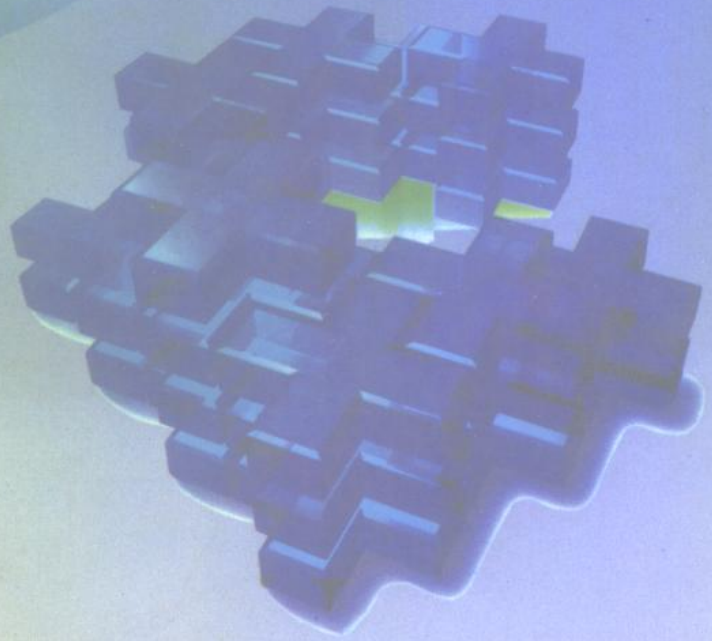
C++下MOTIF编程

任杰 一侠 张强 编著



北京大学出版社

C++下 MOTIF 编程

任 杰 一 侠 张 强 编著

北 京 大 学 出 版 社

新登字(京)159号

图书在版编目(CIP)数据

C++下 MOTIF 编程/任杰等编著. —北京:北京大学出版社,1994.8
ISBN 7-301-02506-8

Ⅰ.C… Ⅱ.任… Ⅲ.C 语言-程序设计 Ⅳ.TP312C

出 版 者: 北京大学出版社
地　　址: 北京大学校内
邮政编码: 100871
排 印 者: 北京成功信息处理有限公司排录
北京飞达印刷厂印刷
发 行 者: 北京大学出版社
经 销 者: 新华书店
版本记录: 787×1092 毫米 16 开本 12.5 印张 300 千字
1994 年 8 月第一版 1994 年 8 月第一次印刷
印数:0001—3000 册
定　　价: 23.00 元

目　录

第1章　3DS3系统

1.1　3DS 软件简介

1.1.1　3D Studio发展简况

3D Studio 是可以在个人计算机上实现高质量三维实体造型及动画创作的图形图像软件系统。它的操作相当简单；对微机的工作环境的要求也不高；所创作的图形质量好、色彩逼真，因而该软件一经推出就受到了广告业、设计业等众多行业的喜爱。

3D Studio 直译成中文的意思是“**立体摄影室**”，简称为**3DS** 。

在现实世界中所观察的景像是二维的平面图像，仅仅由于双眼所视的景像略有差异，反映到大脑的图像是有立体感的。人们在视觉中感受到的物体运动也是由于反映到大脑中的物体图像有着连续的变化。用照像机拍下的每一张照片都是静止的图像，而连续拍摄的照片再经过连续播放，就有了连续变化运动的效果。电影是基于这个原理制作出来的。

3DS 运用这一原理，先在电脑上用3DS 制作一幅图像，通过指定的变化运动生成系列相关的图像，再用连续的播放手段，将这些图像在电脑屏幕上放映出来，就产生了这些图像的动画效果。

3DS 的图形制作过程是:

1．在三维图形编辑模块(3D Editor) 中进行立体图形的造型；或是在二维图形编辑模块(2D Shaper)中制作平面图形，再用三维延伸模块(3D Lofter)将平面图形沿指定的路径适当地向三维延伸生成立体图形，延伸过程中可以加入复杂的变形操作，使生成的立体图形具有多变的效果，最后，将立体图形再传送到三维图形编辑模块当中。

2．用材质编辑模块(Material Editor) 配制三维实体所需的色彩或贴图材质，并在三维图形编辑模块中为各个实体模型附加材质。

3．为三维图形编辑模块中的实体图形加入各种照明用的环境光源，并配上视角与景深可调的照像机。

4．通过对照像机视图的着色处理，产生真实感的彩色图像（也可称之为用照像机拍下的彩色照片）

5．经过上述预创作之后，就可在动画编辑模块(Keyframer) 中制作更详细的一幅幅系列图像，这些图像具有一定的变化运动关系；然后再配合后期剪辑处理和实时播放，就可以在屏幕上观看动画节目。

6．该动画节目可以从VGA 与电视转换接口送到电视或录像带上，供其它播放设备使用。3DS 系统还可与其它图形制作编辑系统进行相关的图形数据的交换，方便了图形的调用和编辑。

7．配合IPAS外部支持模块，3DS 可产生更加丰富多采、变化无穷的特殊效果。

X 应用程序的移植性很强。前面提到了 X-Window 的商业版本 X11 是由 X 联合会支持的。用 X-Window 开发的应用程序不需要什么改动就可以在不同 CPU 的机器上运行。在这里，机器的硬件被 X 协议所屏闭，用户在编写 X-Window 应用程序时，不需要考虑任何机器硬件特征。

X-Window 系统的一大特色是 X 不定义任何特定的用户界面风格。X 只提供一组灵活的窗口操作元素，设置了多种影响界面风格的资源，用户可以对这些资源设定不同的内容以得到不同的风格。原则上，X 不强行采用任何特定的风格，在 X 应用程序系统的逐步发展过程中，形成了几种很流行的风格，其中以 OSF/Motif 以和 SUN 及 AT&T 为代表的 OpenLook 为典型。

1.1 client-sever 机制

X-Window 系统的体系结构是建立在 client-sever 模型基础上的。有一个进程叫做 sever(服务器)，由 sever 来创建并操纵屏幕上的一切图形和文本，用户在使用输入设备进行输入时，诸如敲了一下键盘或按下鼠标按钮，这种输入事件也是由 sever 来接收并处理的。一般地讲，X sever 在工作站或带图形屏幕显示器的电脑上运行。不过，有些厂商也提供了专用 X 终端，它们在硬件或固化软件中实现了部分或全部 X sever。

任何一个使用 X sever 提供的输入输出机制的应用程序进程都叫做 client。client 使用 X 协议，通过一个网络互联与 X sever 通信。多个客户可以同时与一个服务器互联，反过来，多个服务器也可以同时与一个客户互联。例如，两个学生在一台没有联网的 SUN 工作站上利用两个窗口分别运行了一个 X 应用程序，这两个应用程序进程都叫做 client，而此时机器中只有一个 sever 在运行。

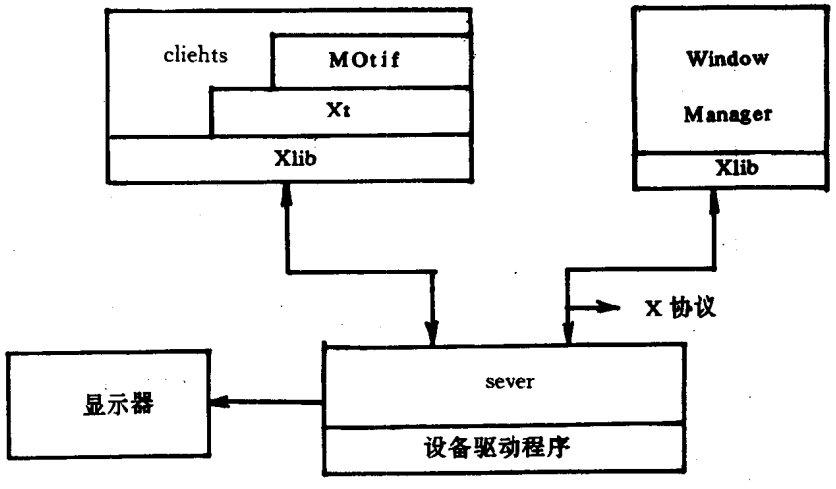


图 1.2

X-Window 将 X sever 中依赖硬件部分实现的大部分细节及它通过 client 所控制的硬件隐蔽起来了。只要 client 和 sever 均遵守 X 协议，那么任何 client 都可以与任何 sever 通信相联。这样便使 X client 程序具有设备独立性。图 1.2 说明了 X client 和 X sever 的通讯。其中 Xlib 和 Xt、Motif 均为 X 的用户接口库，Window Manager(窗口管理器)是一个 client 程序，它们在后面都将讲到。

X 之所以能够使用 client-sever 机制，其关键在于 X 协议，下面就让我们看看什么是 X 协议。

1.2 X 协 议

可以说，X 协议是 X 窗口系统的准确定义，而实现 X-Window 系统的任何语言代码都是 X 协议的实施。X 协议是一种异步双向字节流协议。下面让我们看看 X 协议如何工作。X 系统中 client 与 sever 的关系不是一种过程和系统调用的关系，它们是两个不同的进程，这一点一定要记住。比如说 client 进程现在需要在窗口中画一条线，它便向 sever 发出请求，即一个按照 X 协议要求格式的一个数据包。sever 进程从网络上收到这个包裹之后，在完成请求的同时，给予 client 一个应答。当然，并非所有的请求都需要应答，只有那些要求信息的请求才必须应答。比如恰在此时，client 程序的窗口大小改变了，于是 sever 又给 client 送去一个包裹，里面告之发生的事件。client 又出来一个请求，sever 打开一看，发现请求是不可执行的或存在某种错误，于是 sever 送给 client 一个错误类型数据包，一个错误很像一个事件，只是 client 对其处理方法与事件不同。X 协议指定的可在网络上传送的信息就是以上这四类：请求、应答、事件和错误。

一般地，client 程序实现 X 协议使用的是一个与单个网络协议接口的编程库，这些网络协议典型的有 TCP/IP 和 DECnet。sever 程序通常需要了解许多基础网络协议，以便于同时与多种网络上的客户程序通讯。X 协议设计成异步操作，主要是为了获得高性能。服务器发送事件也是异步的，这样允许那些要求得到连续输入的应用程序优先得到输入。

1.3 显示及屏幕

这小节主要是想解释一下以后常会碰到的两个词：显示(display)和屏幕(screen)。这两个词其英文意义很好理解，但在 X-Window 及 Motif 程序中碰到它们时，程序员常常会不理解它们的意义和两个词的区别。

在 X-Window 系统中 display 是指一个 X 服务器进程，而 screen 则指一个输出设备，是硬件。一个 display 可以支持很多 screen。一个 client 程序在与 X sever 通信之前必须首先打开与该 sever 的互联，然后它就可以使用由 display 所控制的任何 screen。

1.4 资　源

X-Window 系统中有许多资源。资源是一些数据结构，如前面所述，用户正是通过使用各种资源来设置各式各样的窗口界面的。资源是应用程序与工作站交互工作的基础。

图形上下文(GC)

这类资源是 X-Window 系统中最有用的资源，用来记录图形所具有的属性，如前景颜色、背景颜色、线宽、线型、文本字体和点模式。

字体(font)

这类资源含有用来在窗口显示文本的信息，一个字体描述一个字符集的尺寸和形状。

位象图(bitmap 和 pixmap)

这类资源以点的形式存放图形和图像。它在屏幕图形拷贝时很有用。比如我们现在在屏幕上画了一个复杂的图形，我们想让这幅图在屏幕上平移一定距离而又不想重画这么复杂的图形，那我们可以先将这幅图存放在一个位象图资源中，然后再将它拷贝到新的位置。

颜色图(colormap)

这类资源提供你可在你的程序中使用的颜色，使你的用户界面色彩斑烂，引人注目。

以上介绍的几种资源以及其他一些资源均由 X sever 控制管理，client 程序可以请求 sever 创建和毁坏资源。

1.5 事　件

在 X-Window 应用程序中，用户的一切输入，像移动或按下鼠标、在键盘上按下一键等等操作，均为窗口管理器所接受，并产生一个时间片事件，传送给应用程序，则该应用可能需要显式地处理每个鼠标按钮、鼠标移动和键击(包括空格、行删除和换行字符)。除此之外，应用程序窗口的一切变化，诸如你的窗口尺寸发生了变化，或以前被遮掩的窗口现在显露出来，这些均作为事件被窗口管理器所接收，传送给应用程序。一般地，我们在写应用程序时，都要将我们所感兴趣的事件设置在程序中，这样，窗口管理器便只传送我们感兴趣的事件，应用程序就使用这种方式来达到交互输入的目的。

1.6 输 入 设 备

X-Window 系统支持多种输入设备。根据实现的不同，一个服务器可以支持 tablets、track balls、扫描仪及其他数据输入和定位设备。但我们一般只使用两种输入设备：用于文本输入的键盘和用作定位及选择设备的鼠标。

1.7 窗口管理器

在 X-Window 系统中，窗口管理程序是一个普通的客户应用程序。它提供了一个特定的人机接口策略，它允许用户组织和重组织他们的应用窗口。而且，X 也提供了一些旨在让窗口管理程序控制窗口大小及位置的成分。在一般情况下，我们的应用程序在运行时，根窗口的外观特征由窗口管理器控制的。用户要想改变根窗口的特征，需要在命令行上设置特定的参数。

在通常的 X-Window 系统中，窗口管理程序往往不只一个，用户可以根据自己的喜好选择一种窗口管理器。

1.8 X 工具箱层次结构

所谓工具箱就是用户接口子程序库。HP 公司开发和资助的 Xlib 程序库是一很低级的程序库，利用 xlib 编程，每一步的细节都需要程序员来考虑，诸如与 X sever 建立互联，人为地循环查询事件等等，程序员必须对各种主要的数据结构很熟悉，而且一个方面考虑不周就会出错。熟悉微电脑的读者都知道，如果用汇编语言编程，程序员不仅需要对 CPU 内部结构了解清楚，而且需要时刻保持清醒周密的思维。利用 Xlib 编程就类似于微电脑上用汇编语言编程，开出一个窗口便需要上百条 C 语句。

工具箱的目的在于简化了应用用户接口的设计和开发，它们将 Xlib 集成化，提供更高级更直接的输入手段，提供许多完备的，功能强大而灵活的用户界面元素，如菜单(menu)、按钮(button)、对话框(dialog)等，它们被称为 Widget。X 系统的工具箱有许多。一般从各种库连接而成的一个典型的 X 工具箱应用是由四层构成的(见图 1.3)：

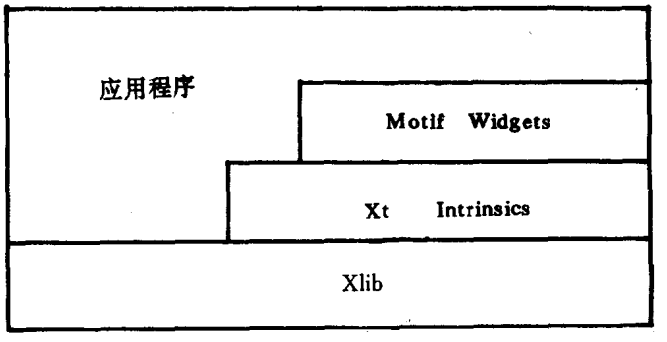


图 1.3 X 工具箱层次结构

- 最低层是 Xlib。Xlib 是标准 X 的一部分。
- 上一层是 X 工具箱的内部函数库，它也是标准 X 的一部分。
- 第三层由目标基(widgets)组成。目前流行着许多目标基集。我们最常见到的有 Motif，OpenLook，Xaw，Xview 等。
- 最顶层是应用代码。

1.9 小　结

本章介绍 X-Window 系统的发展历史、设计目标、关键技术和总体结构。

X-Window 是一个强有力的平台，它允许程序员开发高级的、移植性极强的应用程序。X 是网络透明的，clinet-sever 模型是 X 的构造基础。X 协议是联结 client 和 sever 纽带。X 服务器响应用户的请求，并且发送事件和错误。X 不限定界面风格，应用程序可以通过选择、设定资源来定义自己的界面。窗口管理程序仅仅是一个客户程序，它允许用户移动并操作窗口。在一个工具箱上构造程序比在 Xlib 级别上构造程序容易得多，Widgets 可以被组合起来创造出复杂的用户界面，其方法类似搭积木。

第二章　用 Motif Widgets 编程

从本章开始，我们讨论如何用 Motif Widgets 得到我们想要的用户界面。首先，我们熟悉一下 Motif。

什么是 Motif 呢？读者可能会对 Motif 的概念感到模糊。其实，Motif 是指一套程序库，它是建立在 Xlib 和 Xt 之上的。关于 Xlib 和 Xt，前一章里我们已经有了一个概况的印象。Motif 提供的 Widget 集是一个基于 Xt Intrinsics，可以使应用程序方便快速地访问 X-Window 系统低层的函数和过程，可以使应用程序方便快速地构造用户界面的工具包。Motif 提供的 Widgets 有菜单、按钮、对话框、文本输入区、显示区等，通过这些 Widget，Motif 提供了一套准则，规定了应用程序在屏幕上的显示方式和布局，还规定用户应该如何与应用程序打交道。

Motif 是由 OSF 设计开发的。OSF 是开放软件基金会的简称，它由 IBM、HP、Digtal 等公司联合创建。下面我们从一个小例子入手认识一下 Motif。

2.1　一个示例程序

本节举一个非常简单的 Motif 应用程序 Passwd 来说明如何写 Motif 应用程序。这个小例子在屏幕上开一个窗口并显示一个文本区，提示用户输入，同时将用户的输入与程序中内置的口令比较，若一致则程序退出，否则提示用户重新输入。

下面是 Passwd 程序：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include <Xm/MainW.h>
#include <Xm/Text.h>
#include <Xm/MessageB.h>
#include <Xm/PanedW.h>
#include <Xm/BulletinB.h>
#include <Xm/PushB.h>
#include <Xm/Xm.h>
```

```
Widget      mainw,text,suc,fail;
char        *ch=NULL;
String      string;
void        do_ok(Widget w,XtPointer cli,XtPointer call)
  {
            int n;
            Arg args[10];
            n = 0;
            XtSetArg(args[n],XmNvalue,&ch); n++;
            XtGetValues(text,args,n);
            if(! strcmp(ch,"fpr")) XtManageChild(suc);
            else XtManageChild(fail);
            XtFree(ch);
}
void        do_suc(Widget w,XtPointer cli,XtPointer call)
  {
            exit(0);
}
void        do_cancel(Widget w,XtPointer cli,XtPointer call)
  {
            int n; char ch[40];
            Arg args[2];
            n = 0;
                  sprintf(ch,"lr");
                  string = "";
                  XtSetArg(args[n],XmNvalue,string); n++;
                  XtSetValues(text,args,n);
                  XtFree(string);
}
main(int argc,char *argv[])
  {
            Widget toplevel,form,sp,bt_ok,bt_cal;
            int n;
            Arg args[10];
            char chh[40];
            XmString string;
            toplevel=XtInitialize ("passwd","LR",NULL,0, &argc,argv);
            mainw = XtCreateManagedWidget("MainWindow",
```

```
                    xmMainWindowWidgetClass,
                    toplevel,NULL,0);
        form = XmCreatePanedWindow(mainw,"Pane",NULL,0);
        XtManageChild(form);
        n = 0;
        XtSetArg(args[n],XmNeditable,XmSINGLE_LINE_EDIT);
                n++;
        text = XtCreateManagedWidget("Text",xmTextWidgetClass,
                                        form,args,n);
        XtAddCallback(text,XmNactivateCallback,do_ok,NULL);
        n = 0;
        sp = XtCreateManagedWidget("rowcol",xmBulletinBoardWidget
                                        Class,form,args,n);
        n = 0;
        bt_ok = XmCreatePushButton(sp," ok ",args,n);
        XtManageChild(bt_ok);
        XtAddCallback(bt_ok,XmNactivateCallback,do_ok,NULL);
        n = 0;
        XtSetArg(args[n],XmNx,150); n++;
        bt-cal = XmCreatePushButton(sp,"cancel",args,n);
        XtManageChild(bt_cal);
        XtAddCallback(bt_cal,XmNactivateCallback,do_cancel,
                        NULL);
        n = 0;
        sprintf(chh,"OK,match it");
        string = XmStringCreateLtoR(chh, XmSTRING_DEFAULT
                                        _CHARSET);
        XtSetArg(args[n],XmNmessageString,string); n++;
        suc = XmCreateInformationDialog(mainw,"SUCCESS",
                                        args,n);
        XtAddCallback(suc,XmNokCallback,do_suc,NULL);
        n = 0;
        sprintf(chh,"No match,retry");
        string = XmStringCreateLtoR(chh,XmSTRING_DEFAULT_
                                        CHARSET);
        XtSetArg(args[n],XmNmessageString,string); n++;
        fail = XmCreateErrorDialog(mainw,"FALUTURE",args,n);
//  XmTextFree(string);
```

```
    XtRealizeWidget(toplevel);
    XtMainLoop();
}
```

一般来讲，用 Motif Widgets 编程有如下几步：

- 包含所需头文件；
- 初始化 Xt Intrinsics；
- 增设一个顶端管理窗口；
- 设置 Widget 参数表；
- 建立 Widget；
- 增加回调例程；
- 实现 Widgets；
- 链接相应程序库和建立缺省文件。

下面我们以 Passwd 程序为例，在以下 9 个小节(2.1.1—2.1.9) 中简单介绍一下用 Motif Widgets 编程的步骤。

2.1.1 包含头文件

注意 Passwd 程序的以下几行：

```
#include <Xm/MainW.h>
#include <Xm/Text.h>
#include <Xm/MessageB.h>
#include <Xm/PanedW.h>
#include <Xm/BulletinB.h>
#include <Xm/PushB.h>
#include<Xm/Xm.h>
```

这是 Passwd 程序需要的 X 头文件。我们有过 C 编程经验的读者都知道在写 C 程序时需要包含若干相应的头文件。Motif 是遵循 C 语言接口的一组程序库，Motif 程序中特殊的专用变量类型都在其头文件中定义。在程序中每使用一种类型的 widget 都要在文件头加以相应的头文件说明。在程序中，即使对某一个 Widget 使用了多次，但其头文件只需包含一次。请注意不要只包含了 X 头文件，而忘记包含程序中可能用到的其他头文件(如 stdio.h)。这种狗熊拾棒子的错误在初学者中经常见到。

X 头文件的放置次序是很重要的，应遵循下面的格式：

1. 首先书写一般头文件，如〈stdio.h〉；
2. 再书写 Xlib 和 Xt Intrinsics 头文件，如〈X11/Intrinsic.h〉；
3. 最后书写 Motif 的头文件，程序中使用的每个 Widget 类有一个对应的头文件。Widget 类头文件的次序并不重要。

2.1.2 与 X sever 建立联接

先初始化 XtIntrinsics 才能调用 XtIntrinsics 函数实现用户界面。在 Passwd 程序中使用函数 XtInitialize 生成一个上下文(Content)关系,建立起与 Display 的联系。它对 Passwd 程序的命令行进行语法分析,将相应资源装入资源数据库,并且生成一个 shell widget 作为应用程序 widget 的父 shell。运行中,将应用程序的命令行作为参数传送给 XtAppInitialize,XtAppInitialize 对命令行进行扫描,并清除某些选择项,剩下的与用户界面相关的选择项被保留。然后该函数对相关选择项进行语法分析,定义某些资源并将相应资源装入资源数据库。在程序 Passwd 中 XtAppInitialize 是这样被调用的:

toplevel = XtInitialize ("passwd","Passwd",NULL,0,&arge,argv);

XtInitialize 返回一个 Widget 类型的值,因此,Passwd 中变量 toplevel 也必须定义为 Widget 类型。XtInitialize 的语法一般格式如下:

```
Widget XtInitialize (Shell _name, application _class, options, num options,
                     argc, argv)
Char    * shell name;
Char    * application _class;
Xrm Option Desc Ree   options[ ];
Cardinal   num _options;
Cardinal    * argc;
Char    * argv[ ];
```

我们对其各个参数的功能解释如下:

Shell _name:指出应用程序建立窗口的名字

application _class:规定本应用的类名

options:规定如何进行命令行的语法检查

num _options:指出选择表(options)中的项数

argc:其指针指向变量内容为命令行参数的个数

argv:命令行参数

除了 XtInitialize 函数以外,X 工具箱还提供了其他的函数建立与 X sever 的互联。

XtInitialize()是一个 convenience 函数,即它调用其他 Intrinsics 函数来初始化 Intrinsics 层,打开与 display 的互联,并创建一个 shell widget。

XtInitialize() 调用了三个其他 Intrinsics 函数,这些函数是:

XtToolkitInitialize(),初始化 XtIntrinsics,但它不打开 display,也不生成 shell widget。

XtOpenDisplay(),用给定的或缺省的上下文打开一个与 X sever 的互联。

XtAppCreateShell(),为应用程序创建一个顶层 shell widget。

2.1.3 创建 widget

在初始化工作完成之后,我们下一步可以创建 widget 了。就好比我们盖一幢楼房,先