Hi

This is a long document describing what I have done (well a bit). I aim for it to be high level and not discussing each and every class.

At the outset, I want to implore upon you, I am NOT a developer. Documentation and formats/styles may not be what a Developer would do. I am a Systems Administrator who can script. I am also Australian, so colour is spelt CORRECTLY :-)

Second, I will absolutely NOT be offended or otherwise upset if you do not to progess with this prototype.

There are plenty of bugs. But with the Beta of PoE#2 being released soon I have to stop and ask whether to continue.

## Intro

PyPoB is written in python against the pySide6 library set, to make a GUI implementation of PoB. I used the pyCharm IDE. Attempts to move to pySide6 v6.7.x have resulted in the program hanging or poor quality UI. I have not explored this further.

It was my intention to come up with a program that could do 75% or better of current PoB, deliberately leaving some recent things off, like tattoos and trading.

This is because it needed to be a sufficient prototype that it could be evaluated as a viable option to lua, as stated some years ago.

There is also a make\_exe option using nuitika, which works mostly. It needs to be constantly retested. I don’t percieve that it will be viable and we will probably need a download facility like you currently have and a supplied copy of python so folk don’t need to install a version.

Windows is the target. Linux is supported by python and pySide6 but not really tested. With my most recent change to using luaPoB for stats, ‘nix Operating Systems are now totally untested.

## Statistics

The biggest problem for me has been calculations and not understanding them in lua. Everytime I tried to read and understand it, or even rewrite the lua to extract data from the ggpk, I got nowhere and would stop programming for months at a time.

I have eventually spun a separate thread to call luajit. On my machine this takes 3 seconds and is absolutely not a good idea. It did allow me to print reasonable statistics in the app, achieving so level of compatability. This has not been extensivle tested against lots of build, as it using luaPoB and if it’s wrong, it’s because it’s wrong in luaPoB.

In MainWindow.py is do\_calcs\_v1. This is my first real attempt to get some stats from the items and skills internally to the program. It is in no way complete. It was just a start. As noted, I struggle with this.

There are modules for runing lua in python and data sharing, but as luaPoB’s data sets are complex I haven’t tried them yet.

## File Organisation

The file organisation was mostly setup by [dekkofilms](https://github.com/dekkofilms). I have altered it slightly.

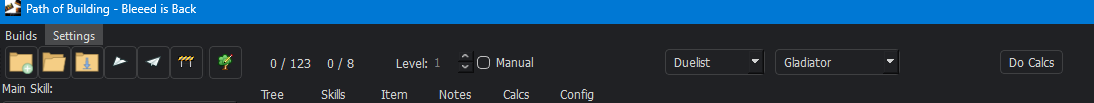
The file type of choice is now json, XML is too slow in python. json and yaml are native and dictionaries can be saved directly to this format. I have them pretty printed for our ease but also git is line based tool. I thought it would handle lots of smaller lines better than one large one.

* The src directory would be what goes out to the public.
  + builds and data: are hopefully obvious. Tree data is direct from GGG with no modification.
  + PoB: is for classes are for PoB functionality and should not feature any UI elements.
  + dialogs: are the processing for a dialog window.
  + widgets: are the bulk of the UI element processing, and mostly represent each tab (eg: Items, Skills). When there is too much inter-communication between these classes, processing is moved up to MainWindow.
  + window: is where the main loop of the program, main\_window.py, resides.
  + ui: is the generated python files of the GUI windows elements. PySide has the ‘QT designer’ which produces .ui files. There is an compiler that turns the .ui files into python.
  + lua: is where a copy of luaPoB is stored for calulations. Either the whole src directory can be copied or these at the minimum:  
     Builds/ Classes/ Data/ Modules/ TreeData/  
     GameVersions.lua Launch.lua UpdateCheck.lua  
    Currently this directory is hard coded. Whilst developers may wish to point it elsewhere, it has no value for the ordinary person (and they may fiddle). Perhaps a setting in settings.json that is manually updated by devs.  
    It is percieved that this situation is fixed by python calculations, but if it isn’t, this directory would get filled from the internet as the current luaPoB does (using python).
  + PathOfBuilding.py: The main program.
  + PoB\_rc.py. This is the QT Designer’s resource file. Fonts, Icons and some tree images are in there. It needs to be in that directory. Moving it breaks the world.
  + settings.json. Current settings (not in git). Equivalent to settings.xml.
* Docs. The beginning of some documentation, including installation and how to get gem and item data from luaPoB.
* scripts. Various scripts to convert ui files to python, make an .exe and scripts to assist in moving the gems and items exported from luaPoB into a format suitable for pyPoB.
* Assets. Basically everthing that is in PoB\_rc.py and the ui files for the QT Designer.

## QT Designer

This is found in .venv\Lib\site-packages\PySide6\designer.exe. It creates limited GUI windows with all the basic functionality provided by most UI widgets. There are lots of limitations. The MainWindow init function has lots of widgets being added as they can’t be done in the designer.

All the elements to the right of the Tree Edit icon, are added in Main\_Window.



Adding, removing or deleting UI elements requires you to break the Layout elements. This is a huge pain in the a\*\*e. You need to know what direction the layout was before you change anything.

Run *devel\_run* after each change or code your IDE to do it.

## Trigger Storms

Many elements have a trigger function (EG: Updating status after a user alters which skill is used). Many classes have a disable and enable triggers as code commonly updates other elements in response to a user input. A good example is clearing all elements and loading a new build, stocking all the UI elements again.

We only want to trigger when the user does something, not our code.

Many functions will reenable trigger and then change an element (EG: dropbox) so as to force all other UI elements to update - Like changing skillsets clears and refills UI elements on the Shkills tab but noowhere else (like the left hand elements).

## What can it do ?

Paste items but not gems yet from PoE game. It cannot copy from luaPoB as luaPoB uses a different format from PoE.

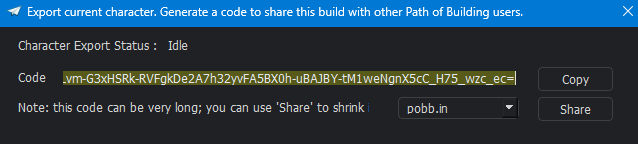
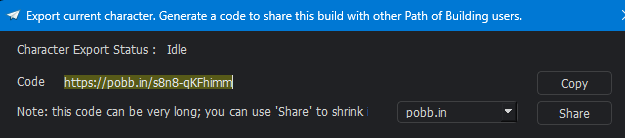
Developer **Note**: Copy/Paste must be done in MainWindow class as it a main thread operation. There could still commented code around when I tried it from the widgets/\*.py classes.

Open and save xml builds (these are version 1). These can be opened in luaPoB.

Open and save json builds (these are version 2).  
 

 Importing builds from the whole generated line (eg: the generate button in luaPoB) or from the internet (eg: pobb.in) – These are xml formats.

 Exporting builds to a whole generated line (eg: the generate button in luaPoB) or to the internet (eg: pobb.in) – These are xml formats.

We need assistance from @dav1d to be able to import a json, but let’s make sure we have the format locked down first – leaving it at xml might be viable too.

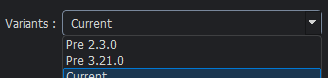
 A fair amount of Management for Trees (or Specs).

A fair amount of Management for Gems and Items and their “Sets”.

Basic changing of trees, item sets and gem sets.

Multiple versions of trees are supported, with auto loading versions as needed.

Item crafting is limited to altering text and changing socket colours. Some accomodation for item versions/variants is available too.



HTML Notes. A simplistic attempt to change ^N and ^#nnnnnn colours to html is made.

Developer **Note**: I have disabled plain Notes saving to XML as I have corrupted it with HTML some how.

Windows Windowing

The elements at the bottom of the Main Window move around as the window is resized. This class will need to be used on the config tab as it come into being.

Some GUI themes are available.

Modal Dialogs

## What can’t it do ?

Ahhh the list ….

Statistics and Crafting are the big ticket items.

The curved lines in the tree tab (ok, I can’t do curved line math)

## Known bugs

A ToDo exists:

https://docs.google.com/spreadsheets/d/1t53Oqg2ckXcusqd85jwsxH1LlRmzb0LVuGKa9K9w\_tw

It has a Bugs tab, with the completed items at the top.