**【软件开发代码规范】**

2017-02-14

良好的软件开发习惯和代码规范可以：便于团体协作开发，有效提高软件开发效率，增强软件**代码质量**，增强软件运行速度，提高软件的通用性可移植性，降低代码出错机率，去除冗余代码，减少软件维护时间和成本，等等。所以，这里大概总结几条代码规范建议，希望可以使我们的团队代码开发更有效率。

1. **分类目录**：

软件代码按功能分类和分子目录，相同或相似功能的子程序文件放在同一目录下。单个程序文件尽量只包含相同目标的子函数。

1. **注释说明**：

每个文件头，每个子程序头，每个函数头应该有简短的功能目的的说明注释。子函数内的关键部分（结构、算法、界面、变量等等）要有简短的描述。注释不需要长，而是应该言简意赅、简明扼要。

1. **层次结构**：

代码希望层次分明、缩进整齐、格式统一，以便调试检错和扩展。

1. **命名规范**：

变量和函数的命名尽量规范化。尽量使用有涵义的简短命名。规范、简短、有涵义、不重复、易区分、易查找、格式统一、尽量归类。同一类型的变量和函数使用类似的变量名字，命名可以加简短相同前缀（比如file\*\*\*\*，layer\*\*\*\*，cell\*\*\*\*），以便于管理和查找。

1. **精简扼要**：

冗余代码即刻删除！所有过时代码，如果希望今后继续参考，可以专门归总在某个目录或某个文件中。但是过时和冗余代码一定不要留在最终代码中。

1. **速度效率**：

代码和算法尽量有效率，注意计算速度。能够用向量和矩阵处尽量用，能够用循环尽量用，减少单个变量和单个操作。任何低效的代码，一经循环调用，对程序的速度影响可以是毁灭性的。

1. **用户体验：**

用户体验至上！降低软件使用所需要的学习时间，减少软件操作所需要的操作次数。比如设定目标“天下没有难做的版图！”，让入门门槛尽量降低。如果你自己都觉得怎样用不方便，那么用户更加会觉得不方便。有人说过，软件代码设计增加一次click会减少10%的用户。

1. **检查优化：**

写完的程序代码一定要检查和优化，你会经常发现原来你可以做得更好，代码可以更规范更有效。尽量把程序当作文章来写，而不是散漫的代码，那么你就自然会该写的写，不该写的不写，该花功夫的地方花功夫，不该花功夫的地方不浪费。

1. **交流沟通：**

及时和团队成员交流沟通。有任何问题需要团体其他成员注意或者解决，可以点名要求回答。

我们可约定在需要检查的地方加入代码注释如下： //CHECK!!

统一思路，我们一定可以做到最好！

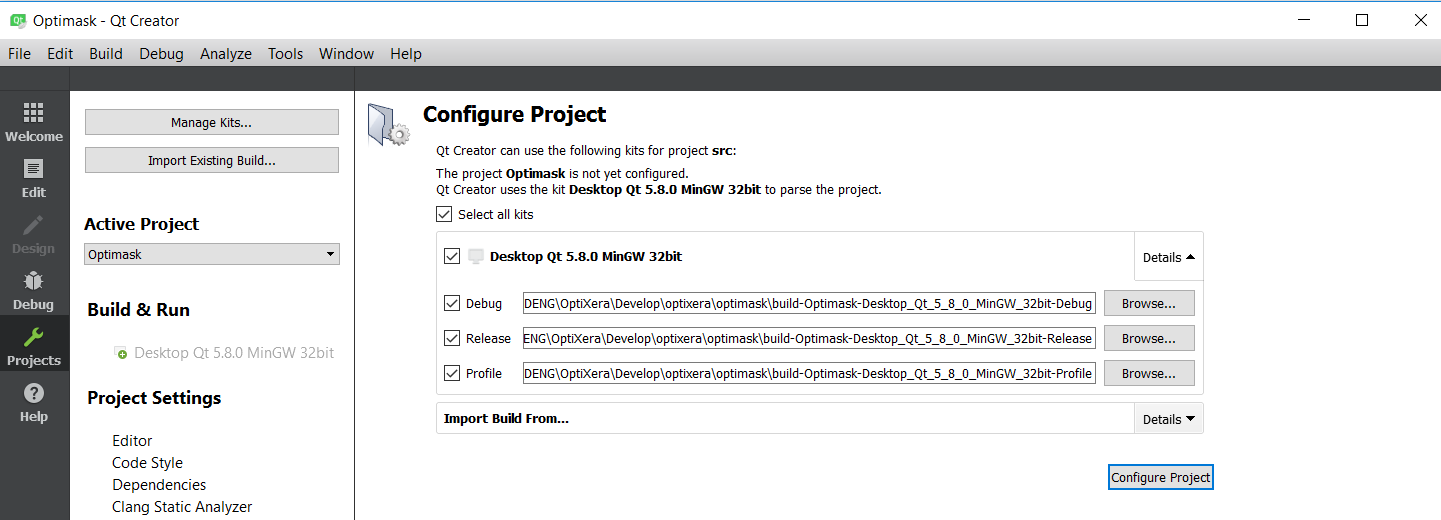
**【统一路径，代码上传】**

2017-02-14

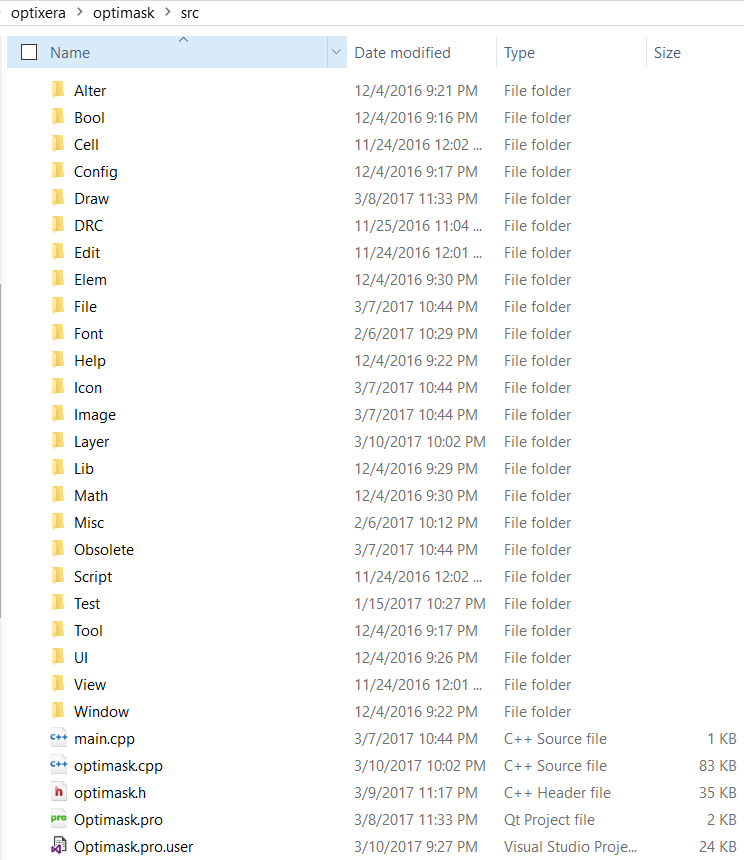
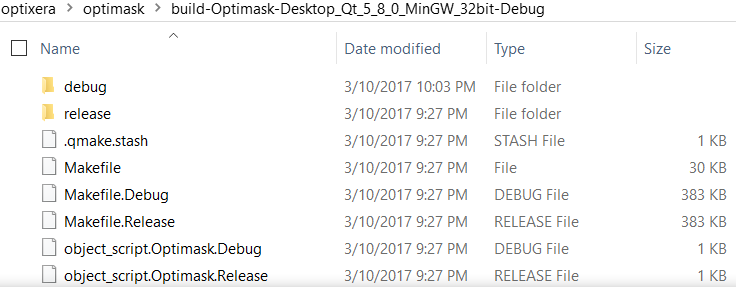
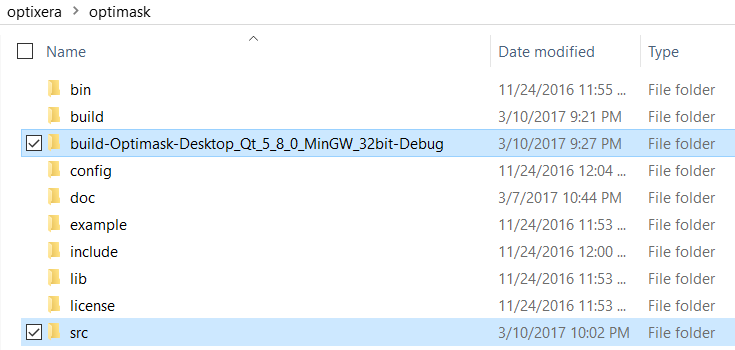
**源代码 \optixera\optimask\src**

**Build目录跟源代码平行 \optixera\optimask\build**

**（如果你喜欢Qt Creator自动设置的长build目录名字也可以，但是一定设置在与源代码目录\src 平行，比如以下）**

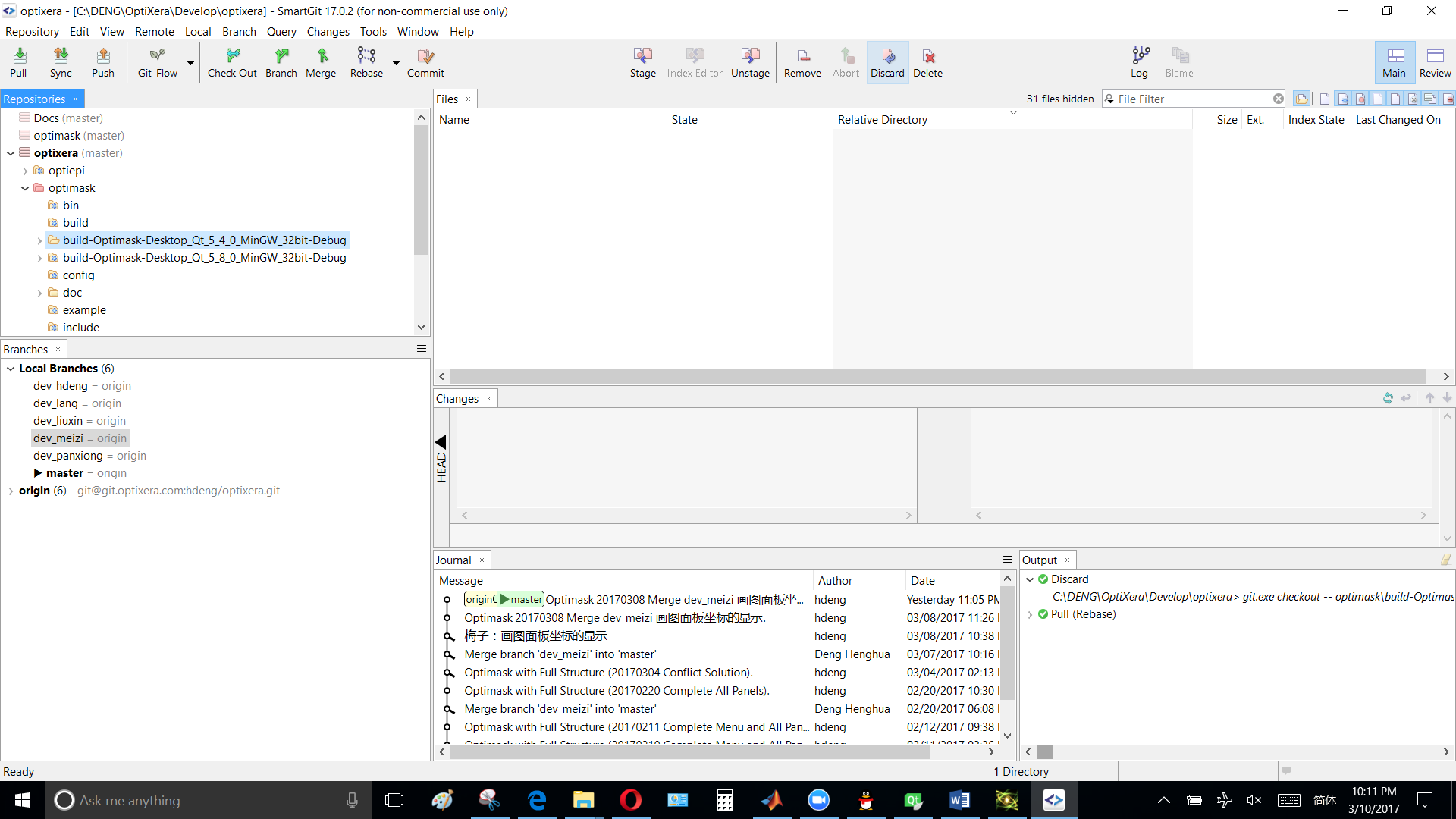


**在资源管理器下，源代码目录和编译目录显示如下**

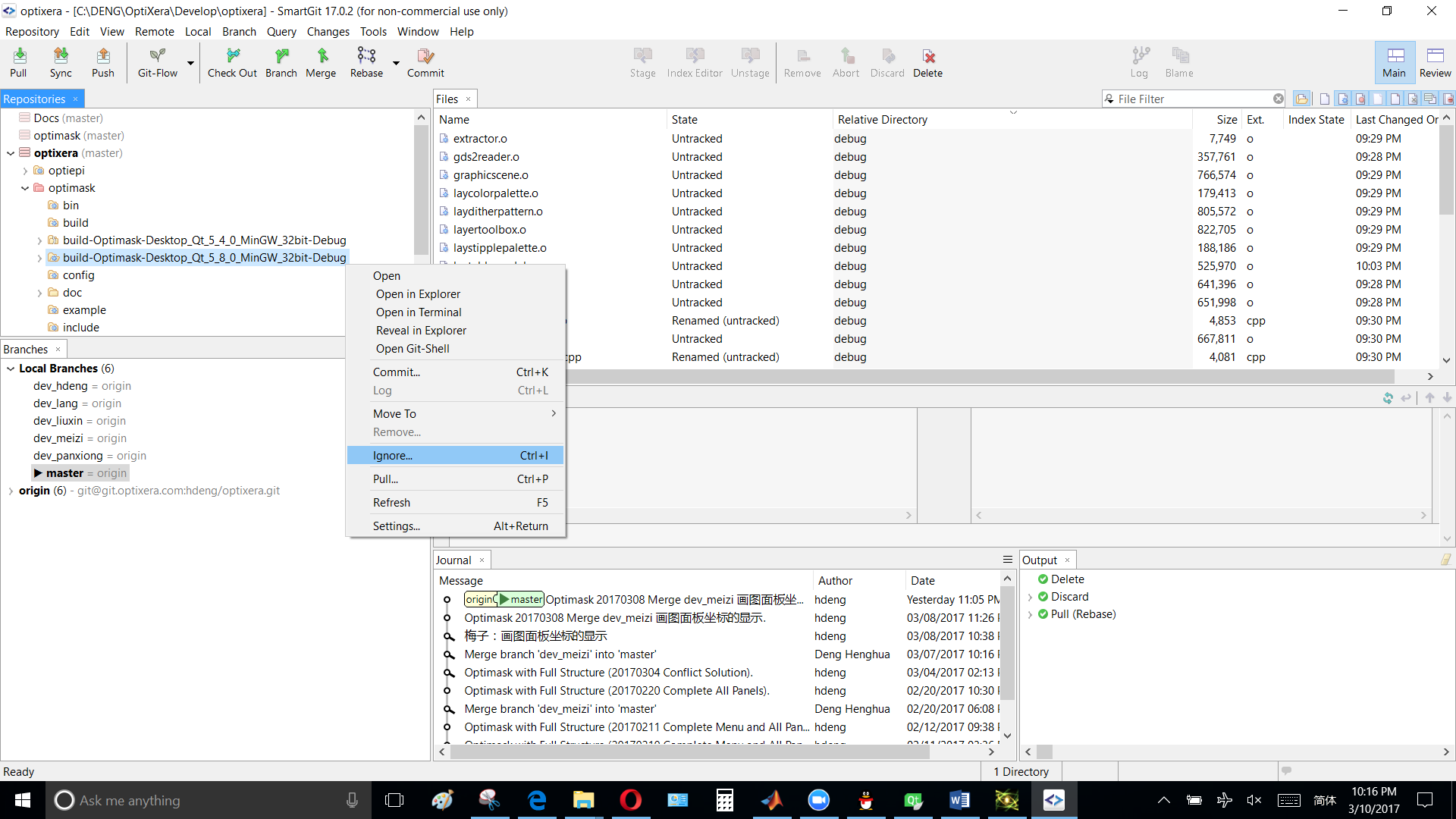


只需要同步源代码目录**\optixera\optimask\src**，所有机器产生的文件都**不**需要上传。

在SmartGit上面，对于远程已有的build目录，可以先“Discard”，再“Delete”（删除），再“Remove”（Un-Version），然后再**Commit**。



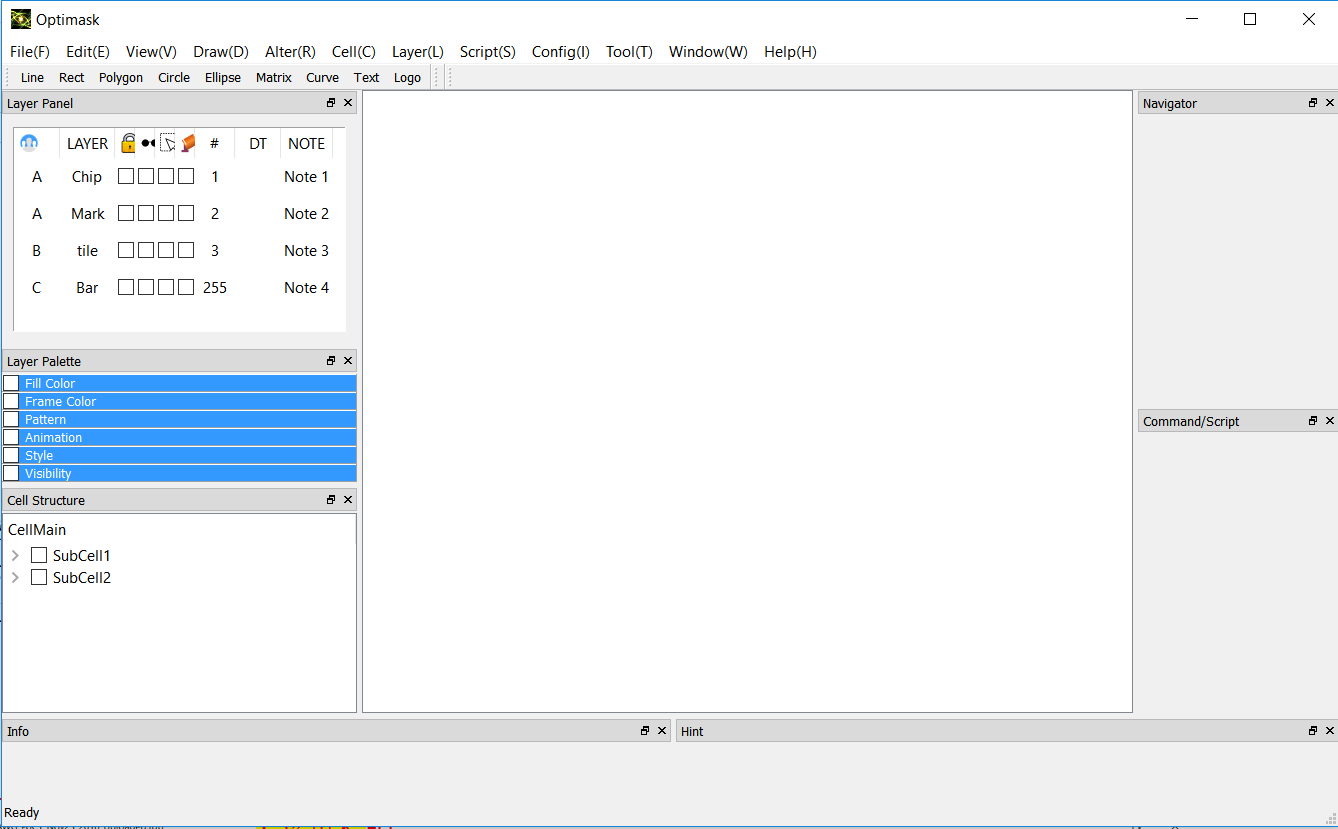
对于本地分支的build目录，可以选择“Ignore”（不要上传）。



**【Optixera光子学版图软件Optimask开发设计进阶】**

2017-02-10

目前我们的基本框架已经定型如下。



CodeDock

7.编程区

或计算区

或程式显示区

Macro/Script

Layer Palette

3b. 构层配置

（色板, 格纹, 动画, 样式, 能见度）

CellDock

4. 模块树

结构区

(需要即时更新)

HintDock

8.报错和提示区

InfoDock

9.状态信息区

Menu 1.菜单栏

Toolbar 2.工具条

WorkDock

5. 版图主视图区

（需要**允许多窗口**）

NaviDock

6.第二视图区或组装区或或关系区或结果显示区

Layer Panel

3a. 构层面板

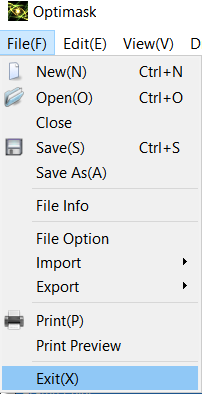
(需要即时更新)

各区域缺省设置如上。用户可在设置界面关闭或者取消指定区域，且可自由移动、放大和缩小区域。

具体到每个子区：

1. 菜单栏（Menu）

菜单栏包括通常的文件输入输出，程序设置，版图绘制，浏览编辑，程序调试，辅助工具，视窗选择，使用帮助等类别功能。

* 1. 文件（File）

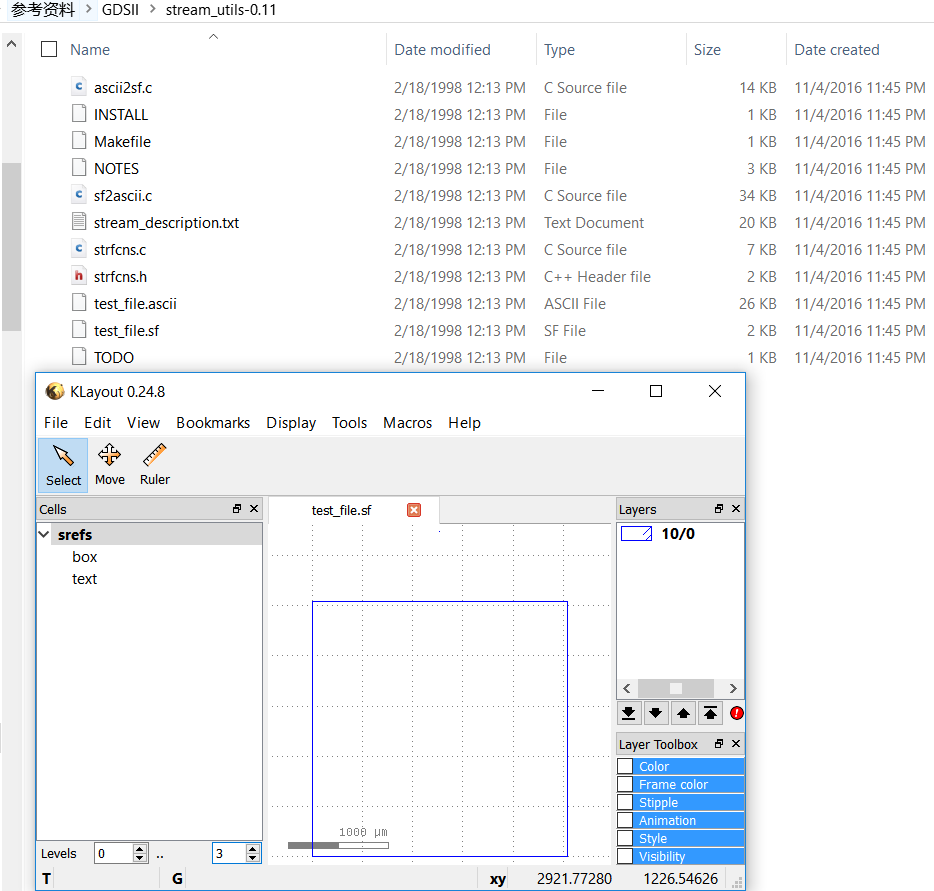
程序的第一步操作是文件操作。

试验创建、打开、关闭文件。

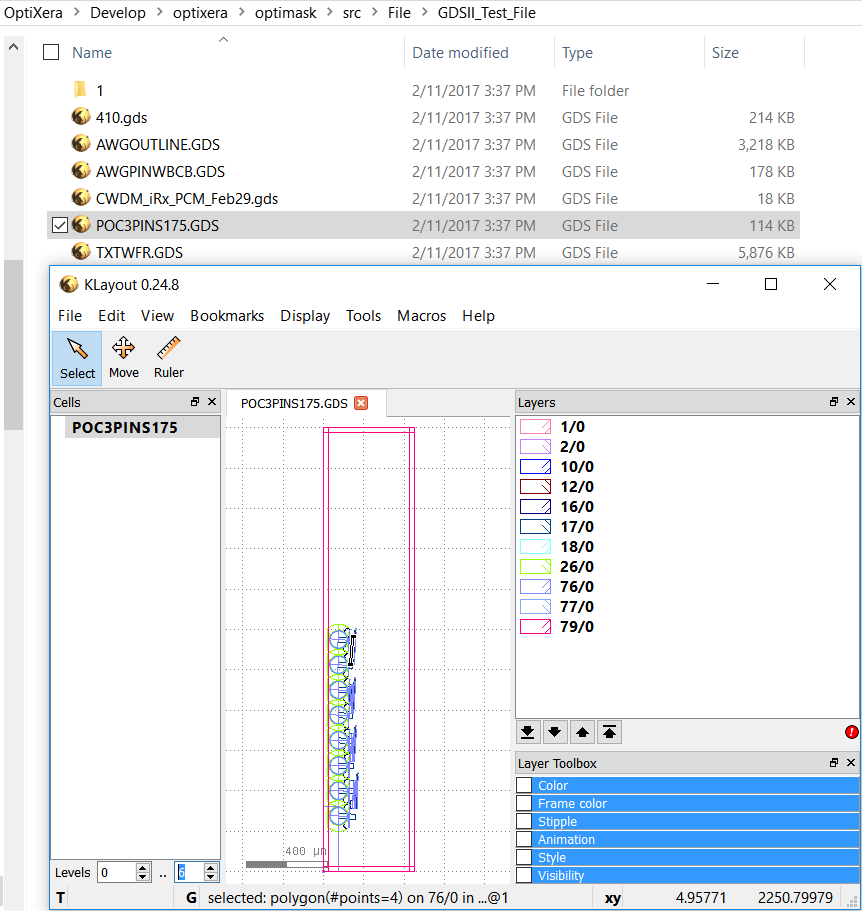
试验导入导出“GDSII”文件。如下图所示。实例GDSII文档在\klayout-0.24.8\testdata\gds 目录和 \klayout-0.24.8\testdata\gds2\_txt 目录。

试验导入导出“OASIS”文件。如下图所示。实例GDSII文档在\klayout-0.24.8\testdata\oasis 目录。OASIS文档和GDSII文档可以互相转换。

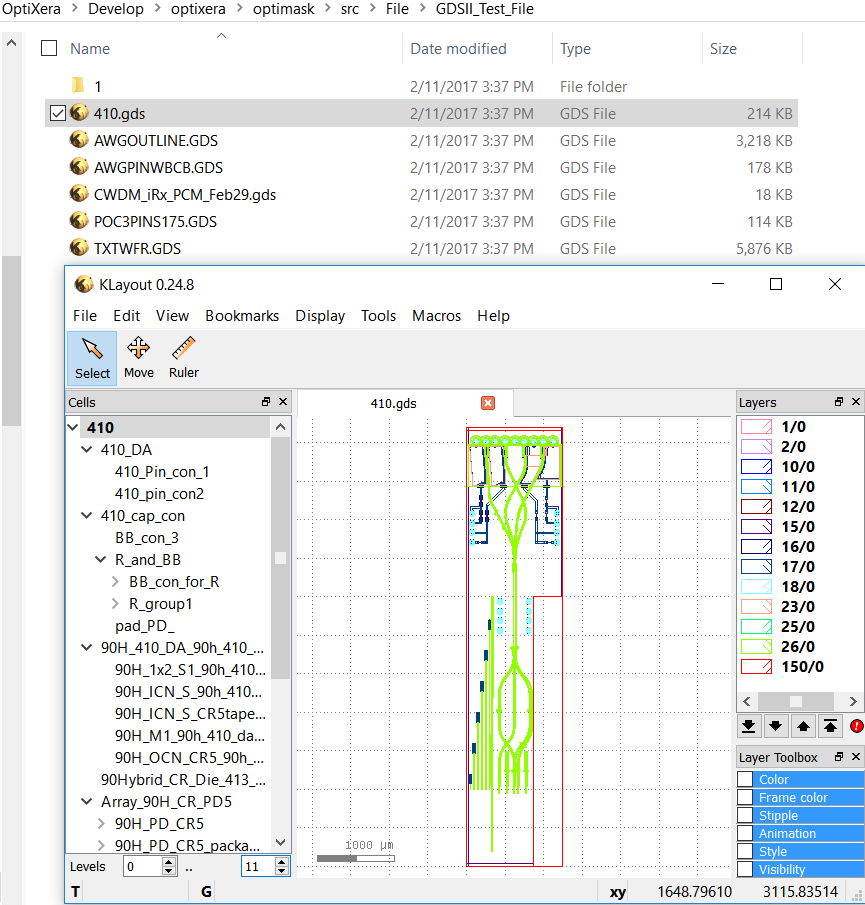
* + 1. File🡪Import🡪GDSII
       1. 测试单个构元，单层。单个构元有多重部件。

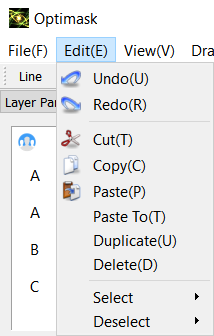
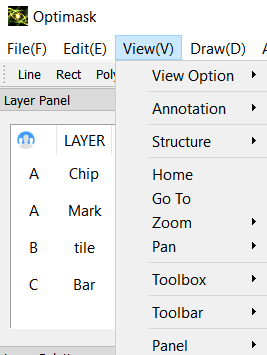
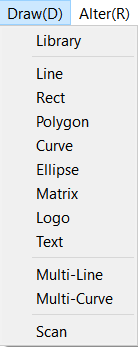
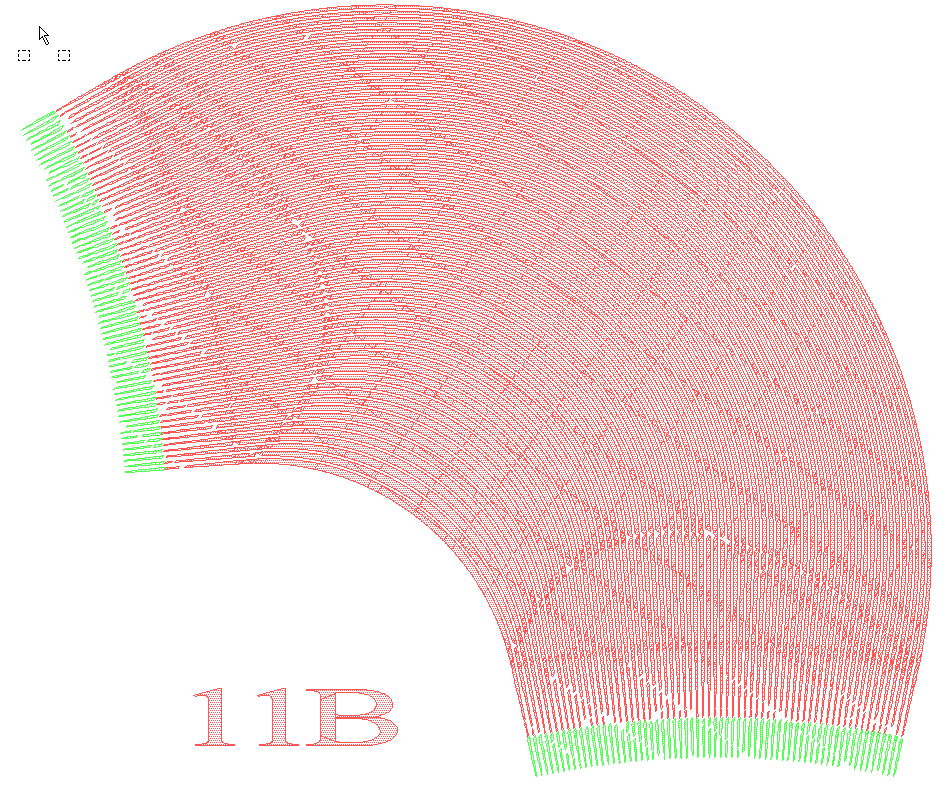
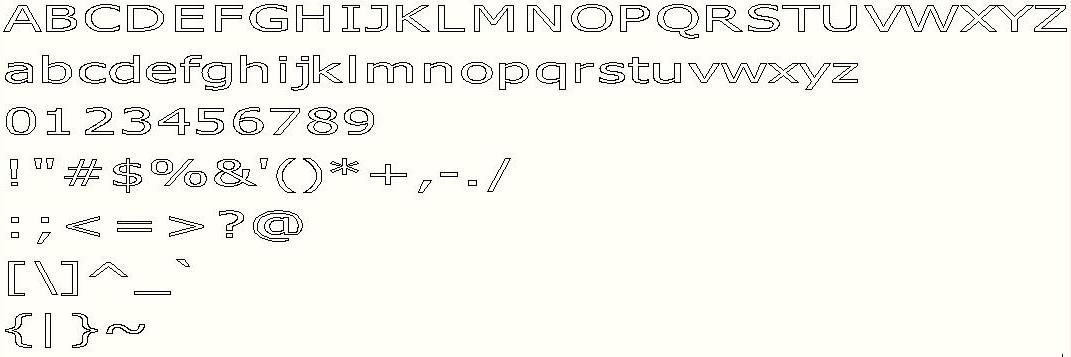
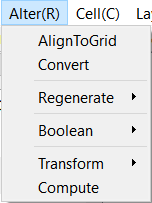
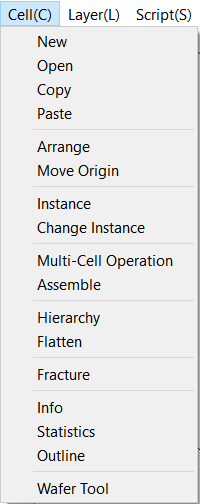
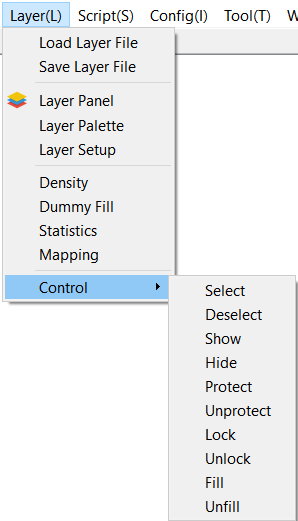
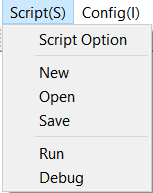
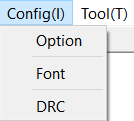
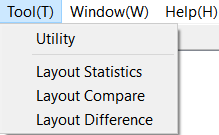
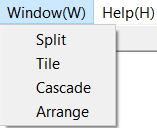
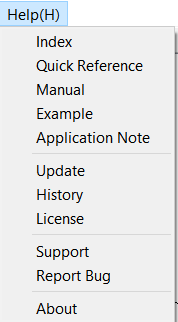


* + - 1. 测试单个构元，但是多层



* + - 1. 测试多个构元，而且多层



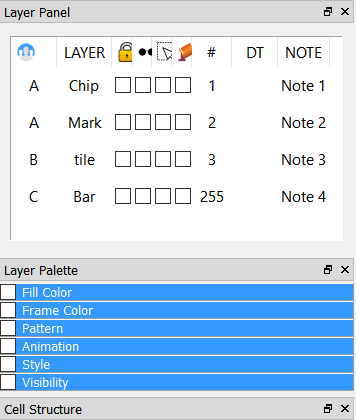
* 1. 编辑（Edit）
  2. 视图（View）
  3. 绘制（Draw）
     1. Polygon, Circle, Wire（用户可简单通过界面画图，鼠标或键盘。这是目前软件都可以做到的基本功能）
     2. Input by Matrix, or calculate polygon points.（这是目前所有版图软件无法独立做到的地方。）
     3. 扫描图形Scan Picture（这是目前版图软件比较难做到的地方）。也可扫描条形码，和二维码。算法存下的文件必须精确，且文件尺寸小。
     4. 图形库（我们慢慢建立）。
     5. 字库。可引入任意字库（C:\Windows\Fonts）字体如上图例（而非限于仅有几个难看的字）。
  4. 变化（Alter）
  5. 构元（Cell）
  6. 构层（Layer）
  7. 编程（Script, Code, Macro, Programming）
  8. 配置（Config）
  9. 工具（Tool）
  10. 窗口（Window）
  11. 帮助（Help）

1. 工具条（Toolbar）

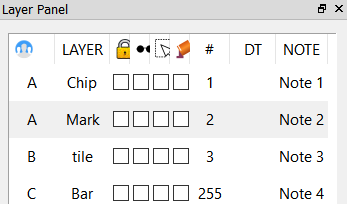
工具条主要是对应菜单栏的所有功能。需要每个功能有对应的图标。每一个菜单栏所有的功能集中在同一个工具条上。

1. 构层面板（Layer Panel）和构层配置（Layer Palette）

构层区和层设置区（Layers Palette）缺省随主界面，但是允许用户自由移动和关闭。（当层设置好后，有时用户需要大窗口进行绘图）



* 1. 构层面板（Layer Panel）：主要是构层的列表。

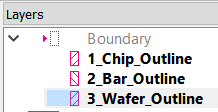


其中表头分别为：Group， LAYER， Lock， Hide， Protect，Fill， GDSII Number （#）， GDSII Data Type (DT)，Note。

表格数据为版图文件所真实包含的构层列表。对于每一构层，如果有对应表头的数据，那么就显示；如果没有表头对应的数据项，那么就显示为空即可。上图显示的表格数据只是一个虚例，真正的显示需要根据版图文件对应显示。

关于表头具体到单项：

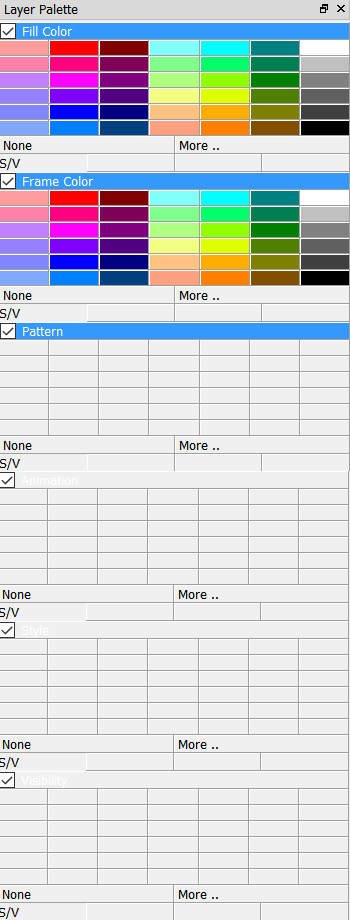
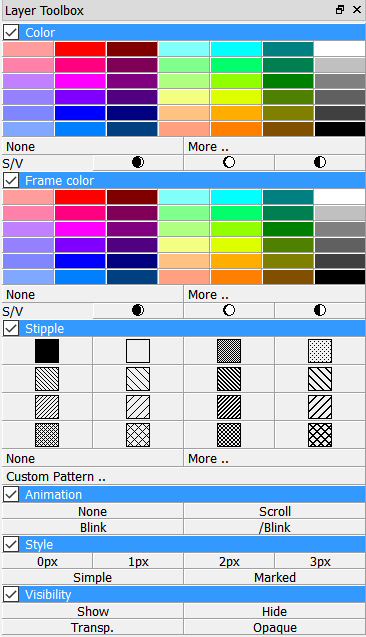
* + 1. 构层组群(Layer Group)或类别(Category)或排序(Order)。这个可以方便用户对多层组织，比如上图的群A有Chip和Mark两层。当然用户可以不组群，此时用户也可以通过这一栏设定特定的值来进行排序。

这里举个K-Layout参考示例，。对层群的操作需要传递到其从属的所有构层。

* + 1. 构层名称(Layer Name)。这里记录构层的名称（必须有）。如果用户不给构层命名，那么程序可以自动设置构层名称为GDSII#对应（读入GDSII文件一定有GDSII#）。比如Layer#005（如果GDSII#为5）。当然也有另外一种情况，有构层名称，但没有GDSII#，这时表示该层不会输出到GDSII文件格式中去。
    2. 锁定（Lock）。是否锁定(LOCK=1加锁, UNLOCK=0解锁)。锁定某构层表示该构层不会被修改，即使用户修改了该层的物件，也不会允许被保存文件（此时可以弹出警告框提示）。锁定可以防止对指定层的误操作。所有可以修改的构层必须在解锁状态。
    3. 隐藏（Hide）。是否隐藏(HIDE=1=UNSHOW隐藏, UNHIDE=0=SHOW显示)。隐藏某构层时，在工作窗口上不会显示该层的任何图像。构层被隐藏时，也同时不可以被选中（即，同时处于PROTect状态）。
    4. 保护（PROTect）。是否保护(PROTect=1=Unselectable不可选中, UNPROT=0=Selectable可选中)
    5. 填充（Fill）。是否填充(FILL=1实心, UNFILL=0空心)。选定是该构层的图形全部实心显示，否则空心显示。
    6. GDSII Number （#）。导入导出时对应的GDSII构层号码。导入GDSII文件一定有GDSII#。如果有构层名称，但没有GDSII#，这时表示该层不会输出到GDSII文件格式中去。
    7. GDSII Data Type (DT)。GDSII数据类型。
    8. Note。对构层的注释（如果有必要）。可以为空白。

表格数据每一行对应一个构层。

* 1. 构层配置(色板,格纹,动画,样式,能见度)

基本套用KLayout

其中(Optimask ：KLayout)名字对应为： Fill Color = Color, Frame Color = Frame color, Pattern = Stipple; 其他不变。

在layertoolbox.cpp（.h）中分别调用

#include "./Layer/laycolorpalette.h" //构层色板(Layer Palette)

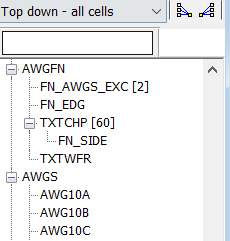
//CHECK!! 可能需要创建再包含这些头文件!! 可多个合并，视情况而优化。

//#include "./Layer/laypattern.h" //构层格纹(Stipple, Pattern)

//#include "./Layer/layanimation.h" //构层动画(Animation)

//#include "./Layer/laystyle.h" //构层样式(Style)

//#include "./Layer/layvisibility.h" //构层能见度(Visibility)

1. CellDock构元结构面板(Cell Structure Tree Panel), 构元组织(Cell Hierarchy)，模块树和组织结构区 显示各结构之间的从属关系。双击特定某个构元时，打开一个新工作窗口来显示该构元图形。
2. WorkDock版图主视图区 ---- 主要版图设计工作区。结构的绘图，调用，几何运算，显示等等。需要允许多重工作区域(Workspace, Work Panel)。
3. NaviDock视图导航区域 (Navigator, Aerial View) (导航区，鸟瞰区，第二视图区，辅助视图区）--- 显示当前结构在整个Wafer组装之后的位置等等。鼠标选中对应的位置市，主工作区可以显示该地方的详尽图。
4. CodeDock编程区域(Command, Script, Macro, Code, Programming)---编程产生结构，以及结构组合在Wafer上。
5. InfoDock信息输出栏(Information, Status, Result, Output) --- 显示程序运行时必要的显示信息，输出结果等等。显示当前结构的统计信息，比如多少层，多少多边型，多少精度等等。
6. HintDock提示报警栏(Hint, Error, Warning, Issue)报错和提示区---如果程序出错，提示如何纠正。

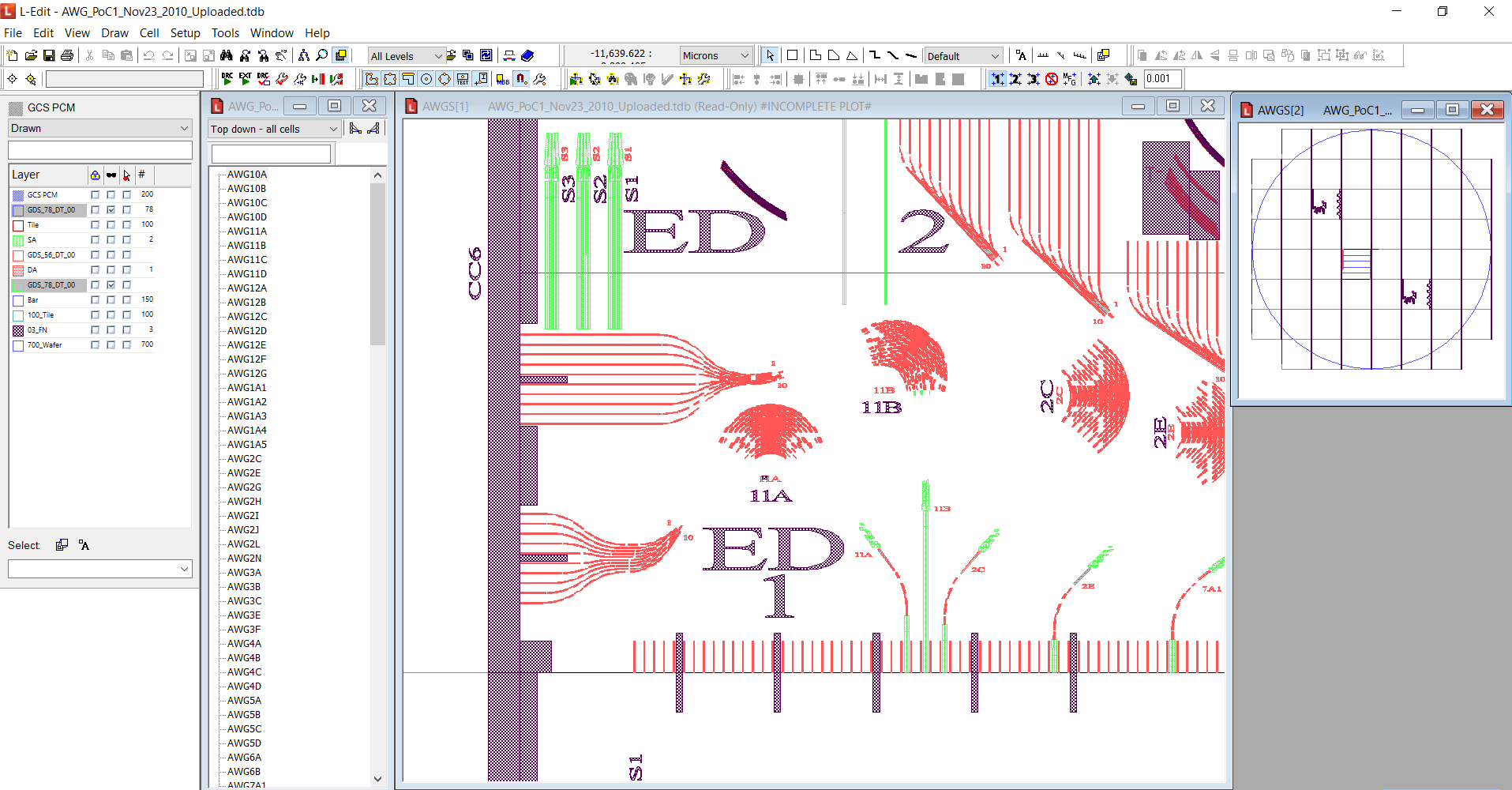
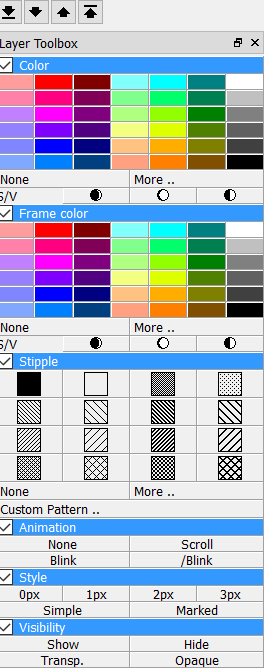
**【光子学版图界面设计需求】**

2016-09-10

如前所述，我们有全套光子学版图设计的高精度计算原代码，基于Matlab和C/C++。但是这些代码需要一个统一的界面系统，以根据用户需求进行自动数据计算和图形处理，并进行版图显示。现有光子学版图软件基本基于电子EDA版图软件，其处理任意曲线的功能不强。因此，希望重新规划光子学版图软件，真正让用户实现简单方便随心所欲高效率高精度的光子学版图设计。

综合常用软件的功能，初步计划光子学版图设计的界面需求如下：

1.菜单栏



7.编程区

或计算区

或程式显示区

Macro/Script

3.构层区

4.模块树

结构区

5.版图主视图区

8.报错和提示区

2.工具条

6.第二视图区或组装区或或关系区或结果显示区

层设置区

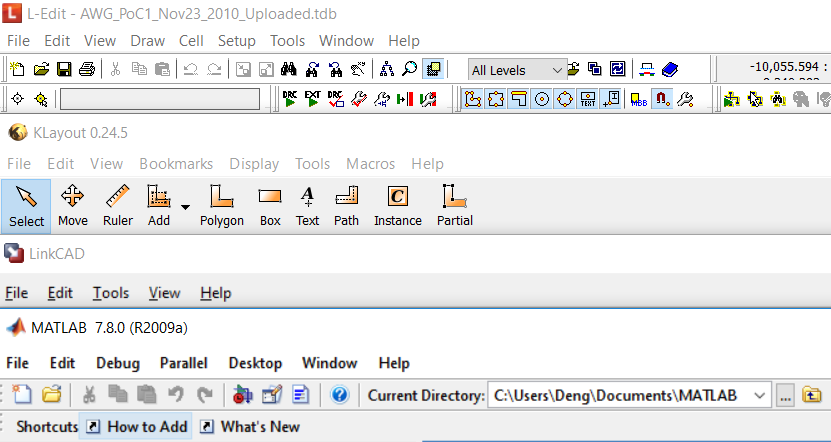
9.状态信息区

各区域缺省设置如上。用户可在设置界面关闭或者取消指定区域，且可自由移动、放大和缩小区域。

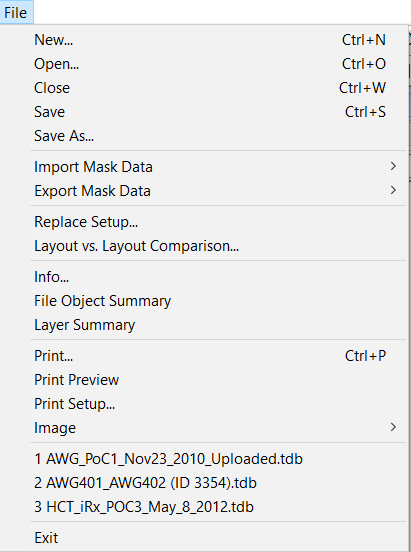
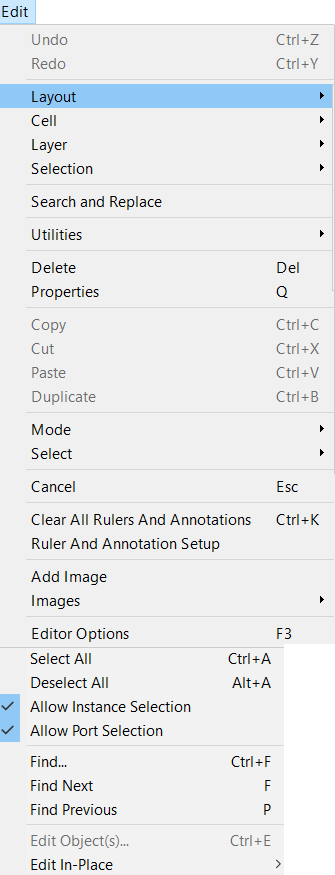
具体到每个子区：

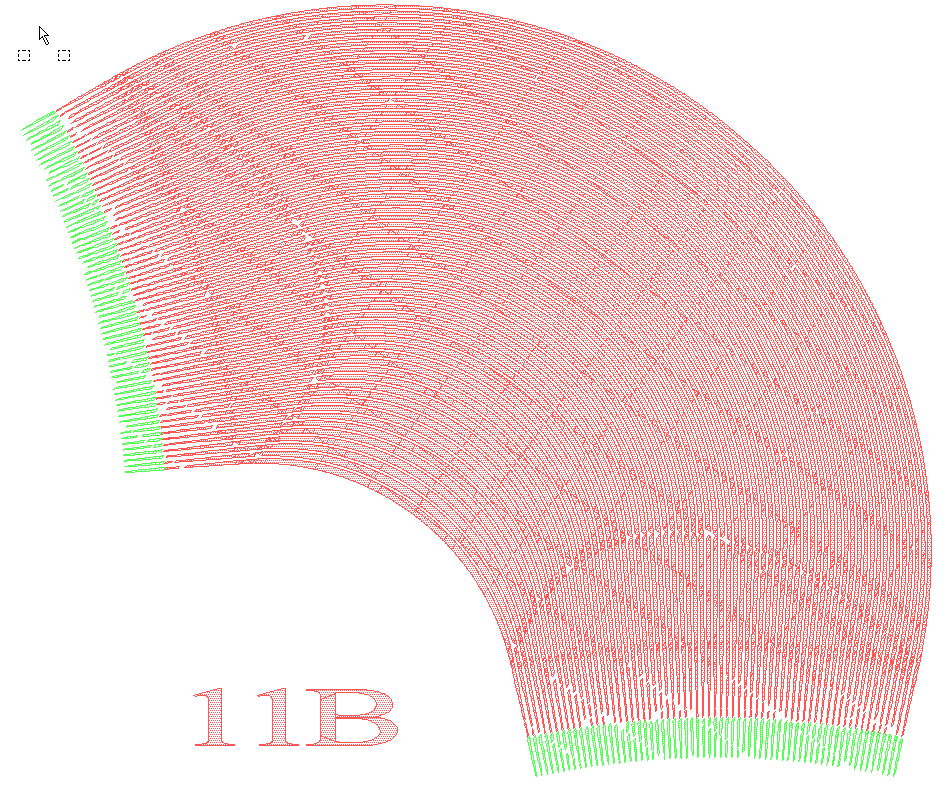
1. 菜单栏

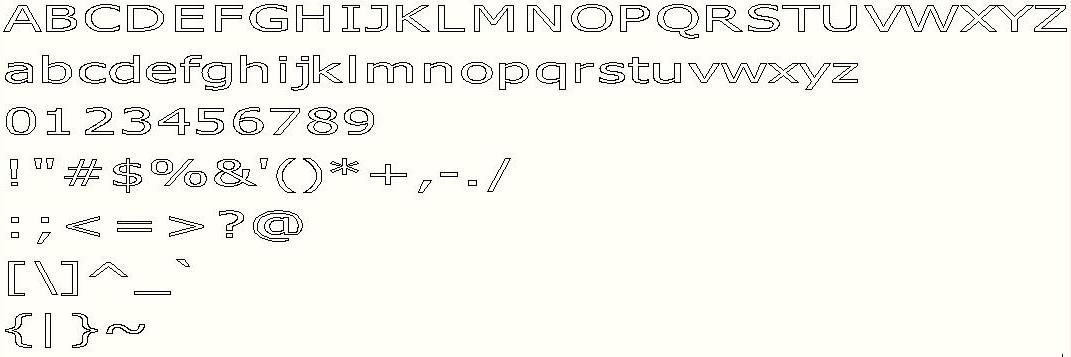
菜单栏包括通常的文件输入输出，程序设置，版图绘制，浏览编辑，程序调试，辅助工具，视窗选择，使用帮助等类别功能。



综合需求，可以设置如下（仅供参考，其实不需要所有命令，我们会再精简优化）：

1. File 
2. Edit 
3. Draw:
   1. Polygon, Circle, Wire, （用户可简单通过界面画图，鼠标或键盘。这是目前软件都可以做到的基本功能）
   2. Input by Matrix, or calculate polygon points.（这是目前所有版图软件无法独立做到的地方，而我们已经有，也是我们的核心之一。）

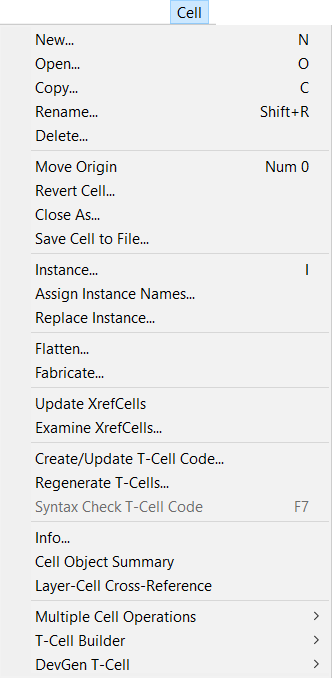
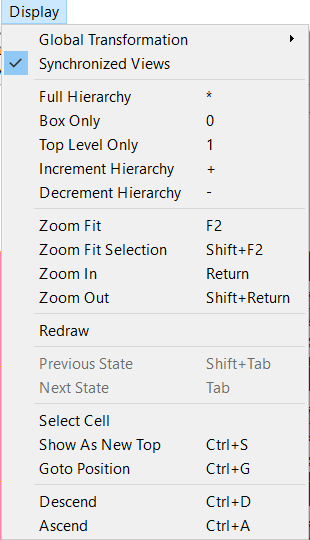
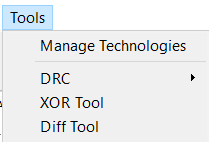
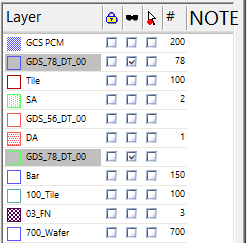
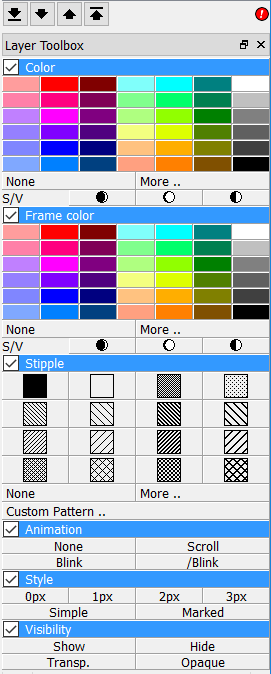
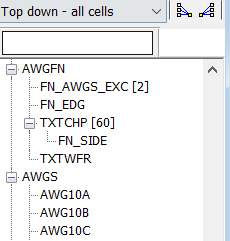


Scan Picture（这是目前所有版图软件无法独立做到的地方）

比如可引入**任意**字库（C:\Windows\Fonts）字体如上图例（而非限于仅有几个难看的字）。这个功能我的程序已经有，不需界面开发者介入。

* 1. Macro/Script（这是目前所有版图软件急需改进的地方，是我们的另外一个核心，详略）

b/c/d三项是我们的核心部分，但是是界面开发人员所不需要关心的问题。

1. Cell: 
2. Display: 
3. Tool: 
4. Navigation: Windows, Cell, Hierarchy
5. Help: Manual, Version, About(CopyRight)
6. 工具条
   1. 菜单栏上每个类别功能对应工具条上图形按钮。
   2. 每个工具条可以自由浮动，移动，关闭和打开。
7. 构层区和层设置区（Layers Palette）
   1. 构层区 需要有Layer Name， GDSII Layer #，显示（开关）Show/Hide，保护（开关）Protect/Unprotect，选择（开关）Select/Unselect，注释信息Note，以及每层的层设置。
   2. 层设置区 包括边框，填充，颜色等。
8. 模块树和组织结构区 显示各结构之间的从属关系
9. 版图主视图区 ---- 主要版图设计工作区。结构的绘图，调用，几何运算，显示等等。
10. 第二视图区（辅助视图区）--- 显示当前结构在整个Wafer组装之后的位置等等
11. 编程区---编程产生结构，以及结构组合在Wafer上。
12. 报错和提示区---如果程序出错，提示如何纠正。
13. 状态信息区---显示当前结构的统计信息，比如多少层，多少多边型，多少精度等等。