Introduction:

This ActiveX control is developped for use with the Vision Command Center from the Lego company.
This software is provided as EmailWare, so you can copy and distribute this ActiveX to all your friends, as long as the persons that are using this software send an email to the author.

Any bugs, comments, mallfunctioning, remarks, requests etc … can be send via email to
snoozysplace@be.tf

Please read this manual carefully before using this ActiveX.
The author of this software can not be held responsible for eventually failures of the software, or any damage that is caused to your other hard- and/or software

I hope you all appreciate my hard work, and let me know if it works fine for you or if you would Like me to change some things in the ActiveX.
(additional features, changing certains features etc …)

Index

ActiveX Control at DesignTime

To Use this ActiveX Control, you have to run the setup.exe for propper installation.
Before running the ActiveX, be sure that your VCS software is also correctly installed,
(only the LegoCam with the necessary drivers should also be enough, note that it wil NOT work with another camera, and don't ask me to change it, just buy the real Lego stuff)
and check if you can see the camera lifefeeds within the orginal VCS software.
If you do not see the camera lifefeeds within the orginal VCS software, you will not be able to use the ActiveX because this indicates that your VCS software is not installed propperply.

This ActiveX can also be used with other languages beside Visual Basic, but in this manual we will only explain how to use it with Visual Basic 6.

1.A Inserting the ActiveX on a form

If you startup Visual Basic, and you open an existing project or you start a new project, the first thing that you have to do is to is to include the ActiveX in you VB project.
To do this, follow the following steps:

1. In the menu of Visual Basic, select "Project" and then "Components".
2. Scroll trough the list of components until you see "LegoCameraControl".
   If you do not find this component in your list, this indicates that the ActiveX is not propperly installed.
3. Click on the small empty box on the left of "LegoCameraControl"
   Now you just enabled this control in your VB project.
4. Click on the "OK" Button and the component-screen will be closed.
   You will see the VCS camera icon in your VB Toolbox.

Now you are ready to use the ActiveX, just put it on a form. Do this by following the next steps:

1. Click on the VCS camera icon in your VB Toolbox.
2. Move your mouse over to the form, and hold down your mousebutton at the top-left position where you want the ActiveX do be displayed.
3. Move your mouse a bit to the left and to the bottom, and release your mousebutton.
   The ActiveX will automatically resize itself to 4800 twips wide and 4125 twips high.
   (that is 320 pixels wide and 275 pixels high)

1.B Using some features in DesignTime

You can use some features at DesignTime for easying up the development of your application.

1. *Switching colordepth*
If the active-property is set to true, you can also change the colordepth in this preview mode.
The changes are directly shown in the preview window.
Settings for this colordepth are:

1.	Low	=>	this is 16 colors.
2.	Medium	=>	this is 256 colors.
3.	High	=>	this is the colorsetting that your windows uses with your display adapter.

1. *Changing the active boolean*
When you set the active boolean to true in DesignTime, you will get the camerafeed !

3. *Changing the debugpreview boolean*
When you set the debugpreview boolean to true, you will get the camerafeed in a double frame, the top frame will show the normal camerafeed and the bottom frame will show
The "converted" camerafeed in the selected colordepth and with area-indicators on it.
An area-indicator will show up in black if not active and in white when color- or motiondetection is triggered.

2. ActiveX Control at RunTime

At RunTime you will also see the lifefeed of your camera if you made a propperly initialise procedure within your VB-code, but with a few differents:

1. The framerate will be a bit lower then in previewmode, this is caused by the calculation-engine for motion- and color-detection.
2. You will see some black outline boxes on the bottom window (if debugpreview is enabled), these are the detectionzones that you defined for the used layer.

2.A Initialise procedure at startup

Before you can actually do something usefull with the ActiveX to let it respond to color- and motion-detection, you have to initialise a lot of things. (see lcc_sampleproject included with the ActiveX)

1. Setting colordepth to the desired working colordepth.
2. Setting debugpreview to true or false
3. Do a clear_all. (for erasing all layers and detectionzones)
4. Setting the active property to true.
5. Setting the Device_id to the LegoCam.
6. Adding detectionzones with the add_motion_detection and the add_color_detection methods.
7. Setting the current used layer via the layer property.

Additional features in RunTime:

1. *Finding a colorvalue as longvariable*
Just move your mouse over the bottom frame and click the left mousebutton on the needed color, a messagebox will then popup with the X coördinate, Y coördinate and colorvalue of the selected pixel.
To use this option, debugpreview need to be set to true.

2.B Setting and getting properties

Here is a list of all properties that can be used, with a brief description:

1. About — Only available at DesignTime, read-only
   This will show an aboutbox with the current version and information on how to contact the author of this application

2. Active — Available at DesignTime and RunTime, read/write
   This property must be set to true in the initialise procedure of your VB code. You can also put this property to false within your code, to hold processing of detectionzones for some time.

3. Colordepth — Available at DesignTime and RunTime, read/write
   This property can be used to set the working colordepth, you may also notice that the bottomframe will also be shown in the selected colordepth.

4. Debugpreview — Available at DesignTime and RunTime, read/write
   This property is used to set or retrieve the boolean that represents if the bottomframe is shown or not.

5. Device_id — Available at DesignTime and RunTime, read/write
   This property is used to set or retrieve the device_id,
   e.g. an integer representing the LegoCam capturedeviceindex.
   Try out different settings, starting with zero, if you are not sure what your capturedeviceindex is. Note that this is system dependent !

6. Height — Read-only

7. Index — Please, do not use more then one instance of the LCC ActiveX, because only one application at a time can have control of the LegoCam.

8. Layer — Use this at Runtime, read/write
   This property is used to set/get the actual used layer.
   This can be changed at any moment in RunTime.

9. Left — Available at DesignTime and RunTime, read/write
   This property indicates the left-position of the lifefeed window.

10. Picture — Only available at RunTime, read-only
    If you set the picture-property of any picturebox equal to the pictureproperty of the lcc control, then the current picture of the camerafeed will be shown in that imagebox or picturebox (in TrueColor, even if you use low or medium colordepth)

11. Top — Available at DesignTime and RunTime, read/write
    This property indicates the top-position of the lifefeed window.

12. Visible — Available at DesignTime and RunTime, read/write
    Make the control Visible or Invisible at RunTime.

13. Width — Read-only

2.C Using methods

This ActiveX does NOT use any pre-defined zones like the original VCS software does, instead you can define freely your own detectionzones.
You can define 64 different layers, with 64 different detectionzones per layer.
So you don't have to change your detectionzones constantly, just put them together in layers, and switch between layers.

1. *Clearing a specified layer*
   LCC1.Clear layer
   Replace layer by an integer, indicating the layer that you want to erase.

2. *Clearing all layers*
   LCC1.Clear_all
   This method will erase all existing layers.

3. *Showing the aboutbox in RunTime*
   LCC1.ShowAbout
   This method will bring up the aboutbox concerning the ActiveX

4. *Adding a color-detectionzone*
   LCC1.add_color_detection layer,zone,left,top,width,height,color,percentage
   This method will add a color detectionzone to a specified layer.
   (for coördinates, see appendix A)

Layer	= Number of the layer where the detectionzone belongs to.	(1 to 64)
Zone	= Number of the detectionzone	(1 to 64)
Left	= leftcoördinate of the detectionzone	(1 to 320)
Top	= topcoördinate of the detectionzone	(1 to 240)
Width	= pixelwidth of the detectionzone	(1 to 320)
Height	= pixelheight of the detectionzone	(1 to 240)
Color	= colorvalue that needs to be detected	(long var)
Percentage	= Percentage of detectionzone that needs to be filled up by the specified color before a color-event is raised.	(0 to 100)

5. *Adding a motion-detectionzone*
   LCC1.add_motion_detection layer,zone,left,top,width,height,percentage
   This method will add a color detectionzone to a specified layer.
   (for coördinates, see appendix A)

Layer	= Number of the layer where the detectionzone belongs to.	(1 to 64)
Zone	= Number of the detectionzone	(1 to 64)
Left	= leftcoördinate of the detectionzone	(1 to 320)
Top	= topcoördinate of the detectionzone	(1 to 240)
Width	= pixelwidth of the detectionzone	(1 to 320)
Height	= pixelheight of the detectionzone	(1 to 240)
Percentage	= Percentage of detectionzone that needs to be changed, compared with the previous frame before a motion-event is raised.	(0 to 100)

6. *Saving the current shown frame to disc*
   lcc1.save_picture filename truecolor

Filename	= The file to save to. (including path), the filename extension needs to be BMP or JPG.
Truecolor	= If you set true for truecolor, then the image will be saved as a truecolor image, otherwise it will be save in the working colordepth.

2.D Intercept events

After all you hard work with initialising and defining detectionzones, it is time that we get something back from the ActiveX, this is done trough events that are raised upon detection …

1. *Event when motion is detected*
When a motion is detected in a motion-detectionzone of the current layer, an event is raised.
Put your code in the following Sub:

Private Sub lcc1_motion(layer As Integer, zone As Integer, status As Boolean, percentage As Integer)

End Sub

Layer	= Current layer
Zone	= Zone where detection happened
Status	= True when motion is detected.
	= False when motion is no longer detected.
Percentage	= an integer between 0 and 100, indicating how much motion is detected.

This event will only raise at the beginning of a motiondetection and will only raise once, even if continously motion is detected. (with status=true)
When there is no longer motion detected, this event is raised again.
(also only once, with status = false)

2. *Event when color is detected*
When a motion is detected in a motion-detectionzone of the current layer, an event is raised.
Put your code in the following Sub:

Private Sub lcc1_color(layer As Integer, zone As Integer, status As Boolean, color As Long, percentage As Integer)

End Sub

Layer	= Current layer
Zone	= Zone where detection happened
Status	= True when color is detected.
	= False when color is no longer detected.
Color	= long variable, indicating the detected colorvalue.
Percentage	= an integer between 0 and 100, indicating how much color is detected.

This event will only raise at the beginning of a colordetection and will only raise once, even if continously color is detected. (with status=true)
When there is no longer color detected, this event is raised again.
(also only once, with status = false)

3. Appendix A

Concerning detectionzones:

1. *detectionarea*
   The detectionarea is 320 pixels width and 240 pixels height (camera view),
   So you need to stay in this area to have a propper detectionzone.
   The topleft position = (1,1)
   The bottomright position = (320,240)

2. *detectionvisualisation*
   Before writing code in the events, you can visually test your zones. (in RunTime)
   When you activate a layer, you can test the rules by watching the bottom frame if debugpreview is set to true.
   When propper motion or color is detected, the black outline boxes that represent the detectionzones will become white when color or motion is activated and will become black when color or motion is no longer detected.

4. Appendix B

In order to run the sampleproject correctly, you download some NQC code to your RCX that does the following things:

if one of the following messages is received by the RCX then
the following things need to be done:

msg 1 : motor C needs to turn left and motor B needs to turn up
msg 2 : motor B needs to turn up
msg 3 : motor C needs to turn right and motor B needs to turn up
msg 4 : motor C needs to turn left
msg 5 : motor C and B needs to stop (play a tone also)
msg 6 : motor C needs to turn right
msg 7 : motor C needs to turn left and motor B needs to turn down
msg 8 : motor B needs to turn down
msg 9 : motor C needs to turn right and motor B needs to turn down

This code is normally included with the sampleproject, but you may need this information if you want To re-write this RCX program with another RCX development tool (like the original RIS software)

5. Appendix C

Work in progress – feature plans:

1. Speeding up the FPS
2. Writing a ZoneGenerator tool (witch automatically generates VB code for you)
3. adding labels to zone-declaration, these labels will then also be returned in the motion and color event.
4. Implementing some Voice Recognition (after all, the LegoCam does have a build-in mic)

For speeding up the FPS I can still use some help. (the other things I can easily do myself)
Anybody that does have some VB code to do the following things as fast as possible can always email it to snoozysplace@be.tf
(You will get credits for it)

1. Convert a 320 x 240 pixels – Truecolor image in a picturebox to a 256 colors and 16 colors image, also in a picturebox. (going via a DIBsect is way to slow)
2. Convert a 320 x 240 pixels image in a picturebox to an array like color(x,y)
   (the picture.point(x,y) method is way to slow)