Updated: 22-04-2021

Max Boy

User Manual

MaxBoy is a “tool suite” for 3ds Max that focuses on speeding up general workflows when working with games. The suite so far contains two main tools; "ToolBoy" and "PipeBoy". Each having their own area of usability, purpose and functions.

ToolBoy is a collection of sub-tools, and other neatly packaged macro functions that tries to help with general asset creation. It's meant as an "assisting tool" that you can leave floating around as you work, fully collapsed and waiting or expanded to your liking.

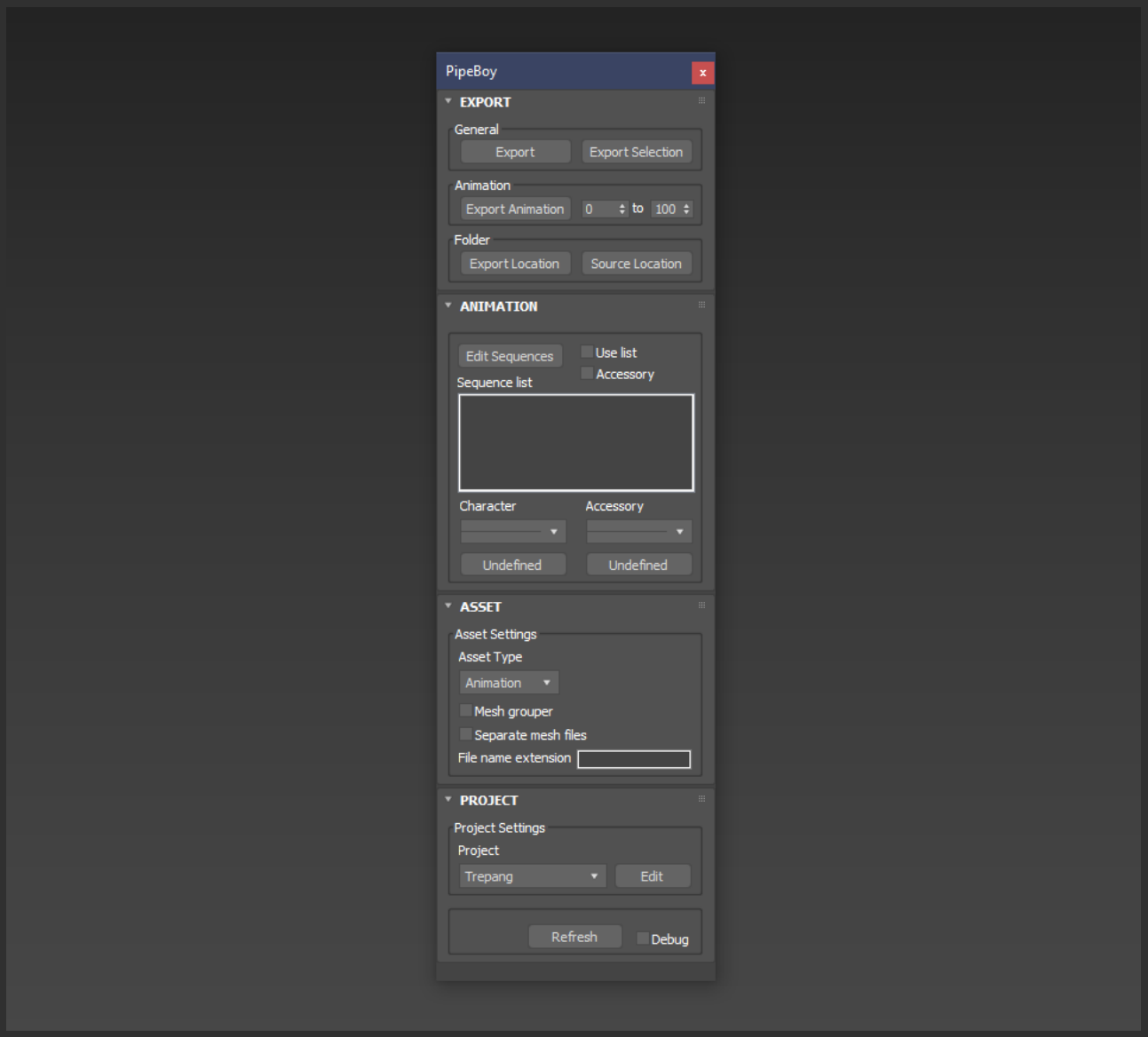
PipeBoy is a pipeline tool, made to assist with iteration times, and keeping your projects neatly structured- by following a standard "directory mirroring" structure (if you want). It has a project manager attached to it to manage and store settings, specific to your projects.

If you have requests, issues you need help with or other feedback, you can contact me directly at jopter@outlook.com

Jacob Andersson - Jopter Interactive 2021

Updated: 22-04-2021

PipeBoy - Pipeline tool



Version from (22-04-2021)

PipeBoy's main function is handling the export process, and general project structure- sort of like a bridge from Max to your project. It has functionality that allows you to work in less destructive ways when working with geometry, and features some "smart functionality" when working with animation.

To simplify the manual structure, I'll go from top to bottom in order of appearance in the tool. To start off, we have the most critical elements, the actual export buttons. But first, I'm gonna have to go into detail how the directory mirroring works, and why it's useful to have this automated.

[Directory mirroring]

The main reason you should mirror your directories is plain and simply <good organization>. Knowing where your files are, and keeping a solid and consistent structure helps out like crazy in the long run. I've spent so much wasted time by digging through crazy nested folders, trying to find where the hell I exported that one asset to and vice-versa, which is completely unnecessary in the first place. The other major reason to do this is automation, which is where PipeBoy comes into the picture. When you have a consistent structure, you can apply rules and processes that follow those rules, enabling you to do some cool stuff, while also staying organized.

The simplest way to explain how this works, is something like this:

Source folder: "C:\GameEngine\Source\Assets\AssetName\YourAsset.source" **<- Export from this**
Project folder: "C:\GameEngine\Game\Assets\AssetName\YourAsset.file" **<- To this**

You know exactly where to look for the corresponding asset files as soon as you find either or.

Another note to this is, when using subversion (Like TortoiseSVN or Perforce)- it really helps managing and maintaining your files when you keep source and project separated. It also allows you to be more selective with how much source content you want to pull out locally on your computer.

[Export buttons]

The buttons "Export" and "Export Selection" are pretty straight forward and self-explanatory. Export selects all the scene content and exports it, and Export selection only exports current selection.

The "Export Animation" button isolates and exports *only* your selected bone structures, using a prefix-based selection system. What this does is cutting out the whole process of isolating and selecting the bone structure each time you want to export animation data. This combined with the sequence list, can really improve iteration times if you're doing a lot of batch exporting, and makes the process much simpler in general.

Tip: You can edit your "preferred prefixes" in the Project Settings. These strings are used for quickly and automatically acquiring your main skeleton structures in the scene!

Updated: 22-04-2021

[Asset Settings]

The Asset settings lets you define what type of asset you're currently working on, and how it should be handled in the exporting process.

[Asset Type]

Setting the "Asset Type" defines what category of asset it is. As an example, if you define it as type "Animation", it will only let you export the asset as such. This is to avoid complications and errors in the export process and how it assumes what to do with your file.

[Mesh Grouper]

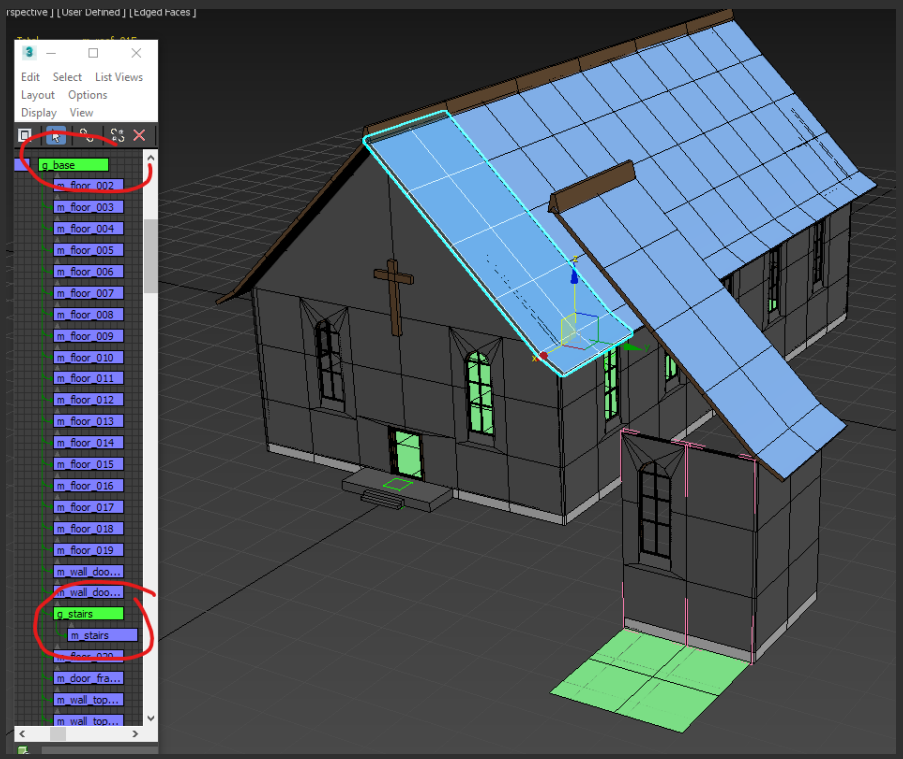
The "Mesh grouper" setting enables a powerful function that lets you work in a "modular" fashion within max that then groups the meshes upon export. This also works with instanced meshes, and I recommend using that if you're building something with a modular set. This feature is not necessary when working with Unreal Engine, as it already features grouping on asset import.

The way it works is that it finds all objects within the scene with a prefix *(default: "g_")* attached in front of the name. Then it goes through each and every group object, and groups the meshes attached to it into a new instance that is named and placed at the same point in the hierarchy as the group object.

This can greatly reduce draw calls within game engines such as Unity where the CPU easily bottlenecks from scene drawing.

Tip: You can nest group objects to split off different areas of the mesh (like an exterior vs interior), that you want to keep within the same instance. Doing this can assist heavily in light baking and culling.

In this image you can see it being used when building a structure using a modular set.



[Separate Mesh Files]

This adds the supplied string after the exported file's name. If the file or sequence is called "a_animation_01" and the supplied string is "_test", the resulting filename will be; "a_animation_01_test". This can be useful when quickly testing something different and comparing it to your originally exported file.

[Animation]

The animation tab is where you can add and define animation sequences and specify your skeletons for export.

[Use List]

This setting enables/ disables the use of the sequence list in this tab. If it's enabled, all of the selected sequences in the list will be exported into their respective files. If it's disabled, only the selected range next to the 'Export Animation' button will be exported, and the exported filename will be based on the scene's filename.

[Accessory]

This setting enables/ disables the inclusion of the selected accessory skeleton for export. Note that if you're using the sequence list, you have to enable Accessory on each sequence you want it included for.

[Sequence List]

The sequence list displays all defined sequences for the current file.

***Note:* You won't be able to use this feature unless the scene has been saved first!**
***Note:* If you rename the file after sequences have already been defined for it, you'll need to go to your local Max folder under AppData.**

This is usually located at:* *"C:\Users\UserName\AppData\Local\Autodesk\3dsMax\2020 - 64bit\ENU\scripts\MaxBoy\AnimSetup"

Under the "AnimSetup" folder, you'll be able to find a file named as your files's previous name. All you need to do is renaming it to your new filename, and PipeBoy will be able to find it yet again!

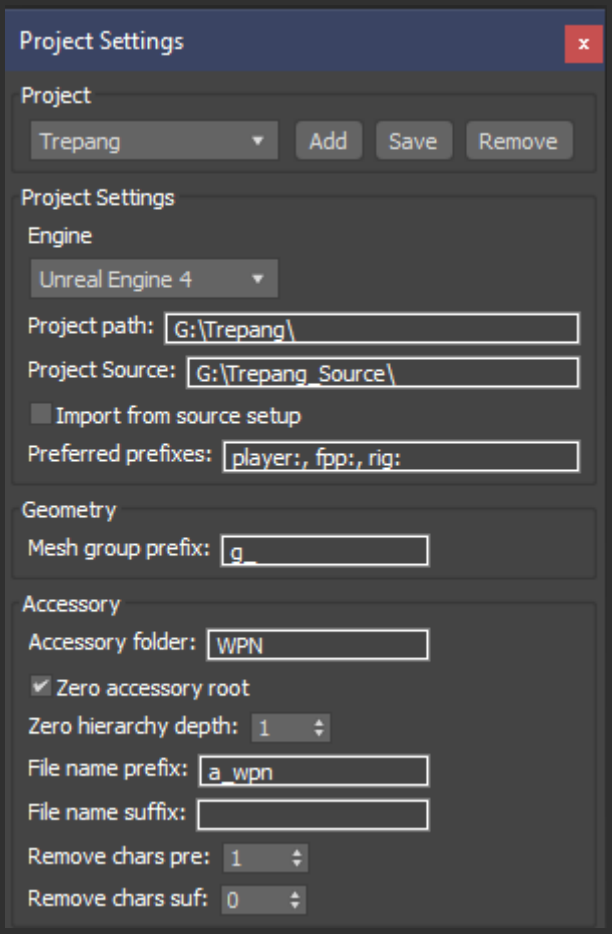
[Character & Accessory]

These settings target the skeleton structures used when exporting your scene's animations. The prefix solution overrides the pick button options, but if no prefix is provided or selected, the pickbutton will be used.

Updated: 22-04-2021

Project Settings - Project settings for PipeBoy

[Editing project settings]



This is a vital step to getting set up. In order to export your files, you need to define where to export the files to, and from where. Most of the settings within this window is self-explanatory, but there are a few things that need some explanation.

At the top, you've got the current project you're editing. You can add and remove projects from here as well. Make sure to save your changed settings before exiting this dialogue!

Depending on what engine you choose for the export, different operations will be applied when exporting your assets. For example, when exporting to Unity, an axis conversion operation will be used to avoid up-axis flipping (this will get expanded upon). There's not a big difference so far depending on which setting you use here, but that's the major one.

Updated: 22-04-2021

Defining your project and source paths, is the most important step here. Insert each path of the **root directory** for each of these, like this:

Source folder: “C:\P\A\T\H\Source\”
Project folder: “C:\P\A\T\H\Game\”

Once this is done, you’re pretty much set to go, but make sure the “Asset folder” corresponds with your current setup. For example, a Unity project has a “Game\Assets\” folder, while Unreal 4 has a “Game\Content\” folder. This means that the directories have to reflect this as well, like this:

Unity:

Source folder: “C:\P\A\T\H\Source\Assets\”
Project folder: “C:\P\A\T\H\Game\Assets\”

Or...

Unreal:

Source folder: “C:\P\A\T\H\Source\Content\”
Project folder: “C:\P\A\T\H\Game\Content\”

This of course being if you’re following that same structure.

“Import from source setup” will cause the exporter to export the files in the source location, rather than to within the project structure. This setting is best suited for unreal, since it’s more or less based to work with this setup and keep the “.uassets” with metadata internally.

“Preferred Prefixes” is a parameter that feeds the “smart algorithms” with your main skeletons “preferred” prefix(es). This function tries to automatically find your skeleton structures in the scene using the provided set of prefixes with the priority being ordered left to right. In the provided example picture above, “player:” would be the prioritized prefix to find. If you don’t want to use this feature, just leave the field empty!

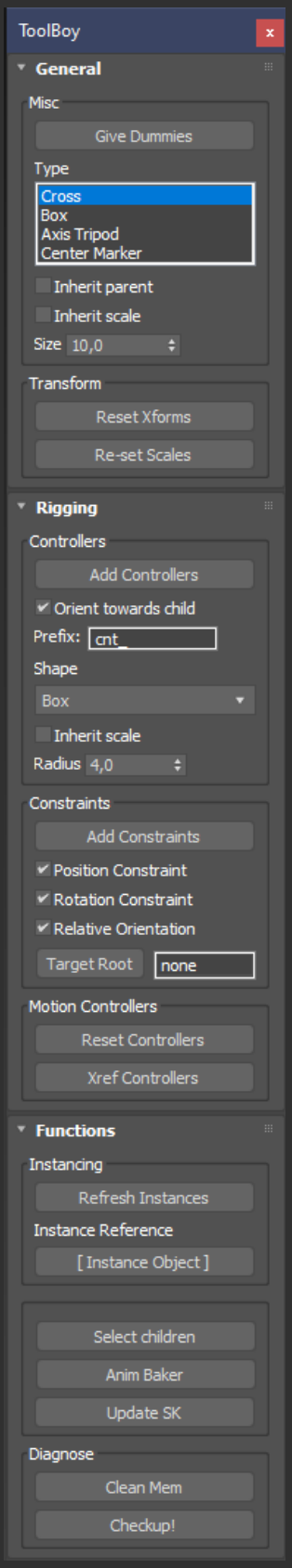
The “Accessory Folder” dictates the name of the export folder for your accessories. This folder gets created automatically inside of your main character’s export folder.

“Zero accessory root” enables a function that removes any motion for the first X amount of nodes in its hierarchy, dictated by the “Zero hierarchy depth” parameter. This happens upon export.

The rest of the settings should be straight forward!

Updated: 22-04-2021

ToolBoy - Macro toolbox



... or

Jacob Andersson - Jopter Interactive 2021

Updated: 22-04-2021

This is a tool I've progressively put together from years of working in max and figuring out what kinds of repeated actions I've been doing the most that could easily be simplified into a macro. The tool focuses on speeding up the asset creation process and assisting with general work. It features tools that assist with miscellaneous tasks like quickly placing out dummies on collections of nodes, rigging macros like "smart controller creation" and "proximity-based constraining", simple transform commands that end up saving lots of time, and more.

There is a lot to cover here, so bear with me- I'll try and be as descriptive, but simple as possible.

[Give dummies]

Give dummies is a quick command for placing dummy/ point nodes on a selected node(s). The settings underneath will be used to create the dummies. Note that you can select multiple items in the "Type" list.

Inherit parent will attempt to recreate the hierarchy in the selected node structure when making the dummies.

Inherit scale will scale the dummy object to be somewhat scaled in relation to the node's scale transform/ bounding box. If you don't use this setting, the "Size" parameter will be used instead.

[Transform Commands]

"Reset Xforms" will reset the node(s) xform, collapse the stack, convert to Editable_Poly, and then re-apply the same pivot as it had before the operation. Pretty neat and a "safer" operation than the regular x-form reset.

"Reset Scales" resets the selected transform's scales. It works with hierarchies too, by temporarily unparenting the nodes, resetting, and then parenting them back after operation is done.

Updated: 22-04-2021

[Rigging]

[Controllers]

“Add Rig Controller” is an automatic controller generator that will create, place and name controllers onto selected bones/ nodes. This can greatly speed up the process of making all the different controllers for a rig, by reducing the process to a single click.

“Orient towards child” will make the controllers point towards the next child in the hierarchy rather than use the bone’s orientation. At the end of the hierarchy, the bone orientation will be used instead.

“Shape” lets you decide what shape the controllers should be. So far there’s only “Box” and “Circle”, but I might add a “custom” setting there in the future.

“Inherit scale” will base the controller’s scale on the bone’s bounding box scale to somewhat match the scales. If this setting is turned off, the “Radius” parameter will be used instead.

[Constraints]

“Add constraints” is a proximity-based constraining tool used to quickly set up a bunch of constraints in a rig. You can choose to do both orientation, and position constraints, or just choose either or.

The “Relative Orientation” will enable the “keep initial offset” setting in the orientation constraints.

“Target Root” is a <pickbutton>, which lets you select a “hierarchy root” node in your scene, that then is used to find all the bones in your rig. Once a node is selected, it will show up in the box next to the button.

[Motion Controllers]

“Re-set Controllers” will re-set the motion controllers of selected node(s), and turn them into the “standard” motion controllers, wiping all current alterations. Pretty useful when re-scaling or doing bigger actions on rigs.

“Xref Controllers” turns the selected node(s) motion controller into an Xref motion controller. A very infrequently used, borderline useless function. . .

Updated: 22-04-2021

[Mesh]

[Instancing]

“Refresh Instances” will take the node selected underneath, using the “Instance Object” pickbutton- and replace the selected node(s) with instances of this “Instance Object”. Really useful if you’ve made changes that didn’t get updated in older instances, or plainly replace copies, and make them instances instead.

[Select Children]

Selects the children in the currently selected node’s hierarchy. This can be natively done by pressing “Shift-PageDown”, but will only select visible nodes!

[Anim Baker]

This sub-tool provides some quick and simple animation baking functionality, as well as options to reverse and bake down current animation controllers into fresh new ones.

[Update SK]

This sub-tool provides a somewhat rough way of replacing a rigged skeleton structure + mesh with a newly imported one. Select root nodes from old & new skeleton structures, and then also select the old meshes to replace. One frequent use case when making this tool was importing newly edited, skinned weapon meshes into several animation files that needed updating.

That’s it!

That’s all the toolset has to offer at this point! I’ll be updating and improving the usability and featureset- most likely for as long as I end up using max. Again, if you have requests, questions, feedback or other input- please feel free to reach out to me personally!

Thank you!

Discord: Kopter#1138
Email: jopter@outlook.com