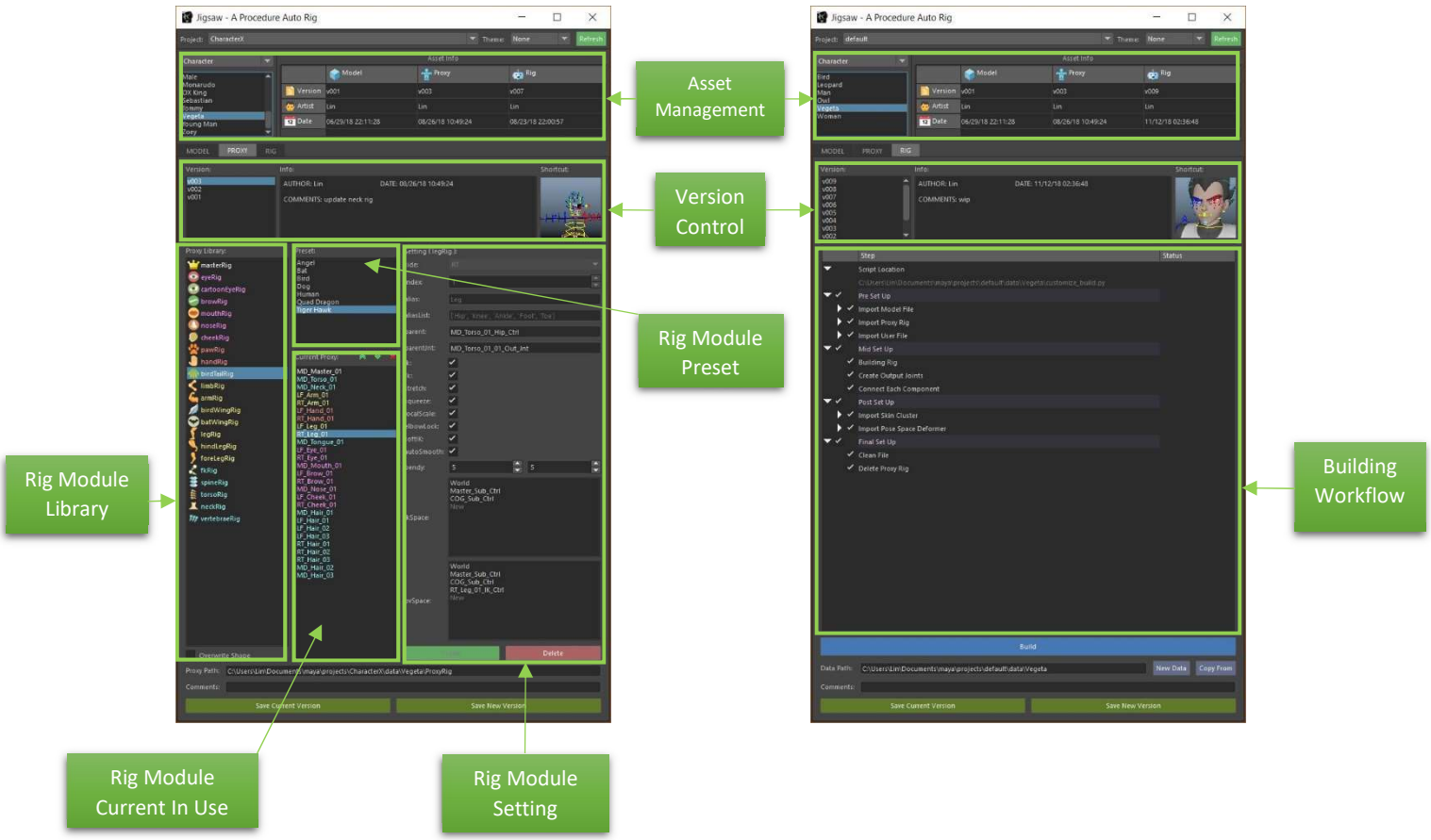
Rig Build Description

-Xiong Lin

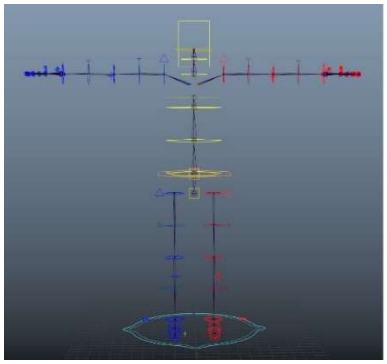
Hello, there! Thank you for looking at my auto rig. In this file I will explain the concept about how I design my rig building workflow.

1

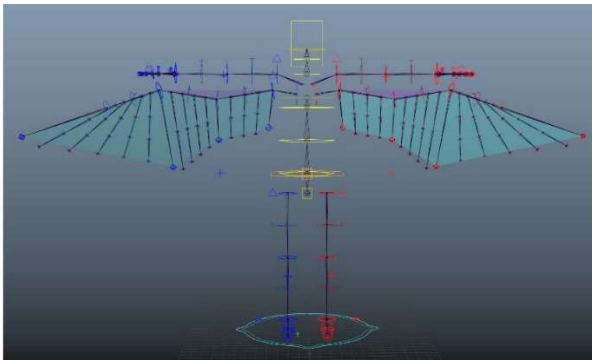
The images below describe the functionalities of my auto rig in the UI part. I use PySide to achieve the overall looking. Basically, my auto rig is module-based procedure auto rig. It also functions as an asset manager.



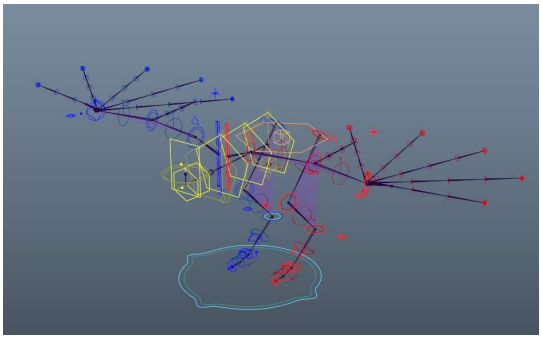
Since it is module based, I call this auto-rig “Jigsaw”. It’s kind like puzzles. I can combine different rig modules together to achieve any type of creatures. For example, I use 2 arm rigs, 2 legs rigs, 2 hand rigs, 1 torso rig and I neck rig to build a basic human character. If I add 2 more wing rigs, I can build an angel character. I have some preselected combination of rig modules, including Angel, Bat, Bird, Dog, Human, Dragon, and a tiger hawk.



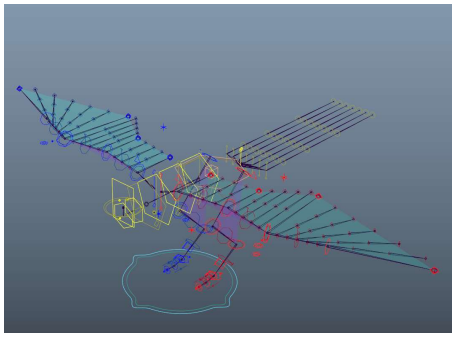
Human template



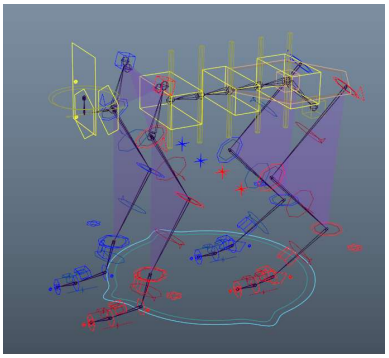
Angele template



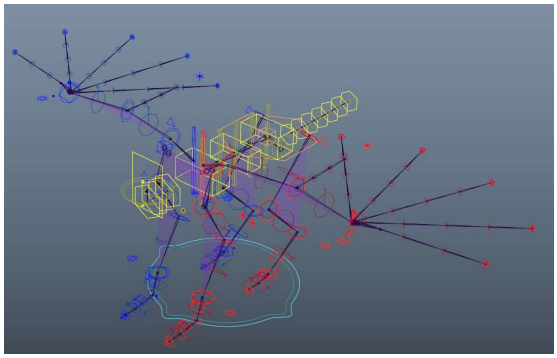
Bat template



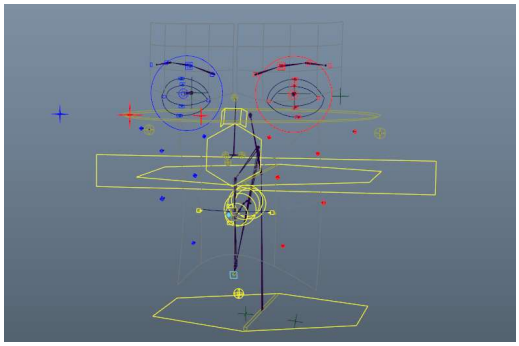
Bird template



Dog template

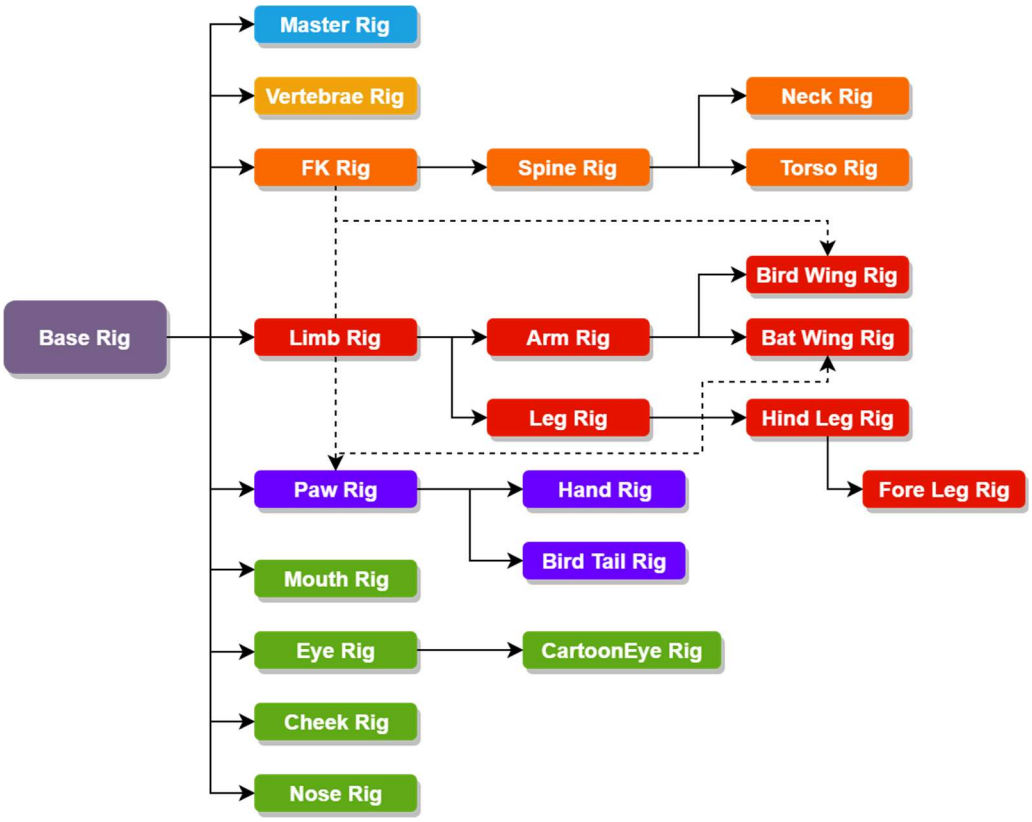


Quad Dragon template



Face template

In total, I have 21 rig modules can be used in production. I use python and object-oriented programming to write all modules. First, I write a basic rig class called “Base Rig”. It only contains some very basic information like naming conventions, connection methods. Then all rest rig modules are inherited from the base rig and further developed to become more complicated rig modules. The image below shows the inheritance relationship of each rig modules.



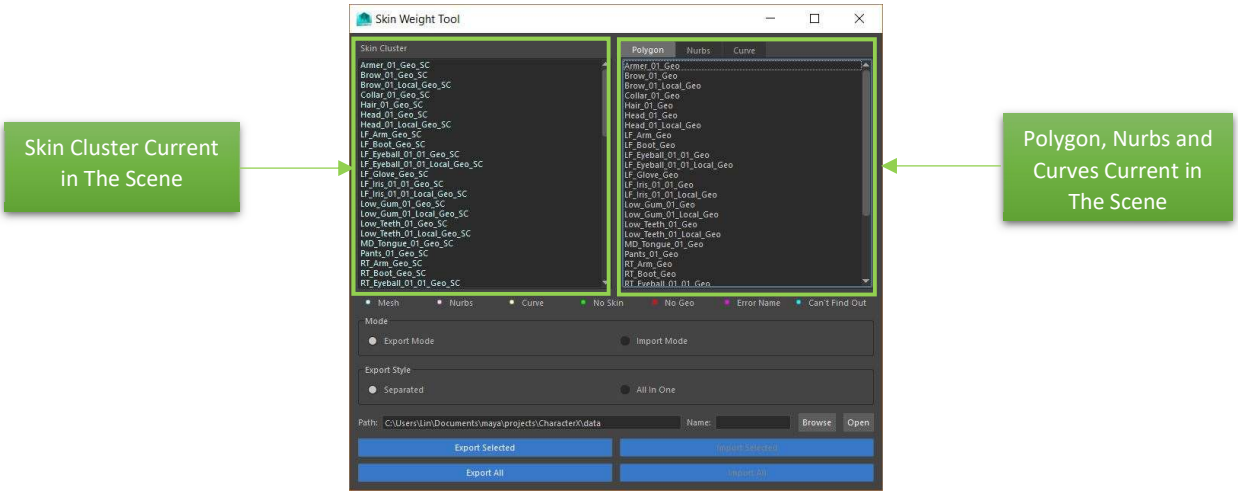
Some of the features I developed for the limb rig:

- IK FK Switch
- IK control auto stretch
- Soft IK
- Elbow offset/lock
- Limb squeeze and squash
- Local scale
- IK control space switch
- FK control rotation space switch
- Limb Length adjust in both FK and IK mode
- Bendy control

All these features will inherit to arm rig, bird wing rig, bat wing rig, leg rig, hind leg rig and fore leg rig.

3

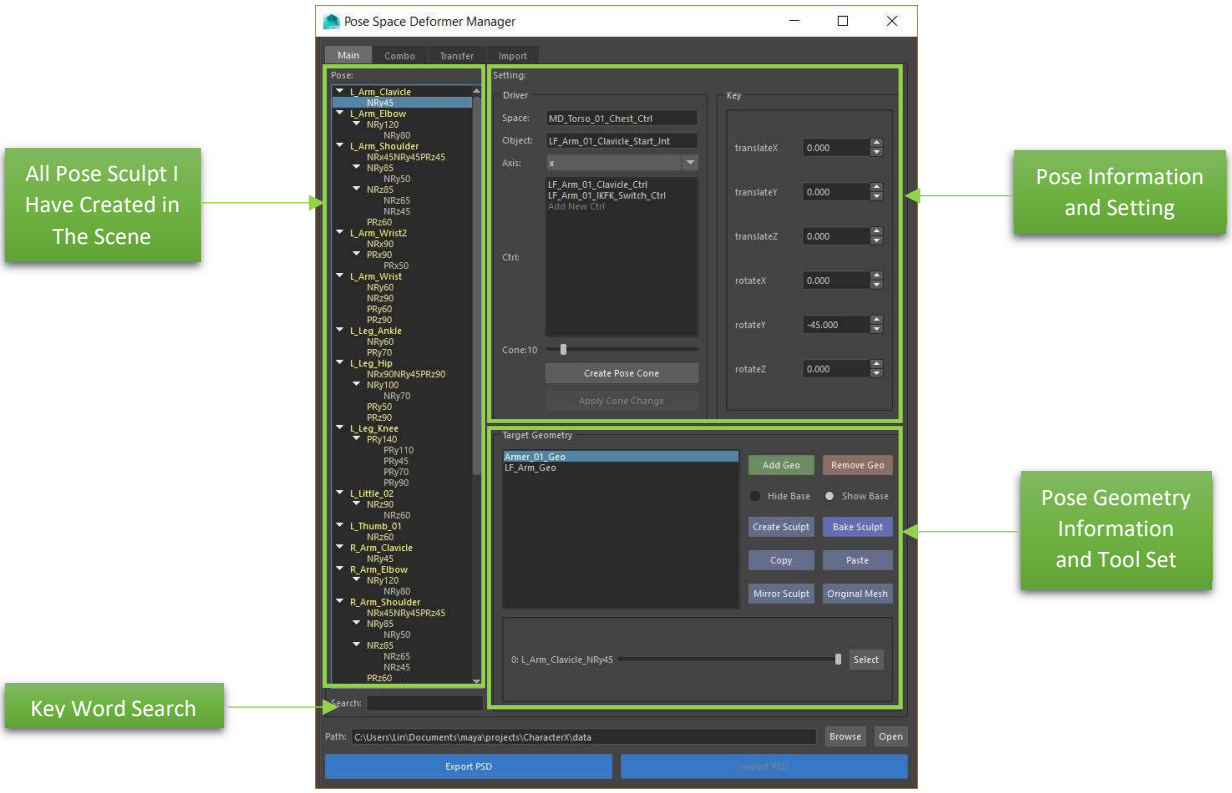
My auto rig is using a procedure workflow, which means I am not only saving the final rig file, I am also saving the progress of building the rig file. In my workflow, I mainly save the skin weights data and pose space deformers.



The image above is the UI of the skin weight tool which I can use to export the skin weighs data. I use Python and PySide to write this tool. My auto rig will automatically pick up the skin data when building the rig.

Features:

1. High speed data transfer
2. Export/Import each skin weight in separate file
3. Export/Import skin weights in one single file
4. Works for polygon, nurbs and curves

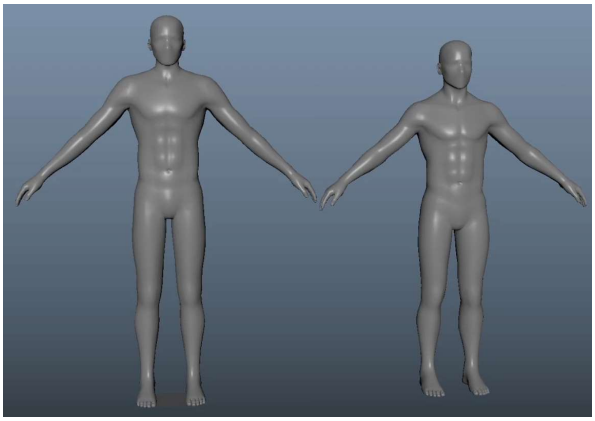


The image above is the UI of the pose space deformer tool which I use to sculpt pose in a better art form and then export my sculpting data. I use Python, PySide and Maya API to write this tool. My auto rig will also automatically pick up these data when building the rig.

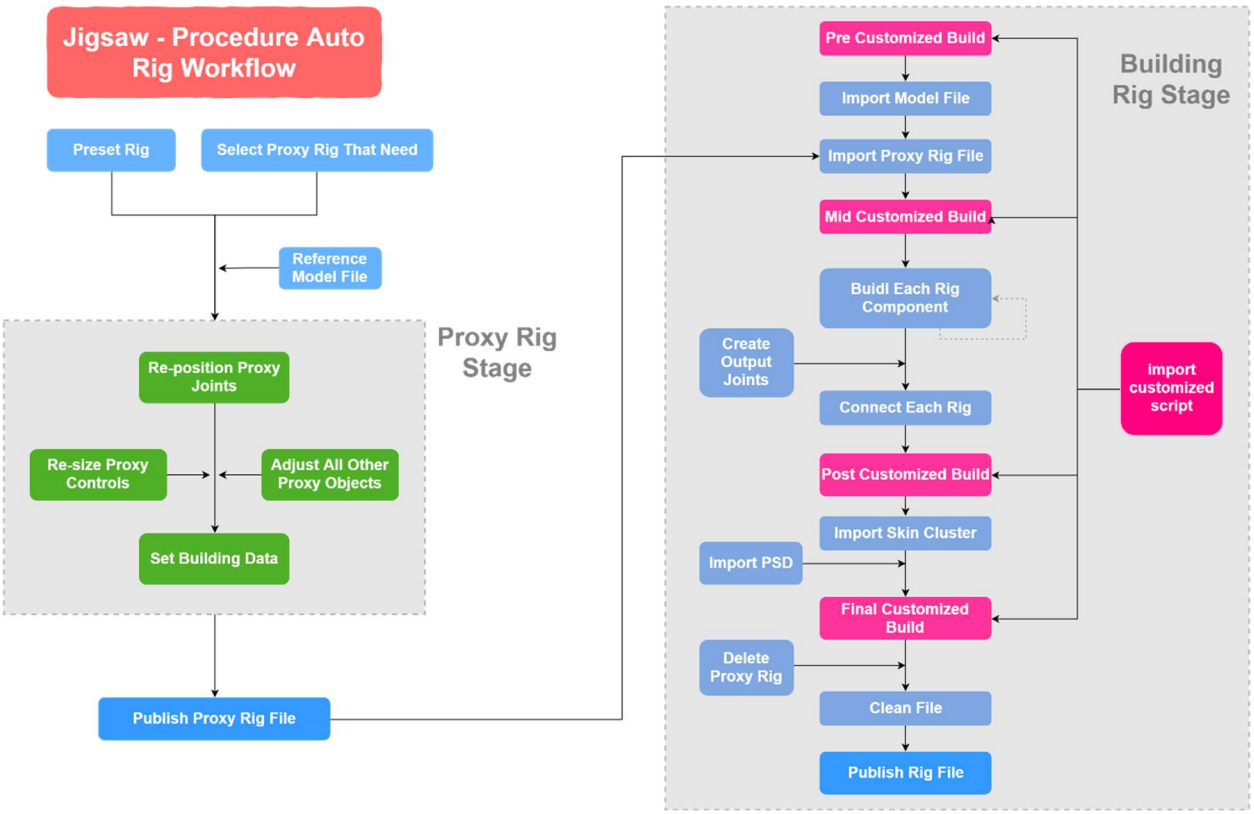
Features:

1. Create pose reader and connect with corrective blendshape
2. Directed sculpt on posed mesh and bake difference
3. Mirror pose space deformer (PSD) from one side to another
4. Pose Management:
    - Easy access to any pose
    - Adjust Pose after PSD is created
    - Adjust PSD influence range
    - Rename PSD
    - Filter PSD in the view list
    - Delete PSD
5. Copy PSD influence from one mesh to another
6. Restore corrective blendshape after target blendshape is deleted
7. Export and Import PSD set up

The biggest advantage of this procedure method is that I can jump back to previous rig building stage to modify or fix the rig without losing the works I have already done. Such as I can reposition the joints and then rebuild the rig with just one click. Besides, if two characters are similar, I can share data between 2 characters like using the same skin weights which saves production a lot of time.



The image below is the entire rig building workflow.



Again, thank for looking at my work!