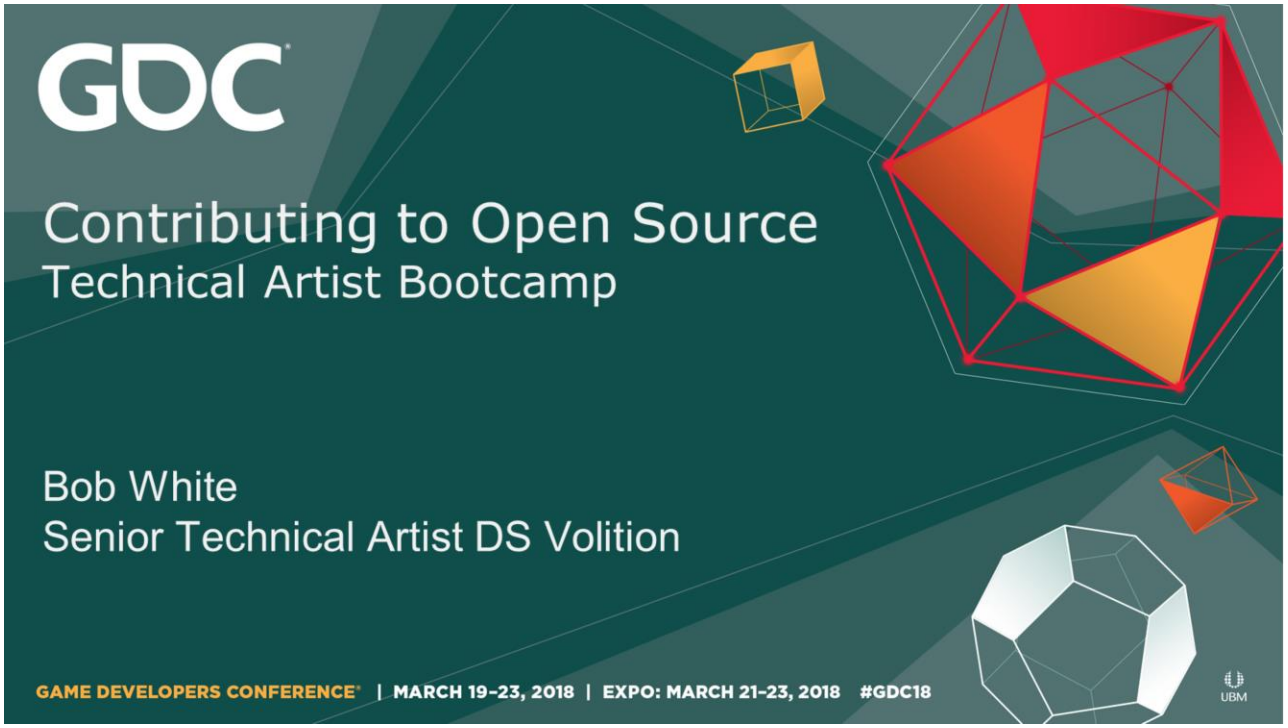

GDC
Contributing to Open Source
Technical Artist Bootcamp
Bob White
Senior Technical Artist DS Volition
GAME DEVELOPERS CONFERENCE® | MARCH 19-23, 2018 | EXPO: MARCH 21-23, 2018 #GDC18
UBM

Quick intro of myself, and a rundown of the talk.

Bob white, Senior Technical Artist at Deep Silver Volition.

This is not a tutorial, as there isn't nearly enough time to walk through a big how-to on the mechanics of contributions.

Also these slides will not only be posted elsewhere, they are also more headings / talking points.

So you really shouldn't feel the need to stress yourself out with any kind of major note taking.

Instead this is more focused on the experience, skills, and other random intangibles that I've gotten from contributing.

Or more simply put. How this helped be a better Technical Artist.

With the overall goal being, that hopefully one of those positives will resonate with you, and at least get you to consider volunteering some of your time and skill towards making these community projects better for everyone.

But before I begin

The major projects I've had the opportunity to contribute to.

This is more here to give them a shout-out specifically because they've let me contribute.

Plus they are awesome projects and just deserve some general props.

I’m pretty sure just about everyone who has worked with Maya has at least heard about PyMEL, but for those that haven’t, it is a library that is bundled with Maya that allows you to write more ‘pythonic’ code when working with Maya.

wxPython is a destop GUI framework, similar to PyQt / PySide.

mGui is the project I’ve put the most effort into these last few years, and is an excellent little library that aims to make writing GUIs in Maya a whole let less painful.

So there are a myriad ways that people can contribute to projects, and this is by no means an exhaustive list.

One of the simplest is reporting any bugs you find, even if you don't know how to fix them, just letting the project know there is an issue helps.

In fact probably half the pymel fixes I've done over the years have come

from things I found in the issue tracker.

So I’ve got a couple of examples of these, just to give some context on how simple they can actually be.

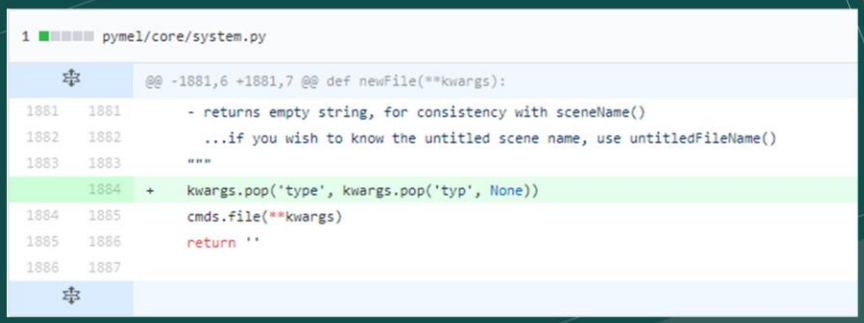
Turns out new features can be rather simple, even just adding a single function.

This was a request I bumped into on the TAO forums, after checking if the functionality was already in pymel (which it wasn't) I went ahead and added it as it was a pretty quick tweak.

I believe you can only use this one if

you’re in 2018 or higher, or backport it yourself.

Fixing Bugs:



1 pymel/core/system.py

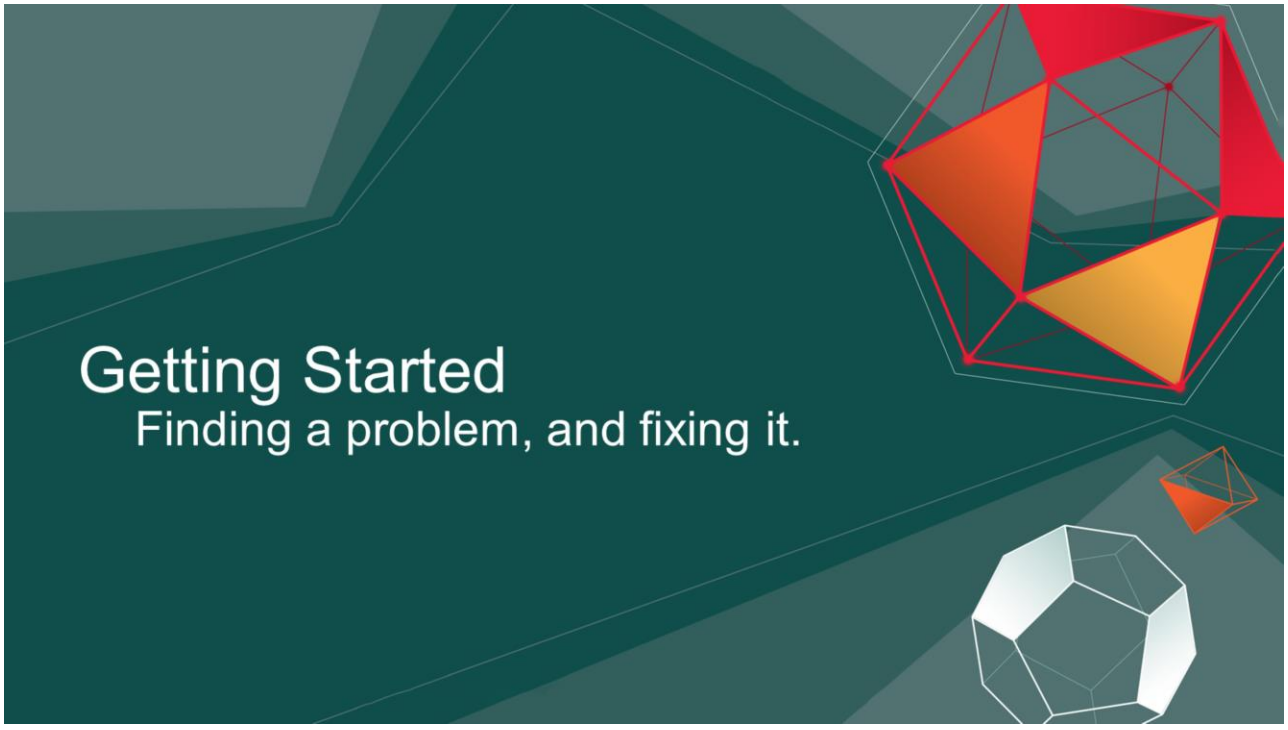
```
@@ -1881,6 +1881,7 @@ def newFile(**kwargs):
    - returns empty string, for consistency with sceneName()
      ...if you wish to know the untitled scene name, use untitledFileName()
    """
+   kwargs.pop('type', kwargs.pop('typ', None))
    cmds.file(**kwargs)
    return ''
```

This was one of those instances where I had some spare time and wanted to write some code.

So I started trolling around in the issue tracker, and realized it was a pretty simple fix.

So this example, is a bit of lie. This isn't a test I just wrote to help out, this was a test that was part of a larger feature.

And really this is just a small part of it, I couldn't fit it on the slide.



So how did this all start?

The same way almost all of my TA related projects seem to start. I ran into a problem, and decided to fix it.

And like many such projects, this caused more problems, with sometimes rather cumbersome fixes.

This led to me maintaining an internal version of pymel with a

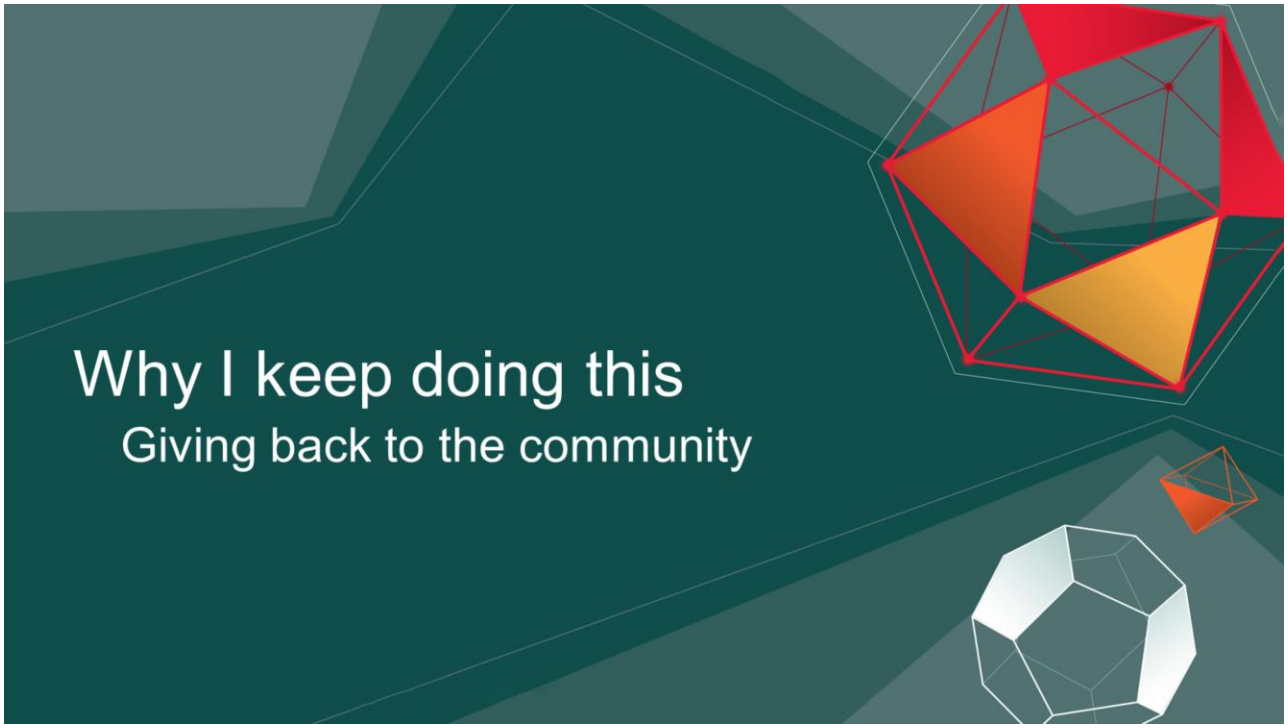
couple of fixes and additions sprinkled throughout.

Which turned out to make version upgrades all the more annoying, as I'd have to merge back in all of my fixes.

So new problem, how do I fix it? By submitting the fixes back to the master project.

But nope, not allowed to share work related code, but at least I wasn't limited on what I could work on at home.

So that became my pattern, hacking on projects at home, and when those fixes were available, use them at work.



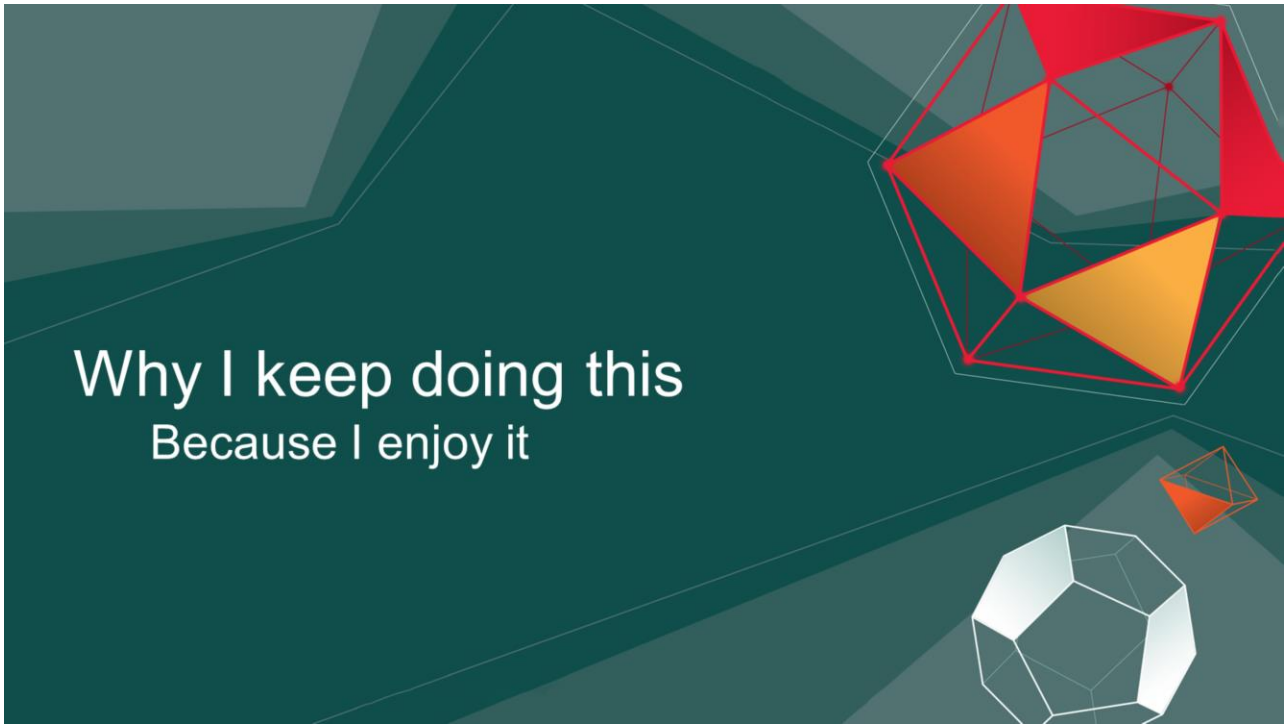
So one of the big reasons I've kept doing this work, is because I like to hope that I've helped others other there using these libraries.

Even more so when it comes to pymel and mGui, because with those, I like to think I've helped other TAs.

Another part is a bit more selfish, because see I'm also a part of these communities, and by getting these

contributions into the main codebase, I can continue to use them no matter where I end up working. You know, unless they ban the use of these libraries, which I guess could technically happen.

But even more than that, I appreciate that others have done a bunch of work that I take advantage of, and it just felt right to do the same when I had both the time and the ability to do so.

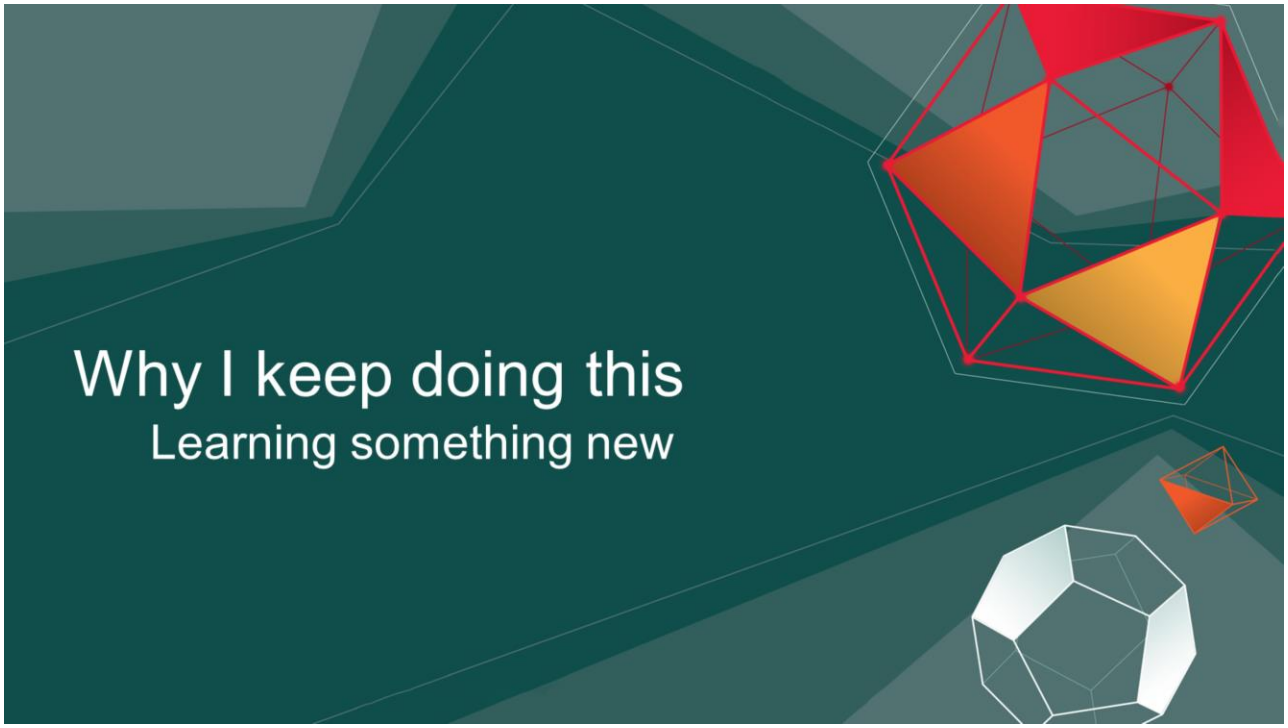


Another big reason is because well I enjoy writing code, I enjoy solving problems, and as previously mentioned I enjoy helping others.

Which is probably why I ended up as a Technical Artist.

I’m not ashamed to admit that as a big nerd I’ve poked around in issue trackers looking to fix bugs in these projects, just for something fun to

do.



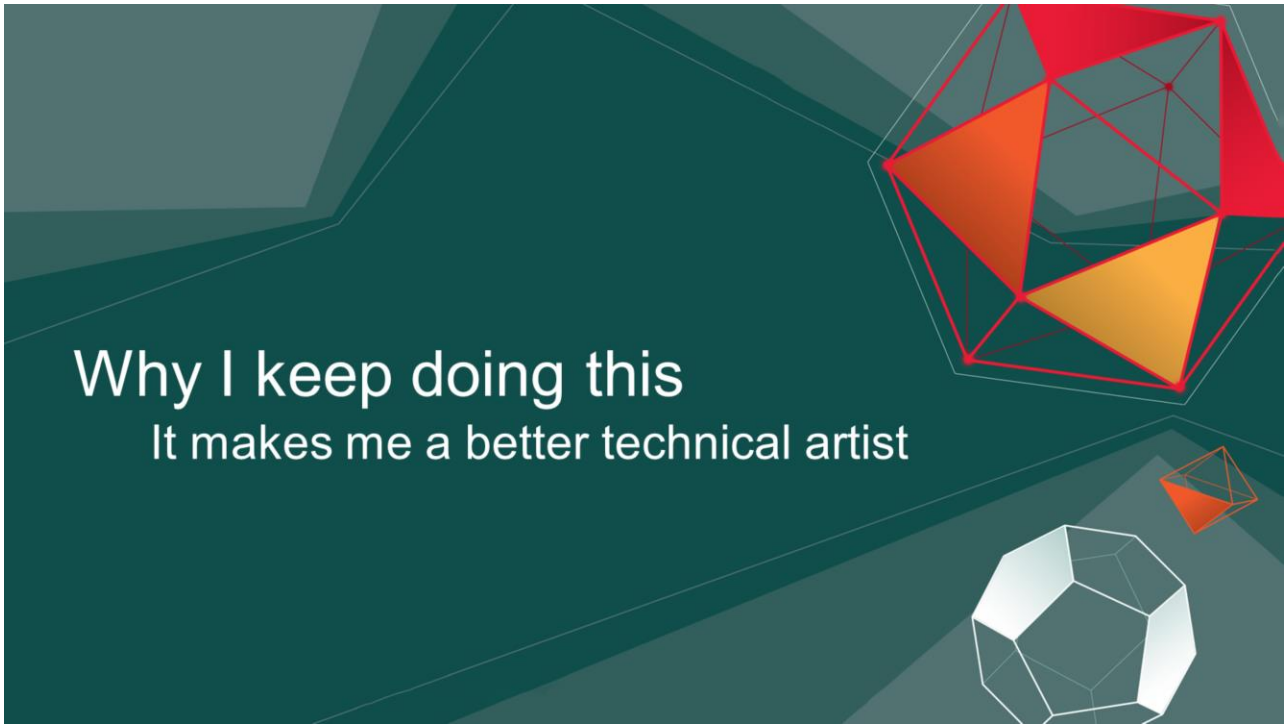
So one of the reasons I was interested in using, and then hacking on mGui is because it is built using a bunch of python's more magical metaprogramming techniques.

The kind of thing I'd bump into in a blog post, or a stack overflow question, but never really had a need to use in my day to day work.

Pymel is built on top of very similar

components, which is probably why I liked poking around in it so much.

But having an actual project that was making use of metaclasses, custom descriptors, a custom event system, was a wonderful way to get some hands on knowledge outside of "lets repeat that example I found on the web that one time".



So this right here, this is why I wanted to give this talk. In fact I might have mentioned that at the start?

Hopefully I wasn't to nervous and dove right through it.

But yeah, I feel the work that I've done on these projects has been invaluable towards my growth as a Technical Artist.

Code readability

Working on these projects has given me a huge confidence boost when it comes to diving into a larger more established code base.

Where I'm not just able to find where a particular problem point is, but to really get a grasp on the surrounding concepts, and in some cases get a decent understanding not just of HOW something was built but WHY it was built the way it was.

These skills have become all the more important for me these last 9 months that I've been with DSVolition, because we've got some code that was started well over a decade ago, that we're still using to this day.

Problem solving

So this harks back to how I got started with this mess in the first place. But really, deciphering bug reports, finding the problem point, and then fixing it?

Totally a huge part of being a TA. Sometimes we even do it without the bug report.

Communication – Using our usual TA communication related skills, but mapped over a very different domain.

Instead of in person white-boarding session, it was cross time zone conversations in an issue tracker.

Instead of meetings, it was a back and forth with pull requests.

Sometimes it was a quick conversation on the TA slack channel, other times it was a random side tangent TAO forums.

Same skills, same concepts, but used

differently. And because they were used differently, I feel immensely more comfortable exercising these skills.

So even if my skills haven't improved all that much, at least my local confidence in them has gotten a decent skill up, which is probably how I tricked myself into thinking that public speaking was in anyway a great idea.

Expanding circle of TAs (networking)

These are two collections of other open source projects in our community.

There are probably others out there, but I literally bumped into one of the lists while I was wrapping these wonderful slides up last week, and it felt proper to include them.

So any questions?