Seamly2D - Release 版本打包发布流程

本文是如何将 GitHub 上的开源项目 Seamly 2D 在二次开发之后打包发布安装包的全流程记录。记录是基于以下文章开发背景的拓展，后面的所有流程都是基于在项目的 debug 成功编译跑通、开发完后续功能后，准备打包发布 Release 版本、数字签名和制作安装包时的后续流程。

> **前提环境：**
>
> Windows11从0开始在Qt6环境下跑通Seamly2D的全流程记录.docx

发布前的环境检查

检查所需软件

- 程序栏搜索打开 Visual Studio Installer，在弹出的窗口中找到 Visual Studio 生成工具，点右边修改；在弹出的窗口右侧找到安装详细信息，点击使用 C++ 的桌面开发左边的小箭头展开列表，在列表中确认 **Windows 10 SDK（版本号）**和 **MSVC v142 - VS 2019 C++ x64/x86 生成工具** 都已经安装好（被勾选），检查完毕后关闭窗口。
- 程序栏搜索打开 Qt Maintenance Tool → 登录 → 选择添加或移除组件 → 下一步 → 等待信息检索完毕（这步请关闭网络代理，要不然会检索失败）→ 在组件名称中展开 Qt，展开 Qt 6.5.3，检查 **MSVC 2019 64-bit**、**Qt 5 Compatibility Module** 和 **Qt Shader Tools** 是否都已经安装好（被勾选），进一步展开 Additional Libraries，检查 **Qt Multimedia** 是否已安装好（被勾选）→ 向下划找到并展开 Qt Creator，检查 **Qt Creator xx.x.x**、**CDB Debugger Support** 和 **Debugging Tools for Windows** 是否已安装好（被勾选），检查完毕后关闭窗口。

检查环境变量

- 打开文件资源管理器，在 **此电脑** 上右键，选择 **属性**，在弹出的界面中找到 **高级系统设置**，再点击 **环境变量**，在弹出来的窗口中 → 下面 **系统变量** 中找到 **Path** 并双击，检查下述目录是否存在，若不存在，将路径加入环境变量（以下仅供参考，需要写入实际所在路径）：

```
#（系统基础工具）
C:\Windows\system32
C:\Windows

#（WMI 工具）
C:\Windows\System32\Wbem

#（PowerShell）
C:\Windows\System32\WindowsPowerShell\r1.0
```

```
#（SSH 客户端）
C:\Windows\System32\OpenSSH

#（Git 版本控制）
C:\Program Files\Git\cmd

#（CMake 构建工具）
C:\Program Files\CMake\bin

#（Qt Creator IDE 本身）
C:\Qt\Tools\QtCreator\bin
```

注：MSVC 工具链（cl、link、nmake 等）通常不在 PATH 中直接配置，而是通过 Visual Studio 提供的开发者命令提示符来设置环境。环境变量中并没有直接包含 MSVC 的路径是正常的，因为 Visual Studio 不建议直接修改全局 PATH，而是通过调用它的脚本来配置环境。

检查 MSVC 工具链

- 在程序列表中找到 **x64 Native Tools Command Prompt for VS xxxx** 并打开。在弹出的命令提示符中输入以下命令确认工具链完备：

  检查编译器 (cl.exe)，输入：

  ```
  cl
  ```

  应返回如下：

  ```
  用于 x64 的 Microsoft (R) C/C++ 优化编译器 [版本号]
  版权所有(C) Microsoft Corporation。保留所有权利。

  用法: cl [ 选项... ] 文件名... [ /link 链接选项... ]
  ```

  检查链接器 (link.exe)，输入：

  ```
  link
  ```

  应返回如下：

  ```
  Microsoft (R) Incremental Linker Version [版本号]
  Copyright (C) Microsoft Corporation.  All rights reserved.

   用法: LINK [选项] [文件] [@commandfile]

    选项: ...
  ```

  检查 nmake（Makefile 工具），输入：

  ```
  nmake /?
  ```

  应返回如下：

  ```
  Microsoft (R) 程序维护实用工具 [版本号]
  版权所有 (C) Microsoft Corporation。  保留所有权利。
  ```

```
用法:  NMAKE @commandfile
       NMAKE [选项] [/f makefile] [/x stderrfile] [macrodefs] [targets]

选项: ...
```

检查 Qt 用的 qmake（Qt for MSVC），输入：

```
qmake -v
```

应返回如下：

```
QMake version 3.1
Using Qt version [版本号] in [绝对路径]
```

检查 Qt Creator 内配置

- 打开 Qt Creator，在欢迎界面中点击打开项目，选择 seamly2d 源代码路径下的 `Seamly2D.pro` 文件打开。项目加载完成后，在左边的图标列表中选择 **项目（扳手图标）**，点击左上角 **管理构建套件**。为防止冲突，在构建套件（Kit）中将 Debug 阶段所用的 `Desktop Qt 6.5.3 MSVC2019 64bit` 克隆一份（选中右边点击克隆），会发现在手动设置中多了一个 `Desktop Qt 6.5.3 MSVC2019 64bit clone`，选中这个 Kit，在下面的列表设置中检查编译器是否为 **MSVC 2019 v142 (BuildTools)**，调试器是否为 **Auto-detected CDB at C:\Program Files (x86)\Windows Kits\10\Debuggers\x64\cdb.exe**，Qt 版本是否为 **Qt 6.5.3 MSVC2019 64bit**，检查完毕后右下角应用确定即可。
- 在 **项目（扳手图表）**右边的 **构建设置** 中，第一行 **编辑构建配置** 右侧应默认为 **Debug**（因为之前一直在用 Debug 构建配置开发），现在开发完成准备发布，需要添加 Release 配置。

  点击 Debug 右侧的 **添加** 按钮，选中 **Release**，逐个检查下方设置：

  Shadow build 已勾选，**构建目录** 与 Debug 目录为同级目录（方便整理，放别处也行），Separate debug info 选择 **Default**，QML debugging and profiling 选择 **Enable**，Qt Quick Compiler 选择 **Default**，qmake system() behavior when parsing 选择 **Use global setting**。

  构建的步骤 中 qmake 右边点击 **详情** 展开，检查 qmake 构建配置为 **Release**，有效的qmake调用中存在 qmake.exe 和 Seamly2D.pro 的绝对路径，还有一系列构建参数，我这里是：

  ```
  C:/Qt/6.5.3/msvc2019_64/bin/qmake.exe C:\Users\123\Desktop\code\Seamly2D\Seamly2D.pro -spec win32-msvc "CONFIG+=qml_debug" && C:/Qt/Tools/QtCreator/bin/jom/jom.exe qmake_all
  ```

  然后检查 Make 右边是否有：`jom.exe in [绝对路径]` 下面 清除的步骤 中 Make 右边是否有 `jom.exe clean in [绝对路径]`，最后检查 Build Environment 中是否为 **使用系统环境变量**。

以上所有内容全部检查完毕后就可以回到项目编辑页面，点击左下角显示器图标的 **Debug**，然后双击选中刚配置好的 **Release**，看到显示器图标下面的字变成了 **Release** 后，就可以点击最左下角的 **小锤子图标** 开始编译 Release 版本了。在经过相当漫长的一段时间等待之后，Qt 会提示编译成功，这时候去刚才设置中的 **构建目录** 所在路径中就会发现多了一个文件夹，其名称默认为 `Desktop_Qt_6_5_3_MSVC2019_64bit_clone-Release`，双击点进这个文件夹，再依次点进 `src\app\seamly2d\bin` 文件夹，在 `bin` 文件夹中往下翻会找到已经编出来的 `seamly2d.exe`，直接双击应用程序，其理应能带着之前开发的功能正常运行。

使用 windeployqt 命令收集依赖

进入 `Desktop_Qt_6_5_3_MSVC2019_64bit_clone-Release\src\app\seamly2d\bin` 文件夹，在 `bin` 文件夹中往下翻找到 `seamly2d.exe`，复制这个文件（单独的 .exe 文件），粘贴到一个全新的自建路径（例如在桌面上新创建一个文件夹，其绝对路径为 `C:\Users\xxx\Desktop\Seamly2D_Release`）中。

现在这个文件夹中仅有一个文件（seamly2d.exe），在程序列表中搜索打开 **x64 Native Tools Command Prompt for VS xxxx**，在弹出来的命令提示符中依次输入：

```
# 进入刚刚创建的里面只有 .exe 一个文件的新目录
cd C:\Users\123\Desktop\Seamly2D_Release

# 使用绝对路径执行 windeployqt，不给环境冲突报错的机会
C:\Qt\6.5.3\msvc2019_64\bin\windeployqt.exe seamly2d.exe
```

等待一段时间收集依赖，windeployqt 运行完成后回到刚才新建的目录，发现里面多了一大堆依赖文件。此时 windeployqt 只收集打包了与 Qt 本身相关的依赖，还没有添加使用到的第三方库，程序暂时还无法运行。幸运的是，seamly2d 应用只调用了一个第三方库 `xerces-c_3_2.dll`， 这是 Apache Xerces-C++ 库的文件，是一个用 C++ 编写的 XML 解析器和工具包。鉴于 Seamly2D 是一个用 XML 格式储存设计数据的服装设计软件，显然，这个库是用来进行 XML 相关的处理操作的。

只缺少了一个第三方库就意味着手动复制粘贴即可，不需要使用额外的脚本程序。经过搜索，这个库藏在以下的相对路径中：

```
{你的项目源码路径\Seamly2D}\src\libs\xerces-c\msvc\lib
```

选中 `xerces-c_3_2.dll` 并 **复制**（不要剪切），回到刚才新建的依赖收集完成的新建路径中，将该第三方库粘贴进来。这时在新建目录中双击 `seamly2d.exe` 时，程序理应带着之前开发的功能运行。

部署数字签名证书（Windows）

签名证书相关信息

软件代码签名证书是由全球公认的、受信任的证书颁发机构颁发的一个数字文件。里面包含了公司信息和一个加密密钥对（一个私钥，一个公钥）。

- **私钥**：由公司严格保密，用来对软件安装包（`.exe`，`.dll`，`.msi` 等）进行“签名”。
- **公钥**：包含在证书里，随软件一起分发。

没有签名的软件，在用户下载运行时会遇到一大堆警告：

- **Windows：**“SmartScreen 筛选器” 警告，会弹出一个吓人的蓝色窗口，提示 “无法验证发布者。你确定要运行此软件吗？”。用户必须点击 “更多信息” → “仍要运行” 才能继续。
- **macOS：** 会提示 “无法打开 ‘XXX’，因为无法验证开发者”。用户需要进入系统设置，手动批准才能运行，体验非常差。
- **杀毒软件：** 没有签名的软件更容易被各大杀毒软件误报为病毒或恶意软件，直接被拦截删除。

至于为什么自己在自己电脑上安装时没有弹出，主要原因如下：

💡 **该软件是在 “本地环境” 下测试的。**

Windows 的 SmartScreen 安全机制主要依赖于**网络声誉系统**。

- **“声誉”的建立**：当一个全新的、没有签名的软件被成千上万的用户从互联网下载并安装时，Microsoft 的云端系统会开始跟踪它。如果在一段时间内（可能几天，可能几周）没有大量用户报告它是恶意的，它就会慢慢积累“声誉”，那个吓人的蓝色警告就会逐渐减少甚至消失。
- **本地情况**：因为在自己的电脑上，直接从本地硬盘运行安装包。对于本地操作，Windows 的限制会宽松很多。它默认你信任自己电脑上的东西。SmartScreen 的主要目标是从 **互联网下载** 的未知文件。

证书部署解决方案

企业商用

一般企业如果要发行商用经过二次开发的开源软件，是必须要购买商业代码签名证书的，不可使用自签名证书，不然可能会丢失商业信誉、安全标准甚至可能承担法律责任。

企业应该选择哪种商业证书?

1. **OV (Organization Validation) 代码签名证书**
   - **适用场景**：绝大多数企业的标准选择。
   - **特点**：证书颁发机构（CA）会严格验证企业的真实性和合法性（如检查工商注册信息、公司电话等）。签名后软件会清晰显示公司名称。
   - **价格**：通常在每年几千元人民币的范围。
2. **EV (Extended Validation) 代码签名证书**
   - **适用场景**：大型企业、金融软件、安全敏感型软件。
   - **特点**：**验证最严格**（可能需要提供更多法律文件），价格最贵。它的最大优势是能立即建立微软 SmartScreen 信誉。用户从下载第一天起就不会看到警告。
   - **硬件要求**：私钥存储在专用的 USB 硬件令牌中，安全性极高，无法被导出和复制。
   - **价格**：更贵，通常在每年上万元人民币的范围。

对企业开发者的建议流程：

1. **购买证书**：从 DigiCert、Sectigo、GlobalSign 或其授权代理商处购买 OV 代码签名证书（或 EV 代码签名证书，视预算和需求而定）。
2. **完成验证**：配合 CA 提交公司资料（营业执照等）完成验证。
3. **安全存储**：收到证书后，将私钥妥善备份并加密存储。若是 EV 证书，硬件令牌必须保管好。
4. **集成签名**：将签名命令集成到 CI/CD（持续集成/部署）流程中（如Jenkins、GitLab CI），确保每一个发布版本的安装包都自动经过签名。
5. **分发发布**：将签好名的软件分发到官网、各大应用商店和合作伙伴。

个人小规模开发

如果是个人开发者，也不会把这个软件卖钱商用 /doge，只想让大家正常安装使用。这是一个非常典型且合理的需求。对于个人项目、二次开发的软件或者免费小工具，花几千块买证书确实没必要。这完全没问题，有 “零成本” 的解决方案。虽然无法像付费证书那样直接显示可验证的公司名，但足以极大地改善用户体验，避免那些吓人的警告。

最佳零成本方案：使用 Windows 的 "自签名证书"。

这个方法的原理是自己充当 "证书颁发机构（CA）"，给自己发一个证书。然后让用户 "安装并信任" 自己充当的 "机构"。优点是一旦用户信任了你的根证书，所有用这个证书签名的软件，在对方的电脑上都会被视为可信任的，**不会再出现 SmartScreen 警告**，只会提示 "发布者：未知"，但不会阻止运行。而缺点就是需要用户多一个 "安装根证书" 的步骤。

鉴于本人的任务是跑通 编译 - 打包 - 签名 - 发布 流程，这里就用第二种方式也就是使用 **自签名证书** 尝试给已经经过二次开发的 Seamly2D 软件部署证书。

自签名证书部署流程

鉴于开发平台为 Windows，我们用 Windows 自带的 PowerShell 生成自签名证书。

步骤 1：以管理员身份运行 PowerShell

- 按 `Win + S` 在弹出来的搜索框输入 `PowerShell`。
- 在搜索结果中的 "Windows PowerShell" 的右侧列表中，选择 **"以管理员身份运行"**。这一步非常重要，否则可能没有权限操作证书存储。

步骤 2：生成根证书 (CA Certificate)

首先，在 C 盘中新建一个名称为 `MyCertificates` 的文件夹（别处也行，写命令时同步更改即可）。

在刚打开的管理员 PowerShell 窗口中，按从上到下的顺序逐个输入每条命令（不带注释）：

```
# 1. 为根证书创建一个变量，证书会直接安装到本地计算机的"个人"存储区
# -Subject 参数：设定根证书的通用名，CN= 后面可以改成其他名字
# -KeyUsage 参数：指定这个证书用途是给其他证书签名
# -NotAfter 参数：设定根证书有效期 10 年
$rootCert = New-SelfSignedCertificate -CertStoreLocation "Cert:\LocalMachine\My" -Subject "CN=MyPersonalRootCA" -KeyUsage CertSign -KeyUsageProperty Sign -KeyAlgorithm RSA -KeyLength 2048 -NotAfter (Get-Date).AddYears(10)

# 2. 将刚生成的根证书导出为一个 .cer 文件，方便分发给用户
# -Cert 参数：设定使用上面生成的根证书变量
# -FilePath 参数：设定证书文件的保存路径和文件名，如果是其他路径需做更改
# -Type 参数：设定导出类型为证书
Export-Certificate -Cert $rootCert -FilePath "C:\MyCertificates\MyPersonalRootCA.cer" -Type CERT
```

执行后会看到刚才创建的文件夹，也就是在 `C:\MyCertificates` 文件夹中会出现一个 `MyPersonalRootCA.cer` 文件，这个就是根证书，将来需要与软件交给用户一并安装。

此外，系统会显示新证书的**指纹**（一长串数字字母序列）。

- 如果不小心关掉了 PowerShell 窗口没看到证书的指纹可以重新打开输入以下指令查看指纹：

  ```
  Get-ChildItem -Path "Cert:\LocalMachine\My" | Where-Object { $_.Subject -eq "CN=MyPersonalRootCA" } | Format-List Subject, Thumbprint, NotAfter
  ```

  返回的内容中的 Thumbprint 就是根证书的指纹，我这里是：

  ```
  Thumbprint : F2F8F3F1BC10FA2B100793F03CD262691F2E5E06
  ```

步骤 3：生成代码签名证书 (Code Signing Certificate)

继续在同一个 PowerShell 窗口逐个输入每条命令（不带注释）：

```
# 1. 指定正确的根证书指纹
# 这里仅是示例，实际请替换成根证书的实际指纹
$rootCertThumbprint = "F2F8F3F1BC10FA2B100793F03CD262691F2E5E06"

# 2. 在证书库中指定用该指纹的根证书并赋值到变量内
$rootCert = Get-ChildItem -Path "Cert:\LocalMachine\My\$rootCertThumbprint"

# 3. 用刚才确定的根证书，来颁发一个代码签名证书
# -Subject 参数：这里可以写软件名，如“Seamly2D Redisigned”
# -KeyUsage 参数：指定这个证书用途是数字签名
# -Type 参数：证书类型为代码签名
# -NotAfter 参数:有效期5年
# -Issuer 参数：指定颁发者是我们刚才创建的根证书
$signingCert = New-SelfSignedCertificate -CertStoreLocation "Cert:\LocalMachine\My" -Subject "CN=Seaml
y2D Redisigned" -KeyUsage DigitalSignature -Type CodeSigning -NotAfter (Get-Date).AddYears(5) -Signer $r
ootCert

# 4. 获取新证书的指纹ID（系统会自动显示，但再用变量存一下以防万一）
$thumbprint = $signingCert.Thumbprint

# 5. 为.pfx文件设置一个密码（请将 "SecurePassword123" 替换成别的复杂密码）
# -String 参数后面用双引号括住的是签名证书的密码，忘记后无法找回，只能重新创建新证书，牢记。
$password = ConvertTo-SecureString -String "SecurePassword123" -Force -AsPlainText

# 6. 将代码签名证书导出为 .pfx 文件（包含私钥），这个文件用于给自己的软件签名
# -Cert 参数：通过指纹找到证书
# -FilePath 参数：设定导出路径和文件名，请确保路径一定存在
# -Password 参数：使用上面设置的密码保护 .pfx 文件
Export-PfxCertificate -Cert "Cert:\LocalMachine\My\$thumbprint" -FilePath "C:\MyCertificates\MySoftware.p
fx" -Password $password
```

执行后会看到在 `C:\MyCertificates\` 文件夹里多了 `MySoftware.pfx` 文件。

至此，证书制作部分全部完成，得到了两个核心文件：

1. `C:\MyCertificates\MyPersonalRootCA.cer` → **与软件一起打包，让用户一并安装的文件**
2. `C:\MyCertificates\MySoftware.pfx` → **自己用来给开发完的程序签名的文件**

使用 Inno Setup 制作软件安装包

Inno Setup 是一个免费的、功能强大且非常流行的 Windows 安装包制作工具。

最后安装包的效果是：用户双击后，会出现一个熟悉的安装界面（下一步、下一步、选择安装目录、创建快捷方式），最后将你的所有文件安装到 `Program Files` 目录中。

操作流程

下载并安装 Inno Setup

打开浏览器，输入官网网址 https://jrsoftware.org/isdl.php（国内加载慢，可能需要网络代理），在打开的网页中部找到 Filename 下的 innosetup-6.5.1.exe 右边的 Download Sites 下的 **US**，点击等待下载完成。下载完成会看到一个名字为 **innosetup-6.5.1.exe** 的文件，双击安装，安装过程很简单，不用动任何配置，一路 next 最后 install、finish 即可。

创建安装脚本及安装包

首先，因为要给软件签名，需要使用 Windows 自带的 `signtool.exe`，在电脑上搜索其路径，它的常见路径是：`C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x86\signtool.exe` 记住这个路径，下面脚本里要用到。

打开 Inno Setup（刚安装完会自己打开），会弹出一个 Welcome 窗口，选择 **Create a new empty script file**，然后右下角 **OK**。如果没弹出 Welcome 窗口，点击左上角 file → new 即可。在空白脚本框中输入以下脚本，用来给软件制作安装包：

```
#define MyAppName "Seamly2D-Redisigned" ; 【必改】软件名称
#define MyAppVersion "1.0"  ; 【必改】软件版本号
#define MyAppPublisher "图灵人工智能研究院" ; 【必改】发布者名称
#define MyAppURL "https://www.ainanjing.org.cn/" ; 【可选】网站
#define MyAppExeName "seamly2d.exe" ; 【必改】主程序文件名

; --- 定义签名工具和证书的路径 ---
#define SignToolPath "C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x86\signtool.exe"
#define PfxPath "C:\MyCertificates\MySoftware.pfx"
#define PfxPassword "SecurePassword123"
; --- 签名定义结束 ---

[Setup]
; 注: AppId的值为单独标识该应用程序。
; 不要为其他安装程序使用相同的AppId值。
; (生成新的 GUID，点击 Tool → Generate GUID 即可)
AppId={{E3CF9F5C-4EF1-4C59-B7F2-0D3A6E13E0A9} ; 【可选】可以不用改，或用工具生成一个新的
AppName={#MyAppName}
AppVersion={#MyAppVersion}
AppVerName={#MyAppName} {#MyAppVersion}
AppPublisher={#MyAppPublisher}
AppPublisherURL={#MyAppURL}
AppSupportURL={#MyAppURL}
AppUpdatesURL={#MyAppURL}
; 安装目录 ("{pf}" 代表 Program Files, "{cf}" 代表 Program Files\Common Files)
DefaultDirName={autopf}\{#MyAppName}
; 关闭安装前是否允许用户修改安装目录
DisableDirPage=no
; 禁用“选择开始菜单文件夹”页
DisableProgramGroupPage=no
; 输出安装包的名字
OutputBaseFilename=Seamly2D_Setup
; 安装包压缩模式
Compression=lzma
SolidCompression=yes
; 安装包图标（可选，可以指向一个ico文件）
;SetupIconFile=C:\Path\To\Your\Icon.ico
; 安装过程中的窗口图标（可选）
;WizardSmallImageFile=C:\Path\To\Your\SmallIcon.bmp

[Languages]
Name: "english"; MessagesFile: "compiler:Default.isl"
; 如果做中文安装界面，可以下载中文语言文件并添加如下一行：
; Name: "chinesesimplified"; MessagesFile: "compiler:Languages\ChineseSimplified.isl"

[Tasks]
; 创建桌面图标任务
Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked
```

```
[Files]
; 这是最重要的部分！告诉安装包要把哪些文件打包进去，以及安装到哪里。
; 上面 windeployqt 整理出来的源文件所在路径：这里是 C:\Users\123\Desktop\Seamly2D_Release
; 目标路径：{app} 代表用户选择的安装目录
Source: "C:\Users\123\Desktop\Seamly2D_Release\*"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs c
reateallsubdirs
; 注意: 不要使用“Flags: ignoreversion”于任何系统文件。 recursesubdirs createallsubdirs 会包含所有子文件夹和
文件。

; --- 将根证书文件添加到安装包 ---
Source: "C:\MyCertificates\MyPersonalRootCA.cer"; DestDir: "{app}"; Flags: ignoreversion deleteafterinstall

[Icons]
; 创建开始菜单快捷方式
Name: "{autoprograms}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"
; 创建桌面快捷方式 (根据用户选择)
Name: "{autodesktop}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"; Tasks: desktopicon

[Run]
; --- 安装后运行命令：为当前用户安装根证书 ---
Filename: "certutil"; Parameters: "-user -addstore ""Root"" ""{app}\MyPersonalRootCA.cer"""; \
    Description: "安装根证书以避免安全警告"; \
    Flags: postinstall nowait skipifsilent; \
    StatusMsg: "正在安装信任证书..."
; 可选：安装完成后自动运行程序
; Filename: "{app}\{#MyAppExeName}"; Description: "{cm:LaunchProgram,{#StringChange(MyAppName, '&',
'&&')}}"; Flags: nowait postinstall skipifsilent

[Code]
procedure CurStepChanged(CurStep: TSetupStep);
var
  ResultCode: Integer;
  SignToolPath: string;
  PfxPath: string;
  PfxPassword: string;
begin
  if CurStep = ssPostInstall then
  begin
    SignToolPath := 'C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x86\signtool.exe';
    PfxPath := 'C:\MyCertificates\MySoftware.pfx';
    PfxPassword := 'SecurePassword123';

    // 签名主程序
    Exec(SignToolPath,
      'sign /tr http://timestamp.digicert.com /td SHA256 /fd SHA256 /f "' + PfxPath + '" /p "' + PfxPassword + '"
"' + ExpandConstant('{app}\{#MyAppExeName}') + '"',
      '', SW_HIDE, ewWaitUntilTerminated, ResultCode);

    // 签名所有 DLL 文件
    Exec(SignToolPath,
      'sign /tr http://timestamp.digicert.com /td SHA256 /fd SHA256 /f "' + PfxPath + '" /p "' + PfxPassword + '"
"' + ExpandConstant('{app}\*.dll') + '"',
      '', SW_HIDE, ewWaitUntilTerminated, ResultCode);
  end;
```

```
end;

// 可选：安装完成提示
procedure CurPageChanged(CurPageID: Integer);
begin
  if CurPageID = wpFinished then
  begin
    MsgBox('已为您的用户账户安装安全证书。', mbInformation, MB_OK);
  end;
end;
```

重要修改：

- 将 `#define SignToolPath` 后面的路径替换为电脑上 `signtool.exe` 的**真实路径**。
- 将 `#define PfxPath` 后面的路径替换为生成的 `MySoftwareSigner.pfx` 文件的**真实路径**。
- 将 `#define PfxPassword` 后面的 `YourPasswordHere` 替换为创建 `.pfx` 时设置的**密码**。

点击菜单栏上的 File → Save，给脚本起个名字（例如 `seamly2d_setup.iss`）并保存。然后点击菜单栏上的 Build → Compile。编译过程中下方会有一个进度条显示进度。成功后，它会提示 “*** Finished. [时间戳，总用时]”。Inno Setup 会默认输出安装包 `Seamly2D_Setup.exe` 到脚本文件（.iss 文件）所在的同一目录下的 `Output` 文件夹中。

手动给安装包进行签名：按 `Win + R` 输入 cmd 并回车，在命令提示符内输入：

```
# 划至最右，最后一个参数需要修改，填入的是创建出来的安装包 Seamly2D_Setup.exe 的绝对路径
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x86\signtool.exe" sign /tr http://timestamp.digicert.com /td SHA256 /fd SHA256 /f "C:\MyCertificates\MySoftware.pfx" /p "SecurePassword123" "[绝对路径]\Seamly2D_Setup.exe"
```

返回如下内容，说明安装包已经被签名成功，可以分发了：

```
Done Adding Additional Store
Successfully signed: F:\XJTLU\\\Seamly2D_addon\Output\Seamly2D_Setup.exe
```

如果觉得手动安装证书较为麻烦，可以写一个 .bat 脚本，让 .iss 文件编译签名自动化完成。

验证签名是否成功

右键点击安装包 → **属性** → **数字签名**。如果这里有一个签名列表，并且点击 “详细信息” 后显示 “已处理证书链”，那就成功了（颁发机构会是 “未受信任”，这是正常的，因为用的是自签名证书）。

至此一个带签名的安装包就制作完成了。

验证安装包

双击制作好的 `Seamly2D_Setup.exe` 安装包，根据引导一步步安装，最后会弹出提示：已为用户安装安全证书，至此证书就都已经部署好了。找到安装时安装的路径（我这里安装到了 `D:\Program Files (x86)\Seamly2D-Redisigned` 路径下），找到 seamly2d.exe 文件右键 → **属性**，发现程序已经带上了数字签名。双击程序，程序能正常运行，至此发布流程结束。

后记

其实按理说应该将做好签名的安装包放在一个什么环境都没有（无 Visual Studio、无 Qt Creator）的 Windows 环境中进行安装验证，但是我手头没有别的 Windows 设备了，条件受限。不过经过这一通折腾，实测在我电脑上编译打包、证书注入、安装运行都是没什么问题的，也就先写到这里。