The art direction on Serpendipity called for extreme, cartoony animation that required a lot of stretching and bending for smear frames. I developed these setups to combat some of the issues that pop up when adding these features to the IK spline setups that AFR teaches. These setups add in more flexibility to the rigs that can better mimic 2D drawn animation with stretch, squash, and smear frames, as well as add extra controls to tweak silhouette. The stretchy squashy setup could have been added to any ribbon spline for volume preservation (Snakes and Limbs). There is also the ability to add a switch to turn the volume preservation on or off through the use of an attribute connected to blend color nodes intercepting the scale of the joints. However, this wasn't necessary as it wasn't as noticeable as the spine and added unnecessary details to keep track of by our animators.

Stretchy Squashy Ribbon Spline

This workflow is to replace the IK spline setup taught in AFR and aims to work in tandem with the rest of the rigging setups taught in AFR. The advantage of a ribbon over the IK spline setup in AFR is there isn't the possibility of flipping caused by Maya's up vector in the IK spline when turning the character around inside of the all_anim.

1. Create the ribbon spline with controls as usual. I'll refer to the nurbs plane as *back_ribbon_plane* and the joints as *back_#_jnt*.

2. Select one of the isoparms along the length of *back_ribbon_plane*.

3. Use MEL code "arclen -ch 1;" to create a curveInfo node for the isoparm.

5. In the node editor, with back_ribbon_plane selected, find *curveFromSurfaceIso1* that is connected to the *back_ribbon_planeShape* node. If nothing is appearing in the node editor, click on the Input and Output Connections button at the top.

6. Click on *curveFromSurfaceIso1* and in the attribute editor, change Isoparm Value to 0.500. This makes it so that the isoparm we're going to be getting data from is at the center of the ribbon spline where the joints are constrained to the nHair.

7. Find and click on *curveInfo1*, it should be connected to *curveFromSurfaceIso1*, and rename it to *back_curveInfo* this carries the Arc Length value that we need to keep track of to see when the spine is stretched or squashed.

8. Create a *multiplyDivide* node and rename it to *normalized_backScale*. To create nodes quickly in the node editor, hit Tab and type in the node you want to create.

.

9. Attach Arc Length from *back_curveInfo* to Input 1X on normalized_backScale by clicking and dragging the output connection to the input connection on the other node . You can expand the nodes you have selected by clicking on the Show Attributes buttons at the top so that the attributes you need are visible.

10. In the attribute editor of *normalized_backScale*, change the operation to Divide. Change Input 2X to the original value of the Arc Length in t-pose (Gordon's is 7.833). This will divide whatever the current length of the spine is by the original length to get the new value for the scale of the joints.

11. Check the orientation of the *back_#_jnts* and see which axis is pointing up. More than likely it is Y as they are oriented to the world axis and constrained to the hair follicles.

12. In the node editor, attach the Output X from normalized_backScale to the Y scale (or whatever axis is pointing up) of *back_#_jnts* 2-8 on the back. Leaving out the first and ninth joints will keep the hips and shoulders at a consistent size and keep the volume preservation only in the spine.

13. The spine should now stretch and squash equally. To keep volume preservation we need to scale the remaining two axis, X and Z, by 1/(sqrt(ScaleY)).

14. Create another multiplyDivide node and rename it to *sqrt_backScale*.

15. Attach Output X from *normalized_backScale* to Input 1X on *sqrt_backScale*.

16. Set the operation to Pow and Input 2X to 0.500. This will give us the square root.

17. Create another multiplyDivide node and rename it *inversed_backScale*. Set the operation to divide

18. Attach Output X from *sqrt_backScale* to Input 2X on *inversed_backScale* and set the value of Input 1X to 1.000. The Output X of *inversed_backScale* is the value we need to plug into the X and Z scale of the *back_#_jnts* to have volume preservation.

19. Attach Output X from *inversed_backScale* to the Scale X and Z of *back_#_jnts* 2-8 on the back.

20. Make sure the stretch and squash is working with all of the controls.

21. Now we need to make this scalable by the all_anim. To do this we're going to take the *normalized_backScale* value and divide it by the *globalScale* value we're going to make on the all_anim group. (Follow AFR for setting up the all_anim group and organization of the outliner before continuing)

22. Once the outliner is organized and the all_anim group is made with *globalScale* attribute, make sure the under the *doNotTouch_grp*, that both *back_follicle_grp* and *back_ribbon_plane* both have Inherit Transforms UNCHECKED. The rig should now work when you move the all_anim around but not when you scale it.

23. To fix this, we need to divide the *normalized_backScale* value by the *globalScale* before feeding it into the joints and the sqrt and inverse nodes we made.

24. Create a new multiplyDivide node, rename it to *globalScale_div* and set the operation to divide.

25. In your outliner click on the *all_anim* group and middle mouse drag it into the node editor to create a node for it.

26. Attach the *globalScale* into Input 2X of *globalScale_div*.

27. Break all of the connections between *normalized_backScale* output X and attach it only to Input 1X of *globalScale_div*.

28. Attach Output X of *globalScale_div* to Scale Y of *back_#_jnts* 2-8 and to Input 1X of *sqrt_backScale*.

29. Once all of the connections are made the rig should be scalable and moveable with a stretchy squashy ribbon spline.

Bendy Ribbon Limbs

This tutorial assumes that you have followed AFR and your characters arms are horizontal along the X axis and legs are vertical along the Y axis in a standard biped rig. If applying this bendy setup to different limbs (i.e. Quadrupeds, Insects, Crustaceans) adjustments to the aim constraint settings or overall setup will most likely be required. There will also be mention of certain scripts provided by AFR such as #### rename (hash rename).

1. Go to Create > NURBS Primitives > Plane Option Box. Reset settings and create a plane with these options:
    a. Width: 1
    b. Length: 17
    c. U patches: 1
    d. V patches: 17

2. Scale the plane to roughly the same length as the arm/leg and position it near the arm or leg. The position and scale being exact isn't important because the control joints that move the ribbon plane will be constrained and moved to the joints in the arm/leg.

3. Under the FX menu set (F5) go to nHair > Create Hair Option Box. Reset settings and create a hair system with these options:
    a. Output: NURBS curves
    b. U count: 1
    c. V count: 17

4. Delete hairSystem1, hairSystem1OutputCurves, and nucleus1. Shift expand the hairSystem1Follicles group and under each follicle delete the curve nodes.

5. Rename your ribbon plane, follicle group, and follicles accordingly with left or R and arm or leg. Use #### rename from AFR for the follicles. (e.x. L_arm_ribbon_plane, L_arm_follicle_grp, and L_arm_#_follicle)

6. Create one joint anywhere in the viewport. Duplicate it 16 times so you now have 17 joints.

7. Use #### rename to name them accordingly. (e.x. L_arm_#_skin_jnt)

8. Select the first follicle, then the first joint and under the rigging menu set (F3) go to Constrain > Parent Option Box. Make sure maintain offset is off.

9. Constrain all the joints to the corresponding follicle by selecting the follicle, then joint and hitting G.

10. Select all the joints and group them together. Rename the group accordingly. (e.x. L_arm_ribbon_jnt_grp)

11. Select joints 1, 5, 9, 13, 17 and duplicate. Set EACH JOINT Rotate values to 0. Delete the constraints under the duplicated joints, name them according to where they line up on the limb, and change their rotation orders to ZXY. (e.x. L_arm_shldr_bend_jnt, L_upArm_bend_jnt, L_arm_elbow_bend_jnt, L_lowArm_bend_jnt, and L_arm_hand_bend_jnt)

12. Create a control curve for EACH of these bend joints with the same pivot. Parent the bend joints under this control curve.

13. Create a group node over EACH of the control curves with the same pivot with the suffix _const.

14. Change the control curve and joints' rotation orders to ZXY.

15. Now it's time to make the bend joints control the nurbs planes.

16. Select all of the bend **joints** and then the nurbs plane. Under the rigging menu set (F3) go to Skin > Bind Skin.

17. Select the nurbs plane, hold right click, and go to Paint Skin Weights Tool. You want to paint the skin weights so that the knee/elbow area has an influence of 1 on the CVs around the skin joint in the knee/elbow area and no influence above that. This will keep a sharp angle in rotation that you would normally see without a bendy ribbon.

18. Now that the ribbon plane is set up we want to set up the constraint system to make this plane act like an arm or leg. It's important to pay attention to the bolded **_anim** or **_const** as selecting the wrong ones will change functionality or create dependency loops. The aim constraint settings are different for the arms and legs so I will separate out those steps for each.

19. In the outliner, select L_leg_knee(arm_elbow)_**anim**, L_leg_hip(arm_shldr)_**anim**, then L_upLeg(upArm)_bend_**const**. Go to Constrain > Point Option Box. Make sure maintain offset is **OFF**.

20. In the outliner, select L_leg_knee(arm_elbow)_**anim**, L_leg_ankle(arm_wrist)_**anim**, then L_lowLeg(lowArm)_bend_**const**. Hit G.

21. **ARMS:** In the outliner, select L_arm_elbow_**anim** then L_lowArm_bend_**const**. Go to Constrain > Aim Option Box. Reset Settings and change them as follows:
    - Maintain offset is **OFF**.
    - Aim Vector: -1, 0, 0 (For the **right** side use: 1, 0, 0)
    - Up Vector: 0, 1, 0
    - World up type: Object rotation up
    - World up vector: 0,1 , 0
    - World up object: L_arm_elbow_bend_jnt

LEGS: In the outliner, select L_leg_knee_**anim** then L_lowLeg_bend_**const**. Go to Constrain > Aim
Option Box. Reset Settings and change them as follows:
- Maintain offset is **OFF**.
- Aim Vector: 0, 1, 0
- Up Vector: 1, 0, 0
- World up type: Object rotation up
- World up vector: 1, 0 , 0
- World up object: L_leg_knee_bend_jnt

22. **ARMS:** In the outliner, select L_arm_elbow_**anim** then L_upArm_bend_**const**. Go to Constrain > Aim Option Box.Reset Settings and and change them as follows:
    - Maintain offset is **OFF**.
    - Aim Vector: 1, 0, 0 (For the **right** side use: -1, 0, 0)
    - Up Vector: 0, 1, 0
    - World up type: Object rotation up

- World up vector: 0,1 , 0
- World up object: L_arm_elbow_bend_jnt

LEGS: In the outliner, select L_leg_knee_**anim** then L_upLeg_bend_**const**. Go to Constrain > Aim
Option Box. Reset settings and change them as follows:

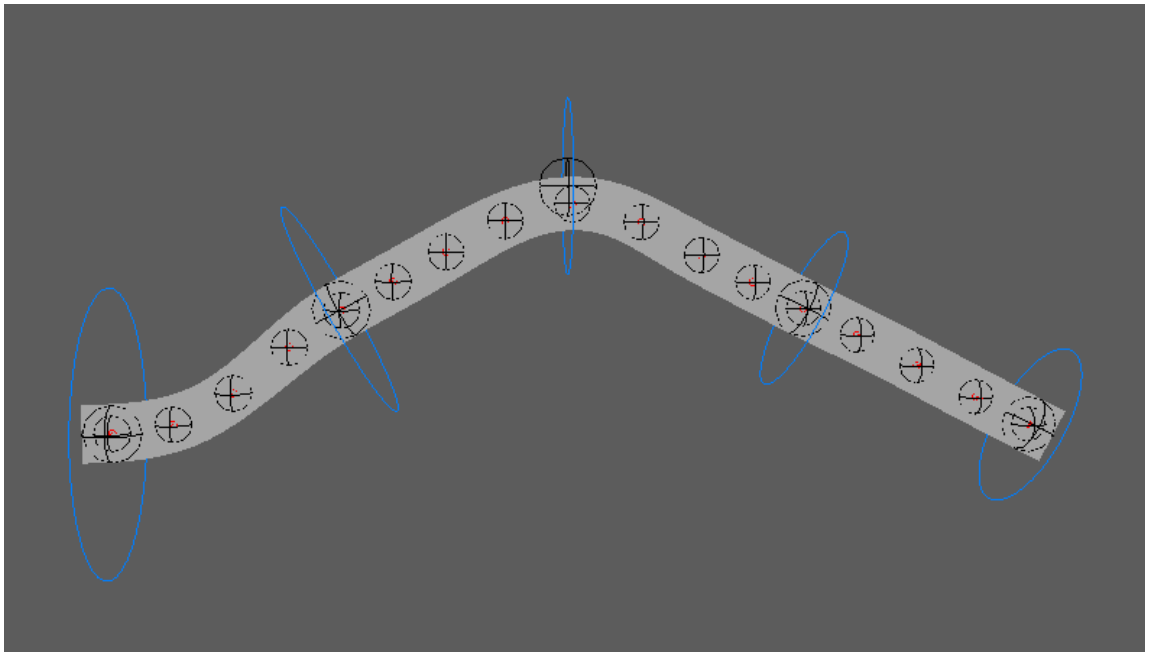
- Maintain offset is **OFF**.
- Aim Vector: 0, -1, 0
- Up Vector: 1, 0, 0
- World up type: Object rotation up
- World up vector: 1, 0 , 0
- World up object: L_leg_knee_bend_jnt

23. **ARMS:** In the outliner, select L_arm_elbow_**anim** then L_arm_wrist_**const**. Go to Constrain > Aim Option Box. Reset settings and and change them as follows:
    - Maintain offset is **OFF**.
    - Aim Vector: -1, 0, 0 (For the **right** side use: 1, 0, 0)
    - Up Vector: 0, 1, 0
    - World up type: Object rotation up
    - World up vector: 0,1, 0
    - World up object: L_arm_elbow_bend_jnt

LEGS: In the outliner, select L_leg_knee_**anim** then L_ankle_bend_**const**. Go to Constrain > Aim
Option Box. Reset settings and change them as follows:

- Maintain offset is **OFF**.
- Aim Vector: 0, 1, 0
- Up Vector: 1, 0, 0
- World up type: Object rotation up
- World up vector: 1, 0 , 0
- World up object: L_leg_knee_bend_jnt

24. Make sure when moving the knee/elbow anim that the other bend controls are following along properly like and arm or leg would and that equal distances are kept between controls. Like this:



25. Now it's time to attach it to the arm/leg.

26. Select your main L_upLeg(Arm) joint (the hip/shldr joint) then L_leg_hip(arm_shldr)_bend_**const** and go to Constrain > Point Option Box. Make sure Maintain Offset is **OFF**.

27. Select your main L_lowLeg(Arm) joint (elbow/knee), then L_knee(elbow)_bend_**const** and go to Constrain > Parent Option Box. Make sure Maintain Offset is **OFF.** If this constraint flips the ribbon at all: delete the constraint, zero out the rotations on the **_const** group, and redo the parent constraint with Maintain Offset **ON.**

28. Repeat step 26 for L_leg_ankle(wrist)**_const. IMPORTANT DIFFERENCE:** For the wrists, change the leader in the point constraint to L(R)_hand_base_const. This is parented under the side and bend locators for the hands.

29. Move around your leg or arm and use some of the switch features to make sure the bend controls are following along correctly. Also check that the aim constraints are rotating the bend controls properly when the character turns around within the all_anim.

30. You'll notice that rotating the wrist in FK and IK mode does not properly twist the forearm as the L(R)_hand_base_const does not actually rotate in the AFR setup. This is purely for the translations of the hand, not the orientation. To remedy this, we need to add an extra group in the hierarchy of the elbow and hand bend controls and make some connections between the X rotations of a few things for the proper twist.

31. In the outliner, select L(R)_lowArm_bend**_anim** and hit Ctrl+G. Rename the group to L(R)_lowArm_bend_**twist_orient**. Make sure the pivot of the new group is the same as L(R)_lowArm_bend**_anim.**

32. In the outliner, select L(R)_arm_hand_bend**_anim** and hit Ctrl+G. Rename the group to L(R)_arm_hand_bend**_twist_orient**. Make sure the pivot of the new group is the same as L(R)_arm_hand_bend**_anim**.

33. In the outliner, select L_lowArm_bend**_twist_orient,** L_arm_hand_bend**_twist_orient**, R_lowArm_bend**_twist_orient**, R_arm_hand_bend**_twist_orient**, L_hand_jnt, and R_hand_jnt.

34. Open the node editor and load the input and output connections of the selected nodes.

35. In the node editor, create a multiplyDivide node and rename it to lowArm_bend_twist_orient_div. Set the operation to Divide and change Input 2X and 2Y to 2.

36. In the node editor, take the Rotate X value from L(R)_hand_jnt and connect it to the Rotate X of L(R)_arm_hand_bend**_twist_orient**.

37. In the node editor, take the Rotate X value from L_arm_hand_bend**_twist_orient** and connect it to Input 1X.

38. In the node editor, take the Rotate X value from R_arm_hand_bend**_twist_orient** and connect it to Input 1Y.

39. In the node editor, take the Output X from lowArm_bend**_twist_orient_div** and connect it the Rotate X of L_lowArm_bend**_twist_orient**.

40. In the node editor, take the Output Y from lowArm_bend**_twist_orient_div** and connect it the Rotate X of R_lowArm_bend_twist_orient.

41. This set of nodes we just made should take the rotate X value from the wrist, apply it to the new group above the wrist bend controller, divide that value in half, and apply it to the new group over the lower arm bend controller for the proper forearm twist. This works under the hierarchy that we set up with constraints to get the ribbon plane to act like a limb, but over the actual controllers so that animation can still be done on them while they work properly.

42. A similar setup is needed for the upper arm as well to mimic the twist of the entire arm from the shoulder since the elbow is only a hinge joint. This setup will flip the shoulder when moving the arm in front of the chest past 90 degrees in either direction but a countermeasure is included called untwist.

43. In the outliner, select L(R)_arm eblbow_bend**_anim** then select L(R)_arm_shldr_bend**_const.** Create an aim constraint with the following settings:
    - Maintain offset is **OFF**.
    - Aim Vector: 1, 0, 0 (-1, 0, 0 for right)
    - Up Vector: 0, 0, 1

- World up type: Object rotation up
- World up vector: 0, 0 , 1
- World up object: L(R)_shoulder_end_jnt (end joint of the clavicle setup)

44. Create a group over the L(R)_upArm_bend_**anim.** Make sure the pivot is the same and name it with the suffix **_rot_grp.**

45. Create a group over L(R)_arm_shldr_bend_**anim.** Make sure te pivot is the same and name it with the suffix **_rot_grp.**

46. Select L(R)_upArm_bend_**rot_grp** and load input and output connections in the node editor.

47. Create a multiplyDivide node and name it L(R)_upArm_twist_div. Change the operation to divide and Input 2X to 2. (You may find you have to use -2 here to have the upArm control rotate the correct direction when twisting.)

48. Connect the Rotate X of L(R)_upArm_jnt (Shoulder joint) to Input 1X of L(R)_upArm_twist_div.

49. Connect Output X of L(R)_upArm_twist_div to the ROtate X of L(R)_upArm_bend_**rot_grp.**

50. This solves the twisting of of the upper arm while the shoulder stays upright, but we still need to add the untwist feature to counter the flipping issue.

51. On your shoulder control (the clavicle controller) create a new attribute that's a float with no limits called Untwist.

52. Connect Untwist to the Rotate X of L_arm_shldr_bend_**rot_grp.** For the right side you will need to create a multiplyDivide node that makes the value negative in between the connection or you can also use Set Driven Key with linear post/pre infinity to control the untwist.

53. The untwist attribute can unflip the shoulder with a value of 180 as well as helps with certain poses if the shoulder should twist a little bit.

54. To incorporate this into the entire rig hierarchy, make sure Inherit Transforms is **OFF** on the ribbon plane and follicle group for each limb.

55. Group all the **_const** groups together and rename it accordingly. This group should be a child of the same group the 17 skin joints are in. (e.x. L_arm_ribbon_jnt_grp)

56. Place the follicle group, ribbon plane, and joint group under the doNotTouch group of the corresponding limb.