OPEN CASCADE 学习笔记
——度量程序执行的进度

著: Roman Lygin

译： George Feng

这是一篇关于开源三维建模软件 OPEN CASCADE 内核的博文：ROMAN LYGIN 是 OPEN CASCADE 的前程序开发员和项目经理，曾经写过许多关于该开源软件开发包的深入文章，可以在他的博客 (HTTP://OPENCASCADE.BLOGSPOT.COM) 上面找到这些文章。

序

在 Open Cascade 的论坛上知道了 Roman Lygin 在他的博客上写了 Open Cascade notes 系列文章，考虑到 Open Cascade 的学习资料并不多，于是从他的博客上下载了其中绝大部分文章，将其翻译过来以方便大家学习交流。如果大家发现文中翻译有错误或不足之处，望不吝赐教，可以发到我的邮箱 fenghongkui@sina.com.cn ，十分感谢。

2012 年 11 月 22 日星期四

第 1 节 OCC 中程序执行的进度

在生活中为自己确定一个可以测度的目标非常重要，这样你就可以不断的检查自己的进度。程序在执行需要花费很长时间的操作时也应该能够显示当前的状态，否则用户可能认为程序死掉了，从而杀死程序。

Open CASCADE 提供了一种方法显示你的应用程序还在执行。它就是 Message_ProgressIndicator ，这是一个用句柄操作的类，可以将该句柄传递给其他算法。当前，IGES 和 STEP 转换器支持该类，BRep 读取器也支持该类 (从 6.3 开始)。好了，虽然讲的不多，但是也是个很好的开始。OCC 团队可能想要

将它扩展到消耗 CPU 比较高的算法中 (例如 Boolean 运算和其他算法)。

但是，借助它丰富的能力，也可以将它用在自己的算法中。

虽然如此，让我们还是研究的更加深入一些吧。

Message_ProgressIndicator 扩展了进度条这个简单概念，成为范围在域 [min, max] 上的整数容器，并具有当前值。

首先比较好的事情是 Message_ProgressIndicator 提供了 double 类型(而不是 int)的范围，并且可以自定义每一步的大小值。

```
Handle(MyProgressIndicator) anIndicator = new MyProgressIndicator;
Standard_Real aStartAngle = 0.5 * PI;
Standard_Real anEndAngle = 1.5 * PI;
Standard_Real aDelta = PI / 6.;
anIndicator->SetRange (aStartAngle, anEndAngle);
anIndicator->SetStep (aDelta);

for (Standard_Real aCurrentAngle = aStartAngle; aCurrentAngle <= aStartAngle; aCurrentAngle += aDelta, anIndicator->Increment()) {
...
}
```

要在全局范围内获取当前值，应该调用函数 Message_ProgressIndicator::GetPosition() ，该函数会返回范围在 [0, 1]之间的一个值。

另外一个比较好的事情是进度条 (indicator) 支持嵌入到某些区域中。这非常有用，假如你的操作是一个非常耗时的操作的一部分，它可能不知道整体的进度。例如你的算法正在进行可视化显示 (AIS_InteractiveObject 的子类), 可能是一个复杂算法的最后一步或者只是从文件中恢复模型。

一个操作可以分为许多子操作， 每一个子操作具有一个子段， 这样子操作就可以报告其在子段中的进展。进度条会将子段中的局部值映射为全局范围的值。 (An operation can allocate a sub-range for its sub-operations so that they report

progress within their sub-ranges. The progress indicator will take care to map local value into a global range.)

```
void MyLongOperation ()
{
Handle(Message_ProgressIndicator) anIndicator = new MyIndicator;
anIndicator->SetRange (0, 100); //100% complete

anIndicator->NewScope (60., "The longest suboperation"); //first sub operation takes 60%
MyLongSubOperation1 (anIndicator);
anIndicator->EndScope ();

anIndicator->NewScope (30.); // 30%
MyLongSubOperation2 (anIndicator);
anIndicator->EndScope ();

anIndicator->NewScope (10.); // 10%
MyLongSubOperation3 (anIndicator);
anIndicator->EndScope ();

}

void MyLongSubOperation1 (const Handle(Message_ProgressIndicator)& theIndicator)
{
Standard_Integer aNbSteps = ...; //small chunks of work
theIndicator->SetRange (0, aNbSteps);
for (int i = 1; i <= aNbSteps; i++, theIndicator->Increment()) {
...
}

}
```

在上面的代码中，全局范围被划分成 3 个不相等的部分，其与估计的每一个 MyLongSubOperation...() 函数执行的时间成正比， 每一个函数都在子段内执行。

这样 MyLongSubOperation1() 中的 SetRange() 仅仅影响为其分配的子段。

第三, 子段可以有他们自己的名字, 显示控件可以使用该名字从而显示当前阶段。名字可以利用函数 SetName() 设置或者直接在 NewScope() 中设置。

用户操作可以发出指令利用标识器打断操作。重新定义和使用虚函数 UserBreak() 。

还有其他一些好的功能特征, 例如支持无限区域(infinite regions) 或者 Message_ProgressSentry 类, 该类的封装非常方便, 而且当标示器为空句柄时特别有用。例如可以查看 IGESToBRep_CurveAndSurface.cxx 文件获知其使用方法。

设计自己的进度标示器

Message_ProgressIndicator 是一个抽象类, 你必须继承该类, 然后重新定义两个纯虚函数—— Show() 和 UserBreak() 。它们定义了进度标示器应该如何显示, 以及用户是否发出取消操作的消息。下一节将介绍一个这样的例子。

(待续… ...)

POSTED BY ROMAN LYGIN AT 23:51, 2009-01-27 

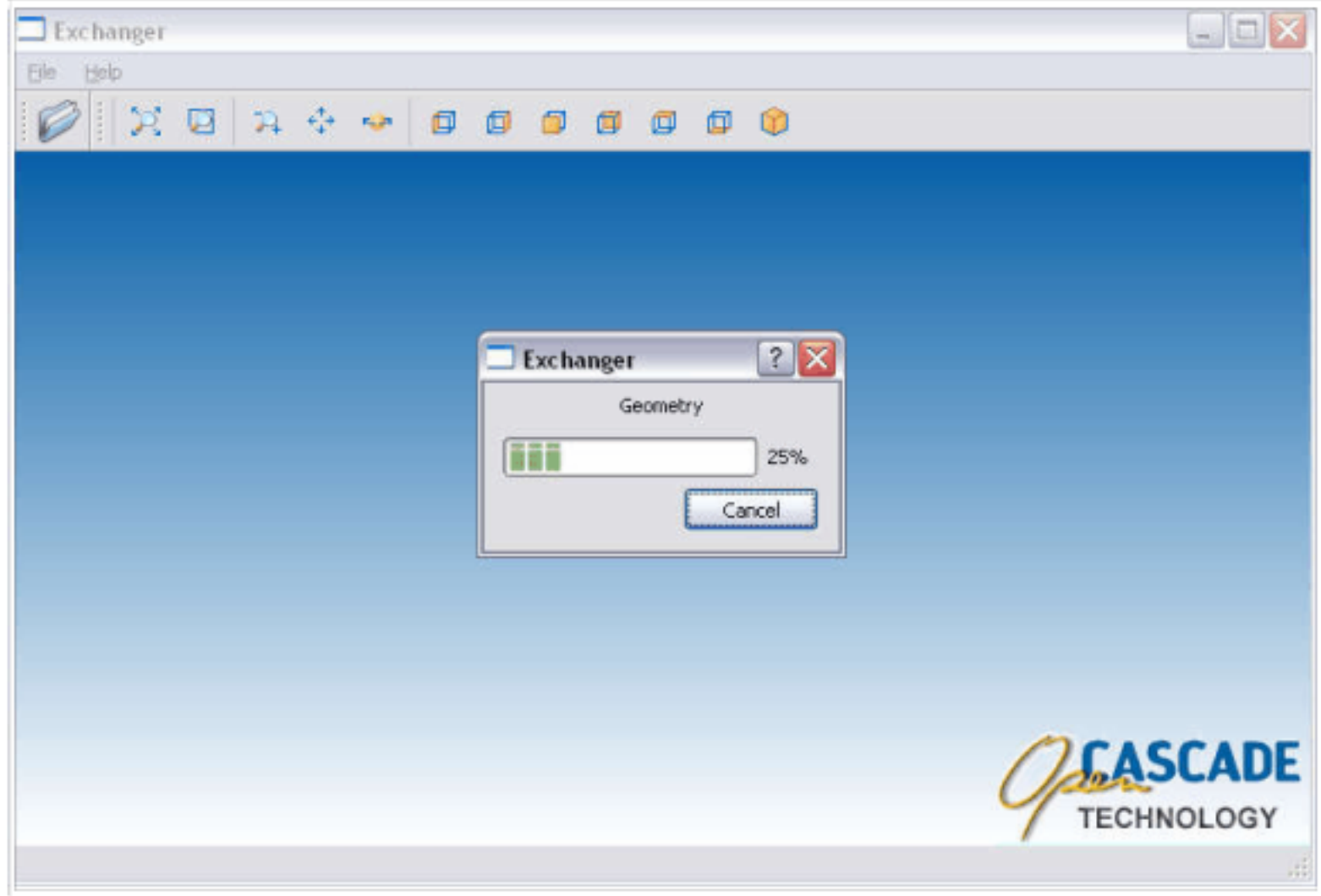
第 2 节 GUI 显示进度

(接上节)

GUI 显示进度

从 Message_ProgressIndicator 继承类, 从而与自己的应用程序 GUI 工具包结合起来工作。另外一种方法是提供简单的文字输出 (例如, 用于调试目的)。不管哪一种方法, 都必须重新定义 Show() 和 UserBreak() 方法。

在我的 QOLib(Qt/Open CASCADE Library) 中, 我使用 Qt 的 QProgressDialog 控件来显示进度条和取消按钮。 你可以使用自己的包含有 QProgressBar, QLabel 和可选项 QButton 控件。



下面是一些来自于　QLib 的代码片段：

```
class QOBase_ProgressIndicator : public Message_ProgressIndicator
{
public:
//!　生成对象。
Standard_EXPORT QOBase_ProgressIndicator (QWidget* theParent,
int theMinVal = 0, int theMaxVal = 100, Qt::WindowFlags theFlags = 0);

//!　删除对象。
Standard_EXPORT virtual ~QOBase_ProgressIndicator ();

//!　更新当前的对象。
Standard_EXPORT virtual Standard_Boolean Show (const
Standard_Boolean theForce);

//!　假如用户按了取消处理按钮，则返回　True。
Standard_EXPORT virtual Standard_Boolean UserBreak();

protected:

QProgressDialog* myProgress;
```

```
public:
DEFINE_STANDARD_RTTI(QOBase_ProgressIndicator)
};

/*!  使用特定的参数生成控件从而初始化  QProgressIndicator 。
theMin 和 theMax 也用于设置进度标识器的范围。
*/
QOBase_ProgressIndicator::QOBase_ProgressIndicator (QWidget* theParent,
int theMinVal, int theMaxVal,
Qt::WindowFlags theFlags)
{
QOLib_ASSERT (theMinVal < theMaxVal);
myProgress = new QProgressDialog (theParent, theFlags);
myProgress->setWindowModality (Qt::WindowModal);
myProgress->setMinimum (theMinVal);
myProgress->setMaximum (theMaxVal);
myProgress->setMinimumDuration  (500);  // 操作时间 >500ms 将会弹出对话
框

SetScale (theMinVal, theMaxVal, 1); //  同步更新 Qt 和 Open CASCADE 的范围
}

/*!  销毁关联的进度对话框。  */
QOBase_ProgressIndicator::~QOBase_ProgressIndicator ()
{
if (myProgress) {
delete myProgress;
myProgress = 0;
}
}

/*!  根据当前的进度更新可视化显示。
文本标识符的内容将根据当前步骤的名字更新。
总是返回 TRUE，从而表明显示已经被更新了
*/
```

```
Standard_Boolean QOBase_ProgressIndicator::Show (const
Standard_Boolean theForce)
{
Handle(TCollection_HAsciiString) aName = GetScope(1).GetName(); // 当前
步骤
if (!aName.IsNull())
myProgress->setLabelText (aName->ToCString());

Standard_Real aPc = GetPosition(); //always within [0,1]
int aVal = myProgress->minimum() + aPc *
(myProgress->maximum() - myProgress->minimum());
myProgress->setValue (aVal);
QApplication::processEvents(); //     重新绘制并保持 GUI 响应

return Standard_True;
}

/*! 返回 True , 假如用户已经按了窗口 QProgressDialog 中的取消按钮
*/
Standard_Boolean QOBase_ProgressIndicator::UserBreak()
{
return myProgress->wasCanceled();
}
```

Open CASCADE 移植了 Tcl/Tk 的实现 (参见 Draw_ProgressIndicator 类)，但是不幸的是 6.3.0 版本中特意删除了 XProgress Draw 命令，原本你可以利用这个命令激活它， 并使用内部的绘制会话 (use inside the Draw session) 。我把这个告诉了 OCC 团队，希望他们在将来的发布版本中恢复成原来的样子。假如你不能忍受，你可以从以前的版本中复制过来。

POSTED BY ROMAN LYGIN AT 23:06, 2009-01-28

RATE THIS POST:

2 COMMENTS:

1. 

QbProg January 28, 2009 11:56 PM

你好 ,

我已经在工程了使用了相似的框架 (对 OCC 非常感兴趣的是可以有设置子段), 最近我对框架也做了一些改进。

- 由于 Vista 的进度条前进的快慢依赖于更新的频率, 在两个更新之间我等了至少 100 毫秒 (我使用两个线程来完成这个任务)。

- 有时候我并不知道一个操作花费的时间 (例如当调用布尔操作时)。所以当工作线程结束之后, 我使用具有下拉框 (Marquee) 的工作线程。

或者下拉框 (Marquee) 可以与无限范围绑定。

- 你对多内核操作比较熟悉。在这种情况下, 你需要多个 “ 子段 ”, 然后将这些子段的进展累加起来来表明所有任务的总进展。 这是基抽象类应该处理的东西 !

再见 !
QbProg

Reply

2. 

Roman Lygin January 29, 2009 10:12 AM
Hi QbProg!

#1. 是的, 在之前版本的 QOLib 中(直到过了一段时间之后, 可视化工具条才更新)我也有同样的缓存问题, 但是只要用户不调用 Show() 太频繁 (因为计算当前的时间以及与增量比较会增加额外的费用)就可以消除这个错误。 QProgresBar 具有这样的机制即阻止第一次弹出太快 (我设置其为 500ms) 。

#2 我也不知道我理解的 "marquee" 是否正确。你能否更为详细的描述它 ?

各个工程的操作时间长短的计算是非常困难的。 我一般将各个操作映射到很多子段中。例如, 导入一个模型分成 80%的部分为导入本身, 5%用于更新数据模型, 15%用于可视化显示。在这 80% 的范围内 (假如整体为 100%的话), 又将其中的 30%用于装载模型, 70%的部分用于转换和修正

模型。这样进度条就会绘制的非常平滑。　你可以多实验几个典型的例子来设置各个子段的百分比。

#3　关于多线程。当前的实现不支持进度条显示　(在第三部分中我将继续介绍)。好消息是 OCC 正在实现关于它的原型系统,　所以很快这个功能会出现在 OCC 中。

多谢关注评论。

Reply