above script, now the objects are pinned to curve points.

Create_Parent_Null.js : creates a null in the bounding box center of current selected meshes and parent the them under the null. I like to call it “fast-parenting”.

ICE_Attribute_To_Component_Selection.js : very useful for rule-based/programmable component selection. Just create a boolean ice attribute on the mesh called “self.select”, tag by whatever condition you want (inside volume for example), then run the script. it will select components based on the context of the variable (per-point will select vertices, per-polygon will select faces, and so on).

Linear_Hierarchy_From_Selection.js : build a hierarchy using selection order. Example, sphere1, sphere2, and sphere3 are selected in order. After the script they become sphere3 (child of) > sphere2 (child of) > sphere1 (top level).

Modulate_Selected_Shape_Keys_With_Weightmap.js : Useful for splitting facial expressions into left/right shapes. Usage: in the explorer select all the shapes you want to modulate plus the weightmap (last selection) > run.

Read_Text_File.js : Simple example of reading a text file.

Replace_In_Text_File.js : Simple example of replacing a text inside a file. Useful for reading templates.

Stick_And_Slide_on_Mesh.js : Fast and easy way to glue rig parts onto a mesh. It follows geometry, aligns to normal and have an offset control that slide/conform to the underlying the mesh. Usage: select control > run > pick geometry > select edge (it will contraint the control to lower vertex and align to the upper vertex)

Strands_to_Curves.js : convert strands to nurbs curves.

Weightmap_Set_Value_For_Selected_Points.js : Useful for precise control over weightmap data. Usage: tag some points on desired object > ctrl+select the weightmap > run > inform new weight value for the points.

Script_Functions:

Those are overall helpers that you can use inside other scripts.

Deactivate_XSI_Log.js : increase the speed of your scripts by turning off xsi command log while they are running.

Get_Screen_Resolution.js : self-explanatory. Useful for many UI stuff.

Log_Params_Props_and_Attrs.js : inspect object data, logs all at once, so you have an overview of the object you are manipulating.

Open_View_on_Second_Monitor.js : launch window maximized on second monitor. *May need to adjust the default 1920x1080 resolution.

Selection_to_Collection.js : stores current selection inside an xsi collection. Example usage: If you are looping over an object selection and create a null, the new selection becomes the null itself, and you loose the original selected objects. Using this script prevents you from losing the original selection.

String_Replace_All.js : Implements “replace all” string function in javascript.

Tracking:

Setup_Boujou_Tracking_Scene.js : For those using Vicon Boujou, this script automates scene importing and setup for boujou tracked shots:

- import boujou camera file (.xsi)
- import image sequence
- plots camera
- group and color code reference nulls
- create image clip and rotoscope from plate
- sets camera and render frame formats

RENDETREE SHADER COMPOUNDS

Ridged Multifractal Shader for Arnold : Create terrains and other fractal stuff. Outputs a scalar value that can be used to drive displacement, bump or gradients.

*(*Bonus Content)*

generate_yeti_caches.mel : Very useful script for those working on Peregrine Labs’ Yeti Hair plugin. It finds all yeti nodes on the scene, caches to specified folder and switches mode to cache mode. In one click you are ready to renderfarm your scenes. Usage: Inside the mel script, change variables in the “cache info” section (scene name, start and end frames).