КАК СТУДЕНТЫ МГТУ ИМ. Н.Э. БАУМАНА ПРИДЕЛАЛИ "СИГНАЛИЗАЦИЮ" к nanoCAD SDK И СТАЛИ РАЗРАБОТЧИКАМИ В КРУПНОЙ РОССИЙСКОЙ ИТ-КОМПАНИИ

Как добавить компиляцию метаданных Qt (moc.exe) в проект Visual Studio? Двое студентов МГТУ им. Н.Э. Баумана – Арсений Пронин и Илья Климов – готовятся стать ИТ-специалистами в области проектирования и разработки программного обеспечения. Проверить полученные знания на практике будущие программисты смогли в ходе стажировки в компании-разработчике инженерного ПО "Нанософт разработка". Благодаря открытому API Платформы nanoCAD стажеры не только собрали нужную компиляцию с примерами SDK nanoCAD, но и создали на ее основе новый проект Visual Studio. По результатам успешной стажировки студенты были зачислены в штат разработчиков компании.

Погружение в API nanoCAD

Проект однокурсников преследовал две цели:

- дополнить компиляцией метаданных текущий проект из SDK с помощью расширения Qt VS Tools и без него;
- создать проект с нуля, используя те же исходники.

На первом этапе студенты изучили API с помощью учебного курса по программированию на .NET и NRX в Платформе nanoCAD. Полученные знания помогли разобраться в технике работы с чертежом через API. Обратную связь по ходу изучения программы обеспечивал руководитель проекта в рамках стажировки Илья Слободин, возглавляющий группу поддержки API.

Авторы проекта – Арсений Пронин и Илья Климов

Следующим шагом стала реализация диалогового окна для копирования объектов чертежа на NRX с использованием Qt – мощного кроссплатформенного инструмента для создания приложений и графического интерфейса. При решении поставленной задачи требовалось разобраться, как в среде разработки Visual Studio использовать Qt вместе с API nanoCAD. Было непонятно, как подключить к проекту Visual Studio метаобъектный компилятор (moc.exe) и компилятор пользовательского интерфейса (uic.exe), которые использует Qt.

Один из вариантов решения задачи – написание собственного .pro-файла, по которому qmake сгенерирует правильный проект, где уже автоматически будет вызываться нужный компилятор. Но этот вариант не был доведен до конца, поскольку нашелся другой, более простой способ использования метаобъектного компилятора Qt: расширение Qt VS Tools. С его помощью легко создавать проекты, использующие moc.exe и uic.exe, прямо в Visual Studio.

В результате проект разработали с нуля, используя исходники из примера HelloQt SDK nanoCAD. Однако при его сборке компилятор выдавал ошибку C2440. Стажеры изучили все параметры командной строки, которые использовались в примере HelloQt из NCadSDK, но не смогли найти флаг, который создавал ошибку. Тогда Илья Слободин порекомендовал воспользоваться утилитой Process Monitor для отслеживания командной строки, чтобы сравнить параметры нового проекта с примером HelloQt. Оказалось, что Visual Studio прячет опции во временный файл, поэтому решено было воспользоваться инструкцией из платформы stackoverflow.

Другой вариант решения – успеть при компиляции вручную перехватить временный файл, скопировав его куда-либо, однако это более сложная процедура. Спустя еще несколько десятков флагов программисты нашли тот самый, из-за которого появлялась ошибка C2440: /Zc:referenceBinding. Он добавлялся к проекту расширением Qt VS Tools. Проблему решили простым добавлением флага /Zc:referenceBinding-.

Далее студенты разбирались, как дополнить компиляцией метаданных существующий проект Visual Studio. Во время исследований Арсений Пронин обнаружил нужную инструкцию, но, поскольку ее версия была устаревшей, пришлось вносить правки.

Возникающую в ходе работы ошибку (рис. 1) исправляли путем добавления переменной среды $(QtToolsPath) = $(QtRoot)\bin, где $(QtRoot) = C:\Qt\5.15.1\msvc2019_64 – это путь до папки Qt версии 5.15.1 для 64-битной версии msvc2019. Таким образом можно дополнить компиляцией метаданных любой проект Visual Studio.

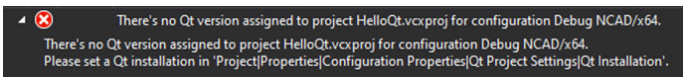

Рис. 1. Ошибка C2440

Когда нужно было обойтись без расширения Qt VS Tools, программисты изменяли .vcxproj-файл вручную: убеждались, что Qt установлен на компьютер, а затем в файле проекта вызвали uic.exe для всех .ui-файлов и moc.exe для всех header-файлов, использующих макрос Q_OBJECT. После этого компилировали все остальное, включая сгенерированные файлы, начинающиеся с moc_.

Процесс отработали на примере C++-проекта Visual Studio, включающего в себя файлы, относящиеся к Qt, и прочие файлы, которые не нужно собирать с помощью компиляторов Qt:

класс QtWidgetsClass с макросом Q_OBJECT (QtWidgetsClass.h, QtWidgetsClass.cpp) и файлы QtWidgetsClass.ui, HelloQt.cpp, stdafx.h.
В файл проекта добавили элемент ItemGroup, в котором делается кастомная сборка и компиляция с вызовом moc.exe и uic.exe:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 <ItemGroup Condition="'$(Configuration)|$(Platform)'=='Debug
NCAD|x64' or '$(Configuration)|$(Platform)'=='Release NCAD|x64'">
   <CustomBuild Include="QtWidgetsClass.ui">
     <AdditionalInputs>%(AdditionalInputs)</AdditionalInputs>
     <Command>$(QtRoot)\bin\uic.exe .\QtWidgetsClass.ui
-o "$(IntDir)\ui_QtWidgetsClass.h"</Command>
     <Message>UIC QtWidgetsClass.ui</Message>
     <Outputs>$(IntDir)\ui_QtWidgetsClass.h;%(Outputs)</Outputs>
   </CustomBuild>
   <CustomBuild Include="QtWidgetsClass.h">
     <AdditionalInputs>%(AdditionalInputs)</AdditionalInputs>
     <Command>$(QtRoot)\bin\moc.exe .\QtWidgetsClass.h -o
"$(IntDir)\moc_QtWidgetsClass.cpp"</Command>
     <Message>MOC QtWidgetsClass.h</Message>
     <Outputs>$(IntDir)\moc_QtWidgetsClass.cpp;%(Outputs)</
Outputs>
   </CustomBuild>
   <ClCompile Include="QtWidgetsClass.cpp" />
 </ItemGroup>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Элемент, приведенный выше, должен быть расположен перед элементом ItemGroup, включающим в себя элементы <ClInclude>, <ClCompile> и предназначенные для файлов проекта, не относящихся к Qt:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

<ItemGroup>
  <ClInclude Include="stdafx.h" />
</ItemGroup>
<ItemGroup>
  <ClCompile Include="HelloQt.cpp" />
  <ClCompile Include="stdafx.cpp">
    <PrecompiledHeader Condition="'$(Configuration)|$(Platform)'=='D
ebug NCAD|Win32'">Create</PrecompiledHeader>
    <PrecompiledHeader Condition="'$(Configuration)|$(Platform)'=='D
ebug NCAD|x64'">Create</PrecompiledHeader>
    <PrecompiledHeader Condition="'$(Configuration)|$(Platform)'=='R
elease NCAD|Win32'">Create</PrecompiledHeader>
    <PrecompiledHeader Condition="'$(Configuration)|$(Platform)'=='R
elease NCAD|x64'">Create</PrecompiledHeader>
  </ClCompile>
</ItemGroup>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Далее в компиляцию был добавлен элемент ItemGroup с элементом ClCompile для файлов, сформированных компилятором Qt:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 <ItemGroup>
   <ClCompile Include="$(IntDir)\moc_QtWidgetsClass.cpp">
     <AdditionalOptions>/FIstdafx.h %(AdditionalOptions)</
AdditionalOptions>
   </ClCompile>
 </ItemGroup>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Затем программисты добавили путь $(IntDir) в Visual Studio: *Menu → Project → Properties → Configuration Properties → C/C++ → General → Additional Include Directories*, чтобы проект смог найти сгенерированные файлы. Следующим шагом стало добавление пути к папке с Qt на компьютере в переменной среде $(QtRoot). При использовании этого способа нет необходимости добавлять флаг /Zc:referenceBinding-.
Воодушевившись успешными результатами проекта, студенты реализовали модули для nanoCAD "Проводник файлов" и "Редактор CSS с подсветкой синтаксиса", используя Qt и свои наработки.
Работа студентов МГТУ им. Н.Э. Баумана получила высокую оценку руководителя. Однокурсникам было предложено развивать свой проект и далее, но уже в штате сотрудников "Нанософт разработка". Предложение принято, сейчас недавние стажеры участвуют в создании модулей и команд для новых версий программных продуктов линейки nanoCAD.

Инструкции по программированию в API nanoCAD

Результаты работы Арсений и Илья оформили в виде инструкций по программированию для примера HelloQt из SDK nanoCAD версии NC_SDK_22.0.5944.3726.6053[1] (в новых версиях SDK компиляция метаданных Qt (moc.exe) уже добавлена). Скачать их можно по ссылкам:

1. Как дополнить компиляцией метаданных проект Visual Studio с примерами из SDK nanoCAD с помощью расширения Qt VS Tools[2].
2. Как дополнить компиляцией метаданных проект Visual Studio с примерами из SDK nanoCAD без расширения Qt VS Tools[3].
3. Как создать новый проект с нуля, используя исходники проекта HelloQt из SDK[4].

Арсений Пронин,
младший программист
Группы сопровождения проектов
ООО "Нанософт разработка"

[1] *https://developer.nanocad.ru/redmine/attachments/525.*
[2] *https://ftp.nanocad.ru/habr/QtSignal/VSExampleProjectWithQtVSTools/VSExampleProjectWithQtVSTools.pdf.*
[3] *https://ftp.nanocad.ru/habr/QtSignal/VSExampleProjectWithOutQtVSTools/VSExampleProjectWithOutQtVSTools.pdf.*
[4] *https://ftp.nanocad.ru/habr/QtSignal/NewProjectExampleWithHelloQt/NewProjectExampleWithHelloQt.pdf.*