jc.clothes

Aug 2008

Stitching

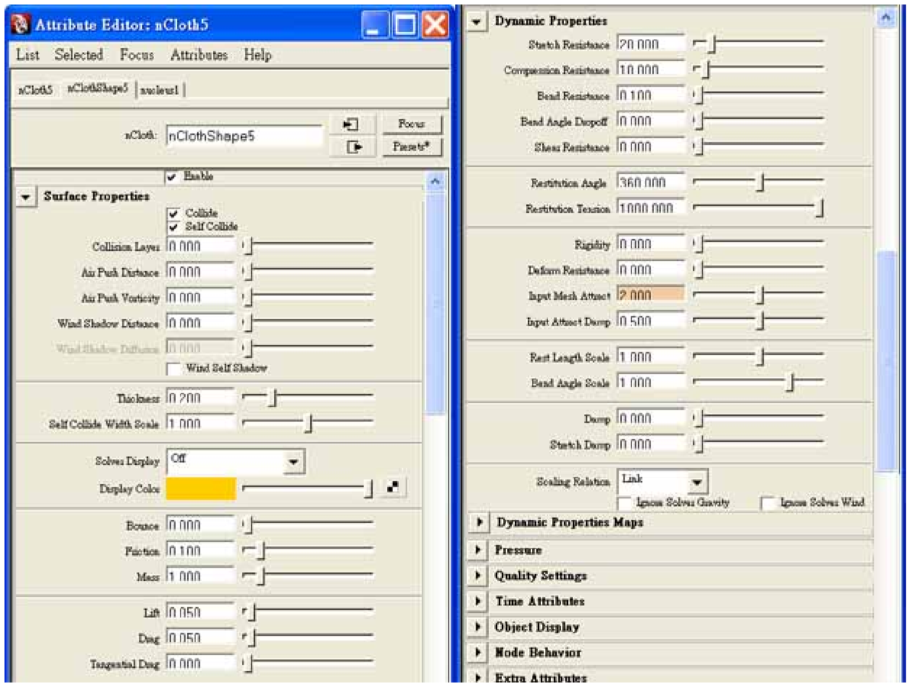
After creating patterns, the next thing to do is to make them 'wrap' around the character and join or stitch the patterns together. The steps are:

- Create joints at the seams in both pattern and measurement curves
- Bind the patterns to the joints (not required in scripted workflow)
- Create nCloth (not required in scripted workflow)
- Create dynamic constraints (not required in scripted workflow)
- Animate the joints, nCloth and constraints

Firstly, create joint chains at the measurement curves corresponding to the seams. For straight seams, two joints (root and end) are enough. For curved seams, there'll be more joints but the number of them involved depends on how precise you want to do stitching. Then duplicate them, 'flatten' them in the front plane (by zero out rotation and joint orient) and match the corresponding edges in the patterns by manually rotating the joints as you can see in Figure 1.

Note that the joints on the patterns must be duplicated from those on the measurement curves because these joints will snap together for the sake of wrapping the patterns around the character. So they must be identical.

Then mirror the joint chains if necessary. Put the joints (affecting the same pattern) into a single group. Bind the garment to the joints (bind to 'Joint hierarchy' in Skin -> Smooth bind). Create nCloth. Make character Passive. Set Input Mesh Attract to 2 in all patterns. See Figure 3 below.


Attribute Editor: nCloth5
List Selected Focus Attributes Help
nCloth5 nClothShape5 nucleus1
Focus
Presets*

nCloth: nClothShape5

✓ Enable

Surface Properties

✓ Collide	
✓ Self Collide	
Collision Layer	0.000
Air Push Distance	0.000
Air Push Vorticity	0.000
Wind Shadow Distance	0.000
Wind Shadow Diffusion	0.000
Wind Self Shadow	
Thickness	0.200
Self Collide Width Scale	1.000
Solver Display	Off
Display Color	🟨
Bounce	0.000
Friction	0.100
Mass	1.000
Lift	0.050
Drag	0.050
Tangential Drag	0.000

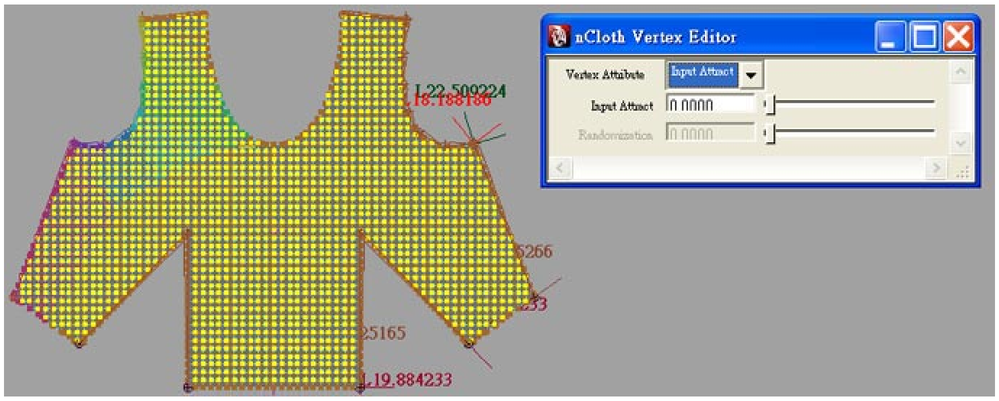
Dynamic Properties

Stretch Resistance	20.000
Compression Resistance	10.000
Bend Resistance	0.100
Bend Angle Dropoff	0.000
Shear Resistance	0.000
Restitution Angle	360.000
Restitution Tension	1000.000
Rigidity	0.000
Deform Resistance	0.000
Input Mesh Attract	2.000
Input Attract Damp	0.500
Rest Length Scale	1.000
Bend Angle Scale	1.000
Damp	0.000
Stretch Damp	0.000
Scaling Relation	Link
Ignore Solver Gravity	
Ignore Solver Wind	

Figure 3. nCloth attributes

Set Input Mesh Attract for the vertices along stitch borders to 1 and the rest 0. Use Duncan's nClothVertexEditor to perform this task. See Figure 4 below.

Those vertices on the border which are outside the seams (not on the joints) should have their Input Attract value set to 0 as they won't follow the deformation caused by the joints.



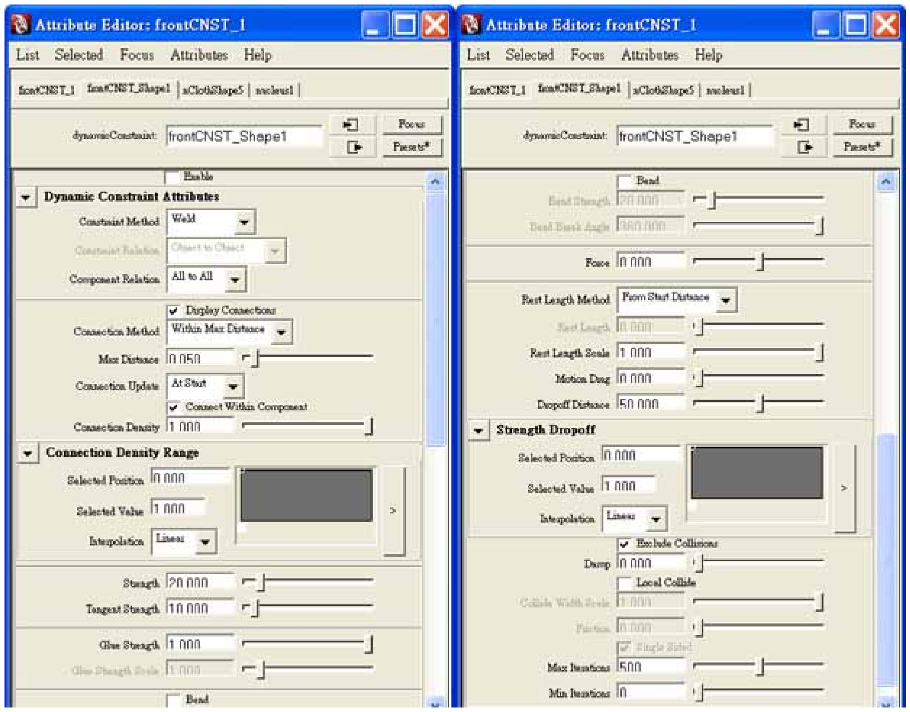
Vertex Attribute: Input Attract
Input Attract: 0.0000
Randomization: 0.0000

Figure 4. nCloth vertex properties

Create weld constraints among the connecting patterns (select both patterns and nConstraint -> Create weld constraint) and within the same pattern (select one pattern and nConstraint -> Create weld constraint) if needed. Those which are required for the blouse are shown in the following table:

Weld constraint	Pattern 1	Pattern 2	seam
1	Front		2 from chest to waist
2	Front	Back	shoulder, body side
3	Left sleeve		inside arm
4	Right sleeve		inside arm
5	Front	Left sleeve	armhole
6	Front	Right sleeve	armhole
7	Back	Left sleeve	armhole
8	Back	Right sleeve	armhole

Then turn on Exclude Collision. Turn off Enable. See Figure 5 below.


Attribute Editor: frontCNST_1
List Selected Focus Attributes Help
frontCNST_1 frontCNST_Shape1 nClothShape5 nucleus1
Focus
Presets*

dynamicConstraint: frontCNST_Shape1

Enable

Dynamic Constraint Attributes

Constraint Method: Weld
Constraint Relation: Object to Object
Component Relation: All to All
✓ Display Connections
Connection Method: Within Max Distance
Max Distance: 0.050
Connection Update: At Start
✓ Connect Within Component
Connection Density: 1.000

Connection Density Range

Selected Position: 0.000
Selected Value: 1.000
Interpolation: Linear

Strength: 20.000
Tangent Strength: 10.000
Glue Strength: 1.000
Glue Strength Scale: 1.000
Bend
Bend Strength: 20.000
Bend Break Angle: 360.000
Force: 0.000
Rest Length Method: From Start Distance
Rest Length: 0.000
Rest Length Scale: 1.000
Motion Drag: 0.000
Dropoff Distance: 50.000

Strength Dropoff

Selected Position: 0.000
Selected Value: 1.000
Interpolation: Linear

✓ Exclude Collisions
Damp: 0.000
Local Collide
Collide Width Scale: 1.000
Friction: 0.000
✓ Single Sided
Max Iterations: 500
Min Iterations: 0

Figure 5. Weld constraint 1

Here comes simulation. The sequence is:

Time	Action
1	Move patterns to initial positions
40	Snap joints
50	Turn on weld constraints

60	Turn off Input Mesh Attract

At time=1, move the patterns by translating and rotating the joint group to initial positions and set key for its translations and rotations. See Figure 6.

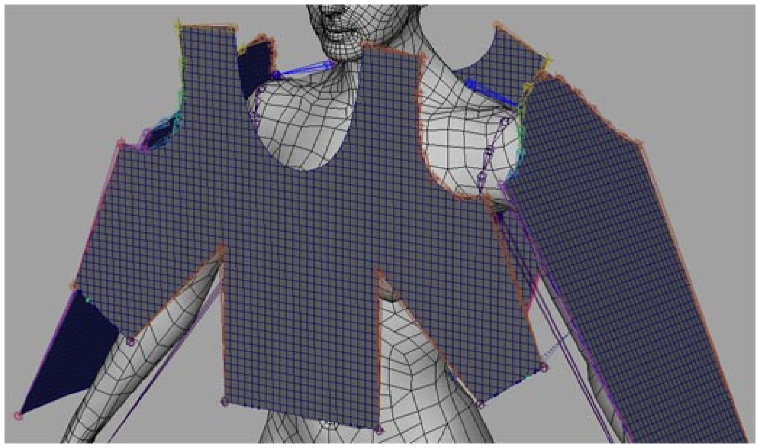


Figure 6. Initial positions

At time=40, snap the joints to the corresponding ones around the body. Then set key for their translations and rotations.

This snapping process is not trivial. There're two ways to accomplish this: by using IKs or orient constraints. Doing this manually is tedious. So this step is automated and can be carried out easily with a single command jc.clothes -> Match Joints which is further encapsulated within jc.clothes -> Set Keyframes. Details will be described later in this section.

At time=49, set key for weld constraints' Enable (which is off). At time=50, turn it on and set key.

At time=59, set key for patterns' (nCloth) Input Mesh Attract (which is 2). At time=60, make it 0 and set key.

The last two steps has been automated. It'll be described later in this section.

Finally, play simulation. It took less than 3 minutes for 100 frames in my P4 machine. See Figure 7.

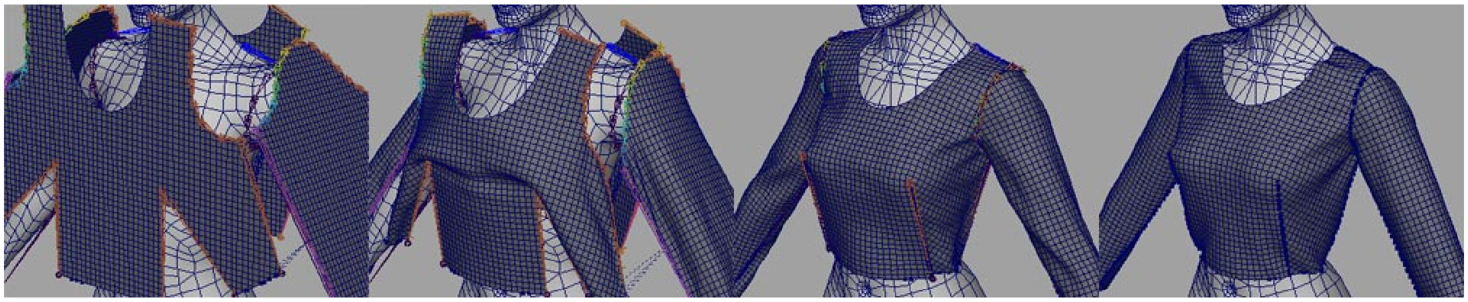
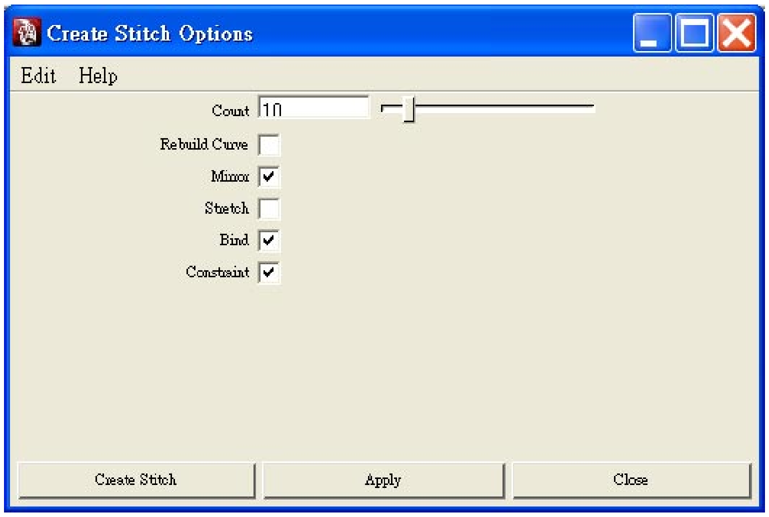


Figure 7. Stitching simulation

It is impossible to achieve the best result in your first attempt. If the garment isn't fit enough, you have to adjust the patterns and do it all over again. So you may want to do it incrementally. For the blouse above, you can make it without sleeves in your first attempt. After you're satisfied with the result, build the sleeves and combine them with the other patterns in your next attempt.

Scripted workflow

With automation, the work above can be simplified tremendously within a few steps. To create joints, select a measurement curve and shift-select the corresponding pattern curve (there can be two of them as you see in the front pattern of the blouse). Then invoke the command jc.clothes -> Create Stitch. See its option box below:



Count 10
Rebuild Curve ☐
Mirror ☑
Stretch ☐
Bind ☑
Constraint ☑

It'll then create the joint chains along the curves, mirror them to the right side, and bind the garment to them (by creating skin cluster or add influence if it already exists). The garment is found by following the jcPattern attributes of the curves.

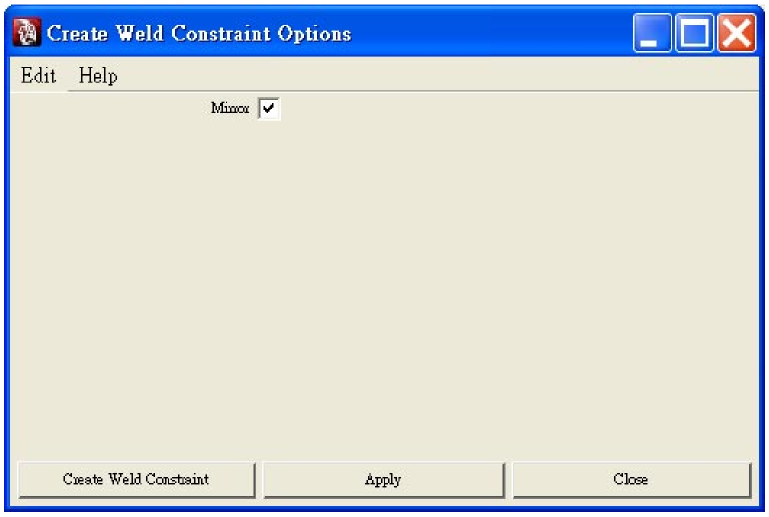
'Count' is the number of joints along the curve. Sometime it fails to create joint chain due to parameterization of the curve. You can turn on 'Rebuild Curve' to fix it. If the curve lengths don't match and your actual intention is to stretch or shrink the pattern seam during stitching, turn on 'Stretch'. If 'Bind' is on, it'll bind the joints to the pattern to which the curves correspond. It will create a weld constraint between the edges along the curves if 'Constraint' is enabled.

Before invoking the command, be careful about the direction of the curves because it determines the direction of the joint chain. In other words, the direction of the measurement curves and that of the pattern curves must match.

This command would also create an extra attribute (called jcDestinationJoint) in the root joint on the pattern curve. It is connected to the corresponding joint on the measurement curve. The command jc.clothes -> Set Keyframes would rely on this connection to find out the matching joint chains so as to do the snapping and set keyframes. So if you create the joint chains manually, you have to do the snapping also manually yourself by calling jc.clothes -> Match Joints.

There's also no need to update per vertex attributes along the pattern border. Because this command can find out the vertices along the border edge and set appropriate values for them. Because of this, it can also find out the connecting edges and create weld constraints between them. So there's no need to create constraints by hand as well.

If weld constraint is created each time you create a stitch, there'll be a number of them because there're a number of stitches. But actually one single constraint is enough to weld all the edges in different stitches. So there's a command jc.clothes -> Create Weld Constraint to do this.

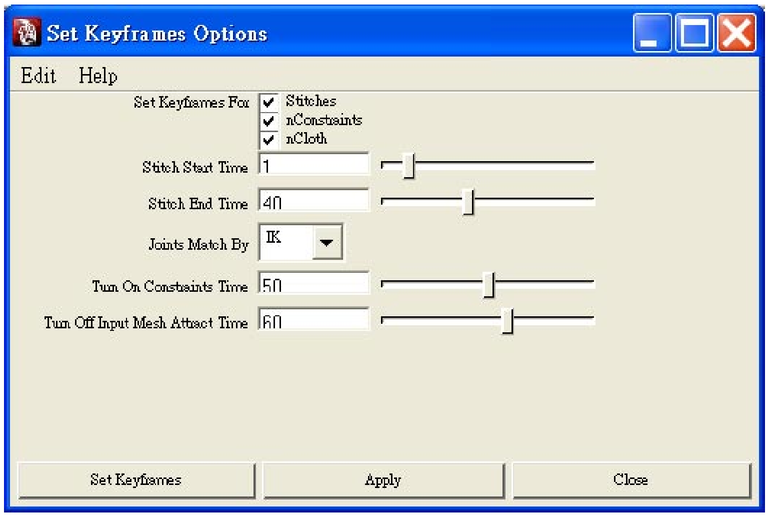


Mirror ☑

If you want to use this command, 'Constraint' should be off when you invoke jc.clothes -> Create Stitch. After all stitches are created, select pattern curves sitting on the edges which are to weld. Then invoke jc.clothes -> Create Weld Constraint and it'll create one dynamic constraint for all stitches.

Then group the pattern joints. Move the groups into start position.

To set keyframes for the stitch joints, nCloth and constraints, select garment and then invoke jc.clothes -> Set Keyframes and it'll clear all keyframes, animate the stitch joints, turn on all constraints and turn off Input Mesh Attract of the nCloth node at the specified times indicated in the option box below:


Set Keyframes Options

Set Keyframes For: ✓ Stitches, ✓ nConstraints, ✓ nCloth

Stitch Start Time: 1

Stitch End Time: 40

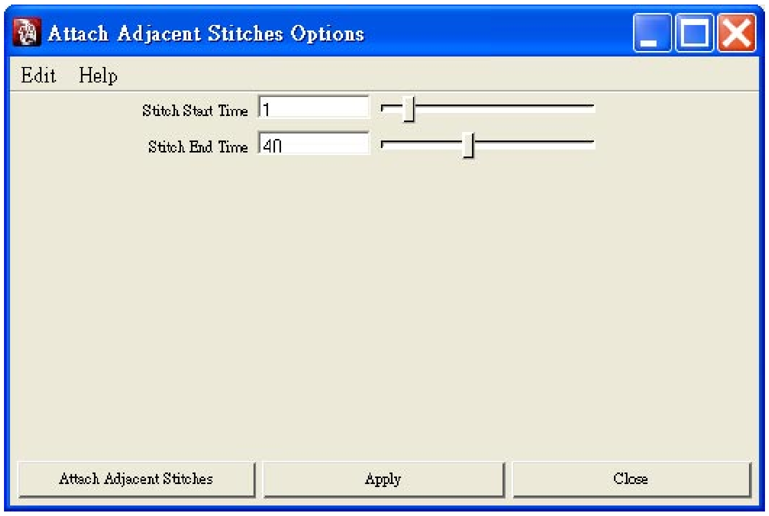
Joints Match By: IK

Turn On Constraints Time: 50

Turn Off Input Mesh Attract Time: 60

If your 'Stich Start Time' is not 1, don't forget to change 'Start Frame' in the nucleus node as well. There're two ways to match the joint chains, either by orient or by IK. If problem happens when joints are matched by orient, it can probably be fixed by using IK. The reverse is also true.

When you play simulation, you may probably found that some stitches which are along the same continues border are snapping to the destinations without following each other as if the border is broken up. This can be fixed by using the command jc.clothes -> Attach Adjacent Stitches.


Attach Adjacent Stitches Options

Stitch Start Time: 1

Stitch End Time: 40

This command would create a point constraint between the ending joint and the starting joint between two joint chains so that the latter will follow the former. If keyframes have already been set on the latter, a pairBlend node will be created automatically and a blendPoint1 attribute will be created on the affected joint to control the weight of the contraint. This blendPoint1 will be keyframed so that the effect of the point contraint will be vanished at the end of simulation. That's why its option box requies the Stitch Start Time and Stitch End Time.

To use this command, select one joint in the leading stitch (joint chain) and then shift-select one joint in the following stitch. It'll find out the ending joint of the former and the starting joint of the latter, and then create the point constraint.

Finally, if you want to throw away everything constructed (except pattern curves) and start it all over again, you can select garment and invoke jc.clothes -> Delete Garment.

Rebuild Garment

When the garment is about finished, you'll find that you need to fine-tune the pattern curves and rebuild the garment repetitively. It is a kind of hassle you want to get rid of. This is possible because the process is scriptable. Before writing the script, you should have built the garment at least once or until you no longer add pattern curves or change pattern structure. Delete the garment (invoke jc.clothes -> Delete Garment). Name the curves properly. Retain the stitch groups (together with their keyframes). Retain Passive object or anything outside the garment. Write the script in Script Editor (in Python pane). Save it to the shelf and execute it after making adjustment to the curves. The basic structure of the script is as follows:

```
# turn off undo queue
undoState = maya.cmds.undoInfo(q=True, state=True)
maya.cmds.undoInfo(state=False)


# create garment
p1 = jc.clothes.createPattern(curves..., u=0.1, v=0.1, mirror=True, \
        division=0, reverseNormal=False)
p2 = jc.clothes.createPattern(curves..., u=0.1, v=0.1, mirror=True, \
        division=0, reverseNormal=True)
...
g = jc.clothes.createGarment(p1, p2, ...)


# create stitches
s1 = jc.clothes.createStitch(curves..., count=10, rebuildCurve=False, mirror=True, \
        stretch=False, bind=True, constraint=True)
s2 = jc.clothes.createStitch(curves..., count=10, rebuildCurve=False, mirror=True, \
        stretch=False, bind=True, constraint=True)
...


# create weld constraint (optional)
# jc.clothes.createWeldConstraint(curves..., mirror=True)


# group stitches
maya.cmds.currentTime(0)
maya.cmds.parent(s1[0], s2[0], ..., "bodyStitchN_1")
maya.cmds.parent(s1[1], s2[1], ..., "stitchN_1")
...

# set keyframes
maya.cmds.currentTime(1)
jc.clothes.setKeyframes(g, setKeyframesFor="Stitches nConstraints nCloth", \
        stitchStartTime=1, stitchEndTime=40, jointsMatchBy="IK", \
        turnOnConstraintsTime=50, turnOffInputMeshAttractTime=60)

# attach adjacent stitches (optional)
# jc.clothes.attachAdjacentStitches(s1[1][0], ..., stitchStartTime=1, stitchEndTime=40)
# jc.clothes.attachAdjacentStitches(s1[1][1], ..., stitchStartTime=1, stitchEndTime=40)
...

maya.cmds.undoInfo(state=undoState)
```

It is necessary to understand the return values of each command before writing the script. Those of jc.clothes.createPattern and jc.clothes.createGarment are trivial. They're the pattern object and the garment object respectively.

The return value of jc.clothes.createStitch is a list of joints. Its sequence depends on the sequence of arguments (curves) you put into the command. The first argument is always the measurement curve. So the first item (index 0) in the output list is always the joint chain created on the measurement curve. Then you should know the next item corresponds to which curve. If 'mirror' is on, the items in the list would be tuples in which the first item (index 0) is the joint chain on the left, the second item (index 1) is that on the right. You can find an example at the end of the walkthrough guide.