Science & Technology Facilities Council

Technical Report

# Auto-AG: Enhancing the User Experience for Access Grid Meetings

Adam Horwich

RAL-TR-2010-015

© Science and Technology Facilities Council

Enquires about copyright, reproduction and requests for additional copies of this report should be addressed to:

Library and Information Services
SFTC Rutherford Appleton Laboratory
Harwell Science and Innovation Campus
Didcot
OX11 0QX
UK
Tel: +44 (0)1235 445384
Fax: +44(0)1235 446403
Email: library@rl.ac.uk

The STFC ePublication archive (epubs), recording the scientific output of the Chilbolton, Daresbury, and Rutherford Appleton Laboratories is available online at:
http://epubs.cclrc.ac.uk/

Neither the Council nor the Laboratory accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigation

# Auto-AG: Enhancing the User Experience for Access Grid Meetings


## Abstract

Access Grid is a multi-site, group-to-group conferencing tool utilised in large scale collaboration projects such as the UK National Grid Service (NGS) and allows the Scientific Computing Technology Group (SCT), which is split between a Rutherford and Daresbury Campus, to communicate cost-free. This report intends to outline some of the key developments which have enabled Auto-AG to become an essential tool for setting up Access Grid meetings within the STFC. This tool is used in all three main Access Grid facilities in the e-Science department and caters for each unique room setup. By providing a rich feature set, encompassing many features found within the Venue Client, Auto-AG ensures meetings commence with little to no interruption, allowing both experienced operators and new users alike to set up meetings. Auto-AG simplifies the set-up process by implementing easy to use 'right click' positioning systems to allow operators to automatically place video feeds, it also automate processes for turning on and off conference room Audio and Video equipment. In order to provide an accessible interface, Auto-AG also de-clutters the user environment and consolidates key Venue Client features. Auto-AG also introduces a 'tutorial mode', guiding users through the process of setting up meetings.

# Table of Contents

# 1. Introduction

The Access Grid [1] is an ensemble of resources including multimedia large-format displays, presentation and interactive environments, and interfaces to Grid middleware and to visualization environments. These resources are used to support group-to-group interactions across the Grid. It started as a research project at the Argonne National Laboratory (ANL) in Chicago. Access Grid is designed to be a scalable technology, supporting operation on laptops as a Personal Instance of Access Grid (PIG) all the way up to full scale conference rooms with three projectors, four cameras and an array of microphones. The software, originally released as an Open Source project, is called Access Grid Toolkit (AGTk) and has been complimented by a commercial version which offers a fully integrated hardware and software solution, produced by IOCOM.

In the UK, Manchester University host the Access Grid Support Centre (AGSC) [2] which is the UK hub for Access Grid, offering services and support to UK Academia. To date there are 255 registered Access Grid 'Nodes' with the AGSC. The Scientific Computing Technology (SCT) group within the Science and Technology Facilities Council (STFC) manage 3 instances. Worldwide there is estimated to be an excess of 500 Access Grid Nodes [3]. Figure 1.1 below shows installations on remote islands in the Pacific as well as nodes in South America and China.

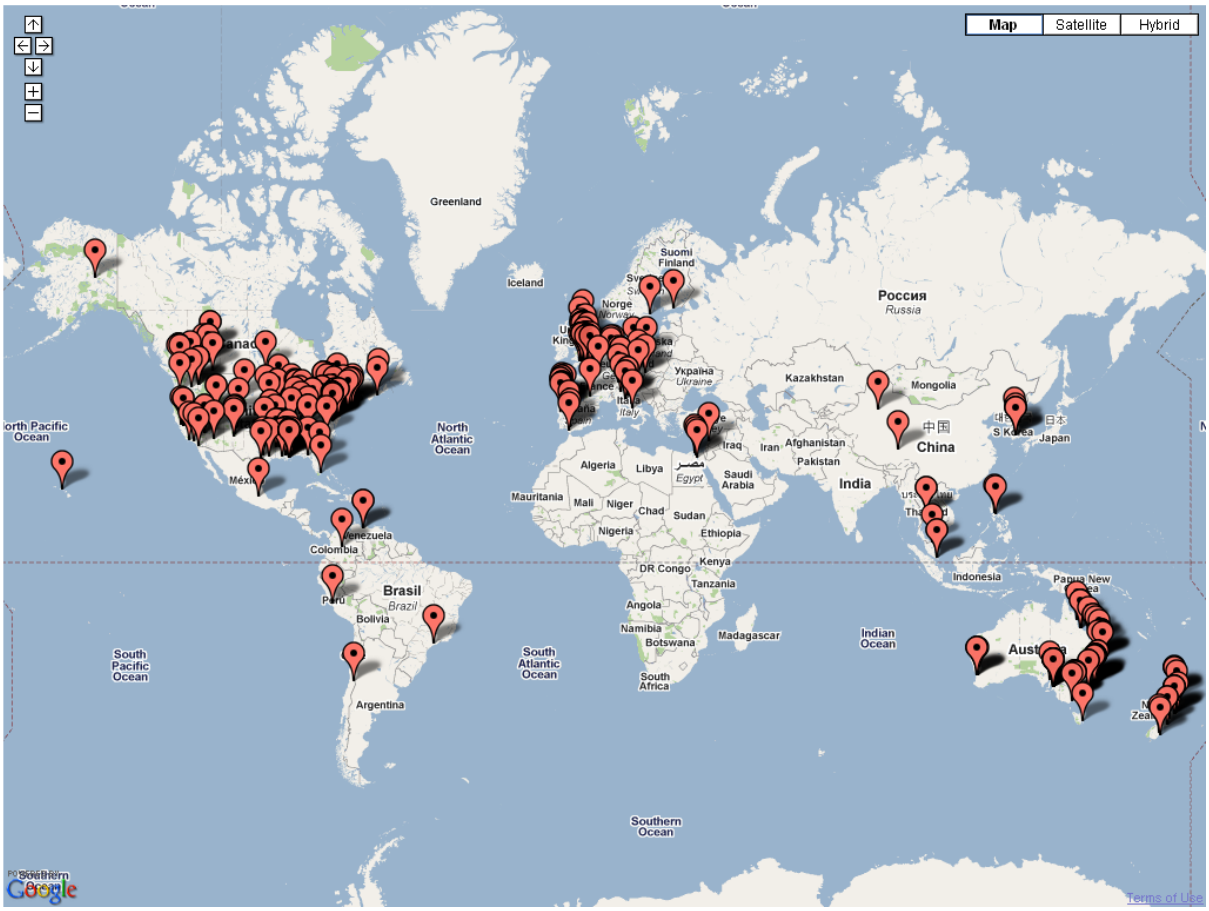

Figure 1.1

The objective of an Access Grid is to provide a 'conference like' experience to videoconferencing, offering multiple cameras and microphones to capture a large array of people and events occurring in a conference room, in high quality. With such capability, comes a wide range of configuration options and actions necessary to be performed when setting up a meeting. To meet these challenges, and offer an experience which any user can interact with, SCT developed an Application called Auto-AG to simplify and automate the process. This report aims to discuss the technical challenges of providing

such a tool and how it has aided the SCT Access Grid service [4]. Further details of Access Grid can be found at the AGSC [5].

## 2. Access Grid Toolkit

The Access Grid Toolkit is an ensemble of tools, (Videoconferencing Client) VIC, (Robust Audio Tool) RAT, and the Venue Client (VC). VIC and RAT are based on legacy tools, currently being maintained and developed by the University College London [6]. The Venue Client and associated software was developed in Python and attempts to be as platform neutral as possible; although each major operating system has its own distribution.

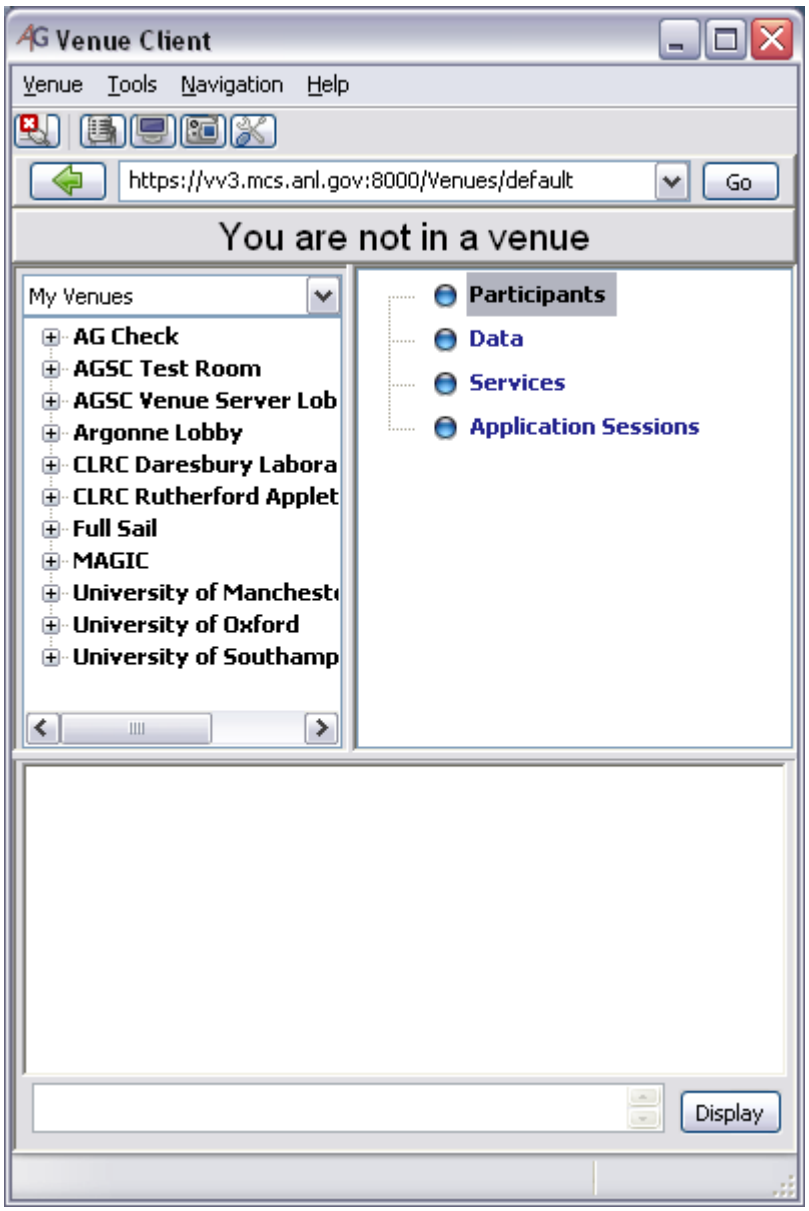


**Figure 1.2**

From the figure above we can see the standard layout of the Access Grid Toolkit. There are four main zones; the first running along the top with a series of icons used for enabling, disabling and configuring video and audio tools, the second a series of expandable trees on the left containing links to all the Access Grid Virtual Venues, third on the right detailing the attendees of a meeting and services available therein, and finally the lower window represents a chat client which becomes available

during meetings. The problem with this configuration is that it is not intuitive to newcomers and can still be cumbersome to those experienced with the software. Understanding where content is and how to configure the service is unfriendly and time consuming. Figure 1.3 below shows a typical display when connecting to a virtual venue from the Access Grid Toolkit.

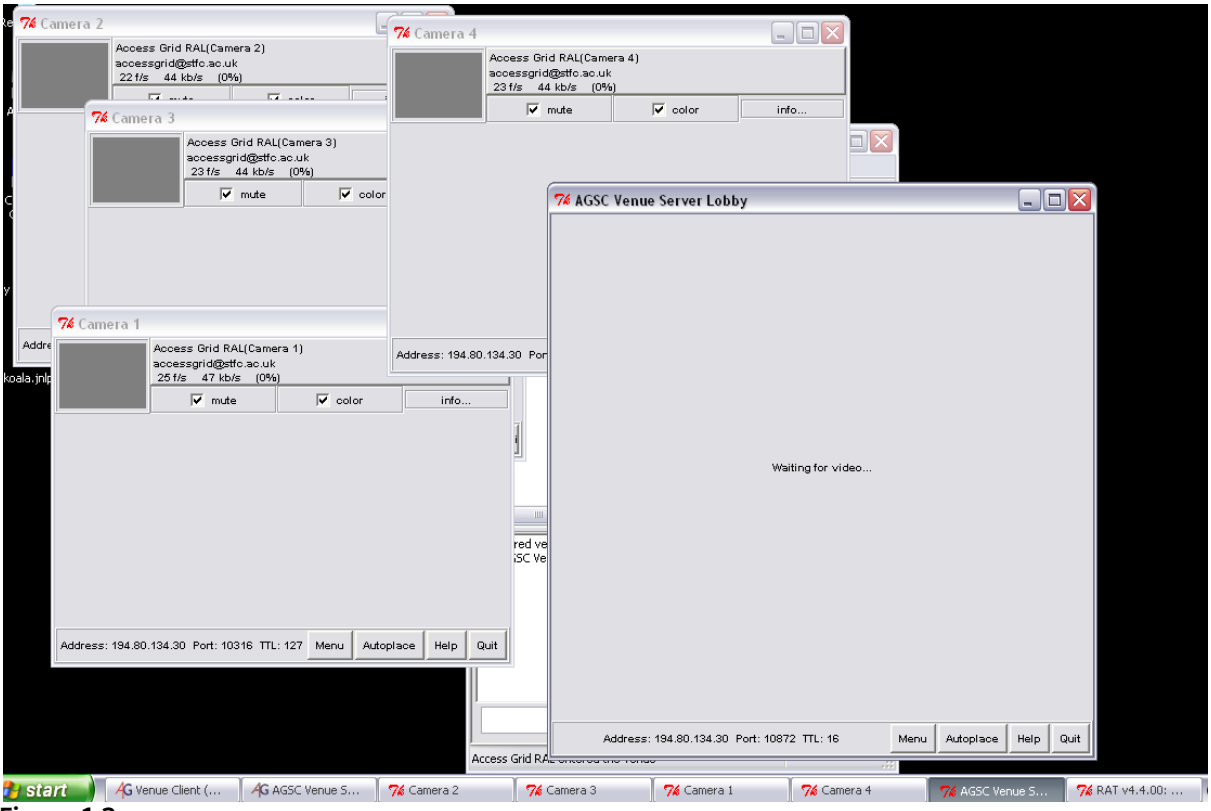


**Figure 1.3**

Here we can see the AGTk Venue Client in the background and six new windows which have appeared after connecting to a Virtual Venue. For each camera that is used in a conference room, a separate 'Video Producer' window will be created, as will an 'Audio Producer' to manage the microphones and speakers, and a 'Video Consumer' for receiving video. Manipulating these windows in order to expose video feeds and display them for an audience is a delicate task. Often, a great deal of time can be wasted at the beginning of a meeting while the Access Grid 'Operator' performs their duties and sets up video feeds, the reason for this is that video feeds by default are 'muted' and microphones are disabled. In order to set up a meeting, each camera feed window must have its 'mute' box unchecked, then the mini-preview feed must be left clicked to pop out a full sized feed, and finally that video window must be positioned onto a display screen for participants to see. This must be done for all cameras and incoming video feeds from remote sites.

# 3. Requirements

In order to best understand how do develop the application, opinions and requests were gathered from SCT Access Grid operators and members of staff who may seek to set up meetings unaided. The primary requirements were as follows:

1. Simple, clean interface - bright colours distinguishing features and workflows
2. Automation for turning on projectors and positioning cameras
3. Automation for shutting down projectors
4. Full independent camera control
5. De-clutter AGTk windows
6. Assisted window positioning
7. Automatic window positioning
8. Audio Tool un-muting (inc control for operator microphone)
9. Teleconference dial-out
10. Virtual Venue Selection
11. Tutorial mode

# 4. Auto-AG

The software tool developed by SCT began life in 2006 to simply turn on projectors, launch the Access Grid Toolkit, and position cameras; addressing the first four requirements. This was achieved through the use of RS232 Serial command interfaces commonly available on projectors and Pan-Tilt-Zoom (PTZ) cameras. While this ensured some aspects of meetings could be automated, it did not address the wider issue of setting up and displaying video feeds. In 2008, additional development effort commenced on Auto-AG to incorporate additional features from the AGTk, and to provide a mechanism to automate video window positioning, thus enabling faster setup times for operators. Figure 1.4 shows the layout used for Auto-AG when configured for the Rutherford Appleton Laboratory (RAL) R89 Access Grid conference room (the default view is shown in Figure 1.5). The principle behind Auto-AG is for it to sit in front of AGTk, driving it through a simplified interface; it does not intend replace it. In its current inception, Auto-AG is inherently tied to operation under Microsoft Windows XP, by virtue of being coded with VB.net and utilising Windows XP centric libraries. It is also tied to the latest stable version of the Access Grid Toolkit (currently 3.1); this is because of subtle changes that are made over time, and backwards compatibility proves problematic, so development effort was best spent ensuring compatibility with the latest version.

## Overview of Main Features

To address the requirements outlined in the previous section, a series of features were developed, detailed in depth in the following sections.

- To rapidly prototype an easy to use interface, VB.net was chosen as the main programming language thanks to Microsoft Visual Studio's Integrated Development Environment. This allowed for the designs shown in Figures 1.4 and 1.5 to be made and modified very easily.
- To address the screen cluttering of AGTk windows, manipulation through Windows XP specific libraries that can be interfaced by VB.net were employed, allowing the simple click of a button on the interface to ensure all windows became visible.
- After enabling the 'right click' positioning system, a simple left then right click on any preview feed or a right click on any open video feed would move it to a pre-defined location based on the number of feeds currently being displayed. This feature also takes account of the number of screens in an Access Grid display wall, and the direction that they all extend to from the primary display.

- Extending the principles of right click positioning allowed the creation of a new button which would place all windows and feeds automatically for the user, devolving all set up effort into a single button. In order to overcome technical limitations in the AGTk applications, control of the mouse needed to be taken away from the user to enable this feature.
- More AGTk manipulation came through the ability to toggle the microphones on and off via Auto-AG, instead of interfacing with the main applications. This was programmatically different to the Venue Client and VIC application interaction.
- Following on from development efforts in controlling cameras and projectors through serial interfaces, Clear One/Gentner audio conferencing hardware support was added into Auto-AG allowing teleconferencing dial-out features and advanced microphone isolation.
- Being able to parse files AGTk uses for configuration and logging allowed Auto-AG to display a loading progress bar and also host a copy of the Virtual Venues bookmarked. This meant that operators could connect to a Virtual Venue without needing to interface with the Venue Client.
- To accommodate entirely new users to Access Grid and Auto-AG, a tutorial mode was developed which posed a guided explanation of how to set up a meeting, specific to the user's needs and remove the opportunity to configure anything else.

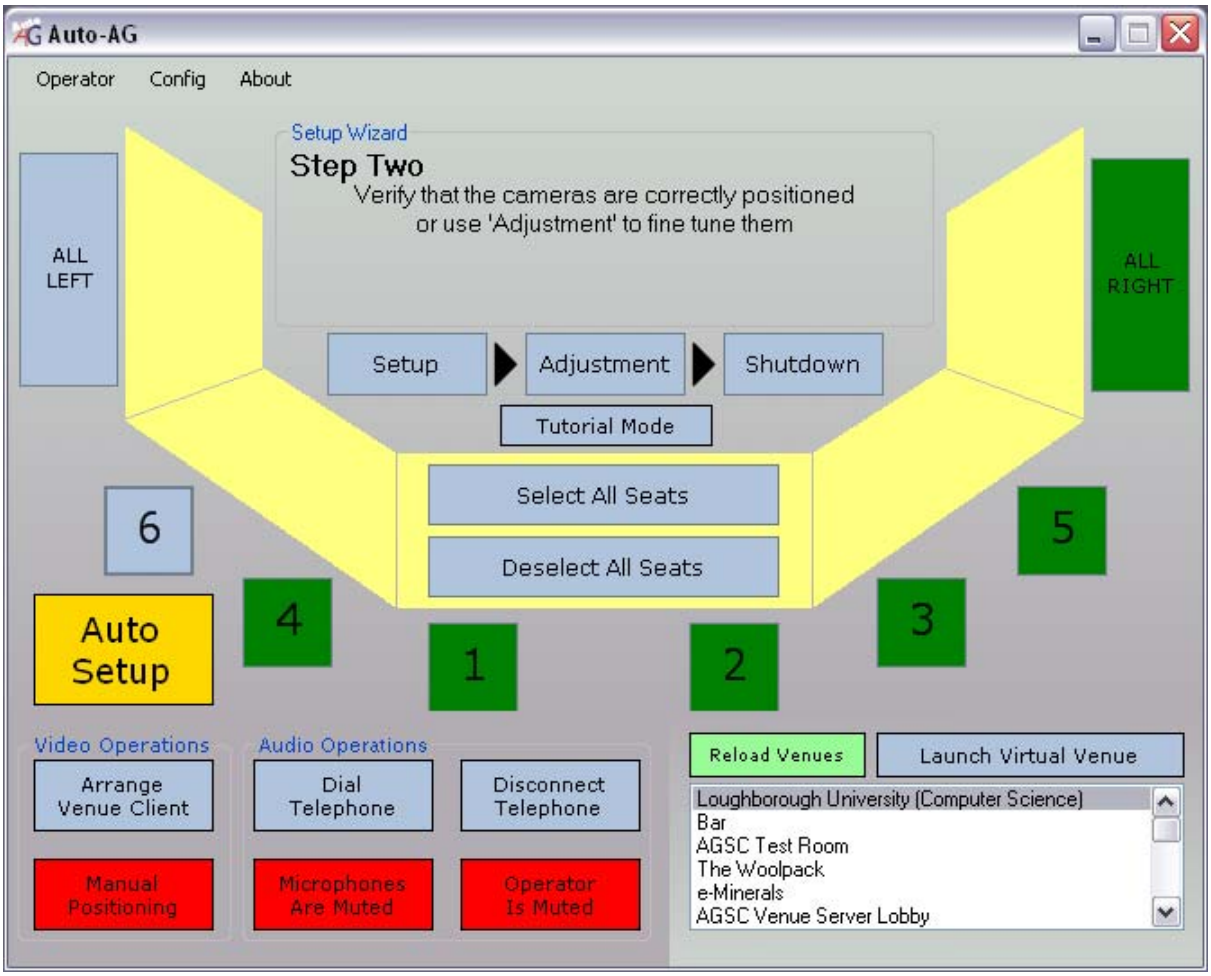


**Figure 1.4**

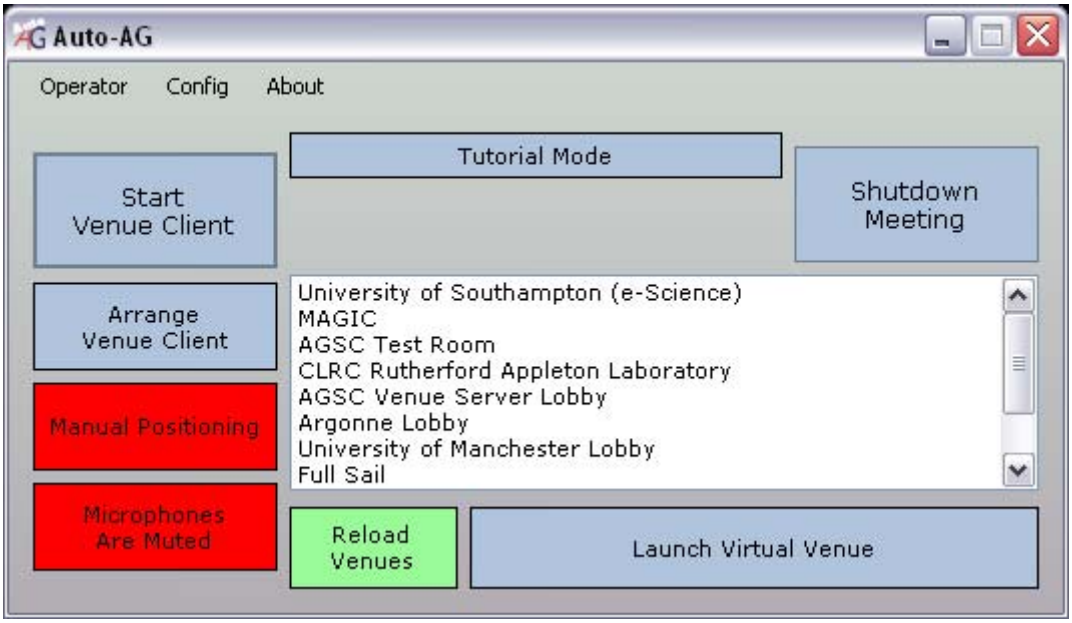


**Figure 1.5**

## De-cluttering AGTk Windows

One of the earliest features to be incorporated into Auto-AG was the ability to position the various windows, which appear when connected to a Virtual Venue, into a sensible order whereby all can be viewed and accessed. A single click on the 'Arrange Venue Client' button will vertically align Video Producer windows along the primary screen. A second click, will determine which VIC window houses the remote feeds, make it larger, and increases the column count so that each incoming video feed can be more easily accessed. Due to a quirk of using Unicast bridging, sometimes incoming video feeds will not appear in the Video Consumer Window (VCW), so in the event of this, Auto-AG will close that window when 'Arrange Venue Client' is activated.

This feature works by utilizing the EnumWindows and FindWindows functions of the user32 library available to VB.net [7]. EnumWindows can generate a list of all the running windows on a system, and by supplying some simple criteria to FindWindows, the window we are looking for can be procedurally determined. Once we know its reference name, it can be manipulated with the SetWindowPos function.

As RAT is always present in a meeting, it is the first window to be moved, using the following command:

```
SetWindowPos(FindWindow("TkTopLevel", "RAT v4.4.01: " + venueName), 0, 0, 0, 352, 321, _
                        SWP_NOZORDER Or SWP_NOOWNERZORDER Or SWP_NOACTIVATE)
```

As RAT is coded in Tk/TCL, it will always have a window name of "TkTopLevel" and it will always consist of the text "RAT v.4.4.01: " followed by the name of the Access Grid Virtual Venue. Knowing these two pieces of information can track down the window reference number with FindWindow, that can then be passed onto SetWindowPos to move the window to the co-ordinates 0,0.

The next stage of the process is to determine whether the VCW is open (this is because the Arrange Venue Client button is designed to be pressed multiple times and based on certain conditions, the VCW may not be present).

```
Dim clientCheck, hWndVCW As IntPtr
Dim videoConsumerWindow As Boolean = False
Dim vcwFeeds As Boolean = False
Dim vicHeight As Integer = 0
Dim posCheck As Integer = 0
Dim winIDs As New ArrayList()
Dim proc As New EnumChildProcDelegate(AddressOf Me.EnumChildProc)
Dim chPos, arrangePos As Rect
Dim vcwName As New System.Text.StringBuilder("", 256)
clientCheck = FindWindow("TkTopLevel", "VIC Members")
Dim arrangeProc As New EnumWindProcDelegate(AddressOf Me.EnumWindowsProc)
EnumWindows(arrangeProc, 0)
Try
        For m As Integer = 0 To winArray.Count - 1
                GetClientRect(winArray(m), arrangePos)
                If (Not (arrangePos.Right - arrangePos.Left) = 352) Then
                        GetWindowText(winArray(m), vcwName, 256)
                        If vcwName.ToString.StartsWith(venueName) Then
                                hWndVCW = winArray(m)
                                videoConsumerWindow = True
                        Exit Try
                        Else
                                hWndVCW = 0
                        End If
                End If
        Next
Catch ex As Exception
        Debug.WriteLine("No Windows open")
End Try

If hWndVCW <> 0 Then
        labels.Add(venueName)
End If

For n As Integer = 0 To labels.Count - 1
        If labels(n).ToString.StartsWith(venueName) Then
                ShowWindow(hWndVCW, SW_RESTORE)
        Else
                winIDs.Add(FindWindow("TkTopLevel", labels(n)))
                ShowWindow(FindWindow("TkTopLevel", labels(n)), SW_RESTORE)
        End If
Next
If hWndVCW <> 0 Then
        labels.Remove(venueName)
End If
```

This code operates in several parts, the first the programme checks whether the VIC windows have launched by finding a window named “VIC Members”, without this, the program will not consider moving any windows. Next, the open windows which are currently being displayed will be scanned to determine which are Video Producer Windows (VPW) (the loop will skip over anything with a width of 352 as this is a video feed window). By using the GetWindowText function, the programme does a text comparison to find the VCW; if this is located, the window reference is recorded; else it is set to zero. The next step is to collect all of the reference numbers for the VIC windows and ensure they are not minimised.

Once this is complete, the process then moves on to determine if any of the video feed windows have more than one feed in them, as if this is the case then we have remote participants attending and the window containing those remote feeds must be allocated extra desktop space. Similar to the way we enumerate the running windows on a system, EnumChildWindows is employed to find all the sub windows of a running process. This is essential so we can find the mini-preview feeds that are displayed in a VIC window and count them.

```
If videoConsumerWindow Then
	EnumChildWindows(hWndVCW, proc, 0)
	For vcw As Integer = 0 To chArray.Count - 1
		GetWindowRect(chArray(vcw), chPos)
		If ((chPos.Right - chPos.Left) = 80 And (chPos.Bottom - chPos.Top) = 60) Then
			vcwFeeds = True
			Exit For
		End If
	Next
End If
```

If the VCW is present, it is scanned for mini-preview feeds and a Boolean is recorded.

```
Dim feednum(labels.Count) As Integer
Dim totalfeed As Integer = 0
For i As Integer = 0 To winIDs.Count - 1
	chArray.Clear()
	EnumChildWindows(winIDs(i), proc, 0)
	For j As Integer = 0 To chArray.Count - 1
		GetWindowRect(chArray(j), chPos)
		If ((chPos.Right - chPos.Left) = 80 And (chPos.Bottom - chPos.Top) = 60) Then
			feednum(i) += 1
			totalfeed += 1
		End If
	Next
Next

If videoConsumerWindow And totalfeed > labels.Count Then
	SendMessage(hWndVCW, WM_CLOSE, 0, 0)
	videoConsumerWindow = False
	vcwFeeds = False
End If
```

'winIDs' is an array which holds reference numbers for all the VIC windows which we collected earlier in the procedure; this is iterated through to count the number of mini-preview feeds in each window. If it is discovered that both the VCW and one of the VPWs contain remote feeds, which is an unusual but not uncommon scenario, the decision is to close the VCW.

```
If totalfeed > winIDs.Count Or vcwFeeds Then
	If vcwFeeds Then
		SetWindowPos(hWndVCW, 0, 444, 0, Screen.PrimaryScreen.Bounds.Width - 444, _
				(Screen.PrimaryScreen.Bounds.Height * 2) / 3, _
				SWP_NOZORDER Or SWP_NOOWNERZORDER Or SWP_NOACTIVATE)
		MakeColumns(hWndVCW)
		posCheck = 1
		For k As Integer = 0 To winIDs.Count - 1
			SetWindowPos(winIDs(k), 0, 0, vicHeight, 444, 110, _
				SWP_NOZORDER Or SWP_NOOWNERZORDER Or SWP_NOACTIVATE)
			vicHeight += 110
		Next
	Else
		For k As Integer = 0 To winIDs.Count - 1
			If feednum(k) > 1 And posCheck = 0 Then
				SetWindowPos(winIDs(k), 0, 444, 0,
					Screen.PrimaryScreen.Bounds.Width - 444, _
					(Screen.PrimaryScreen.Bounds.Height * 2) / 3, _
					SWP_NOZORDER Or SWP_NOOWNERZORDER Or SWP_NOACTIVATE)
				MakeColumns(winIDs(k))
				posCheck = 1
			Else
				SetWindowPos(winIDs(k), 0, 0, vicHeight, 444, 145, _
					SWP_NOZORDER Or SWP_NOOWNERZORDER Or SWP_NOACTIVATE)
				vicHeight += 145
			End If
		Next
	End If
Else
	If videoConsumerWindow Then
		SetWindowPos(hWndVCW, 0, 0, 340, 352, _
			Screen.PrimaryScreen.Bounds.Height - 321, _
			SWP_NOZORDER Or SWP_NOOWNERZORDER Or SWP_NOACTIVATE)
	End If
```

```
        vicHeight = 0
        For l As Integer = 0 To winIDs.Count - 1
                SetWindowPos(winIDs(l), 0, 444, vicHeight, 444, 185, _
                        SWP_NOZORDER Or SWP_NOOWNERZORDER Or SWP_NOACTIVATE)
                vicHeight += 185
        Next
End If
```

The final portion of the procedure is a simple if/else decision, based on whether there are remote feeds or not. If there are remote feeds found in either the VCW or VPW then if the code will determine which window that is and allocate it a large amount of primary display space, with the remaining windows stacked below RAT. If no remote feeds are found then the VCW is given priority and the VPWs are stacked beneath it in the right hand side of the screen. The result of 'Arrange Venue Client' can be seen in Figure 1.6 below, where we can see a Access Grid room which has 4 cameras and one remote feed which has displayed on the third camera's VIC window.

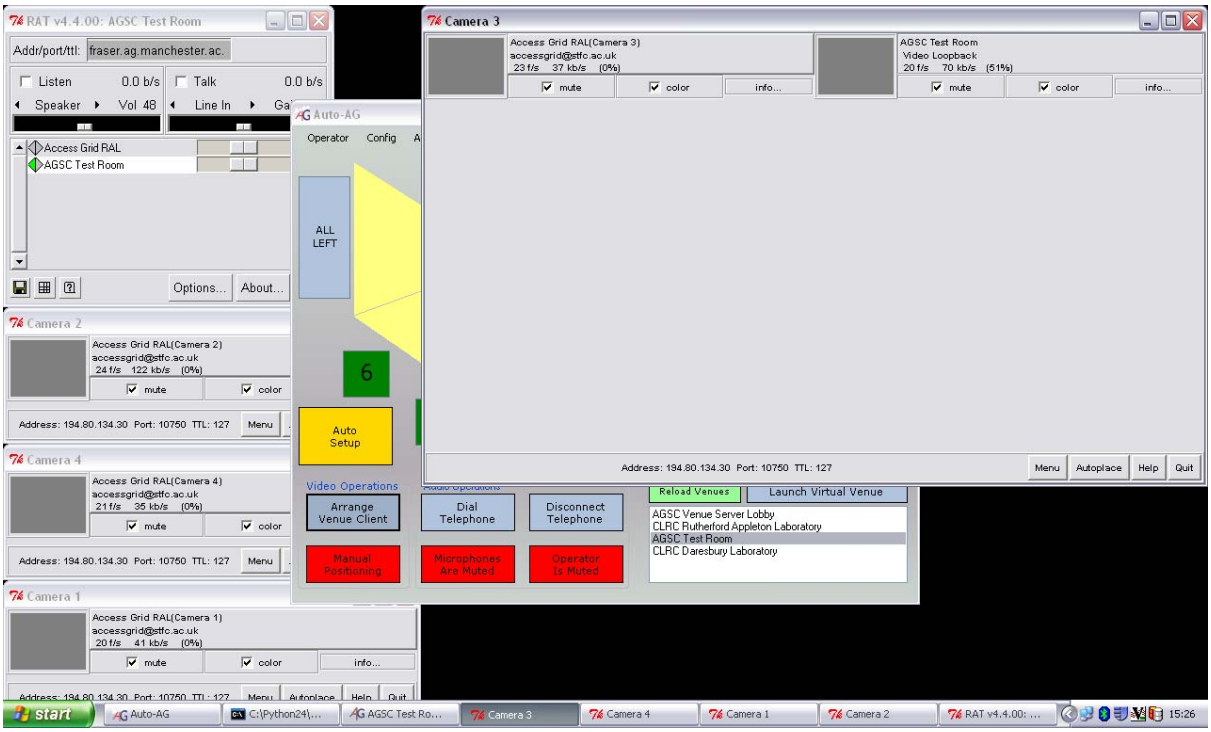


**Figure 1.6**

## Assisting the Window Positioning Process

Making the video windows accessible is only one part of the process, in order to display them, each VCW/VPW mini-preview feed must be left clicked on to pop up a full video window and then dragged to position on a display wall. Auto-AG operates a 'right-click positioning' mode which allows the automatic placement of video feeds in line with a template based on the number of screens available to the Access Grid node, and the direction they extend towards. To achieve this in VB.net, the MouseHook class is leveraged, providing custom definitions when the left and right mouse buttons are pressed. Given this can be potentially dangerous, it is only enabled when the 'right-click positioning' button is activated. Left click functions are employed to unmute VPW feeds (which are muted by default) when anywhere but the mini-preview feeds are clicked, additionally there is a record kept of how many windows are open so that the positioning logic can determine where the next window should be placed.

```
Private Sub m_clsMouseHook_MouseRightDown(ByVal sender As Object, _
                                    ByVal e As System.Windows.Forms.MouseEventArgs) _
                                    Handles m_clsMouseHook.MouseRightDown
        Dim targetWindow, targetChild As IntPtr
        targetWindow = GetForegroundWindow()
        Dim chPos As Rect
        Dim hitPoint As Point
        hitPoint.x = Cursor.Position.X
        hitPoint.y = Cursor.Position.Y
        Dim parPos, cliPos As Rect
        GetWindowRect(targetWindow, parPos)
        GetClientRect(targetWindow, cliPos)
        If (cliPos.Right - cliPos.Left = 352) And (cliPos.Bottom - cliPos.Top = 288) Then
                SetRemotePos(targetWindow)
        Else
                Dim clickName As New System.Text.StringBuilder("", 256)
                GetWindowText(targetWindow, clickName, 256)
                If Not labels.Contains(clickName.ToString) And
                        Not clickName.ToString = venueName Then
                Else
                        If hitPoint.x < parPos.Left Or
                                hitPoint.x > parPos.Right Or
                                hitPoint.y < parPos.Top Or
                                hitPoint.y > parPos.Bottom Then
                            hitPoint.x = -1
                            hitPoint.y = -1
                        End If
                        If hitPoint.x + hitPoint.y > 0 Then
                            chArray.Clear()
                            Dim proc As New EnumChildProcDelegate(
                                                AddressOf Me.EnumChildProc)
                            EnumChildWindows(targetWindow, proc, 0)
                            For j As Integer = 0 To chArray.Count - 1
                                GetWindowRect(chArray(j), chPos)
                                If (chPos.Right > hitPoint.x And _
                                        hitPoint.x > chPos.Left) And _
                                        (chPos.Top < hitPoint.y And _
                                        hitPoint.y < chPos.Bottom) Then
                                        targetChild = chArray(j)
                                End If
                            Next
                            Dim childRect As Rect
                            GetWindowRect(targetChild, childRect)
                            ArrangeFeeds(childRect, targetWindow)
                        End If
                End If
        End If
End Sub
```

Within the above block of code, there is a simple if/else statement to decide what happens when the mouse is right clicked. If the dimensions of the contents of the clicked window are 352x288 (i.e. CIF) then the window is positioned in accordance with the SetRemotePos function. It is essential that the test be done on the contents of the window and not the window itself because Windows XP themes will vary the dimensions subtly. If we have procedurally determined that the window we're right clicking on is a mini-preview screen, Auto-AG will then execute the ArrangeFeeds function, which will tidy up and position all open video feed windows.

## Automating the Window Positioning Process

Due to the Tk/TCL nature of their programming, VIC windows must be left clicked to release the full video window, this cannot be programmatically simulated. As a result, when deploying the 'Auto Setup' button, the operator must relinquish control of the mouse so that Auto-AG can take control of the cursor and click on each mini-preview window. The process of positioning once this automated sequence commences is much the same as the right-click positioning code.

## Audio Software Un-muting

After establishing the foundations for window manipulation in VB.net, the next step was to see what could be done in virtually pressing buttons and clicking checkboxes. The strong motivator for investigating this was through the instability of RAT, which likes to crash if the user interacts with it. By programmatically checking the talk checkbox on RAT, this risk is mitigated; it also provides a visual cue that the microphones are live and broadcasting to remote venues, an often overlooked process in the setup of a meeting.

```
SendMessage(ratHwnd, WM_SETFOCUS, 0, 0)
If tabCheck = False Then
        MakeTab(ratHwnd)
        MakeTab(ratHwnd)
        MakeTab(ratHwnd)
        MakeTab(ratHwnd)
        MakeTab(ratHwnd)
        MakeTab(ratHwnd)
        tabCheck = True
End If
MuteToggle(ratHwnd)
```

Despite VIC and RAT both being developed in Tk/TCL, they have very different designs, preventing the previous methods used for manipulating VIC to be reused. Instead, the programme tries to tab cycle through selectable objects before finding the one we are interested in and then sending the MuteToggle command to the window object to enable or disable microphones. Once this has been completed the first time, a tabCheck Boolean is set, to ensure that RAT keeps this one checkbox highlighted.

## ClearOne Serial Control

Many Access Grid conference suites utilise audio management hardware manufactured by ClearOne/Gentner [8], in order to manage echo cancellation and complex audio routing. At SCT's primary Access Grid room we leverage a Gentner XAP800 in conjunction with a Gentner XAP TH2. These two units combined allow advanced audio routing for microphones, speakers, a DVD player and teleconferencing via the XAP TH2. Until Auto-AG, if a conference wished to dial out into a telephone conference alongside their Access Grid meeting, that meant loading up custom software to interface with the Gentner. Auto-AG uses the standard API ClearOne provide for RS232 serial control in order to provide its own dialling interface for operators to use. From figure 1.7 below, we can see the dialling interface which represents a standard keypad.


**Figure 1.7**

The 'Hang Up' button seen in figure 1.7 and 'Disconnect Telephone' in figure 1.4 are essential buttons for Access Grid, as normally the Gentner Teleconference systems do not hang up after a call is finished, resulting in a loud series of tones being broadcast throughout the conference. Originally, the operator would have to push a button on the unit to ensure the Gentner disconnects itself, but now this can be done through the Auto-AG interface. An additional feature seen in figure 1.4 is the 'Operator is muted' button which will use the Gentner hardware to control the activity of a microphone which may be placed on the desk of the operator, far away from the participants of the conference.

## Virtual Venue Selection

For Auto-AG to be realized as a fully featured assistant to AGTk, it needed to replace the most common functions it would have been used for. One of these is the selection of a Virtual Venue. Auto-AG can read the list of bookmarked Virtual Venues in the Venue Client and allow the Access Grid Operator to connect to one from within the Auto-AG interface. As this list is still maintained by the Venue Client, bookmarks for additional venues cannot be added by Auto-AG; however, once a new Virtual Venue has been added, the 'Reload Venues' button will re-read the Venue Client configuration file and repopulate the Venues list. Interaction with the Venue Client to launch a Virtual Venue is through the SendMessage function within user32. The URL string of the Virtual Venue is passed to a window object and then a 'virtual' return key is pressed to submit it.

## Tutorial Mode

While adding features to make the Access Grid Operator's job easier is very important, the overall objective of providing a tool that anyone could use was not lost. When Auto-AG loads (figure 1.8), it provides the user a choice between entering a tutorial mode or the conventional Auto-AG dashboard (tutorial mode can also be launched at any time in the standard view, however the Venue Client will be closed in the process).

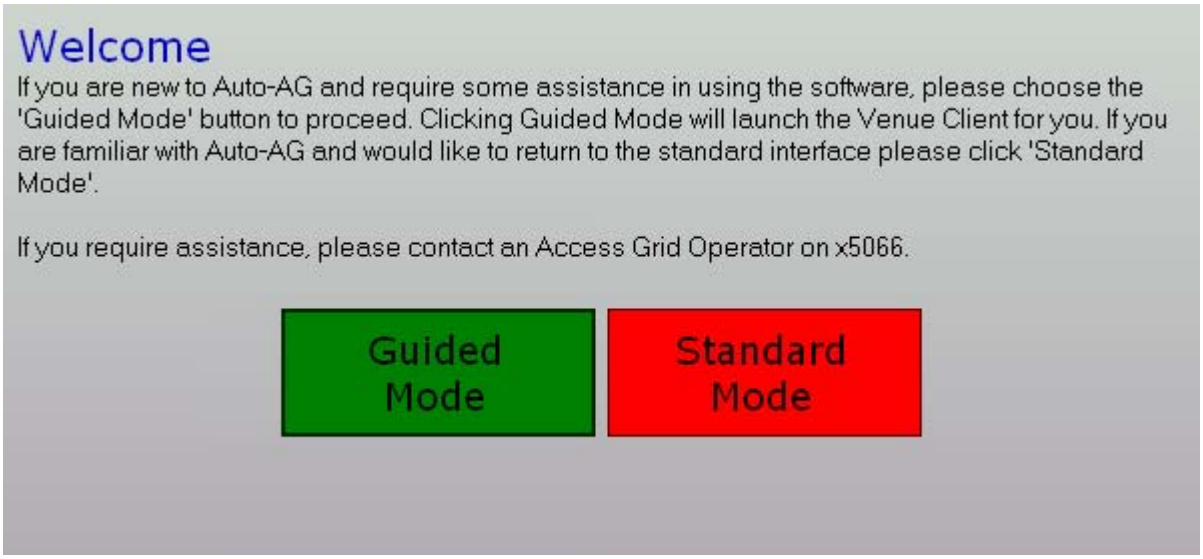


**Figure 1.8**

Once 'Guided Mode' is chosen or 'Tutorial Mode' is selected within Auto-AG, the Venue Client will launch (and depending on serial settings in Auto-AG's configuration, projectors and cameras would be activated) the user is presented with a list of Virtual Venues as in figure 1.9. By eliminating the distraction of other Auto-AG features, the user should be easily guided through the essential processes of entering a Virtual Venue. After the user has entered their Virtual Venue, attempts are made by Auto-AG to ensure the screen does not become cluttered with distracting windows as seen in figure 1.3. The user is then presented with three choices, outlining use cases for Access Grid. The first option shown in figure 1.10 will provide a stripped down Auto-AG interface with just three buttons featured (figure 1.11). Users are expected to display windows with 'Auto Setup' and use 'Adjust Cameras' if fine tuning is necessary; all other features are abstracted away from the user. Once their meeting is complete they can still shut everything down with the 'Shutdown' button.

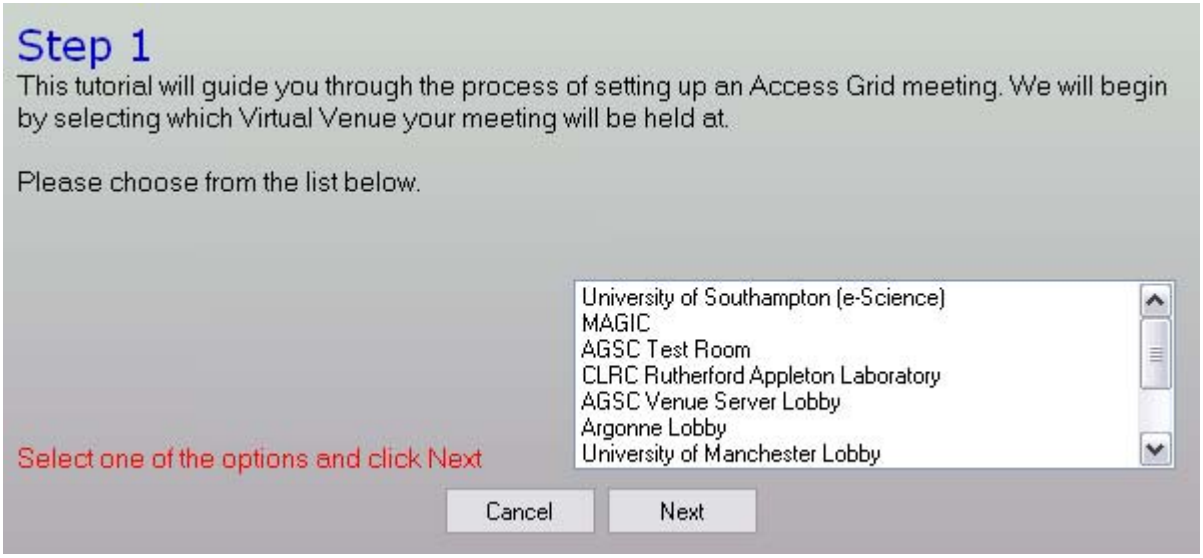


**Figure 1.9**

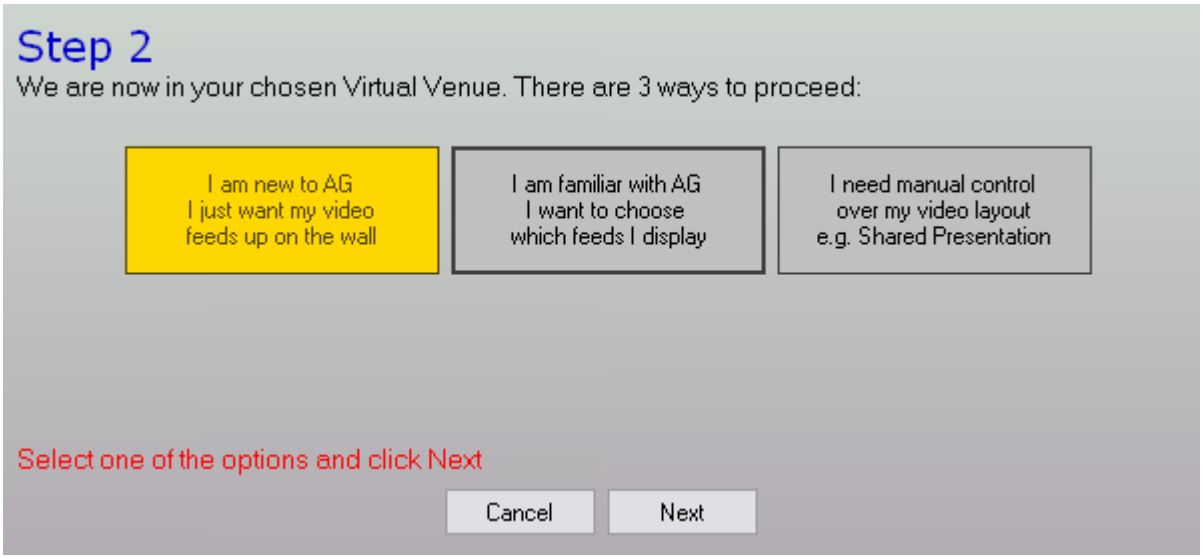


**Figure 1.10**

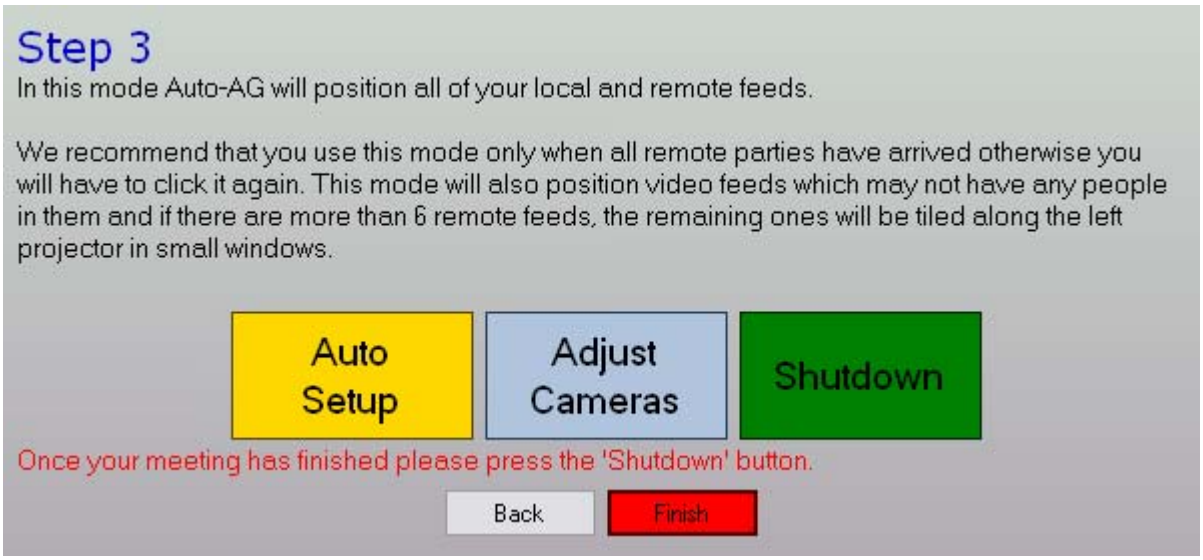


**Figure 1.11**

# 5. Conclusions

The features outlined in this report have greatly enhanced SCT's ability to manage Access Grid facilities. The application has also seen interest from the AGSC as a possible tool to be incorporated into the Access Grid Toolkit. Future development efforts will ensure this takes a step closer, by refactoring the code in python, the native and platform neutral language of the Access Grid Toolkit. Additional effort is also earmarked at incorporating further features from the Venue Client, such as restarting audio and video producer services and switching 'profiles'.

# 6. References

[1] Access Grid
*http://www.accessgrid.org*

[2] Access Grid Support Centre
*http://www.ja.net/services/video/agsc/AGSCHome/*

[3] Access Grid FAQ
*http://www.accessgrid.org/faq*

[4] SCT Access Grid Website
*http://sct.esc.rl.ac.uk/Access_Grid/index.html*

[5] Access Grid Support Centre FAQ
*http://www.ja.net/services/video/agsc/technical-information/faqs.html*

[6] AccessGrid Video and Audio Tools Support (AVATS) Project page
*http://www.cs.ucl.ac.uk/research/avats/index.html*

[7] Win32 API in .NET
*http://www.c-sharpcorner.com/UploadFile/shrijeetnair/win32api12062005005528AM/win32api.aspx*

[8] ClearOne Professional Audio Conferencing website
*http://www.clearone.com/Professional_Conferencing.html*