KtoA 2.3.1.0

Release Date

May 22, 2019

This release includes the beta version of Arnold GPU

> ⚠ **Important information about Arnold GPU (beta)**
>
> - Check the system requirements before using Arnold GPU. If you don't have a supported card or the required drivers, GPU rendering will not work.
> - Review the list of supported features and known limitations before you start using Arnold GPU.
> - If you have any technical problems, questions, or feedback on Arnold GPU, use the beta forum on Arnold Answers

Download and Installation

Arnold, KtoA, and other downloads are available here. Installation instructions come with KtoA, but can also be viewed here: Installation.

Compatibility

- **Arnold:** 5.3.1
- **Katana:** 3.0v1+, 3.1v1+
- **Platforms:**
  - Linux: x86-64, RHEL 6+ or compatible glibc
  - Windows: 7+ on x86-64, with VC++ 2015 redistributable installed
- **GPU (beta)**: see here for detailed information. Required NVIDIA drivers:
  - **Linux:** 418.56 or higher
  - **Windows:** 419.67 or higher

Enhancements

- **Update to Arnold 5.3.1**: Updated to Arnold 5.3.1. See the release notes for more information. (#288)
- **Node creation menus and hotkeys**: Two new pop-up menus have been added for easily creating Arnold nodes in the node editor (default hotkey shift+Z) and Arnold shading nodes (default hotkey alt+Z). You can enable/disable the menus, change hotkeys, and even menu colors with the environment variables `KTOA_DISABLE_NODE_MENU`, `KTOA_NODE_MENU_HOTKEY`, `KTOA_DISABLE_SHADER_MENU`, `KTOA_SHADER_MENU_HOTKEY`, `KTOA_SHADER_COLOR_BUILTIN_GPU`, `KTOA_SHADER_COLOR_BUILTIN_CPU`, and `KTOA_SHADER_COLOR_THIRD_PARTY`. See the README file for more information on the environment variables. (#377)
- **Metadata to customize shader parameters**: Placing a file named `ktoa.mtd` next to your custom shaders allows you to customize a few aspects of shader parameters. For example, you can construct a proper Katana ramp widget, or you can change the default value of a shader parameter, or even hide a shader parameter. For examples of syntax, please see the file `ktoa.mtd` in the `RenderPlugins` directory in your KtoA installation. (#86)
- **Arbitrary data handling**: Arbitrary data translation has improved: (#378)
  - Locations of type `curves` now accept uniform (`face` scope) and varying (`point` scope) arbitrary data properly.
  - All other locations properly check their arbitrary data amounts to make sure they fit within the limits of the underlying primitive. Instanced geometry will also check against the original source geometry. If you previously set invalid arbitrary data, it will now issue warning telling you so and not pass it along to Arnold.
  - `Texture` arbitrary data is accepted as an alias of `st`, as many Katana primitives now set `Texture` data in addition to or instead of `st`.
  - Arbitrary data translation performance has improved substantially and reduces time to prepare the scene for rendering when there is significant amounts of arbitrary data.
- **MaterialX export**: A new node `ArnoldMaterialXBake` will allow exporting a selection of locations with their corresponding materials, labeled with a MaterialX look. This can be brought back in later with an Arnold operator (`arnoldOperator` shader slot, applied to procedurals or to the globals) to apply the look. (#383)
- **Arnold operator export**: A new node `ArnoldOperatorBake` will allow exporting any operators attached to procedurals or to the global options (scene-level operators) to a .ass file. (#385)

- **OSL inline shader node**: The shader `osl` has been implemented, so you can dynamically change the code of an OSL shader and have the parameters update in Katana's UI. Currently known limitations: (#133)
  - Only `int`, `float`, `point`, `vector`, `normal`, `color` and `string` parameters are supported.
  - Default values are always zero for parameters.
  - Widgets are not set for `color` parameters.
  - Only one output parameter is supported.
  - No array or structured input or output parameters.
  - Closures are not supported.
  - Only a single shader can be defined on each node.
  - Including other header files are not supported.
  - Global variables (u, v, etc.) are not supported as parameter defaults.

We plan on removing most of these limitations in future releases.

OSL inline shader workflow:

After editing the code, either externally or in the code editor, the `Compile` button needs to be pressed to generate the parameters. In case of a compilation error, no parameters will be added to the node, and a `compilation_errors` text field will contain the errors from the OSL compiler.

The code and the parameters can be compiled using python, for example:

```
osl = NodegraphAPI.GetNode('myNode')
from ktoa import recompileOSL
recompileOSL(osl)

errors = osl.getParameter('parameters.compilation_errors')
if errors is None:
    # successful compilation
    pass
else:
    print('Errors: \n%s' % errors.getValue(0.0))
```

Bug Fixes

- #379 Update AA_samples_max default to 20 like the core
- #382 Update renderer info and update script versions to 5.3.x.x
- #389 Katana 3.1 live rendering won't show all rendered pixels