[Get-Q3AController]: 10/06/2023

Introduction

The [purpose] of this [document] is to cover the custom [PowerShell] function (Get-Q3AController), and is meant to [supplement this video] on [YouTube]:

Date	Name	Link	Duration
10/05/2023	Q3A Map Controller Utility	https://youtu.be/pANgLvApb4E	02:48:19

Premise

Many, many years ago (2001-2006), I had a website that was hosted on [PlanetQuake] which focused on making maps for [Quake III Arena], though I had also made maps for many [other] various games, such as:

[+] [Doom I + II]
[+] [Duke Nukem 3D + Shadow Warrior]
[+] [Warcraft II + Starcraft]
[+] [Quake I + II]
[+] [Half-Life + CounterStrike]

In order to make the [website], I had to understand how to write [Hyper Text Markup Language].

I won't talk about how I [edited the website for many years], which is technically a level of [programming], believe it or not.

What I WILL talk about, is what this utility is, why I made it, how I made it, and how it applies to [application development], and being a [design artist].

The [premise] of this document, is to cover the [programming] that was featured in the aforementioned video, not necessarily my ability to play [Quake III Arena], nor how to make maps for the game.

Note: (1) of my maps, [Return to Castle: Quake], is featured in that video.

First, I will cover the [function wrapper], such as the [CmdLetBinding()]Param() and the branches at the tail end, and then I will cover [each individual class].

This isn't going to be a [meticulous] breakdown of EVERY single method and object, but it will showcase my [application development + documentation] capabilities and expertise.

Function

```
# Script: This requires [imagemagick] in order to convert (*.tga) -> (*.jpg) for the GUI

Function Get-Q3AController
{
    [CmdLetBinding()]Param(
    [Parameter()][ValidateSet(0,1)][UInt32]$Mode=0,
    [Parameter()][Object[]]$List)

    # <Insert classes here>

    # [Determine whether or not a list was provided as a parameter]
    If ($List)
    {
        $Ctrl = [Q3AController]::New($List)
    }
    Else
    {
        $Ctrl = [Q3AController]::New()
    }

    # [Initialize GUI, or return object]
    Switch ($Mode)
    {
        0 # Initializes the GUI
        {
            $Ctrl.StageXaml()
            $Ctrl.Xaml.Invoke()
        }
```

```
            1 # Returns the object which can still initialize the GUI
            {
                $Ctrl
            }
        }
    }
```

```
                                                                                                                          ___________/
\_________________________________________________________________________________________________________________________/ Function
  Class [Q3AControllerXaml] /-------------------------------------------------------------------------------------------------------\
/---------------------------
```

This particular [section] is [several pages], mainly because of my self-imposed [formatting limitations], in order to avoid [line wrapping]. [Every individual line] considers the [maximum line width] for this document, thus why it is [formatted] the way that it is, and spans [several pages].

```
# // ===================================
# // | Q3A Controller Xaml for the GUI |
# // ===================================

Class Q3AControllerXaml
{
    Static [String] $Content = @(
    '<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"',
    '        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"',
    '        Title="[FightingEntropy()]://Quake III Arena Configuration Utility"',
    '        Height="640"',
    '        Width="800"',
    '        ResizeMode="NoResize"',
    '        Icon="C:\ProgramData\Secure Digits Plus LLC\FightingEntropy\2023.8.0\Graphics\icon.ico"',
    '        HorizontalAlignment="Center"',
    '        WindowStartupLocation="CenterScreen"',
    '        FontFamily="Consolas"',
    '        Background="LightYellow">',
    '    <Window.Resources>',
    '        <Style x:Key="DropShadow">',
    '            <Setter Property="TextBlock.Effect">',
    '                <Setter.Value>',
    '                    <DropShadowEffect ShadowDepth="1"/>',
    '                </Setter.Value>',
    '            </Setter>',
    '        </Style>',
    '        <Style TargetType="ToolTip">',
    '            <Setter Property="Background" Value="#000000"/>',
    '            <Setter Property="Foreground" Value="#66D066"/>',
    '        </Style>',
    '        <Style TargetType="TabItem">',
    '            <Setter Property="Template">',
    '                <Setter.Value>',
    '                    <ControlTemplate TargetType="TabItem">',
    '                        <Border Name="Border"',
    '                                BorderThickness="2"',
    '                                BorderBrush="Black"',
    '                                CornerRadius="5"',
    '                                Margin="2">',
    '                            <ContentPresenter x:Name="ContentSite"',
    '                                              VerticalAlignment="Center"',
    '                                              HorizontalAlignment="Right"',
    '                                              ContentSource="Header"',
    '                                              Margin="5"/>',
    '                        </Border>',
    '                        <ControlTemplate.Triggers>',
    '                            <Trigger Property="IsSelected"',
    '                                     Value="True">',
    '                                <Setter TargetName="Border"',
    '                                        Property="Background"',
    '                                        Value="#4444FF"/>',
    '                                <Setter Property="Foreground"',
    '                                        Value="#FFFFFF"/>',
    '                            </Trigger>',
    '                            <Trigger Property="IsSelected"',
    '                                     Value="False">',
    '                                <Setter TargetName="Border"',
    '                                        Property="Background"',
    '                                        Value="#DFFFBA"/>',
    '                                <Setter Property="Foreground"',
    '                                        Value="#000000"/>',
    '                            </Trigger>',
    '                        </ControlTemplate.Triggers>',
    '                    </ControlTemplate>',
    '                </Setter.Value>',
    '            </Setter>',
```

```
'        </Style>',
'        <Style TargetType="Button">',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="Padding" Value="5"/>',
'            <Setter Property="FontWeight" Value="Heavy"/>',
'            <Setter Property="Foreground" Value="Black"/>',
'            <Setter Property="Background" Value="#DFFFBA"/>',
'            <Setter Property="BorderThickness" Value="2"/>',
'            <Setter Property="VerticalContentAlignment" Value="Center"/>',
'            <Style.Resources>',
'                <Style TargetType="Border">',
'                    <Setter Property="CornerRadius" Value="5"/>',
'                </Style>',
'            </Style.Resources>',
'        </Style>',
'        <Style TargetType="{x:Type TextBox}" BasedOn="{StaticResource DropShadow}">',
'            <Setter Property="TextBlock.TextAlignment" Value="Left"/>',
'            <Setter Property="VerticalContentAlignment" Value="Center"/>',
'            <Setter Property="HorizontalContentAlignment" Value="Left"/>',
'            <Setter Property="Height" Value="24"/>',
'            <Setter Property="Margin" Value="4"/>',
'            <Setter Property="FontSize" Value="12"/>',
'            <Setter Property="Foreground" Value="#000000"/>',
'            <Setter Property="TextWrapping" Value="Wrap"/>',
'            <Style.Resources>',
'                <Style TargetType="Border">',
'                    <Setter Property="CornerRadius" Value="2"/>',
'                </Style>',
'            </Style.Resources>',
'        </Style>',
'        <Style TargetType="{x:Type PasswordBox}" BasedOn="{StaticResource DropShadow}">',
'            <Setter Property="TextBlock.TextAlignment" Value="Left"/>',
'            <Setter Property="VerticalContentAlignment" Value="Center"/>',
'            <Setter Property="HorizontalContentAlignment" Value="Left"/>',
'            <Setter Property="Margin" Value="4"/>',
'            <Setter Property="Height" Value="24"/>',
'            <Style.Resources>',
'                <Style TargetType="Border">',
'                    <Setter Property="CornerRadius" Value="2"/>',
'                </Style>',
'            </Style.Resources>',
'        </Style>',
'        <Style TargetType="ComboBox">',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="Padding" Value="2"/>',
'            <Setter Property="Height" Value="20"/>',
'            <Setter Property="VerticalContentAlignment" Value="Center"/>',
'        </Style>',
'        <Style x:Key="DGCombo" TargetType="ComboBox">',
'            <Setter Property="Margin" Value="0"/>',
'            <Setter Property="Padding" Value="2"/>',
'            <Setter Property="Height" Value="18"/>',
'            <Setter Property="FontSize" Value="10"/>',
'            <Setter Property="VerticalContentAlignment" Value="Center"/>',
'        </Style>',
'        <Style TargetType="CheckBox">',
'            <Setter Property="VerticalAlignment" Value="Center"/>',
'            <Setter Property="HorizontalAlignment" Value="Center"/>',
'        </Style>',
'        <Style TargetType="DataGrid">',
'            <Setter Property="Margin"',
'                    Value="5"/>',
'            <Setter Property="AutoGenerateColumns"',
'                    Value="False"/>',
'            <Setter Property="AlternationCount"',
'                    Value="2"/>',
'            <Setter Property="HeadersVisibility"',
'                    Value="Column"/>',
'            <Setter Property="CanUserResizeRows"',
'                    Value="False"/>',
'            <Setter Property="CanUserAddRows"',
'                    Value="False"/>',
'            <Setter Property="IsReadOnly"',
'                    Value="True"/>',
'            <Setter Property="IsTabStop"',
'                    Value="True"/>',
'            <Setter Property="IsTextSearchEnabled"',
'                    Value="True"/>',
'            <Setter Property="SelectionMode"',
'                    Value="Single"/>',
'            <Setter Property="ScrollViewer.CanContentScroll"',
'                    Value="True"/>',
```

```
'            <Setter Property="ScrollViewer.VerticalScrollBarVisibility"',
'                    Value="Auto"/>',
'            <Setter Property="ScrollViewer.HorizontalScrollBarVisibility"',
'                    Value="Auto"/>',
'        </Style>',
'        <Style TargetType="DataGridRow">',
'            <Setter Property="VerticalAlignment"',
'                    Value="Center"/>',
'            <Setter Property="VerticalContentAlignment"',
'                    Value="Center"/>',
'            <Setter Property="TextBlock.VerticalAlignment"',
'                    Value="Center"/>',
'            <Setter Property="Height"   Value="20"/>',
'            <Setter Property="FontSize" Value="12"/>',
'            <Style.Triggers>',
'                <Trigger Property="AlternationIndex"',
'                         Value="0">',
'                    <Setter Property="Background"',
'                            Value="#F8FFFFFF"/>',
'                </Trigger>',
'                <Trigger Property="AlternationIndex"',
'                         Value="1">',
'                    <Setter Property="Background"',
'                            Value="#FFF8FFFF"/>',
'                </Trigger>',
'                <Trigger Property="AlternationIndex"',
'                         Value="2">',
'                    <Setter Property="Background"',
'                            Value="#FFFFF8FF"/>',
'                </Trigger>',
'                <Trigger Property="AlternationIndex"',
'                         Value="3">',
'                    <Setter Property="Background"',
'                            Value="#F8F8F8FF"/>',
'                </Trigger>',
'                <Trigger Property="AlternationIndex"',
'                         Value="4">',
'                    <Setter Property="Background"',
'                            Value="#F8FFF8FF"/>',
'                </Trigger>',
'                <Trigger Property="IsMouseOver" Value="True">',
'                    <Setter Property="ToolTip">',
'                        <Setter.Value>',
'                            <TextBlock Text="{Binding Description}"',
'                                       TextWrapping="Wrap"',
'                                       FontFamily="Consolas"',
'                                       Width="400"',
'                                       Background="#000000"',
'                                       Foreground="#00FF00"/>',
'                        </Setter.Value>',
'                    </Setter>',
'                    <Setter Property="ToolTipService.ShowDuration"',
'                            Value="360000000"/>',
'                </Trigger>',
'            </Style.Triggers>',
'        </Style>',
'        <Style x:Key="xTextBlock" TargetType="TextBlock">',
'            <Setter Property="TextWrapping"',
'                    Value="WrapWithOverflow"/>',
'            <Setter Property="FontFamily"',
'                    Value="Consolas"/>',
'            <Setter Property="FontWeight"',
'                    Value="Heavy"/>',
'            <Setter Property="Background"',
'                    Value="#000000"/>',
'            <Setter Property="Foreground"',
'                    Value="#00FF00"/>',
'        </Style>',
'        <Style x:Key="xDataGridRow"',
'               TargetType="DataGridRow">',
'            <Setter Property="VerticalAlignment"',
'                    Value="Center"/>',
'            <Setter Property="VerticalContentAlignment"',
'                    Value="Center"/>',
'            <Setter Property="TextBlock.VerticalAlignment"',
'                    Value="Center"/>',
'            <Setter Property="Height"',
'                    Value="20"/>',
'            <Setter Property="FontSize"',
'                    Value="12"/>',
'            <Setter Property="FontWeight"',
'                    Value="Heavy"/>',
```

```
'        </Style>',
'        <Style TargetType="DataGridColumnHeader">',
'            <Setter Property="FontSize"   Value="10"/>',
'            <Setter Property="FontWeight" Value="Normal"/>',
'        </Style>',
'        <Style TargetType="TabControl">',
'            <Setter Property="TabStripPlacement" Value="Top"/>',
'            <Setter Property="HorizontalContentAlignment" Value="Center"/>',
'            <Setter Property="Background" Value="LightYellow"/>',
'        </Style>',
'        <Style TargetType="GroupBox">',
'            <Setter Property="Foreground" Value="Black"/>',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="FontSize" Value="12"/>',
'            <Setter Property="FontWeight" Value="Normal"/>',
'        </Style>',
'        <Style TargetType="Label">',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="FontWeight" Value="Bold"/>',
'            <Setter Property="Background" Value="Black"/>',
'            <Setter Property="Foreground" Value="White"/>',
'            <Setter Property="BorderBrush" Value="Gray"/>',
'            <Setter Property="BorderThickness" Value="2"/>',
'            <Style.Resources>',
'                <Style TargetType="Border">',
'                    <Setter Property="CornerRadius" Value="5"/>',
'                </Style>',
'            </Style.Resources>',
'        </Style>',
'        <Style x:Key="LabelGray" TargetType="Label">',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="FontWeight" Value="Bold"/>',
'            <Setter Property="Background" Value="DarkSlateGray"/>',
'            <Setter Property="Foreground" Value="White"/>',
'            <Setter Property="BorderBrush" Value="Black"/>',
'            <Setter Property="BorderThickness" Value="2"/>',
'            <Setter Property="HorizontalContentAlignment" Value="Center"/>',
'            <Style.Resources>',
'                <Style TargetType="Border">',
'                    <Setter Property="CornerRadius" Value="5"/>',
'                </Style>',
'            </Style.Resources>',
'        </Style>',
'        <Style x:Key="LabelRed" TargetType="Label">',
'            <Setter Property="Margin" Value="5"/>',
'            <Setter Property="FontWeight" Value="Bold"/>',
'            <Setter Property="Background" Value="IndianRed"/>',
'            <Setter Property="Foreground" Value="White"/>',
'            <Setter Property="BorderBrush" Value="Black"/>',
'            <Setter Property="BorderThickness" Value="2"/>',
'            <Setter Property="HorizontalContentAlignment" Value="Center"/>',
'            <Style.Resources>',
'                <Style TargetType="Border">',
'                    <Setter Property="CornerRadius" Value="5"/>',
'                </Style>',
'            </Style.Resources>',
'        </Style>',
'        <Style x:Key="Line" TargetType="Border">',
'            <Setter Property="Background" Value="Black"/>',
'            <Setter Property="BorderThickness" Value="0"/>',
'            <Setter Property="Margin" Value="4"/>',
'        </Style>',
'    </Window.Resources>',
'    <Grid>',
'        <Grid.RowDefinitions>',
'            <RowDefinition Height="140"/>',
'            <RowDefinition Height="280"/>',
'            <RowDefinition Height="*"/>',
'            <RowDefinition Height="40"/>',
'        </Grid.RowDefinitions>',
'        <Grid Grid.Row="0">',
'            <Grid.ColumnDefinitions>',
'                <ColumnDefinition Width="*"/>',
'                <ColumnDefinition Width="200"/>',
'            </Grid.ColumnDefinitions>',
'            <Grid Grid.Column="0">',
'                <Grid.RowDefinitions>',
'                    <RowDefinition Height="40"/>',
'                    <RowDefinition Height="*"/>',
'                </Grid.RowDefinitions>',
'                <Grid Grid.Row="0">',
'                    <Grid.ColumnDefinitions>',
```

```
'                    <ColumnDefinition Width="90"/>',
'                    <ColumnDefinition Width="*"/>',
'                    <ColumnDefinition Width="25"/>',
'                </Grid.ColumnDefinitions>',
'                <Label Grid.Column="0"',
'                       Content="[Game]:"/>',
'                <TextBox Name="Game"',
'                         Grid.Column="1"/>',
'                <Image Name="GameIcon"',
'                       Grid.Column="2"/>',
'            </Grid>',
'            <DataGrid Grid.Row="1"',
'                      Name="Property"',
'                      HeadersVisibility="None">',
'                <DataGrid.Columns>',
'                    <DataGridTextColumn Header="Name"',
'                                        Width="60"',
'                                        Binding="{Binding Name}"/>',
'                    <DataGridTextColumn Header="Value"',
'                                        Width="*"',
'                                        Binding="{Binding Value}"/>',
'                </DataGrid.Columns>',
'            </DataGrid>',
'        </Grid>',
'        <Grid Grid.Column="1">',
'            <Grid.RowDefinitions>',
'                <RowDefinition Height="40"/>',
'                <RowDefinition Height="*"/>',
'            </Grid.RowDefinitions>',
'            <Grid Grid.Row="0">',
'                <Grid.ColumnDefinitions>',
'                    <ColumnDefinition Width="*"/>',
'                    <ColumnDefinition Width="*"/>',
'                </Grid.ColumnDefinitions>',
'                <Button Name="Browse"',
'                        Content="Browse"',
'                        Grid.Column="0"/>',
'                <Button Name="Set"',
'                        Grid.Column="1"',
'                        IsEnabled="False"',
'                        Content="Set"/>',
'            </Grid>',
'            <Grid Grid.Row="1">',
'                <Grid.RowDefinitions>',
'                    <RowDefinition Height="10"/>',
'                    <RowDefinition Height="40"/>',
'                    <RowDefinition Height="40"/>',
'                    <RowDefinition Height="10"/>',
'                </Grid.RowDefinitions>',
'                <Border Grid.Row="0" Background="Black" Margin="4"/>',
'                <Grid Grid.Row="1">',
'                    <Grid.ColumnDefinitions>',
'                        <ColumnDefinition Width="10"/>',
'                        <ColumnDefinition Width="120"/>',
'                        <ColumnDefinition Width="*"/>',
'                        <ColumnDefinition Width="10"/>',
'                    </Grid.ColumnDefinitions>',
'                    <Label Grid.Column="1"',
'                           Content="[Archive(s)]:"/>',
'                    <TextBox Name="ArchiveCount"',
'                             Grid.Column="2"/>',
'                </Grid>',
'                <Grid Grid.Row="2">',
'                    <Grid.ColumnDefinitions>',
'                        <ColumnDefinition Width="10"/>',
'                        <ColumnDefinition Width="120"/>',
'                        <ColumnDefinition Width="*"/>',
'                        <ColumnDefinition Width="10"/>',
'                    </Grid.ColumnDefinitions>',
'                    <Label Grid.Column="1"',
'                           Content="[Level(s)]:"/>',
'                    <TextBox Grid.Column="2"',
'                             Name="LevelCount"/>',
'                </Grid>',
'                <Border Grid.Row="3" Background="Black" Margin="4"/>',
'            </Grid>',
'        </Grid>',
'    </Grid>',
'    <TabControl Grid.Row="1">',
'        <TabItem Header="Archive(s)">',
'            <Grid>',
'                <Grid.RowDefinitions>',
```

```
'                    <RowDefinition Height="*"/>',
'                </Grid.RowDefinitions>',
'                <DataGrid Name="Archive">',
'                    <DataGrid.RowStyle>',
'                        <Style TargetType="{x:Type DataGridRow}"',
'                           BasedOn="{StaticResource xDataGridRow}">',
'                            <Style.Triggers>',
'                                <Trigger Property="IsMouseOver" Value="True">',
'                                    <Setter Property="ToolTip">',
'                                        <Setter.Value>',
'                                            <TextBlock Text="{Binding Fullname}"',
'                                                       Style="{StaticResource xTextBlock}"/>',
'                                        </Setter.Value>',
'                                    </Setter>',
'                                    <Setter Property="ToolTipService.ShowDuration"',
'                                            Value="360000000"/>',
'                                </Trigger>',
'                            </Style.Triggers>',
'                        </Style>',
'                    </DataGrid.RowStyle>',
'                    <DataGrid.Columns>',
'                        <DataGridTextColumn Header="#"',
'                                            Width="40"',
'                                            Binding="{Binding Index}"/>',
'                        <DataGridTextColumn Header="Name"',
'                                            Width="*"',
'                                            Binding="{Binding Name}"/>',
'                    </DataGrid.Columns>',
'                </DataGrid>',
'            </Grid>',
'        </TabItem>',
'        <TabItem Header="Level(s)">',
'            <Grid>',
'                <Grid.ColumnDefinitions>',
'                    <ColumnDefinition Width="2*"/>',
'                    <ColumnDefinition Width="*"/>',
'                </Grid.ColumnDefinitions>',
'                <Grid Grid.Column="0">',
'                    <Grid.RowDefinitions>',
'                        <RowDefinition Height="40"/>',
'                        <RowDefinition Height="*"/>',
'                        <RowDefinition Height="40"/>',
'                    </Grid.RowDefinitions>',
'                    <Grid Grid.Row="0">',
'                        <Grid.ColumnDefinitions>',
'                            <ColumnDefinition Width="90"/>',
'                            <ColumnDefinition Width="*"/>',
'                        </Grid.ColumnDefinitions>',
'                        <Label Grid.Column="0"',
'                               Content="[Level]:"/>',
'                        <TextBox Grid.Column="1"',
'                                 Name="LevelName"/>',
'                    </Grid>',
'                    <DataGrid Grid.Row="1"',
'                              Name="Level">',
'                        <DataGrid.RowStyle>',
'                            <Style TargetType="{x:Type DataGridRow}"',
'                                   BasedOn="{StaticResource xDataGridRow}">',
'                                <Style.Triggers>',
'                                    <Trigger Property="IsMouseOver" Value="True">',
'                                        <Setter Property="ToolTip">',
'                                            <Setter.Value>',
'                                                <TextBlock Text="{Binding Fullname}"',
'                                                           Style="{StaticResource xTextBlock}"/>',
'                                            </Setter.Value>',
'                                        </Setter>',
'                                        <Setter Property="ToolTipService.ShowDuration"',
'                                                Value="360000000"/>',
'                                    </Trigger>',
'                                </Style.Triggers>',
'                            </Style>',
'                        </DataGrid.RowStyle>',
'                        <DataGrid.Columns>',
'                            <DataGridTextColumn Header="#"',
'                                                Width="40"',
'                                                Binding="{Binding Index}"/>',
'                            <DataGridTextColumn Header="Date"',
'                                                Binding="{Binding Date}"',
'                                                Width="75"/>',
'                            <DataGridTextColumn Header="Time"',
'                                                Binding="{Binding Time}"',
'                                                Width="90"/>',
```

```
'                            <DataGridTextColumn Header="Name"',
'                                        Width="*"',
'                                        Binding="{Binding Name}"/>',
'                            <DataGridTemplateColumn Header="[+]" Width="25">',
'                                <DataGridTemplateColumn.CellTemplate>',
'                                    <DataTemplate>',
'                                        <CheckBox IsChecked="{Binding Profile,',
'                                                              Mode=TwoWay,',
'                                                              NotifyOnSourceUpdated=True,',
'                                                              NotifyOnTargetUpdated=True,',
'                                                              UpdateSourceTrigger=PropertyChanged}">',
'                                            <CheckBox.LayoutTransform>',
'                                                <ScaleTransform ScaleX="0.9" ScaleY="0.9"/>',
'                                            </CheckBox.LayoutTransform>',
'                                        </CheckBox>',
'                                    </DataTemplate>',
'                                </DataGridTemplateColumn.CellTemplate>',
'                            </DataGridTemplateColumn>',
'                        </DataGrid.Columns>',
'                    </DataGrid>',
'                    <Grid Grid.Row="2">',
'                        <Grid.ColumnDefinitions>',
'                            <ColumnDefinition Width="90"/>',
'                            <ColumnDefinition Width="*"/>',
'                            <ColumnDefinition Width="25"/>',
'                            <ColumnDefinition Width="90"/>',
'                        </Grid.ColumnDefinitions>',
'                        <Label Grid.Column="0"',
'                               Content="[Config]:"/>',
'                        <TextBox Grid.Column="1"',
'                                 Name="ConfigName"/>',
'                        <Image Grid.Column="2"',
'                               Name="ConfigNameIcon"/>',
'                        <Button Grid.Column="3"',
'                                Name="Create"',
'                                Content="Create"',
'                                IsEnabled="False"/>',
'                    </Grid>',
'                </Grid>',
'                <Grid Grid.Column="1">',
'                    <Grid.RowDefinitions>',
'                        <RowDefinition Height="*"/>',
'                        <RowDefinition Height="40"/>',
'                    </Grid.RowDefinitions>',
'                    <Image Grid.Column="0"',
'                           Name="Image"/>',
'                    <Grid Grid.Row="1">',
'                        <Grid.ColumnDefinitions>',
'                            <ColumnDefinition Width="*"/>',
'                        </Grid.ColumnDefinitions>',
'                        <Button Grid.Column="1"',
'                                Name="Clear"',
'                                Content="Clear"/>',
'                    </Grid>',
'                </Grid>',
'            </Grid>',
'        </TabItem>',
'        <TabItem Header="Config">',
'            <TabControl>',
'                <TabItem Header="Queue">',
'                    <Grid>',
'                        <Grid.RowDefinitions>',
'                            <RowDefinition Height="*"/>',
'                            <RowDefinition Height="40"/>',
'                            <RowDefinition Height="40"/>',
'                        </Grid.RowDefinitions>',
'                        <DataGrid Grid.Row="0"',
'                                  Name="Config">',
'                            <DataGrid.RowStyle>',
'                                <Style TargetType="{x:Type DataGridRow}"',
'                                       BasedOn="{StaticResource xDataGridRow}">',
'                                    <Style.Triggers>',
'                                        <Trigger Property="IsMouseOver" Value="True">',
'                                            <Setter Property="ToolTip">',
'                                                <Setter.Value>',
'                                                    <TextBlock Text="{Binding Title}"',
'                                                               Style="{StaticResource xTextBlock}"/>',
'                                                </Setter.Value>',
'                                            </Setter>',
'                                            <Setter Property="ToolTipService.ShowDuration"',
'                                                    Value="360000000"/>',
'                                        </Trigger>',
```

```
'                                </Style.Triggers>',
'                            </Style>',
'                        </DataGrid.RowStyle>',
'                        <DataGrid.Columns>',
'                            <DataGridTextColumn Header="#"',
'                                                Width="40"',
'                                                Binding="{Binding Index}"/>',
'                            <DataGridTextColumn Header="Date"',
'                                                Binding="{Binding Date}"',
'                                                Width="75"/>',
'                            <DataGridTextColumn Header="Time"',
'                                                Binding="{Binding Time}"',
'                                                Width="90"/>',
'                            <DataGridTextColumn Header="Name"',
'                                                Width="150"',
'                                                Binding="{Binding Name}"/>',
'                            <DataGridTextColumn Header="Title"',
'                                                Width="*"',
'                                                Binding="{Binding Title}"/>',
'                            <DataGridTextColumn Header="Mode"',
'                                                Width="60"',
'                                                Binding="{Binding ModeStr}"/>',
'                            <DataGridTextColumn Header="Rating"',
'                                                Width="60"',
'                                                Binding="{Binding Rating}"/>',
'                        </DataGrid.Columns>',
'                    </DataGrid>',
'                    <Grid Grid.Row="1">',
'                        <Grid.ColumnDefinitions>',
'                            <ColumnDefinition Width="160"/>',
'                            <ColumnDefinition Width="*"/>',
'                            <ColumnDefinition Width="90"/>',
'                            <ColumnDefinition Width="90"/>',
'                            <ColumnDefinition Width="60"/>',
'                        </Grid.ColumnDefinitions>',
'                        <Label Grid.Column="0"',
'                               Content="[Title/Mode/Rating]:"/>',
'                        <TextBox Grid.Column="1"',
'                                 Name="MapTitle"/>',
'                        <TextBox Grid.Column="2"',
'                                 Name="MapMode"/>',
'                        <TextBox Grid.Column="3"',
'                                 Name="MapRating"/>',
'                        <Button Grid.Column="4"',
'                                Name="Apply"',
'                                Content="Apply"/>',
'                    </Grid>',
'                    <Grid Grid.Row="2">',
'                        <Grid.ColumnDefinitions>',
'                            <ColumnDefinition Width="*"/>',
'                            <ColumnDefinition Width="*"/>',
'                            <ColumnDefinition Width="*"/>',
'                        </Grid.ColumnDefinitions>',
'                        <Button Grid.Column="0"',
'                                Name="Randomize"',
'                                Content="Randomize"/>',
'                        <Button Grid.Column="1"',
'                                Name="Export"',
'                                Content="Export"/>',
'                        <Button Grid.Column="2"',
'                                Name="Import"',
'                                Content="Import"/>',
'                    </Grid>',
'                </Grid>',
'            </TabItem>',
'            <TabItem Header="Content">',
'                <TextBox Name="Content"',
'                 Height="220"/>',
'            </TabItem>',
'        </TabControl>',
'    </TabItem>',
'</TabControl>',
'<Grid Grid.Row="2">',
'    <Grid.ColumnDefinitions>',
'        <ColumnDefinition Width="*"/>',
'    </Grid.ColumnDefinitions>',
'    <DataGrid Grid.Row="2"',
'              Name="Console"',
'              SelectionMode="Extended">',
'        <DataGrid.RowStyle>',
'            <Style TargetType="{x:Type DataGridRow}"',
```

```
        '                            BasedOn="{StaticResource xDataGridRow}">',
        '                        <Style.Triggers>',
        '                            <Trigger Property="IsMouseOver" Value="True">',
        '                                <Setter Property="ToolTip">',
        '                                    <Setter.Value>',
        '                                        <TextBlock Text="{Binding String}"',
        '                                                   Style="{StaticResource xTextBlock}"/>',
        '                                    </Setter.Value>',
        '                                </Setter>',
        '                                <Setter Property="ToolTipService.ShowDuration"',
        '                                        Value="360000000"/>',
        '                            </Trigger>',
        '                        </Style.Triggers>',
        '                    </Style>',
        '                </DataGrid.RowStyle>',
        '                <DataGrid.Columns>',
        '                    <DataGridTextColumn Header="#"',
        '                                        Binding="{Binding Index}"',
        '                                        Width="50"/>',
        '                    <DataGridTextColumn Header="Elapsed"',
        '                                        Binding="{Binding Elapsed}"',
        '                                        Width="125"/>',
        '                    <DataGridTextColumn Header="State"',
        '                                        Binding="{Binding State}"',
        '                                        Width="50"/>',
        '                    <DataGridTextColumn Header="Status"',
        '                                        Binding="{Binding Status}"',
        '                                        Width="*"/>',
        '                </DataGrid.Columns>',
        '            </DataGrid>',
        '        </Grid>',
        '        <Grid Grid.Row="3">',
        '            <Grid.ColumnDefinitions>',
        '                <ColumnDefinition Width="*"/>',
        '                <ColumnDefinition Width="90"/>',
        '                <ColumnDefinition Width="*"/>',
        '            </Grid.ColumnDefinitions>',
        '            <Button Grid.Column="1"',
        '                    Name="Launch"',
        '                    Content="Launch"/>',
        '        </Grid>',
        '    </Grid>',
        '</Window>' -join "`n")
    }

                                                                                    ___________________________/
\__________________________________________________________________________________/ Class [Q3AControllerXaml]
  Class [XamlProperty] /---------------------------------------------------------------------------------------\
/---------------------

    # // ==========================================================
    # // | Provides an individual Xaml property access to controls |
    # // ==========================================================

    Class XamlProperty
    {
        [UInt32]   $Index
        [String]    $Name
        [Object]    $Type
        [Object] $Control
        XamlProperty([UInt32]$Index,[String]$Name,[Object]$Object)
        {
            $This.Index   = $Index
            $This.Name    = $Name
            $This.Type    = $Object.GetType().Name
            $This.Control = $Object
        }
        [String] ToString()
        {
            Return $This.Name
        }
    }
                                                                                         ______________________/
\_______________________________________________________________________________________/ Class [XamlProperty]
  Class [XamlWindow] /-----------------------------------------------------------------------------------------\
/-------------------

    # // =================================================================
    # // | Creates an object that (processes/controls) the Xaml +  Window |
    # // =================================================================
```

```
    Class XamlWindow
    {
        Hidden [Object]        $Xaml
        Hidden [Object]         $Xml
        [String[]]            $Names
        [Object]              $Types
        [Object]               $Node
        [Object]                 $IO
        [String]          $Exception
        XamlWindow([String]$Xaml)
        {
            If (!$Xaml)
            {
                Throw "Invalid XAML Input"
            }

            [System.Reflection.Assembly]::LoadWithPartialName('presentationframework')

            $This.Xaml           = $Xaml
            $This.Xml            = [XML]$Xaml
            $This.Names          = $This.FindNames()
            $This.Types          = @( )
            $This.Node           = [System.Xml.XmlNodeReader]::New($This.Xml)
            $This.IO             = [System.Windows.Markup.XamlReader]::Load($This.Node)

            ForEach ($X in 0..($This.Names.Count-1))
            {
                $Name            = $This.Names[$X]
                $Object          = $This.IO.FindName($Name)
                $This.IO         | Add-Member -MemberType NoteProperty -Name $Name -Value $Object -Force
                If (!!$Object)
                {
                    $This.Types += $This.XamlProperty($This.Types.Count,$Name,$Object)
                }
            }
        }
        [String[]] FindNames()
        {
            Return [Regex]::Matches($This.Xaml,"( Name\=\`"\w+`")").Value -Replace "( Name=|`")",""
        }
        [Object] XamlProperty([UInt32]$Index,[String]$Name,[Object]$Object)
        {
            Return [XamlProperty]::New($Index,$Name,$Object)
        }
        [Object] Get([String]$Name)
        {
            $Item = $This.Types | ? Name -eq $Name
            If ($Item)
            {
                Return $Item.Control
            }
            Else
            {
                Return $Null
            }
        }
        Invoke()
        {
            Try
            {
                $This.IO.Dispatcher.InvokeAsync({ $This.IO.ShowDialog() }).Wait()
            }
            Catch
            {
                $This.Exception = $PSItem
            }
        }
        [String] ToString()
        {
            Return "<FEModule.XamlWindow[Q3AControllerXaml]>"
        }
    }

                                                                                                    ____________________/
\__________________________________________________________________________________________________/ Class [XamlWindow]
  Class [Q3AProperty] /-----------------------------------------------------------------------------------------------\
/---------------------

    # // ====================================================
    # // | Meant to contain DataGrid properties for the game |
    # // ====================================================
```

```
    Class Q3AProperty
    {
        [String] $Name
        [String] $Value
        Q3AProperty([String]$Name,[String]$Value)
        {
            $This.Name  = $Name
            $This.Value = $Value
        }
    }

                                                                                                    ____________________/
\__________________________________________________________________________________________________/ Class [Q3AProperty]
  Class [Pk3FileBsp] /-------------------------------------------------------------------------------------------------------\
/--------------------

    # // ===================================================================
    # // | Used as a template for any (*.bsp) to populate (Level + Config) |
    # // ===================================================================

    Class Pk3FileBsp
    {
        [UInt32]         $Index
        Hidden [DateTime] $Real
        [String]          $Date
        [String]          $Time
        [String]          $Name
        [UInt32]       $Profile
        [String]         $Image
        Pk3FileBsp([UInt32]$Index,[Object]$Bsp)
        {
            $This.Index    = $Index
            $This.Real     = $Bsp.LastWriteTime.ToString("MM/dd/yyyy HH:mm:ss")
            $This.Date     = $This.Real.ToString("MM/dd/yyyy")
            $This.Time     = $This.Real.ToString("HH:mm:ss")
            $This.Name     = $Bsp.Name -ireplace "\.bsp",""
        }
        SetProfile([UInt32]$xProfile)
        {
            $This.Profile = $xProfile
        }
        SetImage([String]$Image)
        {
            $This.Image  = $Image
        }
        [String] ToString()
        {
            Return "{0} {1} {2}" -f $This.Date, $This.Time, $This.Name
        }
    }

                                                                                                     ___________________/
\___________________________________________________________________________________________________/ Class [Pk3FileBsp]
  Class [Pk3FileEntrySize] /-----------------------------------------------------------------------------------------------\
/-------------------------

    # // =======================================================
    # // | Represents the compressed size of a particular file |
    # // =======================================================

    Class Pk3FileEntrySize
    {
        [String]   $Name
        [UInt64]  $Bytes
        [String]   $Unit
        [String]   $Size
        Pk3FileEntrySize([UInt64]$Bytes)
        {
            $This.Name   = "Compressed"
            $This.Bytes  = $Bytes
            $This.GetUnit()
            $This.GetSize()
        }
        GetUnit()
        {
            $This.Unit   = Switch ($This.Bytes)
            {
                {$_ -lt 1KB}                 {     "Byte" }
                {$_ -ge 1KB -and $_ -lt 1MB} { "Kilobyte" }
                {$_ -ge 1MB -and $_ -lt 1GB} { "Megabyte" }
                {$_ -ge 1GB -and $_ -lt 1TB} { "Gigabyte" }
                {$_ -ge 1TB}                 { "Terabyte" }
            }
```

```
        }
        GetSize()
        {
            $This.Size   = Switch -Regex ($This.Unit)
            {
                ^Byte     {     "{0} B" -f  $This.Bytes      }
                ^Kilobyte { "{0:n2} KB" -f ($This.Bytes/1KB) }
                ^Megabyte { "{0:n2} MB" -f ($This.Bytes/1MB) }
                ^Gigabyte { "{0:n2} GB" -f ($This.Bytes/1GB) }
                ^Terabyte { "{0:n2} TB" -f ($This.Bytes/1TB) }
            }
        }
        [String] ToString()
        {
            Return $This.Size
        }
    }

\_____________________________________________________________________________________________/ Class [Pk3FileEntrySize] /---------------------------------------------------------------------------------------------\
  Class [Pk3FileEntry] /----------------------
/

    # // ================================================================
    # // | Template for each individual entry in all selected (*.pk3) files |
    # // ================================================================

    Class Pk3FileEntry
    {
        [UInt32]         $Index
        Hidden [DateTime] $Real
        [String]          $Date
        [Object]          $Size
        [String]          $Name
        [String]      $Fullname
        Pk3FileEntry([UInt32]$Index,[Object]$Entry)
        {
            $This.Index    = $Index
            $This.Real     = $Entry.LastWriteTime.ToString("MM/dd/yyyy HH:mm:ss")
            $This.Date     = $This.Real.ToString("MM/dd/yyyy HH:mm:ss")
            $This.Size     = $This.Pk3FileEntrySize($Entry.CompressedLength)
            $This.Name     = $Entry.Name
            $This.Fullname = $Entry.Fullname
        }
        [Object] Pk3FileEntrySize([UInt64]$Bytes)
        {
            Return [Pk3FileEntrySize]::New($Bytes)
        }
    }

\_____________________________________________________________________________________________/ Class [Pk3FileEntry] /---------------------------------------------------------------------------------------------\
  Class [Pk3FileArchive] /------------------------
/

    # // ============================================================
    # // | Template for each individual (*.pk3) file in the base path |
    # // ============================================================

    Class Pk3FileArchive
    {
        [UInt32]    $Index
        [String]     $Name
        [String] $Fullname
        [Object]  $Archive
        [Object]   $Output
        Pk3FileArchive([UInt32]$Index,[Object]$File)
        {
            $This.Index    = $Index
            $This.Name     = $File.Name
            $This.Fullname = $File.Fullname
            $This.Archive  = [System.IO.Compression.ZipFile]::Open($This.Fullname,"Read")
            $This.Refresh()
        }
        Clear()
        {
            $This.Output   = @( )
        }
        [Object] Pk3FileEntry([UInt32]$Index,[Object]$Entry)
        {
            Return [Pk3FileEntry]::New($Index,$Entry)
        }
        Refresh()
```

```
        {
            $This.Clear()

            ForEach ($Entry in $This.Archive.Entries | Sort-Object Fullname)
            {
                $This.Output += $This.Pk3FileEntry($This.Output.Count,$Entry)
            }
        }
    }

\_____________________/
\_____________________/ Class [Pk3FileArchive]
  Class [Q3AMapItem] /----------------------------------------------------------------------------------------------\
/---------------------

    # // ================================================================
    # // | Template for all (queued/selected) maps to include in the configuration |
    # // ================================================================

    Class Q3AMapItem
    {
        [UInt32]          $Index
        Hidden [DateTime]  $Real
        [String]           $Date
        [String]           $Time
        [String]           $Name
        [String]          $Title
        [Int32[]]          $Mode
        Hidden [String] $ModeStr
        [Float]          $Rating
        Q3AMapItem([UInt32]$Index,[Object]$Map)
        {
            $This.Index  = $Index
            $This.Real   = $Map.Real.ToString("MM/dd/yyyy HH:mm:ss")
            $This.Date   = $Map.Date
            $This.Time   = $Map.Time
            $This.Name   = $Map.Name

            $This.SetTitle("<Not set>")
            $This.SetMode(-1)
            $This.Rating = 0.00
        }
        SetTitle([String]$Title)
        {
            $This.Title  = $Title
        }
        SetMode([String]$Mode)
        {
            $This.Mode    = @(Invoke-Expression $Mode)
            $This.ModeStr = $This.Mode -join ","
        }
        SetRating([String]$Rating)
        {
            If ($Rating -match "\d+\.\d+")
            {
                $This.Rating = $Rating
            }
            If ($Rating -match "\d+\/\d+")
            {
                $This.Rating = Invoke-Expression $Rating
            }
        }
        [String] ToString()
        {
            Return "{0} {1} {2}" -f $This.Date, $This.Time, $This.Name
        }
    }

\_____________________/
\_____________________/ Class [Q3AMapItem]
  Class [Q3AMapConfig] /----------------------------------------------------------------------------------------------\
/----------------------

    # // ================================================================
    # // | Template for a newly created configuration, or an existing one |
    # // ================================================================

    Class Q3AMapConfig
    {
        [String]    $Name
        [String]    $Path
        [Object] $Content
        [Object]  $Output
```

```
Q3AMapConfig([String]$Name,[String]$Path)
{
    $This.Name    = $Name
    $This.Path    = $Path
    $This.Content = $Null
    $This.Clear()
}
Clear()
{
    $This.Output  = @( )
}
[Object] Q3AMapItem([UInt32]$Index,[Object]$Map)
{
    Return [Q3AMapItem]::New($Index,$Map)
}
[UInt32] GetRandom([UInt32]$Max)
{
    Return Get-Random -Maximum $Max
}
Randomize()
{
    $Total = $This.Output.Count
    $Out   = @( )
    ForEach ($X in 0..($Total-1))
    {
        Do
        {
            $Number = $This.GetRandom($Total)
        }
        Until ($Number -notin $Out)

        $Out += $Number
    }

    ForEach ($X in 0..($Total-1))
    {
        $This.Output[$X].Index = $Out[$X]
    }

    $This.Output = $This.Output | Sort-Object Index
}
Add([Object]$Map)
{
    $Item         = $This.Q3AMapItem($This.Output.Count,$Map)

    $This.Output += $Item
}
Remove([UInt32]$Index)
{
    If ($Index -gt $This.Output.Count)
    {
        Throw "Invalid index"
    }

    $This.Output = $This.Output | ? Index -ne $Index
    $This.Rerank()
}
Rerank()
{
    $X = 0
    ForEach ($Item in $This.Output)
    {
        $Item.Index = $X
        $X ++
    }
}
WriteConfig()
{
    # [Write Config]
    $Total        = $This.Output.Count
    $D            = ([String]$Total).Length

    $This.Content = @("set g_gametype 0;","set fraglimit 10;","set timelimit 0;")

    ForEach ($X in 0..($Total-1))
    {
        $Item     = $This.Output[$X]
        $Label    = "lvl{0}" -f $X
        $Next     = @("lvl{0}" -f ($X + 1);"lvl0")[$X -eq ($Total-1)]

        $Template = "echo $Label [Name]: {0}, [Rank]: ({1:d$D}/{2}), [Build]: {3} {4}"
        $Say      = $Template -f $Item.Name, ($X+1), $Total, $Item.Date, $Item.Time
```

```
            $This.Content += "seta $Label `"$Say;wait 500;map $($Item.Name); kick allbots; addbot hunter 5; set nextmap vstr $Next`""
        }

        $This.Content += "vstr lvl0"

        [System.IO.File]::WriteAllLines($This.Path,$This.Content)
    }
    ReadConfig()
    {
        $This.Content = [System.IO.File]::ReadAllLines($This.Path)
    }
    [String] ToString()
    {
        Return "<Q3A.Map.Config>"
    }
}
```

```
                                                                                               ______________________/
\_____________________________________________________________________________________________/ Class [Q3AMapConfig]
  Class [Q3AValidatePath] /-------------------------------------------------------------------------------------------\
/------------------------
```

```
    # // ========================================================================
    # // | Simply meant to validate a given path, and which object is referenced |
    # // ========================================================================

    Class Q3AValidatePath
    {
        [UInt32]   $Status
        [String]     $Type
        [String]     $Name
        [Object] $Fullname
        Q3AValidatePath([String]$Entry)
        {
            $This.Status       = [UInt32]($Entry -match "^\w+\:\\")
            $This.Fullname     = $Entry
            If ($This.Status -eq 1)
            {
                Try
                {
                    If ([System.IO.FileInfo]::new($Entry).Attributes -match "Directory")
                    {
                        $This.Type    = "Directory"
                    }
                    Else
                    {
                        $This.Type    = "File"
                    }

                    $This.Name       = Split-Path -Leaf $Entry

                    If (!(Test-Path $This.Fullname))
                    {
                        $This.Status = 2
                    }
                }
                Catch
                {

                }
            }
        }
        [String] ToString()
        {
            Return $This.Fullname
        }
    }
```

```
                                                                                            _________________________/
\__________________________________________________________________________________________/ Class [Q3AValidatePath]
  Class [Q3AControllerFlag] /-----------------------------------------------------------------------------------------\
/--------------------------
```

```
    # // ================================================
    # // | Only meant to categorize Xaml testing criteria |
    # // ================================================

    Class Q3AControllerFlag
    {
        [UInt32] $Index
        [String] $Name
```

```
        [UInt32] $Status
        Q3AControllerFlag([UInt32]$Index,[String]$Name)
        {
            $This.Index  = $Index
            $This.Name   = $Name
            $This.SetStatus(0)
        }
        SetStatus([UInt32]$Status)
        {
            $This.Status = $Status
        }
    }

  ______________________________________________________________________________________________/
\_____________________________________________________________________________________________/ Class [Q3AControllerFlag]
  Class [Q3AInputObject] /--------------------------------------------------------------------------------------------\
/-----------------------

    # // ==============================================================
    # // | Provides the option to import map information as an input object |
    # // ==============================================================

    Class Q3AInputObject
    {
        [UInt32]  $Index
        [String]   $Name
        [String]  $Title
        [Int32[]]  $Mode
        [String] $Rating
        Q3AInputObject([UInt32]$Index,[Object]$Entry)
        {
            $This.Index  = $Index
            $This.Name   = $Entry[0]
            $This.Title  = $Entry[1]
            $This.Mode   = $Entry[2]
            $This.Rating = $Entry[3]
        }
        [String] ToString()
        {
            Return "<Q3A.Input.Object>"
        }
    }

  ______________________________________________________________________________________________/
\_____________________________________________________________________________________________/ Class [Q3AInputObject]
  Class [Q3AController] /---------------------------------------------------------------------------------------------\
/----------------------

    # // ========================================================================
    # // | Controller of the entire utility and all of the above classes, acts as a factory |
    # // ========================================================================

    Class Q3AController
    {
        [Object]      $Module
        [Object]        $Xaml
        [Object]    $Property
        [Object]     $Archive
        [Object]       $Level
        [Object]      $Config
        Hidden [Object] $Flag
        Hidden [Object] $List
        Q3AController()
        {
            $This.Initialize()

            $This.List = @( )
        }
        Q3AController([Object[]]$List)
        {
            $This.Initialize()

            $This.List = @( )

            ForEach ($Item in $List)
            {
                $This.List += $This.Q3AInputObject($This.List.Count,$Item)
            }
        }
        Initialize()
        {
            $This.Module    = Get-FEModule -Mode 1
            $This.Module.Console.Reset()
            $This.Module.Console.Initialize()
            $This.Xaml      = [XamlWindow][Q3AControllerXaml]::Content
            $This.Property  = @( )
```

```
    $This.Flag      = @( )
    $This.Flag     += $This.Q3AControllerFlag($This.Flag.Count,"Game")
    $This.Flag     += $This.Q3AControllerFlag($This.Flag.Count,"ConfigName")
}
Update([Int32]$Status,[String]$Message)
{
    $This.Module.Update($Status,$Message)
    $Last = $This.Module.Console.Status
    If ($This.Module.Mode -ne 0)
    {
        [Console]::WriteLine($Last)
    }

    $This.Xaml.IO.Console.Items.Add($Last)
}
Main([String]$Game)
{
    # [Validate existence of game directory]
    If (![System.IO.Directory]::Exists($Game))
    {
        Throw "Invalid directory"
    }

    $This.Property += $This.Q3AProperty(   "Game","$Game")
    $This.Property += $This.Q3AProperty(   "Base","$Game\baseq3")
    $This.Property += $This.Q3AProperty( "Engine","$Game\quake3.exe")

    # [Validate quake3.exe hash value]
    $Engine         = $This.GetProperty("Engine")
    If ((Get-FileHash $Engine).Hash -ne $This.Q3AHash())
    {
        Throw "Invalid game engine"
    }

    # [Validate/create temporary directory]
    $This.Property += $This.Q3AProperty( "Temp","$Env:Temp\Q3A")

    $Temp           = $This.GetProperty("Temp")

    If (![System.IO.Directory]::Exists($Temp))
    {
        [System.IO.Directory]::CreateDirectory($Temp)
    }

    # [Populate the class with archives and maps]
    $This.Refresh()
}
[Object] Q3AProperty([String]$Name,[String]$Value)
{
    Return [Q3AProperty]::New($Name,$Value)
}
[Object] Pk3FileArchive([UInt32]$Index,[Object]$File)
{
    Return [Pk3FileArchive]::New($Index,$File)
}
[Object] Pk3FileBsp([UInt32]$Index,[Object]$Bsp)
{
    Return [Pk3FileBsp]::New($Index,$Bsp)
}
[Object] Q3AMapConfig([String]$Name,[String]$Path)
{
    Return [Q3AMapConfig]::New($Name,$Path)
}
[String] Q3AHash()
{
    Return "1DDF68B5B5314A39325A9362B1564D417A18B2B111BE7F8728CD808353829CC0"
}
[Object] Q3AValidatePath([String]$Entry)
{
    Return [Q3AValidatePath]::New($Entry)
}
[Object] Q3AControllerFlag([UInt32]$Index,[String]$Name)
{
    Return [Q3AControllerFlag]::New($Index,$Name)
}
[Object] Q3AInputObject([UInt32]$Index,[Object]$Entry)
{
    Return [Q3AInputObject]::New($Index,$Entry)
}
[String] IconStatus([UInt32]$Flag)
{
    Return $This.Module._Control(@("failure.png","success.png","warning.png")[$Flag]).Fullname
}
[String] GetProperty([String]$Name)
{
    Return $This.Property | ? Name -eq $Name | % Value
}
Clear()
{
```

```
    $This.Archive = @( )
    $This.Level   = @( )
    $This.Config  = @( )
}
Refresh()
{
    $This.Clear()
    $Base = $This.GetProperty("Base")
    $xList = Get-ChildItem $Base | ? Extension -eq .pk3 | ? Name -notmatch ^pak\d

    $This.Update(0,"Archive [~] ($($xList.Count)) files found")

    ForEach ($File in $xList)
    {
        $This.Update(0,"Archive [~] $($File.Name)")

        $This.Archive += $This.Pk3FileArchive($This.Archive.Count,$File)
    }

    $This.Update(1,"Archive [+] Complete")

    $This.ExtractBsp()
}
ExtractBsp()
{
    $This.Level = @( )

    $Magick = Get-ChildItem $Env:ProgramFiles | ? Name -match ImageMagick | % { "{0}\magick.exe" -f $_.Fullname }

    $Filter = $This.Archive.Archive.Entries | ? Fullname -match "(^maps/.+\.bsp$|^levelshots/.+\.(jpg|tga)$)"
    $Bsp    = $Filter | ? Fullname -match ^maps\/.+$      | Sort-Object Name | Select-Object -Unique
    $Image  = $Filter | ? Fullname -match ^levelshots/.+$ | Sort-Object Name | Select-Object -Unique
    $Temp   = $This.GetProperty("Temp")

    $This.Update(0,"Level [~] ($($Bsp.Count)) maps detected")

    ForEach ($Entry in $Bsp)
    {
        $Item   = $This.Pk3FileBsp($This.Level.Count,$Entry)
        $String = "\/{0}\." -f [Regex]::Escape($Item.Name)
        $Shot   = $Image | ? Fullname -imatch $String
        $Target = "{0}\{1}" -f $Temp, $Shot.Name

        $This.Update(0,"Level [~] Name: $($Item.Name)")

        If (![System.IO.File]::Exists($Target))
        {
            [System.IO.Compression.ZipFileExtensions]::ExtractToFile($Shot,$Target)

            Switch -Regex ($Target)
            {
                \.jpg$
                {
                    $Splat = @{

                        FilePath         = $Magick
                        ArgumentList     = "{0} -size 640x480" -f $Target
                        WorkingDirectory = Split-Path $Magick
                    }

                    Start-Process @Splat -Wait -WindowStyle Hidden
                }
                \.tga$
                {
                    $Source = $Target
                    $Target = $Target -Replace "tga","jpg"

                    If (![System.IO.File]::Exists($Target))
                    {
                        $Splat = @{

                            FilePath         = $Magick
                            ArgumentList     = "{0} -size 640x480 {1}" -f $Source, $Target
                            WorkingDirectory = Split-Path $Magick
                        }

                        Start-Process @Splat -Wait -WindowStyle Hidden

                        If ([System.IO.File]::Exists($Target))
                        {
                            [System.IO.File]::Delete($Source)
                        }
                    }
                }
            }
        }

        If ($Target -match "\.tga")
        {
```

```
            $Target = $Target -Replace "tga","jpg"
        }

        $Item.SetImage($Target)

        $This.Level += $Item
    }

    $This.Update(1,"Level [+] Complete")
}
StartProcess()
{
    If (!$This.Config)
    {
        Throw "Invalid configuration"
    }

    $Splat     = @{

        Filepath         = $This.GetProperty("Engine")
        ArgumentList     = "+exec {0}" -f $This.Config.Name
        WorkingDirectory = $This.GetProperty("Game")
    }

    Start-Process @Splat -NoNewWindow
}
NewConfig([String]$Name)
{
    If ($Name -match ".cfg$")
    {
        $Name = $Name -Replace "\.cfg", ""
    }

    $Base  = $This.GetProperty("Base")
    $xPath = "{0}\{1}.cfg" -f $Base, $Name

    $This.Config = $This.Q3AMapConfig($Name,$xPath)
}
Selection()
{
    $xList = $This.Level | ? Profile | Sort-Object Real

    $This.Update(0,"Configuring [~] ($($xList.Count)) map(s)")

    ForEach ($Map in $xList)
    {
        If ($Map.Name -notin $This.Config.Output)
        {
            $This.Config.Add($Map)

            $This.Update(1,"Config/Map [+] $($Map.Name)")
        }

        If ($Map.Name -in $This.List.Name)
        {
            $Item   = $This.Config.Output | ? Name -eq $Map.Name
            $Object = $This.List | ? Name -eq $Map.Name
            $Item.SetTitle($Object.Title)
            $Item.SetMode($Object.Mode)
            $Item.SetRating($Object.Rating)
        }
    }

    $This.Update(1,"Configured [+] ($($xList.Count)) map(s)")
}
Randomize()
{
    If (!$This.Config)
    {
        Throw "Invalid configuration"
    }

    $This.Config.Randomize()
}
WriteConfig()
{
    If (!$This.Config)
    {
        Throw "Invalid configuration"
    }

    $This.Config.WriteConfig()
}
ReadConfig()
{
    If (!$This.Config)
    {
        Throw "Invalid configuration"
    }
```

```
    $This.Config.ReadConfig()
}
Reset([Object]$xSender,[Object]$Object)
{
    $xSender.Items.Clear()
    ForEach ($Item in $Object)
    {
        $xSender.Items.Add($Item)
    }
}
FolderBrowse([String]$Name)
{
    $This.Update(0,"Browsing [~] Folder: [$Name]")

    $Object            = $This.Xaml.Get($Name)
    $Item              = [System.Windows.Forms.FolderBrowserDialog]::New()
    $Item.SelectedPath = [Environment]::GetFolderPath("ProgramFilesX86")
    $Item.ShowDialog()

    $Object.Text       = @("<Select a path>",$Item.SelectedPath)[!!$Item.SelectedPath]
}
CheckPath()
{
    $Item         = $This.Xaml.Get("Game")
    $Icon         = $This.Xaml.Get("GameIcon")
    $xFlag        = $This.Flag | ? Name -eq Game

    $xFlag.Status = $This.Q3AValidatePath($Item.Text).Status

    $Icon.Source  = $This.IconStatus($xFlag.Status)
}
CheckConfig()
{
    $Item         = $This.Xaml.Get("ConfigName")
    $Icon         = $This.Xaml.Get("ConfigNameIcon")
    $xFlag        = $This.Flag | ? Name -eq ConfigName
    $xText        = $Item.Text -Replace "\.cfg", ""
    $xList         = $This.Level | ? Profile

    If ($xText -eq "")
    {
        $xFlag.Status = 0
        $Icon.Source  = $This.IconStatus(0)
        $This.Xaml.IO.Create.IsEnabled = 0
    }
    Else
    {
        $xPath        = "{0}/{1}.cfg" -f $This.GetProperty("Base"), $xText
        $xFlag.Status = $This.Q3AValidatePath($xPath).Status
        $Icon.Source  = $This.IconStatus($xFlag.Status)
        $This.Xaml.IO.Create.IsEnabled = [UInt32]($xList.Count -gt 0)
    }
}
StageXaml()
{
    $Ctrl = $This

    $Ctrl.Xaml.IO.Browse.Add_Click(
    {
        $Ctrl.FolderBrowse("Game")
    })

    $Ctrl.Xaml.IO.Game.Add_TextChanged(
    {
        $Ctrl.CheckPath()
        $xFlag                     = $Ctrl.Flag | ? Name -eq Game
        $Ctrl.Xaml.IO.Set.IsEnabled = $xFlag.Status
    })

    $Ctrl.Xaml.IO.Game.Text = "${env:ProgramFiles(x86)}\Quake III Arena"

    $Ctrl.Xaml.IO.Set.Add_Click(
    {
        $Ctrl.Main($Ctrl.Xaml.IO.Game.Text)
        $Ctrl.Reset($Ctrl.Xaml.IO.Property,$Ctrl.Property)

        # [Archive]
        $Ctrl.Reset($Ctrl.Xaml.IO.Archive,$Ctrl.Archive)
        $Ctrl.Xaml.IO.ArchiveCount.Text = $Ctrl.Archive.Count

        # [Level]
        $Ctrl.Reset($Ctrl.Xaml.IO.Level,$Ctrl.Level)
        $Ctrl.Xaml.IO.LevelCount.Text   = $Ctrl.Level.Count
    })

    $Ctrl.Xaml.IO.Level.Add_SelectionChanged(
    {
        $Ctrl.Xaml.IO.Image.Source = $Ctrl.Xaml.IO.Level.SelectedItem.Image
```

```
    $Ctrl.CheckConfig()
})

$Ctrl.Xaml.IO.LevelName.Add_TextChanged(
{
    $Text   = $Ctrl.Xaml.IO.LevelName.Text
    Start-Sleep -Milliseconds 25

    $Result = $Ctrl.Level | ? Name -match ([Regex]::Escape($Text))
    $Ctrl.Reset($Ctrl.Xaml.IO.Level,$Result)
})

$Ctrl.Xaml.IO.ConfigName.Add_TextChanged(
{
    $Ctrl.CheckConfig()
})

$Ctrl.Xaml.IO.Create.Add_Click(
{
    $Ctrl.NewConfig($Ctrl.Xaml.IO.ConfigName.Text)
    $Ctrl.Selection()
    $Ctrl.Reset($Ctrl.Xaml.IO.Config,$Ctrl.Config.Output)
    $Ctrl.Xaml.IO.Launch.IsEnabled = 1
})

$Ctrl.Xaml.IO.Clear.Add_Click(
{
    $Ctrl.Level | % { $_.Profile = 0 }
    $Ctrl.Reset($Ctrl.Xaml.IO.Level,$Ctrl.Level)
})

$Ctrl.Xaml.IO.Config.Add_SelectionChanged(
{
    $Index                          = $Ctrl.Xaml.IO.Config.SelectedIndex
    If ($Index -gt -1)
    {
        $Item                           = $Ctrl.Xaml.IO.Config.SelectedItem

        $Ctrl.Xaml.IO.MapTitle.Text     = $Item.Title
        $Ctrl.Xaml.IO.MapTitle.IsEnabled  = 1

        $Ctrl.Xaml.IO.MapMode.Text      = $Item.ModeStr
        $Ctrl.Xaml.IO.MapMode.IsEnabled   = 1

        $Ctrl.Xaml.IO.MapRating.Text    = $Item.Rating
        $Ctrl.Xaml.IO.MapRating.IsEnabled = 1

        $Ctrl.Xaml.IO.Apply.IsEnabled   = 1
    }
    Else
    {
        $Ctrl.Xaml.IO.MapTitle.Text     = $Null
        $Ctrl.Xaml.IO.MapTitle.IsEnabled  = 0

        $Ctrl.Xaml.IO.MapMode.Text      = $Null
        $Ctrl.Xaml.IO.MapMode.IsEnabled   = 0

        $Ctrl.Xaml.IO.MapRating.Text    = $Null
        $Ctrl.Xaml.IO.MapRating.IsEnabled = 0

        $Ctrl.Xaml.IO.Apply.IsEnabled   = 0
    }
})

$Ctrl.Xaml.IO.Apply.Add_Click(
{
    $Item        = $Ctrl.Xaml.IO.Config.SelectedItem
    $Item.SetTitle($Ctrl.Xaml.IO.MapTitle.Text)
    $Item.SetMode($Ctrl.Xaml.IO.MapMode.Text)
    $Item.SetRating($Ctrl.Xaml.IO.MapRating.Text)

    $Ctrl.Reset($Ctrl.Xaml.IO.Config,$Ctrl.Config.Output)
})

$Ctrl.Xaml.IO.Randomize.Add_Click(
{
    $Ctrl.Randomize()
    $Ctrl.Reset($Ctrl.Xaml.IO.Config,$Ctrl.Config.Output)
})

$Ctrl.Xaml.IO.Export.Add_Click(
{
    $Ctrl.WriteConfig()
    $Ctrl.Xaml.IO.Content.Text = $Ctrl.Config.Content -join "`n"
})

$Ctrl.Xaml.IO.Import.Add_Click(
{
    $Ctrl.ReadConfig()
```

```
            $Ctrl.Xaml.IO.Content.Text = $Ctrl.Config.Content -join "`n"
        })

        $Ctrl.Xaml.IO.Launch.Add_Click(
        {
            $Ctrl.StartProcess()
        })
    }
    [String] ToString()
    {
        Return "<Q3A.Controller>"
    }
}
```

Class [Q3AController]

Breakdown

What is featured in the beginning of the video, is essentially the [code behind] the [graphical user interface].
I didn't cover the [code behind] all that much in the video, although I stated that I would, or that I wanted to.

Making a video [INTERESTING] is a bit of a [balancing act], and viewers will typically need a REASON to
(start/keep) watching. So, if they think [programming] is [boring], [lame], [not interesting], [sucks],
and they'd prefer to watch something else ... ?

Then, lingering on that aspect of the video for too long may cause them to miss out on the
[graphical user interface], and the [output] of the [utility] controlling the game for a majority of the video.

While I DID cover some of the [code behind] at the tail end of the video,
as well as some [basic code troubleshooting] ... it wasn't at length.

Here's where I will [break down] and [explain] what's going on in the code behind, as well as showing some of
the [output] from the [console]. The processes involved in getting the [code behind] to (work with/drive) the
[graphical user interface] requires [certain criteria] to be [executed] in proper [chronological order].

Teaching people how that [chronological order] has to be done, requires some [planning].

Take for instance, [Lego blocks].
When I was a kid, I played with [Lego blocks] all the time.
Even my son played with [Lego blocks].
I would imagine that even [General Kenneth J. McKenzie Jr.] has probably played with [Lego blocks], among
millions of other (adults + children).

There's a pretty simple answer as to WHY so many people have played with [Lego blocks], and they understand
them, but the same can be said for [so many other things], such as (programming + graphics).

In every [Lego] box, there's an [instruction manual] with [pictures] that show what [color] blocks you'll need,
as well as what [size], [width], and [shape] they are, and for [each step], the [correct pieces] need to be put
into the [correct place] ...

... otherwise the kid won't be able to [build] what's on the [front of the box].

In many cases, this is ok, [maybe they'll make something else]…

But- if you want to have an [intended result], you need to start with things in certain [chronological order],
and this video below is a [great start] to understanding what's going on in the [code behind].

Date	Name	Link	Duration
03/09/2021	A Deep Dive: PowerShell/XAML	https://youtu.be/NK4NuQrraCI	00:57:15

And, here is the [associated lesson plan] for [that video].

Date	Link
01/28/2021	https://github.com/mcc85sx/FightingEntropy/blob/master/Documentation/2021_0128-A_Deep_Dive.pdf

Breakdown

Output

Alright, so, we'll begin with the [script area].

```
# [If list ... ]
$List = ("20kdm2"         , "Return to Castle: Quake"          , "0,1"    ,  "8/10"),
("acid3dm9"       , "Chlorophyl"                       , "0,1,2"  ,  "8/10"),
("auh3dm1"        , "Overwhelming Hostility"           , "0,1"    ,  "8/10"),
("devdm3"         , "Infernal Genesis"                 , "0,1"    , "10/10"),
("dubenigma"      , "Lucid Enigma"                     , "0,1,2"  ,  "9/10"),
```

```
("fr3dm1"         , "Iron Yard"                        , "0,1,2" , "10/10"),
("geit3dm6"       , "Black Shining Leather"            , "0,1,2" ,  "8/10"),
("hub3aeroq3"     , "Aerowalk"                         , "1"     , "10/10"),
("hub3dm1"        , "Dismemberment"                    , "1"     , "10/10"),
("ik3dm2"         , "Meatball Grinder"                 , "0,1"   ,  "9/10"),
("jof3dm2"        , "The Forlorn Hope"                 , "0,1"   ,  "8/10"),
("kamq3dm2"       , "Discontent"                       , "0,1"   ,  "8/10"),
("lun3dm2"        , "Lets Drink Beer and Shoot Things" , "0,1"   , "10/10"),
("lun3_20b1"      , "Ludonarrative Dissonance"         , "0,1"   ,  "8/10"),
("mrcq3t4"        , "Prophecy"                         , "0,1"   ,  "8/10"),
("mrcq3t6"        , "Bitter Embrace"                   , "1"     , "10/10"),
("phantq3dm1_rev" , "Battleforged"                     , "0,1"   , "10/10"),
("pro-q3tourney7" , "Almost Lost"                      , "0,1,2" ,  "9/10"),
("pukka3tourney6" , "Kamasutra"                        , "0,1"   ,  "9/10"),
("qfraggel3ffa"   , "Swiss Cheese Trickster"           , "0,2"   ,  "8/10"),
("rdogdm4"        , "Epoch"                            , "0,1,2" ,  "8/10"),
("rota3dm3"       , "Marilyn"                          , "0,1,2" ,  "8/10"),
("rota3dm4"       , "Rashmi"                           , "0,2"   ,  "8/10"),
("storm3tourney1" , "Gloom"                            , "0,1"   ,  "8/10"),
("storm3tourney2" , "Nuclear Wasteland"                , "0,1"   ,  "9/10"),
("storm3tourney5" , "Cajun Hell"                       , "0,1,2" , "10/10"),
("storm3tourney8" , "Devilish"                         , "0,1,2" ,  "9/10"),
("tig_den"        , "Tigs Den"                         , "0,1"   , "10/10"),
("ts_dm4"         , "Perfect Place"                    , "0,1,2" ,  "9/10"),
("ts_t6"          , "Brains War"                       , "0,1,2" ,  "9/10"),
("tymo3dm5"       , "Stabilized Warfare: Resurrection" , "0,1,2" ,  "9/10"),
("unitooldm4"     , "Peccary of Destiny"               , "0,1,2" ,  "8/10"),
("unitooldm6"     , "Dumb All Over"                    , "0,1,2" ,  "9/10"),
("uzul3"          , "Uzuldaroum III"                   , "0,1,2" ,  "8/10"),
("wvwq3dm6"       , "Kryptonite"                       , "0,1"   ,  "8/10"),
("ztn3dm1"        , "Blood Run"                        , "0,1"   , "10/10"),
("ztn3dm2"        , "Beatbox"                          , "0,1"   , "10/10")

# [Load GUI]
$Ctrl      = Get-Q3AController -Mode 1 -List $List
```

So, this information up above stores a bunch of map names to the $List variable, and then it calls the function (Get-Q3AController) with -Mode set to (1) (*returns the object*), and -List set to variable $List.

What this does is it instantiates the [controller] class, as can be seen *below*.

```
PS Prompt:\> $Ctrl.GetType()

IsPublic IsSerial Name                                     BaseType
________ ________ ____                                     ________
True     False    Q3AController                            System.Object

PS Prompt:\> $Ctrl

Module   : <FightingEntropy.Module.Controller>
Xaml     : <FEModule.XamlWindow[Q3AControllerXaml]>
Property : {}
Archive  :
Level    :
Config   :

PS Prompt:\>
```

(1st) [PS Prompt:\>] calls the type, which is [Q3AController], the (controller/factory) class.
(2nd) [PS Prompt:\>] calls the [variable] $Ctrl and retrieves its' [output].

I will repeat this (1) → (2) [console entry] behavior, below.

We can see the [Module] property says <FightingEntropy.Module.Controller>, and the [Xaml] property says <FEModule.XamlWindow[Q3AControllerXaml]>… but the rest of the [properties] are $Null for the time being.

That's because it needs [additional input] from the (Window/GUI), in order to [populate] those [properties].

The GUI would check and validate this string "${env:ProgramFiles(x86)}\Quake III Arena" and then pass it to:

[Method] → $Ctrl.Main("${env:ProgramFiles(x86)}\Quake III Arena")

```
PS Prompt:\> $Ctrl.Main("${env:ProgramFiles(x86)}\Quake III Arena")
[00:00:03.1354683] (State: 0/Status: Archive [~] (99) files found)
...
[00:00:04.5470025] (State: 1/Status: Archive [+] Complete)
[00:00:06.3051263] (State: 0/Status: Level [~] (108) maps detected)
...
[00:00:07.1357696] (State: 1/Status: Level [+] Complete)
```

```
PS Prompt:\>
```

At this point, the [console] has logged a number of [items] and [information] ...

```
PS Prompt:\> $Ctrl

Module   : <FightingEntropy.Module.Controller>
Xaml     : <FEModule.XamlWindow[Q3AControllerXaml]>
Property : {Game, Base, Engine, Temp}
Archive  : {2023_0717-(testmap3).pk3, 20kdm3.pk3, addict.pk3, akutadm1.pk3 ... }
Level    : {07/17/2023 19:07:26 2023_0717-(testmap3), 03/08/2003 05:12:30 20kctf1 ... }
Config   : {}

PS Prompt:\>
```

Now, this information won't be THIS [specific] or [particular] to just anybody who runs this script, as the information that is showing is resultant to the number of files I have in MY [baseq3] directory.

There are some [prerequisites] in order, to be able to use this.

(1st), the module [FightingEntropy(π)] (... *needs* to be installed, in order to even get this far ...)
(2nd), [ImageMagick] (... to translate any (*.tga) files to (*.jpg) files that the [Xaml] engine can process…)

Here is the information behind the [Module] property…

```
PS Prompt:\> $Ctrl.Module.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     False    ModuleController                         System.Object

PS Prompt:\> $Ctrl.Module

Source      : https://www.github.com/mcc85s/FightingEntropy
Name        : [FightingEntropy(π)]
Description : Beginning the fight against ID theft and cybercrime
Author      : Michael C. Cook Sr.
Company     : Secure Digits Plus LLC
Copyright   : (c) 2023 (mcc85s/mcc85sx/sdp). All rights reserved.
Guid        : 4b564727-b84b-4033-a716-36d1c5e3e62d
Date        : 8/7/2023 8:52:08 PM
Version     : 2023.8.0
OS          : <FightingEntropy.Module.OS[Controller]>
Root        : <FightingEntropy.Module.Root[Controller]>
Manifest    : <FightingEntropy.Module.Manifest[Controller]>
Registry    : <FightingEntropy.Module.Registry[Key]>

PS Prompt:\>
```

And, here is the information behind the [Xaml] property…

```
PS Prompt:\> $Ctrl.Xaml

Names     : {Border, Game, GameIcon, Property ... }
Types     : {Game, GameIcon, Property, Browse ... }
Node      : System.Xml.XmlNodeReader
IO        : System.Windows.Window
Exception :

PS Prompt:\>
```

Now, here's what's behind the [Property] property…

```
PS Prompt:\> $Ctrl.Property

Name   Value
----   -----
Game   C:\Program Files (x86)\Quake III Arena
Base   C:\Program Files (x86)\Quake III Arena\baseq3
Engine C:\Program Files (x86)\Quake III Arena\quake3.exe
Temp   C:\Users\mcadmin\AppData\Local\Temp\Q3A

PS Prompt:\>
```

Since that returned an array of [Object[]], we have to select (1) item in the array, to get its' type.
We'll select the (1st) object in the array, via $Ctrl.Property[0]…

```
PS Prompt:\> $Ctrl.Property[0].GetType()

IsPublic IsSerial Name                                     BaseType
──────── ──────── ────                                     ────────
True     False    Q3AProperty                              System.Object

PS Prompt:\> $Ctrl.Property[0]

Name Value
──── ─────
Game C:\Program Files (x86)\Quake III Arena

PS Prompt:\>
```

Now here's what's behind the [Archive] property, which as it was in the case of [Property] ...
... is an array of [Object[]], and this trend continues for the properties [Archive] and [Level].

We'll select the (1st) object in the array, via $Ctrl.Archive[0]…

```
PS Prompt:\> $Ctrl.Archive[0].GetType()

IsPublic IsSerial Name                                     BaseType
──────── ──────── ────                                     ────────
True     False    Pk3FileArchive                           System.Object

PS Prompt:\> $Ctrl.Archive[0]

Index    : 0
Name     : 2023_0717-(testmap3).pk3
Fullname : C:\Program Files (x86)\Quake III Arena\baseq3\2023_0717-(testmap3).pk3
Archive  : System.IO.Compression.ZipArchive
Output   : {, 2023_0717-(testmap3).jpg, , 2023_0717-(testmap3).aas ... }

PS Prompt:\>
```

Now here's what's behind the [Level] property.

We'll select the (1st) object in the array via $Ctrl.Level[0]…

```
PS Prompt:\> $Ctrl.Level[0].GetType()

IsPublic IsSerial Name                                     BaseType
──────── ──────── ────                                     ────────
True     False    Pk3FileBsp                               System.Object

PS Prompt:\> $Ctrl.Level[0]

Index   : 0
Date    : 07/17/2023
Time    : 19:07:26
Name    : 2023_0717-(testmap3)
Profile : 0
Image   : C:\Users\mcadmin\AppData\Local\Temp\Q3A\2023_0717-(testmap3).jpg

PS Prompt:\>
```

Now, to explain what all of that code did up above by stripping the code to its' basic elements.

```
$xList = Get-ChildItem $Base | ? Extension -eq .pk3 | ? Name -notmatch ^pak\d

ForEach ($File in $xList)
{
    $This.Archive += $This.Pk3FileArchive($This.Archive.Count,$File)
}
```

The controller class accepts the path to [Quake III Arena], and it looks for specific files that would typically belong to a healthy installation of the game.

That variable $Base is actually one of the properties, particularly the one named "Base" for /baseq3

$xList looks through that path using (Get-ChildItem) for any file with the extension (*.pk3) that does not have a name that matches ^pak\d, which is a [regular expression] which filters the default [id Software] pak files.

Every entry that comes back from this command is fed through to the class [Pk3FileArchive], which filters certain properties, while also retaining the actual underlying class type, [System.IO.Compression.ZipArchive], which allows access to its' stored [System.IO.Compression.ZipArchiveEntry] entries for [file extraction].

```
$Magick = Get-ChildItem $Env:ProgramFiles | ? Name -match ImageMagick | % { "{0}\magick.exe" -f $_.Fullname }
```

```
$Filter = $This.Archive.Archive.Entries | ? Fullname -match "(^maps/.+\.bsp$|^levelshots/.+\.(jpg|tga)$)"
$Bsp    = $Filter | ? Fullname -match ^maps\/.+$       | Sort-Object Name | Select-Object -Unique
$Image  = $Filter | ? Fullname -match ^levelshots/.+$ | Sort-Object Name | Select-Object -Unique
$Temp   = $This.GetProperty("Temp")

ForEach ($Entry in $Bsp)
{
    $Item   = $This.Pk3FileBsp($This.Level.Count,$Entry)
    $String = "\/{0}\." -f [Regex]::Escape($Item.Name)
    $Shot   = $Image | ? Fullname -imatch $String
    $Target = "{0}\{1}" -f $Temp, $Shot.Name

    If (![System.IO.File]::Exists($Target))
    {
        [System.IO.Compression.ZipFileExtensions]::ExtractToFile($Shot,$Target)

        Switch -Regex ($Target)
        {
            \.jpg$
            {
                $Splat = @{

                    FilePath         = $Magick
                    ArgumentList     = "{0} -size 640x480" -f $Target
                    WorkingDirectory = Split-Path $Magick
                }

                Start-Process @Splat -Wait -WindowStyle Hidden
            }
            \.tga$
            {
                $Source = $Target
                $Target = $Target -Replace "tga","jpg"

                If (![System.IO.File]::Exists($Target))
                {
                    $Splat = @{

                        FilePath         = $Magick
                        ArgumentList     = "{0} -size 640x480 {1}" -f $Source, $Target
                        WorkingDirectory = Split-Path $Magick
                    }

                    Start-Process @Splat -Wait -WindowS
                    If ([System.IO.File]::Exists($Target))
                    {
                        [System.IO.File]::Delete($Source)
                    }
                }
            }
        }
    }

    If ($Target -match "\.tga")
    {
        $Target = $Target -Replace "tga","jpg"
    }

    $Item.SetImage($Target)

    $This.Level += $Item
}
```

Here's a crash course in what's happening there.

$Magick looks for the executable for [ImageMagick], to use in the command line.
$Filter looks through all of the archive entries for a fullname that matches (maps + levelshots)
$Bsp filters out (*.bsp) entries that are unique.
$Image does the same thing, but for (*.jpg/*.tga) entries.

$Temp is the temporary location for the image files to be extracted to.

Then, for each item that is in variable $Bsp, it's going to create a new [Pk3FileBsp] object, trim the file name to its' base name, and then filter out the corresponding levelshot from $Image based on the base name.

If the file matches (*.jpg), it extracts the image, saves it to the temp directory, and assigns the path to the [Pk3FileBsp] object.

If the file matches (*.tga), it extracts the image, launches [ImageMagick] and converts it to a (*.jpg), removes the (*.tga) file, and then assigns the path to the new (*.jpg) to the [Pk3FileBsp] object.

Now, at this point, the [DataGrid] populates with the entries that were saved to $Ctrl.Level.
The [event handlers] in the [controller class] will respond to any item that is selected,
and show its' image in the [GUI].

However, if you're just using the command line, all of that work is [relatively pointless].

Either way, in order to make a map selection, you have to do so manually in the [GUI], or…

```
ForEach ($Item in $Ctrl.List)
{
    $Map = $Ctrl.Level | ? Name -eq $Item.Name
    If ($Map)
    {
        $Map.Profile = 1
    }
}
```

What this does, is it looks for [each item] in the $Ctrl.List property, and checks $Ctrl.Level for a [name] that is [literally equal] to the [item]'s. This isn't using a -match operand here, so if it is NOT equal, it does NOT add that map to the [config queue].

Now, I intentionally left this out because doing this sort of defeats the purpose of the utility, but obviously this would allow the initial $List to be used to select and import into the [config queue].

However, BEFORE adding things to the [config queue], the [config] needs a name and then it needs to be passed through to:

[Method] → $Ctrl.NewConfig("basenameofconfigfile")

```
$Name = "basenameofconfigfile"

If ($Name -match ".cfg$")
{
    $Name = $Name -Replace "\.cfg", ""
}

$Base  = $This.GetProperty("Base")
$xPath = "{0}\{1}.cfg" -f $Base, $Name

$This.Config = $This.Q3AMapConfig($Name,$xPath)
```

So, the [GUI] would validate whether the projected file [exists], in order to figure out if it needs to throw an error, or to [overwrite the target file]. It's important to note that all of this occurs in [memory], so it doesn't actually [overwrite the file] until the button [Export] is clicked on in the [GUI], or the method $Ctrl.WriteConfig() is activated.

Still, neither one of those will work if the [Config] object hasn't been made first, which requires a [name].

```
PS Prompt:\> $Ctrl.NewConfig("lvlworld5")
PS Prompt:\> $Ctrl.Config.GetType()

IsPublic IsSerial Name                                     BaseType
──────── ──────── ────                                     ────────
True     False    Q3AMapConfig                             System.Object

PS Prompt:\> $Ctrl.Config

Name      Path                                                        Content Output
────      ────                                                        ─────── ──────
lvlworld5 C:\Program Files (x86)\Quake III Arena\baseq3\lvlworld5.cfg         {}

PS Prompt:\>
```

Now, in order to import the selected maps to the [config] object, we can call the method:

[Method] → $Ctrl.Selection()

```
$xList = $This.Level | ? Profile | Sort-Object Real

ForEach ($Map in $xList)
{
    If ($Map.Name -notin $This.Config.Output)
    {
        $This.Config.Add($Map)
    }

    If ($Map.Name -in $This.List.Name)
    {
        $Item   = $This.Config.Output | ? Name -eq $Map.Name
        $Object = $This.List | ? Name -eq $Map.Name
```

```
        $Item.SetTitle($Object.Title)
        $Item.SetMode($Object.Mode)
        $Item.SetRating($Object.Rating)
    }
}
```

So, here's another crash course.

$xList looks for anything in $Ctrl.Level that has the property [Profile] set to (1).

Then, for each $Map in $xList, the loop checks to see if the map is in the [config output] already.
If it isn't, it adds it.

Also, during the same loop, if the current $Map has a name that is in $Ctrl.List…?
The loop enters a [second area] where it injects all of the information such as [title], [mode], and [rating].

```
PS Prompt:\> $Ctrl.Selection()
[00:00:14.1354784] (State: 0/Status: Configuring [~] (37) map(s))
 ...
[00:00:14.3374804] (State: 1/Status: Configured [+] (37) map(s))
PS Prompt:\>
```

Finally, at this point, you can either press the button [Export] if you're using the [GUI], or you can use the method $Ctrl.WriteConfig().

Conclusion / Output

Once you've done all of this, you can hit the launch button to get [Quake III Arena] to launch the custom [map configuration]. Please consider that this is not quite complete, as I've considered adding [other options] and such. But- I felt that this was worthy of a [video], and a [lesson plan]…

Michael C. Cook Sr.
Security Engineer
Secure Digits Plus LLC