Getting Started with Contouring Scripting

Grayden MacLennan
MBA, MSM, MS, CIIP
Disclosures

• Grayden worked for MIM Software from 2007-2013, but no longer has financial ties to the company.

• This presentation contains screenshots from several commercial software packages:
  – MIM
  – Velocity
  – RayStation
  – Pinnacle
My Background

• I’m a tech geek, but NOT a natural born programmer

• Enthusiast more than expert

• I learned most of these systems while making this presentation

• I had a lot of help
Table of Contents

• The What and the Why
• Tools of the Trade
• Two Simple Script Ideas
• Writing Scripts in the Real World
  – MIM
  – Velocity
  – RayStation
  – Pinnacle
  – Eclipse
• Next Steps
What is Scripting?

- Identifying a **specific sequence of actions** that a user normally performs in the same order **every time**

- Creating a **set of instructions** that makes sense to a computer so that it can do those actions **for the user**
You Mean Templates?

• No, not the same thing

• Templates create or load a list of things
  – ROIs with specific names/colors
  – Isodose line settings/colors
  – Beam arrangements
  – Report sections

• Scripts are MUCH easier to implement if you also use templates
  – Predictable names for things are important
What Kinds of Actions?

- Contour creation and manipulation
- Statistics processing
- Image loading and unloading
- Image creation, processing, editing
- Image registration
- Report generation
- Dose evaluation
- Display management
- Beam setup
- Plan calculation
- lots more...
Establishing the Steps

• Pay close attention to the order of your actions
  – Do you **REALLY** do it the same way every time?
  – Can you write out the instructions in everyday language?

• Length vs. complexity
  – Is it truly a complex task?
  – Maybe just a lot of simple tasks back to back?
Making Instructions for a Lego Set

• Lots of different kinds of pieces
  – Each one is relatively simple
  – Each one has different capabilities

• Identify which pieces you'll need
• Identify how they will attach to each other
• Identify the order in which they need to be added

• Create a document that describes the process
Simple Instructions for a Lego Car

1. 1x brick
2. 2x black, 4x black tire
3. 2x brick, 1x transparent window, 1x black
4. 1x brick, 1x transparent window, 1x black, 2x black

Image ©2014 Jessica Brown
We Can Use More Complex Parts...
...to Build Something Better

Image ©2012 Nathanaël Kuipers
Just Watch Out For Logic Problems

Penrose Triangle

1

2

3

5x

4

Image adapted from comic by Peter Lea
Approaches to Script-Making

- Writing code by hand
- “Copy My Clicks” style script recorder
- Drag & drop visual script builder
# Different Systems, Different Options

<table>
<thead>
<tr>
<th>System</th>
<th>Scripting Method</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>XiO/Monaco</td>
<td>None</td>
<td>n/a</td>
</tr>
<tr>
<td>Eclipse</td>
<td>Hand written code</td>
<td>C#</td>
</tr>
<tr>
<td>Velocity</td>
<td>Hand written code</td>
<td>XML</td>
</tr>
<tr>
<td>Pinnacle</td>
<td>&quot;Copy My Clicks&quot; recorder, Hand written code</td>
<td>Proprietary</td>
</tr>
<tr>
<td>RayStation</td>
<td>&quot;Copy My Clicks&quot; recorder, Hand written code</td>
<td>Python</td>
</tr>
<tr>
<td>MIM</td>
<td>Drag &amp; drop visual builder (Workflows)</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>Hand written code (MIMextensions)</td>
<td>Java, MATLAB</td>
</tr>
</tbody>
</table>
Differing Attitudes Towards Scripting

Boundless Optimism

Crushing Pessimism

You'll shoot your eye out, kid
Differing Attitudes Towards Scripting

It's just a Physics toy
How I Feel Most Days

Do you want to script some contours?
Arguments Against Scripting

• It takes time to set up

• It doesn't do **EXACTLY** what I want

• It takes time to learn new skills
Arguments For Scripting

• Time is **invested**, not spent

• Evaluating a process can help find inefficiencies, even if you don't end up using a script
  – Helps to standardize your work process

• You **learn a new skill!**
Psychological Hurdles

• The key to finishing is starting

• Avoid analysis paralysis
  – Pick something simple
  – "Hello World" is usually the first program you write in every language for a reason
Setting Expectations for Today

• We will **NOT** be learning to program today
  – Tons of good tutorials online for every language

• Each system uses a different language
  – Explanations will dive in assuming you already have some knowledge of that language

• Open/edit/save functions will be glossed over
  – They are either intuitive or in the manual
No Need to Take Coding Notes

• I will be moving quickly

• **Remember:** these slides will be on the AAMD website after the conference

• You can come back and re-read the code later
Table of Contents

• The What and the Why
• Tools of the Trade
• Two Simple Script Ideas
• Writing Scripts in the Real World
  – MIM
  – Velocity
  – RayStation
  – Pinnacle
  – Eclipse
• Next Steps
Tools of the Trade

• **Text Editor**
  – **NOT** a Word Processor like Microsoft Word
    • Spaces and tabs matter
    • Autocorrect and Autoformat will ruin your day
  – Writing prose and writing code have different demands
    • A word processor operates on the principle that appealing layout of printed pages is the ultimate goal
Useful Text Editor Features

- **Syntax highlighting** (code coloring)
- Powerful find/replace tools
- Will not wrap text to next line unless you say so
- Autocomplete offered but not imposed
- Smart indentation
Syntax Highlighting

- Good editors can apply color based on code content

```xml
<VelocityScript name="Excerpt from IMRT script">
  <!-- Demo of syntax highlighting -->
  <Crop
    source="Bladder"
    target="Bladder_Avoid"
    cropSpecification="RemoveInside"
    marginDirection="Grow"
    margin="5"
    cropList="PTV"
  />
</VelocityScript>

<VelocityScript name="Excerpt from IMRT script">
  <!-- Demo of syntax highlighting -->
  <Crop
    source="Bladder"
    target="Bladder_Avoid"
    cropSpecification="RemoveInside"
    marginDirection="Grow"
    margin="5"
    cropList="PTV"
  />
</VelocityScript>
```
Good Text Editors

• **Windows**: Notepad++

• **Mac**: TextMate, BBEdit, TextWrangler

• **Linux/Solaris/UNIX**: emacs, vim, nano
More Tools of the Trade

- Integrated Development Environment (IDE)
  - Editors with extra tools for making software
  - Examples: Visual Studio, Eclipse (no relation), Xcode
  - Serious overkill for most of these projects
Table of Contents

• The What and the Why
• Tools of the Trade
• Two Simple Script Ideas
• Writing Scripts in the Real World
  – MIM
  – Velocity
  – RayStation
  – Pinnacle
  – Eclipse
• Next Steps
Two Simple Script Ideas

• We are going to pick 2 simple script ideas
  – Not super useful by themselves, but they demonstrate the clinical usefulness of expanding on the ideas

• We will see how to make those 2 scripts on several different systems
Script Idea #1

- Rename ROIs to RTOG-style names
- For simplicity, let's just rename 5 contours

<table>
<thead>
<tr>
<th>Old Style</th>
<th>New Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIGHT ORBIT</td>
<td>Eye_R</td>
</tr>
<tr>
<td>LEFT ORBIT</td>
<td>Eye_L</td>
</tr>
<tr>
<td>RIGHT OPTIC NERVE</td>
<td>OpticNerve_R</td>
</tr>
<tr>
<td>LEFT OPTIC NERVE</td>
<td>OpticNerve_L</td>
</tr>
<tr>
<td>OPTIC CHIASM</td>
<td>Chiasm</td>
</tr>
</tbody>
</table>
Script #1 Relevance

• Some contour atlases provide little control over ROI naming conventions

• Some clinical trials demand specific naming of ROIs

• In both cases, one known set of names must be mapped to another
  – **Atlas** standard names → **Clinic** standard names
  – **Clinic** standard names → **Trial** standard names
Script #1 Process

• Steps in script
  – Find a contour
  – Rename it

Lather, rinse, repeat

• Assumptions for the demo
  – Source contours exist
  – They are named exactly as we expect them to be
Script Idea #2

• Create a few IMRT optimization structures
  
  – **Bladder_Avoid** and **Rectum_Avoid**
    • Portion of each organ at least 5 mm outside PTV
  
  – **Ring1**
    • Dose gradient control structure
    • 2 cm thick shell 1 cm away from the PTV
Script #2 Starting Contours
Bladder_Avoid & Rectum_Avoid
Ring1
Many plans use optimization or dose gradient control structures.

These structures are usually created the same way every time.

Perfect for automation.
Script #2 Process

• Steps in Script (varies slightly by system)
  – **Crop** Bladder 5 mm from PTV
    • Call the result "Bladder_Avoid"
  – **Crop** Rectum 5 mm from PTV
    • Call the result "Rectum_Avoid"
  – **Expand** from PTV 3 cm
    • **Subtract** 1 cm margin around PTV
    • Call the result "Ring1"

• Assumptions for the demo
  – Bladder, Rectum, and PTV all **exist**
  – They are named **exactly** as we expect them to be
Cool.
Let's Write Some Scripts.
Table of Contents

• The What and the Why
• Tools of the Trade
• Two Simple Script Ideas
• Writing Scripts in the Real World
  – MIM
  – Velocity
  – RayStation
  – Pinnacle
  – Eclipse
• Next Steps
Scripting in MIM

• Scripting is built in

• Scripts are called Workflows

• Drag & Drop interface
  – No code writing necessary

• Most software functions are scriptable
MIM Documentation/Support

- 23 page manual supplement
  - Mostly just a list of all the tools available

- Training video on support webpage

- Many sample workflows

- Support will help with writing and troubleshooting
MIM Workflow Interface
Workflow Builder is designed to help automate a diverse set of tasks that would normally require a great deal of time and input from a user. MIM exposes numerous Actions that may be placed together to achieve an end result in a fraction of the time spent performing the task manually.

Let's begin by providing the Workflow a name, indicative of its use. Optionally, a description may be given, possibly to assist others in determining a workflow's general use.

**Workflow Name:** AAMD2016 Renamer

**Description:**

*Workflow Icon:*

*Workflow Labels:*

- AAMD2016 Renamer
Page 2 of 3 in Workflow Builder
Giant List of Scriptable Actions Data to Feed Into Actions
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM
Script #1 (Renamer) in MIM

Find a Contour          Rename the Contour
Find a Contour          Rename the Contour
Find a Contour          Rename the Contour
Find a Contour          Rename the Contour
Find a Contour          Rename the Contour
Find a Contour          Rename the Contour
Find a Contour          Rename the Contour
Find a Contour          Rename the Contour
Find a Contour          Rename the Contour
Find a Contour          Rename the Contour
Script #1 Workflow Steps
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM

![Workflow Builder](image)

Each multicontour variable is treated as the union of all its member contours.
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM
Script #2 (IMRT ROIs) in MIM

- Same process for Rectum_Avoid

- Ring1 is one PTV expansion minus another
Complete IMRT ROI Script
Tangent Topic: MIMextentions

- Extension API lets you manipulate the application from behind the scenes
  - Goes beyond stringing a series of steps together

- MIM doesn’t have a feature you need?
  - Make it yourself

- Java, JavaScript, or link to MATLAB
Table of Contents

• The What and the Why
• Tools of the Trade
• Two Simple Script Ideas
• Writing Scripts in the Real World
  – MIM
  – Velocity
  – RayStation
  – Pinnacle
  – Eclipse
• Next Steps
Scripting in Velocity

• Scripting is **built in**

• Scripts must be hand-written
  – **Easy** to learn language: XML

• Limited capabilities
  – Only has 6 most commonly used contouring tools
    • Margin, Crop, Intersection, Union, Fill Holes, Remove Pieces
    • Can also specify color of output ROI
Velocity Documentation/Support

• 4 pages in User Manual with sample code for a few of the actions

• Full syntax reference guide is in the form of a hastily edited (but complete) example script
  – Need Velocity Admin User access to see the reference guide

• Support will help with writing and troubleshooting
Velocity Script Skeleton

`<VelocityScript name="Script name for menu" >`

`<!-- Freeform text to help with code documentation -->`

`<Action parameter="value" parameter="value" ... />`
`<Action parameter="value" parameter="value" ... />`
`<Action parameter="value" parameter="value" ... />`
`<Action parameter="value" parameter="value" ... />`

`</VelocityScript>`
Script #1 (Renamer) in Velocity

• Contour renaming is **not** one of the 6 tools
  – We can improvise…
  – Use the margin tool with **0 cm expansion** to create a new contour with correct name

• Need a strategy for the old contour
  – There is **no delete tool**, but we can empty it out
  – It will be easier to know which one to delete later
Improvising "Copy" with "Margin"

- Copy = Margin with no expansion

```xml
<Margin source="LEFT ORBIT" target="Eye_L" margin="0" marginDirection="Grow" marginSpecification="Symmetric" />
```

- Code can be split onto separate lines for readability

```xml
<Margin
   source="LEFT ORBIT"
   target="Eye_L"
   marginDirection="Grow"
   marginSpecification="Symmetric"
   margin="0"
/>```
Improvising "Erase" with "Crop"

- Erase by cropping ROI from itself

```xml
<Crop
    source="LEFT ORBIT"
    cropSpecification="RemoveInside"
    cropList="LEFT ORBIT"
    marginDirection="None"
/>
```
<VelocityScript name="AAMD 2016 Contour Renaming in Velocity">

  <Margin source="RIGHT ORBIT" target="Eye_R" marginDirection="Grow"
    marginSpecification="Symmetric" margin="0" />
  <Crop source="RIGHT ORBIT" cropSpecification="RemoveInside" cropList="RIGHT ORBIT"
    marginDirection="None" />

  <Margin source="LEFT ORBIT" target="Eye_L" marginDirection="Grow"
    marginSpecification="Symmetric" margin="0" />
  <Crop source="LEFT ORBIT" cropSpecification="RemoveInside" cropList="LEFT ORBIT"
    marginDirection="None" />

  <Margin source="RIGHT OPTIC NERVE" target="OpticNerve_R" marginDirection="Grow"
    marginSpecification="Symmetric" margin="0" />
  <Crop source="RIGHT OPTIC NERVE" cropSpecification="RemoveInside" cropList="RIGHT OPTIC NERVE"
    marginDirection="None" />

  <Margin source="LEFT OPTIC NERVE" target="OpticNerve_L" marginDirection="Grow"
    marginSpecification="Symmetric" margin="0" />
  <Crop source="LEFT OPTIC NERVE" cropSpecification="RemoveInside" cropList="LEFT OPTIC NERVE"
    marginDirection="None" />

  <Margin source="OPTIC CHIASM" target="Chiasm" marginDirection="Grow"
    marginSpecification="Symmetric" margin="0" />
  <Crop source="OPTIC CHIASM" cropSpecification="RemoveInside" cropList="OPTIC CHIASM"
    marginDirection="None" />

</VelocityScript>
How to Access Scripting

Normal Users

Admin Users

Create Set
Deselect
Edit
Center
Duplicate
Margin
Run Contour Scripts
Post Processing
Crop
Smooth
Primary Volume Histogram
Delete
Properties
Create and Edit Contour Scripts
Post Processing
Crop
Smooth
Primary Volume Histogram
Delete
Properties
Creating and Editing Scripts

![Image of a software interface for creating and editing contour scripts]

<table>
<thead>
<tr>
<th>Script File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRProstatePTVs.xml</td>
<td>LR Prostate PTV Expansions</td>
</tr>
<tr>
<td>MarginWithCrop.xml</td>
<td>Margins and Cropping</td>
</tr>
<tr>
<td>PatientContour.xml</td>
<td>Patient Processing</td>
</tr>
<tr>
<td>PostProcess.xml</td>
<td>Post Processing</td>
</tr>
<tr>
<td>VelocityOptiContours...</td>
<td>AAMD 2016 IMRT Contours in Velocity</td>
</tr>
</tbody>
</table>

- New
- Delete
- Help

MacLennan 2016
Select a Script to Edit

```xml
<VelocityScript name="AAMD 2016 Contour Renaming in Velocity">
    <!-- by Gregory MacLennan, Seattle Proton Therapy Center, last edited 2016-05-28 -->
    <Margin source="RIGHT ORBIT" target="Eye_R" marginDirection="Grow"
             marginSpecification="Symmetric" margin="0" />
    <Crop source="RIGHT ORBIT" cropSpecification="RemoveInside" cropList="RIGHT ORBIT"
           marginDirection="None" />
    <Margin source="LEFT ORBIT" target="Eye_L" marginDirection="Grow"
             marginSpecification="Symmetric" margin="0" />
    <Crop source="LEFT ORBIT" cropSpecification="RemoveInside" cropList="LEFT ORBIT"
           marginDirection="None" />
    <Margin source="RIGHT OPTIC NERVE" target="OpticNerve_R" marginDirection="Grow"
             marginSpecification="Symmetric" margin="0" />
    <Crop source="RIGHT OPTIC NERVE" cropSpecification="RemoveInside" cropList="RIGHT OPTIC NERVE"
           marginDirection="None" />
    <Margin source="LEFT OPTIC NERVE" target="OpticNerve_L" marginDirection="Grow"
             marginSpecification="Symmetric" margin="0" />
    <Crop source="LEFT OPTIC NERVE" cropSpecification="RemoveInside" cropList="LEFT OPTIC NERVE"
           marginDirection="None" />
    <Margin source="OPTIC CHIASM" target="Chiasm" marginDirection="Grow"
             marginSpecification="Symmetric" margin="0" />
    <Crop source="OPTIC CHIASM" cropSpecification="RemoveInside" cropList="OPTIC CHIASM"
           marginDirection="None" />
</VelocityScript>
```
Script #1 Loaded into Velocity

<table>
<thead>
<tr>
<th>Command</th>
<th>Source</th>
<th>Target</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Margin</td>
<td>RIGHT OR... Eye_R</td>
<td>Grow</td>
<td>Symmetric 0 mm</td>
<td></td>
</tr>
<tr>
<td>Crop</td>
<td>RIGHT OR...</td>
<td>None</td>
<td>Crop voxels inside (RIGHT ORBIT)</td>
<td></td>
</tr>
<tr>
<td>Margin</td>
<td>LEFT ORBIT   Eye_L</td>
<td>Grow</td>
<td>Symmetric 0 mm</td>
<td></td>
</tr>
<tr>
<td>Crop</td>
<td>LEFT ORBIT</td>
<td>None</td>
<td>Crop voxels inside (LEFT ORBIT)</td>
<td></td>
</tr>
<tr>
<td>Margin</td>
<td>RIGHT OR... OpticNerv...</td>
<td>Grow</td>
<td>Symmetric 0 mm</td>
<td></td>
</tr>
<tr>
<td>Crop</td>
<td>RIGHT OR...</td>
<td>None</td>
<td>Crop voxels inside (RIGHT OPTIC NERVE)</td>
<td></td>
</tr>
<tr>
<td>Margin</td>
<td>LEFT OPTI... OpticNerv...</td>
<td>Grow</td>
<td>Symmetric 0 mm</td>
<td></td>
</tr>
<tr>
<td>Crop</td>
<td>LEFT OPTI...</td>
<td>None</td>
<td>Crop voxels inside (LEFT OPTIC NERVE)</td>
<td></td>
</tr>
<tr>
<td>Margin</td>
<td>OPTIC CH... Chiasm</td>
<td>Grow</td>
<td>Symmetric 0 mm</td>
<td></td>
</tr>
<tr>
<td>Crop</td>
<td>OPTIC CH...</td>
<td>None</td>
<td>Crop voxels inside (OPTIC CHIASM)</td>
<td></td>
</tr>
</tbody>
</table>
Comparing Before and After
Script #2 (IMRT ROIs) in Velocity

• Margins and Cropping are what Velocity's scripting systems was designed for

• **Bladder_Avoid** and **Rectum_Avoid** can both be done with a single Margin command

• **Ring1** needs two actions: a Margin then a Crop
Complete IMRT ROIs Script

<VelocityScript name="AAMD 2016 IMRT Contours in Velocity">

  <Crop source="Bladder" target="Bladder_Avoid" cropSpecification="RemoveInside" cropList="PTV" marginDirection="Grow" margin="5" />

  <Crop source="Rectum" target="Rectum_Avoid" cropSpecification="RemoveInside" cropList="PTV" marginDirection="Grow" margin="5" />

  <Margin source="PTV" target="Ring1" marginDirection="Grow" marginSpecification="Symmetric" margin="30" />

  <Crop source="Ring1" cropSpecification="RemoveInside" cropList="PTV" marginDirection="Grow" margin="10" />

</VelocityScript>

• Note: comment tags can provide documentation
Table of Contents

• The What and the Why
• Tools of the Trade
• Two Simple Script Ideas
• Writing Scripts in the Real World
  – MIM
  – Velocity
  – RayStation
  – Pinnacle
  – Eclipse
• Next Steps
RayStation

- Scripting is built-in
- Most program functions are scriptable

- Has a "Copy my Clicks" style script recorder
- Hand-coding to edit recorded scripts
- Hand-coding to make your own scripts

- Language is Python
  - Must install IronPython interpreter if it isn't already there
RayStation Documentation/Support

• 36 page manual supplement with code samples

• Videos on RaySearch website & YouTube

• Official discussion forums

• Training workshops (one coming up next weekend)

• Support will help with writing and troubleshooting
Script #1 (Renamer) in RayStation

- Start the script recorder
- Manually rename each of the ROIs
- Stop the script recorder
Open the Scripting Tab
Open the Script Creation Interface
Script Creation Window
Now That We Are Recording...
Examine the ROI List
Start Editing ROIs Manually

Double Click on ROI
Let's Rename this ROI

ROI Properties

Select ROI: [RIGHT OPTIC NERVE]

ROI Properties

Name: [RIGHT OPTIC NERVE]
Type: Organ
Color:
Tissue name:
Organ type: Organ at risk
Compression ratio: 0.48
Material: [None]
Create new material...

Mass density [g/cm³]:

Geometries:

<table>
<thead>
<tr>
<th>Image set</th>
<th>Representation</th>
<th>Volume [cm³]</th>
<th>Intensity Min</th>
<th>Avg</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT 1</td>
<td>Contours</td>
<td>0.73</td>
<td>-94</td>
<td>31.55</td>
<td>181</td>
<td>HU</td>
</tr>
</tbody>
</table>
Much Better
Do That a Few More Times...
Hit Stop Button When Done
Here's the Code it Created for Us

```python
# Script recorded 11 Apr 2016
# RayStation version: 4.7.2.5
# Selected patient: ...
from connect import *
patient = get_current("Patient")

with CompositeAction('Apply ROI changes (RIGHT ORBIT)'):
    patient.PatientModel.RegionsOfInterest['RIGHT ORBIT'].Name = "Eye_R"
# CompositeAction ends

with CompositeAction('Apply ROI changes (LEFT ORBIT)'):
    patient.PatientModel.RegionsOfInterest['LEFT ORBIT'].Name = "Eye_L"
# CompositeAction ends

with CompositeAction('Apply ROI changes (RIGHT OPTIC NERVE)'):
```

MacLennan 2016
Script #1 Code in a Text Editor

```python
# Script recorded 11 Apr 2016

# RayStation version: 4.7.2.5
# Selected patient: ...

from connect import *

patient = get_current("Patient")

with CompositeAction('Apply ROI changes (RIGHT ORBIT)'):
    patient.PatientModel.RegionsOfInterest['RIGHT ORBIT'].Name = "Eye_R"
    # CompositeAction ends

with CompositeAction('Apply ROI changes (LEFT ORBIT)'):
    patient.PatientModel.RegionsOfInterest['LEFT ORBIT'].Name = "Eye_L"
    # CompositeAction ends

with CompositeAction('Apply ROI changes (RIGHT OPTIC NERVE)'):
    patient.PatientModel.RegionsOfInterest['RIGHT OPTIC NERVE'].Name = "OpticNerve_R"
    # CompositeAction ends

with CompositeAction('Apply ROI changes (LEFT OPTIC NERVE)'):
    patient.PatientModel.RegionsOfInterest['LEFT OPTIC NERVE'].Name = "OpticNerve_L"
    # CompositeAction ends

with CompositeAction('Apply ROI changes (OPTIC CHIASM)'):
    patient.PatientModel.RegionsOfInterest['OPTIC CHIASM'].Name = "Chiasm"
    # CompositeAction ends
```
The Code is Ready to Use

- In this case, there is no need to edit the code at all
- Let's break it down anyway, just to see how it works
# Script recorded 11 Apr 2016
# RayStation version: 4.7.2.5
# Selected patient: ...

```python
from connect import *

patient = get_current("Patient")

with CompositeAction('Apply ROI changes (RIGHT ORBIT)'):
    patient.PatientModel.RegionsOfInterest['RIGHT ORBIT'].Name = "Eye_R"
    # CompositeAction ends

with CompositeAction('Apply ROI changes (LEFT ORBIT)'):
    patient.PatientModel.RegionsOfInterest['LEFT ORBIT'].Name = "Eye_L"
    # CompositeAction ends

with CompositeAction('Apply ROI changes (RIGHT OPTIC NERVE)'):
    patient.PatientModel.RegionsOfInterest['RIGHT OPTIC NERVE'].Name = "OpticNerve_R"
    # CompositeAction ends

with CompositeAction('Apply ROI changes (LEFT OPTIC NERVE)'):
    patient.PatientModel.RegionsOfInterest['LEFT OPTIC NERVE'].Name = "OpticNerve_L"
    # CompositeAction ends

with CompositeAction('Apply ROI changes (OPTIC CHIASM)'):
    patient.PatientModel.RegionsOfInterest['OPTIC CHIASM'].Name = "Chiasm"
    # CompositeAction ends
```

Breaking Down the Code

Time stamp and version record

Linkage to RayStation active session

MacLennan 2016
CompositeAction = Undo Step

- The "CompositeAction" and its closing comment simply indicate the breadth of one Undo step.

```python
with CompositeAction('Apply ROI changes (RIGHT ORBIT)):
    patient.PatientModel.RegionsOfInterest['RIGHT ORBIT'].Name = "Eye_R"
# CompositeAction ends
```

- RayStation's Undo menu after running the script.
Breaking Down the Code

```python
# Script recorded 11 Apr 2016
# RayStation version: 4.7.2.5
# Selected patient: ...

from connect import *

patient = get_current("Patient")

with CompositeAction('Apply ROI changes (RIGHT ORBIT)'):
    patient.PatientModel.RegionsOfInterest['RIGHT ORBIT'].Name = "Eye_R"
# CompositeAction ends

with CompositeAction('Apply ROI changes (LEFT ORBIT)'):
    patient.PatientModel.RegionsOfInterest['LEFT ORBIT'].Name = "Eye_L"
# CompositeAction ends

with CompositeAction('Apply ROI changes (RIGHT OPTIC NERVE)'):
    patient.PatientModel.RegionsOfInterest['RIGHT OPTIC NERVE'].Name = "OpticNerve_R"
# CompositeAction ends

with CompositeAction('Apply ROI changes (LEFT OPTIC NERVE)'):
    patient.PatientModel.RegionsOfInterest['LEFT OPTIC NERVE'].Name = "OpticNerve_L"
# CompositeAction ends

with CompositeAction('Apply ROI changes (OPTIC CHIASM)'):
    patient.PatientModel.RegionsOfInterest['OPTIC CHIASM'].Name = "Chiasm"
# CompositeAction ends
```
• We could strip out all of the extra CompositeAction lines and make the whole thing one undoable action

# AAMD 2016 Contour Renaming Script in RayStation

from connect import *

patient = get_current("Patient")

with CompositeAction('Run Renamer Script'):

    patient.PatientModel.RegionsOfInterest['RIGHT ORBIT'].Name = "Eye_R"
    patient.PatientModel.RegionsOfInterest['LEFT ORBIT'].Name = "Eye_L"
    patient.PatientModel.RegionsOfInterest['RIGHT OPTIC NERVE'].Name = "OpticNerve_R"
    patient.PatientModel.RegionsOfInterest['LEFT OPTIC NERVE'].Name = "OpticNerve_L"
    patient.PatientModel.RegionsOfInterest['OPTIC CHIASM'].Name = "Chiasm"

# CompositeAction ends
A Note About Python Code

• Whitespace (tabs, spaces, returns) matters in Python

• Long code lines can be split onto multiple lines, but it must be done with care
  – Indentation level is part of how the interpreter knows what's going on

• Style guide at:
  https://www.python.org/dev/peps/pep-0008/
Script #2 (IMRT ROIs) in RayStation

• Start the script recorder

• Create **Bladder_Avoid, Rectum_Avoid**, and **Ring1**
  – They can all be done with the ROI Algebra tool

• Stop the script recorder
Find the ROI Algebra Tool
Find the ROI Algebra Tool
ROI Algebra Window

Expression A

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bladder</td>
<td>Organ</td>
</tr>
<tr>
<td>CTV</td>
<td>CTV</td>
</tr>
<tr>
<td>FemoralHead_L</td>
<td>Organ</td>
</tr>
<tr>
<td>FemoralHead_R</td>
<td>Organ</td>
</tr>
<tr>
<td>PTV</td>
<td>PTV</td>
</tr>
<tr>
<td>Prostate</td>
<td>GTV</td>
</tr>
<tr>
<td>Rectum</td>
<td>Organ</td>
</tr>
<tr>
<td>patient</td>
<td>External</td>
</tr>
</tbody>
</table>

Expression B

<table>
<thead>
<tr>
<th>Source ROIs</th>
<th>Margins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expand</td>
<td>None</td>
</tr>
<tr>
<td>Contract</td>
<td></td>
</tr>
<tr>
<td>Use uniform margin</td>
<td></td>
</tr>
</tbody>
</table>

Superior [cm] 0.50

Right [cm] 0.50

Posterior [cm] 0.50

Anterior [cm] 0.50

Left [cm] 0.50

Inferior [cm] 0.50

OK  Cancel
Set the Output ROI
Script #2 Generated Code

```
# Script recorded 20 May 2016
# RayStation version: 4.7.2.5
# Selected patient: ...
from connect import *
patient = get_current("Patient")
examination = get_current("Examination")

with CompositeAction('ROI Algebra (Bladder_Avoid) '):
    retval_0 = patient.PatientModel.CreateRoi(Name="Bladder_Avoid", Color="Blue", Type="Organ", TissueName=None, RoiMaterial=None)
    retval_0.CreateAlgebraGeometry(Examination=examination, Algorithm="Auto", ExpressionA=[ 'Operation': "Union", 'SourceRoiNames': ['Bladder'], ])
    # CompositeAction ends

with CompositeAction('ROI Algebra (Rectum_Avoid) '):
    retval_1 = patient.PatientModel.CreateRoi(Name="Rectum_Avoid", Color="Yellow", Type="Organ", TissueName=None, RoiMaterial=None)
```

MacLennan 2016
from connect import *

patient = get_current("Patient")

examination = get_current("Examination")

with CompositeAction("ROI Algebra (Bladder_Avoid)"):
    retval_0 = patient.PatientModel.CreateRoi(Name="Bladder_Avoid", Color="Blue", Type="Organ", TissueName=None, RoiMaterial=None)

    retval_0.CreateAlgebraGeometry(
        Examination=examination,
        Algorithm="Auto",
        Expression=[
            ("Operation": "Union", "SourceRoiNames": ["Bladder"], "MarginSettings": 
                {"Type": "Expand", "Superior": 0, "Inferior": 0, "Anterior": 0, "Posterior": 0, "Right": 0, "Left": 0}),
            ("Operation": "Union", "SourceRoiNames": ["PTV"], "MarginSettings": 
                {"Type": "Expand", "Superior": 0.5, "Inferior": 0.5, "Anterior": 0.5, "Posterior": 0.5, "Right": 0.5, "Left": 0.5}),
        ],
        ResultOperation="Subtraction",
        ResultMarginSettings=[
            ("Type": "Expand", "Superior": 0, "Inferior": 0, "Anterior": 0, "Posterior": 0, "Right": 0, "Left": 0)
        ]
    )

    # CompositeAction ends

with CompositeAction("ROI Algebra (Rectum_Avoid)"):

    retval_1 = patient.PatientModel.CreateRoi(Name="Rectum_Avoid", Color="Yellow", Type="Organ", TissueName=None, RoiMaterial=None)

    retval_1.CreateAlgebraGeometry(
        Examination=examination,
        Algorithm="Auto",
        Expression=[
            ("Operation": "Union", "SourceRoiNames": ["Rectum"], "MarginSettings": 
                {"Type": "Expand", "Superior": 0, "Inferior": 0, "Anterior": 0, "Posterior": 0, "Right": 0, "Left": 0}),
            ("Operation": "Union", "SourceRoiNames": ["PTV"], "MarginSettings": 
                {"Type": "Expand", "Superior": 0.5, "Inferior": 0.5, "Anterior": 0.5, "Posterior": 0.5, "Right": 0.5, "Left": 0.5}),
        ],
        ResultOperation="Subtraction",
        ResultMarginSettings=[
            ("Type": "Expand", "Superior": 0, "Inferior": 0, "Anterior": 0, "Posterior": 0, "Right": 0, "Left": 0)
        ]
    )

    # CompositeAction ends

with CompositeAction("ROI Algebra (Ring)"):

    retval_2 = patient.PatientModel.CreateRoi(Name="Ring!", Color="Pink", Type="Organ", TissueName=None, RoiMaterial=None)

    retval_2.CreateAlgebraGeometry(
        Examination=examination,
        Algorithm="Auto",
        Expression=[
            ("Operation": "Union", "SourceRoiNames": ["PTV"], "MarginSettings": 
                {"Type": "Expand", "Superior": 3, "Inferior": 3, "Anterior": 3, "Posterior": 3, "Right": 3, "Left": 3}),
            ("Operation": "Union", "SourceRoiNames": ["PTV"], "MarginSettings": 
                {"Type": "Expand", "Superior": 1, "Inferior": 1, "Anterior": 1, "Posterior": 1, "Right": 1, "Left": 1}),
        ],
        ResultOperation="Subtraction",
        ResultMarginSettings=[
            ("Type": "Expand", "Superior": 0, "Inferior": 0, "Anterior": 0, "Posterior": 0, "Right": 0, "Left": 0)
        ]
    )

    # CompositeAction ends
The Code is Good! (But Long)

- The generated code is ready to save and reuse

- The 9 longest lines are extra long because they specify ant/post/sup/inf/left/right margins
  - They are required, so we can't chop them out

- Could chop out the two extra CompositeActions, but there's really no point
Table of Contents

• The What and the Why
• Tools of the Trade
• Two Simple Script Ideas
• Writing Scripts in the Real World
  – MIM
  – Velocity
  – RayStation
  – Pinnacle
  – Eclipse
• Next Steps
Pinnacle Scripting

• Scripting is built-in
• Many program functions are scriptable

• Has a "Copy my Clicks" style script recorder
• Hand-coding to edit recorded scripts*
• Hand-coding to make your own scripts*

• Language is proprietary
Pinnacle Documentation/Support

- Script recorder is a **supported** feature
  - Manual tells you how to record actions and save scripts
  - Support will help you use the script recorder

- You are **on your own** if you want to **edit** the code
  - No official documentation of the language
  - Support will not help you
Editing Pinnacle Scripts
Made to be Reverse Engineered?

• The language syntax and capabilities can be reverse engineered from recorded scripts and system logs

• Many people have spent a lot of time doing this

• The knowledge pool remains **fragmented**
  – Several places to look for these collections of knowledge
Unofficial Community Support

• Pinnacle Users Google Group
  – [https://groups.google.com/d/forum/pinnacle3-users](https://groups.google.com/d/forum/pinnacle3-users)

• Guide by [Sean Geoghegan](https://groups.google.com/d/forum/pinnacle3-users), Royal Perth Hospital
  – 44 page scripting guide published in 2007
  – Covers core concepts, but no ROI manipulation details
  – Bundled with dozens of sample scripts
Unofficial Community Support

- Phone a friend?
Pinnacle Script Skeleton

- There is no skeleton necessary for Pinnacle scripts
- Just start listing commands
Approach for Our Scripts

- We'll use the *Script Recorder* first
- We'll see if we can figure out what do to from there
Finding the Script Recorder

![Image of script recorder interface]

- Utilities: Cutplanes, Profile, Scripting, Preferences, Data Sets, Image Fusion, Transform
- View: Setup, Co
HotScripts Window
Attesting Script #1

- Script recorder is running
- Rename each of the ROIs
- Stop the recorder
End Recording
Script #1 Output in a Text Editor

RoiList.Current = 0;

RoiList.Current.Name = "Eye_R";
RoiList.Current = 1;

RoiList.Current.Name = "Eye_L";
RoiList.Current = 2;

RoiList.Current.Name = "OpticNerve_R";
RoiList.Current = 3;

RoiList.Current.Name = "OpticNerve_L";
RoiList.Current = 4;

RoiList.Current.Name = "Chiasm";

• More than we need

• We don't care if the ROI can use SUV

• Scrap these lines
The Part of the Code We Need

RoiList.Current = 0;
RoiList.Current.Name = "Eye_R";

Translation:

• Make the 1st ROI in the list the active ROI
• Set name of the active ROI to "Eye_R"

Problem:

• What if the right eye isn't the 1st ROI in the list?
"Off By One" Error
Can We Fix It?

• The Google Group is a useful resource
  – Forum contributor **Wyatt Smith** posted code for a script that does almost exactly what we are doing
    
    • [https://groups.google.com/d/msg/pinnacle3-users/kANhYx3tXVE/MJjP3dWE-y8J](https://groups.google.com/d/msg/pinnacle3-users/kANhYx3tXVE/MJjP3dWE-y8J)

  – He searches **by ROI name**, not by **position in the list**
  – He searches for every common name variant and tries to perform a rename action on each one
  – He also specifies a standard color for each ROI
Excerpt from Wyatt's Code

- Surely one of these will work...

```plaintext
RoiList ."Eye R" .Name = "eye_R";
RoiList ."Eye_R" .Name = "eye_R";
RoiList ."R Eye" .Name = "eye_R";
RoiList ."R_Eye" .Name = "eye_R";
RoiList ."Rt Eye" .Name = "eye_R";
RoiList ."Rt_Eye" .Name = "eye_R";
RoiList ."RT EYE" .Name = "eye_R";
RoiList ."RT_EYE" .Name = "eye_R";
RoiList ."R_EYE" .Name = "eye_R";
RoiList ."R_EYE" .Name = "eye_R";
RoiList ."eye RT" .Name = "eye_R";
RoiList ."eye_RT" .Name = "eye_R";
RoiList ."eye_R" .Color = "lightorange";
```

- Doesn't include "RIGHT ORBIT", but that's easy to fix

```plaintext
RoiList ."RIGHT ORBIT" .Name = "Eye_R";
```
Script #1 Wyatt's Way

RoiList ."RIGHT ORBIT" .Name = "Eye_R";
RoiList ."LEFT ORBIT" .Name = "Eye_L";
RoiList ."RIGHT OPTIC NERVE" .Name = "OpticNerve_R";
RoiList ."LEFT OPTIC NERVE" .Name = "OpticNerve_L";
RoiList ."OPTIC CHIASM" .Name = "Chiasm";
Script #2 (IMRT ROIs) in Pinnacle

- Get the Script Recorder up and running
- Do the expansions and subtractions and ring
- Stop the Script Recorder
- See what we have
Note: We Need 2 Extra ROIs

- **PTV_05** = PTV + 5 mm
- **Bladder_Avoid** = Bladder – PTV_05
- **Rectum_Avoid** = Rectum – PTV_05
- **PTV_10** = PTV + 10 mm
- **Ring1** = 20 mm ring from PTV_10
Finding the ROI Expansion Tools
The ROI Expansion Window

<table>
<thead>
<tr>
<th>ROI</th>
<th>Source</th>
<th>Avoid Interior</th>
<th>Avoid Exterior</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLADDER</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECTUM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTV_05</td>
<td></td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>

**ROI Group**

- Destination ROI:
  - Select ROI to modify
  - Create new ROI

**Name:** Bladder_Avoid

- Uniform margin (cm): 0
- Variable margin (cm):
  - Right: 0
  - Left: 0
  - Anterior: 0
  - Posterior: 0
  - Superior: 0
  - Inferior: 0

- Expand
- Contract
- Create Ring ROI
- Dismiss
- Help
Script #2 Output in a Text Editor

WindowList .RoiExpandWindow .Create = "ROI Expansion/Contraction...";
RoiList .F"". ResetRoiExpandState = "Clear All";
RoiExpandControl .CheckTargetRoi = RoiList .F"2" .Address;
RoiExpandControl .ConstantPadding = " 0.5";
RoiExpandControl .TargetRoiName = "PTV 05";
RoiExpandControl .CreateNewTarget = "1";
RoiExpandControl .Expand = "1";
RoiExpandControl .DoExpand = "Expand";
WindowList .RoiExpandWindow .Create = "ROI Expansion/Contraction...";
RoiList .F"0" .ResetRoiExpandState = "Clear All";
RoiExpandControl .CheckTargetRoi = RoiList .F"0" .Address;
RoiExpandControl .CheckTargetRoi = RoiList .F"3" .Address;
RoiExpandControl .ConstantPadding = " 0.5";
RoiExpandControl .TargetRoiName = "Bladder_Avoid";
RoiExpandControl .CreateNewTarget = "1";
RoiExpandControl .Expand = "1";
RoiExpandControl .DoExpand = "Expand";
RoiExpandControl .CheckTargetRoi = RoiList .F"1" .Address;
RoiExpandControl .CheckTargetRoi = RoiList .F"1" .Address;
RoiExpandControl .TargetRoiName = "Rectum_Avoid";
RoiExpandControl .CreateNewTarget = "1";
RoiExpandControl .Expand = "1";
RoiExpandControl .DoExpand = "Expand";
RoiList .F"". ResetRoiExpandState = "Clear All";
RoiExpandControl .CheckTargetRoi = RoiList .F"2" .Address;
RoiExpandControl .ConstantPadding = " 1";
RoiExpandControl .TargetRoiName = "PTV 10";
RoiExpandControl .CreateNewTarget = "1";
RoiExpandControl .Expand = "1";
RoiExpandControl .DoExpand = "Expand";
RoiList .F"6". ResetRoiExpandState = "Clear All";
RoiExpandControl .CheckTargetRoi = RoiList .F"6" .Address;
RoiExpandControl .ConstantPadding = " 1";
RoiExpandControl .TargetRoiName = "Ring";
RoiExpandControl .CreateNewTarget = "1";
RoiExpandControl .DoRingExpansion = "Create Ring ROI";
WindowList .RoiExpandWindow .Create = "ROI Expansion/Contraction...";

RoiList ."*" .ResetRoiExpandState = "Clear All";

IF .RoiList ."#2" .RoiExpandState .Is .Source
  THEN .RoiList ."#2" .ResetRoiExpandState
  ELSE .RoiList ."#2" .RoiExpandState = "Source";

RoiExpandControl .CheckTargetRoi = RoiList ."#2" .Address;

RoiExpandControl .ConstantPadding = " 0.5";
RoiExpandControl .UseConstantPadding = "1";

RoiExpandControl .TargetRoiName = "PTV_05";
RoiExpandControl .CreateNewTarget = "1";

RoiExpandControl .Expand = "1";
RoiExpandControl .DoExpand = "Expand";
Code Snippet 1 of 7

WindowList .RoiExpandWindow .Create = "ROI Expansion/Contraction...";

• Open the ROI Expansion Window
• Clear all checkboxes (if not already clear)
IF .RoiList ."#"2" .RoiExpandState .Is .Source
THEN .RoiList ."#"2" .ResetRoiExpandState
ELSE .RoiList ."#"2" .RoiExpandState = "Source";

• IF Source checkbox for the 3rd ROI is already checked
  – THEN uncheck it
  – ELSE check it
• In other words, **toggle** the Source checkbox on PTV
• Why not just say we want it checked and reference the ROI by name while we're at it...

<table>
<thead>
<tr>
<th>ROI</th>
<th>Source</th>
<th>Avoid Interior</th>
<th>Avoid Exterior</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLADDER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECTUM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTV</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RoiList ."PTV"
Code Snippet 4 of 7

```csharp
RoiExpandControl.CheckTargetRoi = RoiList.#"#2".Address;
```

- No idea, but it looks important (no documentation)

- Let's at least reference the ROI by name

```csharp
RoiExpandControl.CheckTargetRoi = RoiList.#"PTV".Address;
```
Code Snippet 5 of 7

RoiExpandControl .ConstantPadding = " 0.5";
RoiExpandControl .UseConstantPadding = "1";

• Set 5 mm margin
• Uniform margin = True

• More readable with the order swapped

RoiExpandControl .UseConstantPadding = "1";
RoiExpandControl .ConstantPadding = " 0.5";
**Code Snippet 6 of 7**

```csharp
RoiExpandControl .TargetRoiName = "PTV_05";
RoiExpandControl .CreateNewTarget = "1";
```

- Set target name to PTV_05
- Create new ROI = True

- More readable with the order swapped

```csharp
RoiExpandControl .CreateNewTarget = "1";
RoiExpandControl .TargetRoiName = "PTV_05";
```
• We are indeed doing an expansion
• Do the expansion

RoiExpandControl .Expand = "1";
RoiExpandControl .DoExpand = "Expand";
Is the Script #2 Code Any Good?

• It's a LOT to take in
  – Mimics the way a human would do it rather than using a more straightforward computer method

• Needs some tweaks to make it useful

• Hard if you're just beginning (took me many hours)
  – There might be a better way to do this
Table of Contents

- The What and the Why
- Tools of the Trade
- Two Simple Script Ideas
- Writing Scripts in the Real World
  - MIM
  - Velocity
  - RayStation
  - Pinnacle
  - Eclipse
- Next Steps
Eclipse Scripting API (ESAPI)

- **API** = Application Programming Interface
  - Back end access to Eclipse session and patient database

- Not built-in. Have to run the Eclipse Scripting API installer to add files that are necessary for script development.
  - **NOTE**: Only Varian personnel may make changes to clinical systems. Must use a different workstation for script development.
Eclipse Support

• 31 page API Reference Guide
  – Explains main objects and their relationships
  – Includes instructions for installing API

• Varian Developer Website
  – https://variandev.developer.codeplex.com

• Training workshops
  – Example: Pre-conference workshop at AAPM
Types of ESAPI Scripts

• Plug-In Scripts
  – Accessed while in Eclipse

• Stand-Alone Scripts
  – Separate programs that can access Eclipse data
using System;
using System.Text;
using System.Windows;
using VMS.TPS.Common.Model.API;
namespace VMS.TPS
{
    class Script
    {
        public Script()
        {
        }
        public void Execute(ScriptContext context)
        {
            if (context.Patient != null)
            {
                MessageBox.Show("Patient id is " + context.Patient.Id);
            }
            else
            {
                MessageBox.Show("No patient selected");
            }
        }
    }
}
Current Varian Scripting Philosophy

- Giving users with unknown training direct access to the database can make serious errors possible

- A badly written script could damage the patient database and alter treatment
Not a Likely Outcome of Making Rings

YOU MANIACS!
YOU BLEW IT UP!
Look But Don't Touch

• For now, Eclipse script actions are **READ-ONLY**
  – No ROI editing
  – No ROI creation
  – No beam editing
  – Etc...
That's Going to Make This Hard

• **Script #1**
  – Can't rename anything

• **Script #2**
  – Can't edit or create new ROIs
Can We Do Anything ROI-Related?

- Could run a check to evaluate ROI names
  - Produce a list of name changes that the user should make?
Is There Anything Useful it CAN Do?

• Lot's of stuff!
  – Reports like **Clinical Goal Scorecards**
  – Plan quality checks
  – **Data mining**
    • E.g. - Create a report of rectal V70 dose for all prostate cases in the last 6 months
  – Research of many varieties

• But... today we are only talking about ROIs, so that will have to wait for a future presentation
Table of Contents

• The What and the Why
• Tools of the Trade
• Two Simple Script Ideas
• Writing Scripts in the Real World
  – MIM
  – Velocity
  – RayStation
  – Pinnacle
  – Eclipse
• Next Steps
Can I Really Do This?

- You are a technical person
- You learned to plan
- You can learn to make scripts
## Scripts the SPTC Uses Regularly

### Prostate Planning Structures

<table>
<thead>
<tr>
<th>Starting ROIs</th>
<th>Script-Generated ROIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prostate</td>
<td>PTVEval</td>
</tr>
<tr>
<td>CTV</td>
<td>Bladder_Avoid</td>
</tr>
<tr>
<td>PTV</td>
<td>BladderWall</td>
</tr>
<tr>
<td>Bladder</td>
<td>Rectum_Avoid</td>
</tr>
<tr>
<td>Rectum</td>
<td>RectalEval</td>
</tr>
<tr>
<td>FemoralHead_R</td>
<td>RectalWallEval</td>
</tr>
<tr>
<td>FemoralHead_L</td>
<td>Air_In_Rectum</td>
</tr>
<tr>
<td>Coil_R</td>
<td>Coil_R_01</td>
</tr>
<tr>
<td>Coil_L</td>
<td>Coil_L_01</td>
</tr>
<tr>
<td>Coil_Apex</td>
<td>Coil_Apex_01</td>
</tr>
</tbody>
</table>
Scripts the SPTC Uses Regularly

• Proton-compatible patient outline
Scripts the SPTC Uses Regularly

• Better Brain ROI
  – 7 Step script that makes a better Brain than built-in tools

• QA Scan Resample and Adaptive Recontour

• CSI PTV Chopper

• Head & Neck OptiPTV Maker
Other Ideas?

- Lexie Smith Raymond's 18 Opti Structures
- Thomas Constantino's SBRT ROIs
- Yolanda King's Dose Gradient Control Structures
Other Ideas?

- Pamela Lemish's 1 cm junction slices for **Low Gradient Field Matching**
- 2 hours to write
- Runs in 20 seconds
Your Homework

• Start thinking about where you waste time doing repetitive tasks

• Figure out if a script might help

• **TRY IT!**
Acknowledgements

• Sara St. James, PhD, DABR
• Eric Ford, PhD, FAAPM
• Mark Daniels, BS, RT(T), CMD
• Maggie Stauffer, MS
• Hazel Ramirez, MS, DABR
• Matthew Schmidt
• Sean Hames, MS
Questions?