I don't know if my data are representative because I found that the bug I had without OGCB was much easier to unmask than the one I had with OGCB

---

i didn't get that the breakpoint on the onColor variable was reset with the restart button until a few try.
I also ended up using the normal debugguer to find the bug, i found the haltIf: method more appropriate to spot the corner coordinate than then the new debbuguer but i do think it will be useful.

I've used VisualWorks for 30 years, Pharo/Squeak only a few times. Finding my way around the Pharo UI was slow at times, and some browser and debugger functionality I'm used to seemed to be missing. OCDbg itself seemed simple enough to use. VW has similar functions to add breakpoints to all assignments or reads, and these can be made conditional on the object identity in the normal condition mechanism. The latter requires two small extra manual steps compared to OCDbg ("MyGlobal := self" for the desired object, and a breakpoint condition "self == MyGlobal"). In some cases the VW approach would have been better, e.g. for the 248th Person I lost 20 seconds on my second scenario, as I forgot that my Person with the breakpoint was no longer used, as each new run created new Person instances. In VW I would have set the global to the desired Person in the method that singled out that Person, so the object-centred breakpoint would have updated each time to the current Person of interest.

My results on Ammolite and Lights Out aren't really useful for you, because I didn't use OCDbg facilities, just normal unconditional breakpoints (which I assume Pharo already has) and code inspection or searches. Overall I think OCDbg is a useful tool, but not one that would often be the best approach in the debugging situations I tend to face. Conditional breakpoints using globals may be better, if needed that rarely: the added value may be less than the extra UI weight. But knowing how to use object-specific breakpoints in debugging is certainly a useful skill.

Sorry for the long interruption in the last task!

During the tutorial the breakpoints did not work as explained. I tried it several time, but the breakpoint was hit on every object.

Last survey question, I think OCDb does not supplement classic debugging but complement it.

First of all - if break is related to object. I need to see hash of this object (in order to visually verify that object is on given instance). Biggest trouble for me was that bug occured during initialization, didn't find the way, when state of this object is wrongly set. When you restart the entire app/or game, breakpoint is no longer valid, because instance is different (new obect is created). So maybe I missed whole point, when to setup "halt on state access". When instance is created and it has wrong name (or color in case of LoCell), i didn't find a way, how OCD can be helpful. Maybe it is just lack of my experience. Also confusing was, when I closed game or test, where instance occured, it didn't remove from breakpoint navigator. It needs to be garbage colllected (image cleanup) and then it is no longer visible. I would really like to see the video, how procedure should be done ideally using OCD.

The operating instructions for object centric debugging failed to inform that the breakpoint on the object instance must be inserted each time we execute the code. This is counterintuitive, as developers are used to adding the breakpoint once and running the code several times to exercise and analyze the piece of code

with the breakpoint. The breakpoint functionality on the instance is useful when we have objects of the same class in a collection of objects. This idea works well in Pharo because like the LightsOutGame example, we can inspect the object instance in just one click. However, in traditional languages such as C# and Java, it would be necessary to debug in a traditional way until the object is created, and then add the breakpoint to the instance, it would be very laborious and perhaps not very efficient. In these cases of collections of objects, in traditional debugging tools, we have conditional breakpoint functionality that normally meet this need. For Pharo, I suggest an improvement: making it not necessary to re-add the breakpoint in the new instance of the same object once the previous instance is lost. This may be possible by creating a history of object instances for the class once the instance breakpoint is added to an instance.

I like the experiment. One think the most difficult part in the last task was to inspect the object I want to debug. For instance, I realize that the problem was that the color was the same, but then i wanted to track when this object change this variable, but i could not find a way to inspect the object before it change its state.
Other think to consider is that when you run a test, or open a windows other different objects may be created. So, the debugger will not jump because is other object. So, I need to be able to inspect the object before the method that cause the bad beharvior is executed, and then hope that the break point works in the same execution. Otherwise, a second execution may create another objects that I need to tag again.

I was surprised at the Lights Out task, as OCD does not particularly help there. Once you have the object in question, it is too late to set any breakpoint as the damage has been done. Unless I am missing something, you need to use other techniques to locate the dirty method.

I have not used Pharo in long and had to struggle in remembering the basic apis. Also, I got interrupted too often.

Personally, I spent a lot of time looking in the wrong place because I expected the bug in the LO* classes 😃
That kind of biased me a bit.

---

for the last part I got stuck because of a bug which meant that sometimes my 4 sides were colored and sometimes not, whereas there should always have been only 3 of them colored.

It's a pleasure to debug 😃

Good experiment! We hit a bug but we could resolved easily

I tend to look at bugs as "fault -> error -> failure" where a failure is a service that has failed, the cause of the failure is fault (bad instructions in the code). An error is the manifestation of the fault. I see OCDbug as giving more help to find the objects wher an "error" has occurred. Sadly, the complexity of black-box frameworks (like morph) doesn't really make debugging much easier even if we know the object. The last experiment seemed to have a non-deterministic bug, and that is really hard to trace back to the framework (or it was for me). That is, I suspected onColor was being corrupted, but didn't really have time to go back

and find out when it was being called on the 100 squares at initialization (or wherever), especially when that item was in the morphic framework.

I feel I couldn't finish the OCD debugging task because I was tired after the first part of the experiment. (I didn't sleep much last night)

I think I have participated in the experiment, because I remember the Madec name.

Cool experiment.
About the last task: because the bug occurs at initalization of the game, it's not easy to use object-centric debugger.
The 'wrong' state is already here when one can install a breakpoint., so breakpoints only indicate that it *does not* break.
This is an information of course, but then, when one understands this, the object-centric is not useful anymore and one needs to switch to standard debugging and static analysis.
