Notes for a future developer working on the Blender Importer/Exporter

Changes 2.55 → 2.93

export_anm.py

- bpy.context.scene.update() → bpy.context.view_layer.update() (Ref)

- Vector.normalize() → Vector.normalized()
    - normalize() doesn't **return** a normalized vector.
    - normalized() returns a **copy** of the vector normalized

- Matrix multiplication and multiplication between vector and matrices uses " @ " instead of " * "
    - Ex: mat1 * mat2 * mat3 → mat1 @ mat2 @mat3
    - Ex: mat1 * vec1 → mat1 @ vec1

- Matrix.invert() → Matrix.inverted_safe()
    - Inverted() is the canon replacement but some matrices cannot be inverted safely so we use inverted_safe()
    - inverted_safe() returns a **copy** of the matrix inverted.

- Matrix initialization by using a list-of-lists had to be transposed.
    - Using transpose() to transpose the matrix before use

- Matrix.resize4x4() → Matrix.resize_4x4()
    - resize_4x4() does **not** return a resized matrix, it manipulates the inside.
    - So to use a 4x4 resized matrix, you need to do the resize_4x4() first.

- Matrix.translation_part() → Matrix.to_translation()

- Matrix.to_quat() → Matrix.to_quaternion()

- Pose.bone.matrix_local → Pose.bone.matrix_basis

- Bpy.context.scene.objects.active → bpy.context.view_layer.objects.active

- Pose.bone.head → Pose.bone.bone.head_local
- Pose.bone.tail → Pose.bone.bone.head_tail
    - These two above are questionable changes, if any bugs would appear then check these two first. Through testing, they are equivalent when comparing the data in 2.55 and 2.93.

- **Selecting an active object:**
  bpy.context.scene.objects.active = my_obj +
  bpy.context.scene.objects.active.select = True →

  arm_obj.select_set(True)

- Array.array.fromstring → Array.array.frombytes (**fromstring deprecated**)

- ANMExporter operator now includes 'ExportHelper' which helps with receiving file path and other from the window manager

__init__.py

- Importer disabled since it doesn't work both in the previous version and not the current version either.

- Default path disabled: automatic ".anm" is extension is handled by ExportHelper

- Bpy.types.register → bpy.utils.register_class
- bpy.types.unregister → bpy.utils.unregister_class

- bpy.types.INFO_MT_file_import →
  bpy.types.TOPBAR_MT_file_import.remove

- bpy.types.INFO_MT_file_export →
  bpy.types.INFO_MT_file_export

r_rig_props.py

- This was a part of the BlendFile for rabbit_rig and rabbit_rig253 and was removed. They did not execute on the Blender 2.93 version and had no impact on the animations being exported. This file still exists in the old rigs and may be crucial for the Phoenix Bones. So it is still retrievable.

General notes

The update is primarily based on this video which I used as a foundation on to understand *in what way* the community uses the rig in general. I worked with a limited knowledge on Overgrowth, Blender, status of OG's Blender tools and Animation/Rigging in general so the result of the port has primarily been to ensure that data is consistent and piped through the script correctly with respect to the API and Python changes. Personally, I've had to experiment with the Overgrowth editor, read up on the custom file format for the animation, how Blender works, how scripts are integrated and topics related to animation/rigging such as bones and armature so there may be room for improvement in the port.

Below, I will write a few notes that may be useful for future developers that may visit this code.

- The script **manipulates** a **copy** of the rig to be exported during the script.
- If there are any bone dependency cycles detected by Blender (check console), then it most likely has something to do with the ***connector bones*** which are added in *AddConnectorBones*.
    - Dependency cycles **have** to be resolved or this will possibly lead to keyframes lagging behind as the bones evaluation order is undefined.
- A trick to check whether your rig is working correctly on export is to ***not*** delete the **copy** of the rig that is manipulated (should be at the end of *GetAnimation)*
    - This rig will be deformed a bit, but if you do the same thing on the old version then you can compare both versions to see if they are deformed in the same manner.
        - It is not a ***definitive*** way of confirming results, but it's one way of confirming that has been useful to me.
    - This rig will also help you more easily see if you have any keyframes that seem to be lagging behind (for example due to dependency cycles)
- BakeBoneConstraints is a ***recursive function***. It is recommended that you select a **SPECIFIC bone** at a **SPECIFIC time** that you follow as the data transforms through the function.
- GetBoneMatrixRotation is also a recursive function. Same applies here as above.
- If you want to backtrack data, the easiest first place to do it is at the end of GetAnimation where data is being appended to a helper structure (keyframes). Comparing the data in the new version vs the old version will give you a hint on *where* the incorrect data is.
    - KeyframeFromPose is the next-easiest place where you can evaluate the data from GetBoneMatrixRotation/Translation.
        - If data is wrong, then you will have to look into BakeBoneConstraints.
- It is recommended that you test a script through the Blender scripting interface and NOT through an addon button. You can then rapidly test new changes on the script as you work with it.

You can diff the old version and the new version of the exporter_anm.py to see exactly the changes that were made if you do find a bug or see that something needs an extension. It will then be clear what comes from the old version and is new for the port.