Generative Imaging

Zsolt Korai
Abstract

The Generative Imaging curriculum’s main subject is the graphic programming language called VVVV. This is a visual programming language mostly used for art installations and various types of projections. Through these lessons, you will get to know the VVVV program's interface, simple mathematics and the processing of different message types, 2D and 3D imaging and transformations, picture and video detection, the controlling of external peripheries (like cameras, Kinect, MIDI controllers, Wacom tablets, Arduino and DMX) and the OSC communication with Android and IOS. Advanced programming knowledge is not required for the use of VVVV, only the basics of computer usage and graphic-video editing.
# Table of Contents

1. This material is about learning how to use VVVV ................................................................. 1  
2. User interface, language, menu system, the Inspektor and the use of help files ................................ 20  
   2.1. First steps, installation and useful tips ............................................................................. 20  
   2. Video Lesson ........................................................................................................................ 28  
3. Message types, inlets and outlets ....................................................................................... 29  
   1. Video Lesson ........................................................................................................................ 34  
4. Komplexebb matematikai feladatok, fájlia írás és olvasás ..................................................... 35  
   1. Video Lesson .......................................................................................................................... 46  
5. Rendering types .................................................................................................................... 47  
   1. Video Lesson .......................................................................................................................... 51  
6. Layers ...................................................................................................................................... 52  
   1. Video Lesson .......................................................................................................................... 57  
7. Rendering 3D Objects ............................................................................................................. 58  
   1. Video Lesson .......................................................................................................................... 62  
8. Transformation and grouping ............................................................................................... 63  
   1. Video Lesson .......................................................................................................................... 70  
9. Effects and Plug-ins .............................................................................................................. 72  
   1. Video Lesson .......................................................................................................................... 80  
10. Audio Analysis ...................................................................................................................... 81  
    1. Video Lesson .......................................................................................................................... 85  
11. Video and tracking ............................................................................................................... 86  
    1. Video Lesson .......................................................................................................................... 95  
12. External controllers ............................................................................................................... 96  
    1. Video Lesson .......................................................................................................................... 105  
13. Arduino and some basic electronics .................................................................................... 106  
    1. Video Lesson .......................................................................................................................... 127  
14. Communication between computers and different control methods ................................... 128  
    1. Video Lesson .......................................................................................................................... 143  
15. User interface design .......................................................................................................... 144  
    1. Video Lesson .......................................................................................................................... 157  
    2. Node list ................................................................................................................................ 157
List of Figures

1.1. max ......................................................................................................................... 2
1.2. PD ......................................................................................................................... 2
1.3. Eyesweb ................................................................................................................. 3
1.4. Touchdesigner ....................................................................................................... 3
1.5. VVVV .................................................................................................................... 4
1.6. Cybear_screenshot ............................................................................................... 4
1.7. Lightmare_screens ............................................................................................... 5
1.8. Lanvideosource ..................................................................................................... 6
1.9. Microode_screens ................................................................................................. 6
1.10. u7angel_still ........................................................................................................ 7
1.11. Unc_screenshot13 ................................................................................................. 7
1.12. Mr_prudence_Parhelia ....................................................................................... 8
1.13. Elliottwoods_Asse .............................................................................................. 8
1.14. Patxi7 Ehime Daruma ....................................................................................... 9
1.15. Unc_screenshot 13 ............................................................................................. 10
1.16. Unc_screenshot13 ............................................................................................... 11
1.17. Lecloneur_screens ............................................................................................. 11
1.18. Sebl ...................................................................................................................... 12
1.19. Lecloneur_screens ............................................................................................. 12
1.20. Lorenz_2 ........................................................................................................... 13
1.21. Brozzmical_screen ............................................................................................. 13
1.22. Lecloneur_screens ............................................................................................. 14
1.23. Lecloneur_screens ............................................................................................. 14
1.24. Vux_screenshot ................................................................................................. 15
1.25. Vux_screenshot ................................................................................................. 15
1.26. WirmachenBunt, Mutabor Design G ................................................................. 16
1.27. Stain_MIMPI ...................................................................................................... 16
1.28. Eletromeier_screens ......................................................................................... 17
1.29. Vux_screenshot ................................................................................................. 17
1.30. Dottore_screensh .............................................................................................. 18
1.31. mr prudence_Isograms ..................................................................................... 18
2.1. Downloading VVVV ............................................................................................ 20
2.2. crack.exe ............................................................................................................. 20
2.3. Administrator mode ............................................................................................ 21
2.4. Running VVVV.exe in administrator mode ......................................................... 21
2.5. VVVV after launching ....................................................................................... 22
2.6. Two left clicks .................................................................................................... 23
2.7. Two left clicks + a right click ............................................................................ 24
2.8. Two right clicks .................................................................................................. 24
2.9. Main menu ......................................................................................................... 25
2.10. Lfo + Inspektor ................................................................................................. 26
2.11. Lfo help file ..................................................................................................... 27
3.1. The Message menu .......................................................................................... 29
3.2. Bang and Toggle ................................................................................................. 30
3.3. Vector types ...................................................................................................... 30
3.4. String and number message .......................................................................... 30
3.5. Text module ...................................................................................................... 31
3.6. Hello Renderer .................................................................................................. 31
3.7. Resolutions 10 and 300 .................................................................................. 32
3.8. Scale X Y .......................................................................................................... 32
3.9. Moving text with mouse ................................................................................. 33
3.10. Hello in colour ............................................................................................... 33
3.11. Adjusting HSL with mouse position .............................................................. 33
4.1. Basic mathematical operations ...................................................................... 35
4.2. Complex mathematical operations ............................................................... 35
4.3. Using logic gates .............................................................................................. 36
<table>
<thead>
<tr>
<th>Page</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.9</td>
<td>Resolution setup in DX9Texture</td>
</tr>
<tr>
<td>9.10</td>
<td>Alpha texture out</td>
</tr>
<tr>
<td>9.11</td>
<td>Sample effect</td>
</tr>
<tr>
<td>9.12</td>
<td>Contribution site</td>
</tr>
<tr>
<td>9.13</td>
<td>Root patch</td>
</tr>
<tr>
<td>10.1</td>
<td>AudioAnalysis</td>
</tr>
<tr>
<td>10.2</td>
<td>Recording device</td>
</tr>
<tr>
<td>10.3</td>
<td>Windows audio recording settings</td>
</tr>
<tr>
<td>10.4</td>
<td>AudioIn and FFT</td>
</tr>
<tr>
<td>10.5</td>
<td>TagPoint plug-in</td>
</tr>
<tr>
<td>10.6</td>
<td>Mouse and MouseState for TagPoint</td>
</tr>
<tr>
<td>10.7</td>
<td>FFT patch</td>
</tr>
<tr>
<td>11.1</td>
<td>VideoIn DirectShow</td>
</tr>
<tr>
<td>11.2</td>
<td>VideoIn Property page</td>
</tr>
<tr>
<td>11.3</td>
<td>VideoIn Ueye</td>
</tr>
<tr>
<td>11.4</td>
<td>VideoIn OpenCV DirectShow</td>
</tr>
<tr>
<td>11.5</td>
<td>VideoIn OpenCV Ueye Property page</td>
</tr>
<tr>
<td>11.6</td>
<td>IDS Ueye</td>
</tr>
<tr>
<td>11.7</td>
<td>Imaging Source</td>
</tr>
<tr>
<td>11.8</td>
<td>Basler cameras</td>
</tr>
<tr>
<td>11.9</td>
<td>Point Grey Flea</td>
</tr>
<tr>
<td>11.10</td>
<td>Point Grey 3d cameras</td>
</tr>
<tr>
<td>11.11</td>
<td>Asus Xtion, Xtion Pro</td>
</tr>
<tr>
<td>11.12</td>
<td>Microsoft Kinect</td>
</tr>
<tr>
<td>11.13</td>
<td>Contour Tracking</td>
</tr>
<tr>
<td>11.14</td>
<td>Color Tracking</td>
</tr>
<tr>
<td>11.15</td>
<td>Kinect Depth</td>
</tr>
<tr>
<td>11.16</td>
<td>Kinect user detection</td>
</tr>
<tr>
<td>12.1</td>
<td>Behringer BCR Midi interface</td>
</tr>
<tr>
<td>12.2</td>
<td>Korg Nano series</td>
</tr>
<tr>
<td>12.3</td>
<td>Wacom tablet</td>
</tr>
<tr>
<td>12.4</td>
<td>Tablet Switches and Sliders</td>
</tr>
<tr>
<td>12.5</td>
<td>Tablet Tube2d patch</td>
</tr>
<tr>
<td>12.6</td>
<td>Tablet Fluid patch</td>
</tr>
<tr>
<td>12.7</td>
<td>Wiimote</td>
</tr>
<tr>
<td>12.8</td>
<td>Infra pen</td>
</tr>
<tr>
<td>12.9</td>
<td>Wiimote plug-in</td>
</tr>
<tr>
<td>12.10</td>
<td>Miscellaneous Wiimote settings</td>
</tr>
<tr>
<td>12.11</td>
<td>Kontrolleur interface</td>
</tr>
<tr>
<td>12.12</td>
<td>Kontrolleur patch</td>
</tr>
<tr>
<td>12.13</td>
<td>Kontrolleur setting</td>
</tr>
<tr>
<td>13.1</td>
<td>Standard Arduino</td>
</tr>
<tr>
<td>13.2</td>
<td>Standard Arduino back side</td>
</tr>
<tr>
<td>13.3</td>
<td>Arduino Bluetooth</td>
</tr>
<tr>
<td>13.4</td>
<td>Arduino Mini</td>
</tr>
<tr>
<td>13.5</td>
<td>Arduino Mega</td>
</tr>
<tr>
<td>13.6</td>
<td>Lilypad</td>
</tr>
<tr>
<td>13.7</td>
<td>Arduino Fio</td>
</tr>
<tr>
<td>13.8</td>
<td>Arduino Micro</td>
</tr>
<tr>
<td>13.9</td>
<td>Arduino Ethernet</td>
</tr>
<tr>
<td>13.10</td>
<td>Serial port</td>
</tr>
<tr>
<td>13.11</td>
<td>Arduino Game</td>
</tr>
<tr>
<td>13.12</td>
<td>Arduino Wifi</td>
</tr>
<tr>
<td>13.13</td>
<td>Arduino GSM shield</td>
</tr>
<tr>
<td>13.14</td>
<td>Lilypad MP3 player</td>
</tr>
<tr>
<td>13.15</td>
<td>Arduino Midi shield</td>
</tr>
<tr>
<td>13.16</td>
<td>Arduino Ethernet shield</td>
</tr>
<tr>
<td>13.17</td>
<td>Arduino Proto shield</td>
</tr>
<tr>
<td>13.18</td>
<td>Arduino Zigbee shield</td>
</tr>
<tr>
<td>13.19</td>
<td>Arduino breadboard shield</td>
</tr>
</tbody>
</table>
13.20. Arduino Motor shield  .......................................................... 118
13.21. Lilypad circuit ................................................................. 119
13.22. Lilypad climate dress ......................................................... 119
13.23. Lilypad climate dress ......................................................... 120
13.25. Arduino Fritz software ....................................................... 122
13.27. Single Led Calculator ......................................................... 123
13.28. Serial Led Calculator ......................................................... 123
13.29. Arduino interface ............................................................. 124
13.30. Arduino and Led .............................................................. 125
13.31. Arduino and Switch ......................................................... 126
13.32. Arduino and Potentiometer ............................................... 126
14.1. Touch Osc Ipad Interface ................................................... 129
14.2. Touch Osc in Iphone ......................................................... 130
14.3. OSC Receiver ................................................................. 130
14.4. OSC Sender ................................................................. 130
14.5. UDP Sender ................................................................. 131
14.6. UDP Receiver ............................................................... 131
14.7. Sunlite USB DMX ............................................................ 131
14.8. Sunlite USB DMX ............................................................ 132
14.9. Enttec USB DMX ............................................................ 133
14.10. Enttec USB DMX clone ..................................................... 134
14.11. Enttec USB DMX clone ..................................................... 135
14.12. Dimmer ................................................................. 135
14.13. Compuite Controller ....................................................... 135
14.15. DMX PWM converter ....................................................... 137
14.16. LED Lamp ................................................................. 137
14.17. DL3 Projector ............................................................... 138
14.18. Enttec USB DMX receiver ................................................ 138
14.19. Enttec USB DMX interface ............................................... 139
14.20. Cheap Artnet input ........................................................ 140
14.21. Cheap Artnet DMX output ............................................... 141
14.22. Enttec Artnet Dmx converter ........................................... 142
14.23. Artnet Sender ............................................................... 142
14.24. Artnet Receiver ............................................................. 143
14.25. Little Cat Midi software ................................................... 143
15.1. Toggle Button Gui ........................................................... 145
15.2. Toggle Button Gui Multitouch .......................................... 146
15.3. XY Slider Gui ............................................................... 146
15.4. XY Slider Gui Multitouch ................................................ 147
15.5. Slider Gui ................................................................. 148
15.6. Slider Gui Multitouch ....................................................... 149
15.7. Rotary Slider Gui .......................................................... 150
15.8. Rottary Slider Gui Multitouch .......................................... 151
15.9. Radio Button Gui .......................................................... 152
15.10. Button 3D Quad ........................................................... 153
15.11. Button 3D Mesh ............................................................ 154
15.12. PicPoints ................................................................. 154
15.13. TagPoints ................................................................. 155
15.14. Write Raw ................................................................. 155
15.15. Write String .............................................................. 156
15.16. Write and read automatically ............................................ 156
Chapter 1. This material is about learning how to use VVVV

VVVV is a visual programming environment developed in Germany. Sebastian Oschatz and Max Wolf started it in 1998 within the framework of MESO (http://www.meso.net/vvvv). Sebastian Gregor joined a little later, and in 2000 Joreg started to work on the user interface. The first public release was in 2002. Joreg and Sebastian Gregory have been leading the development ever since.

You may have seen other video, animation or 3D software with similar interfaces, like Eyeon Fusion, Blender, 3DS Max, Nuke, UDK, Unity etc. There are other programs that look more like VVVV, including the multiplatform Max/MSP and PureData (PD), Quartz Composer (which is integrated into OS X), EyesWeb and the Windows only TouchDesigner.

Max/MSP (http://cycling74.com/products/max/) and PD (http://puredata.info/) are almost brothers because they are both developed by Miller Puckette. The difference between them is that Max is proprietary software and has a user-friendly interface, while PD is an Open Source application. They are usually used for sound generation and control, but many video devices have been integrated in them over the years (like a node-based Shader editor in Max).

EyesWeb used to be the best software for recognizing and tracking a human shape, so it was mainly used in theatres and dance performances. Its most popular use case was motion detection, with the data transferred to another machine responsible for the visuals, generated in a different environment. Unfortunately this software is often overlooked / has been surpassed by the development of alternatives.

Touch Designer is the closest to VVVV regarding its performance (http://www.derivative.ca/). In contrast with , its user interface is reminiscent of 3D softwares'. It is also more expensive than 0. It costs between 600 and 2200 USD while VVVV is free for private use. Touch Designer’s great innovation is the use of Cuda for processing effects which provides for a higher performance than any competitors, since calculations are done on the graphics card rather than on the computer’s processor.

VVVV is primarily used for art projects, video-installations, theatre productions, exhibition interfaces, building and object projections (mapping) and projections at concerts. Beyond real time audio and video synthesis, it also serves as a versatile controller for physical objects: motors, relays, LEDs, lights (with Arduino). Moreover, it is useful for controlling theatre and concert lighting, as it is compatible with Artnet and DMX systems. It also has a built-in Timeline, which makes it an ideal tool for music animations, interactive systems, and sound or sensor data reactive installations.

Furthermore, VVVV can be controlled with external devices. These can be internal or external sound (based on volume or frequency changes), tablets, joysticks, Wii controllers, Microsoft Kinect or Primesens 3D cameras, conventional cameras (light or motion detection), MIDI controllers, sensors and many other things. In fact, users are continuously uploading new external software to the forums and to the software’s own website, expanding the range of compatible input devices. Additionally, uploads include new effects and simulations (like Bullet Physics which is used in games, physical simulations used in 3D modelling, or the popular GPU Particle Systems). If you are an advanced programmer then you can also write your own nodes and effects in C# or calculate the different transformations on the GPU (similarly to Cuda or Hlsl Shader).

The real advantage of VVVV compared to other 3D software is that it performs the rendering in real time, which makes the development process much faster. Conversely, real time rendering means that the render quality suffers, especially compared to other software which focuses on superior rendering. A lot of features are missing (like shading, blurring and other effects) that we are used to in other programs. However, VVVV is actively developed and constantly improving.

VVVV is currently a Windows only application because it uses the .Net and DirectX9 frameworks for visualization. Interestingly, at the latest developer conference, entitled Node Workshop, an OpenGL-based initiative was presented, which means that VVVV might become multiplatform in the future.

The most important development of recent years, which was started by a user called Vux, is the implementation of DirectX 11 support for VVVV. This is important because the program is based on DirectX 9 at the moment.
This material is about learning how to use VVVV

which is a relatively old visualizing framework. The newest games are all written in DirectX 11, since it can produce more realistic effects, offers better performance and it is more efficient. Unfortunately, the aforementioned VVVV support is still in Alpha state, so we have to wait for it a little longer.

In the attached video you can see the most significant VVVV works of recent years.

Enjoy the movie and have fun learning!

**Figure 1.1. max**

![Diagram of max](image1.png)

**Figure 1.2. PD**

![Diagram of PD](image2.png)
This material is about learning how to use VVVV

The (arduino) object works with the Firmata!

Arduino (previously known as Fduino firmware)

Figure 1.3. Eyesweb

Figure 1.4. Touchdesigner
This material is about learning how to use VVVV.

**Figure 1.5. VVVV**

**Figure 1.6. Cybear_screenshot**
This material is about learning how to use VVVV.

Figure 1.7. Lightmare_screens
This material is about learning how to use VVVV.

Figure 1.8. Lanvideosource

Figure 1.9. Microdee_screens
This material is about learning how to use VVVV.

Figure 1.10. u7angel_still

Figure 1.11. Unc_screenshot13
This material is about learning how to use VVVV.

Figure 1.12. Mr_prudence_Parhelia

Figure 1.13. Elliotwoods_Asse
This material is about learning how to use VVVV.

Figure 1.14. Patxi7 Ehime Daruma
This material is about learning how to use VVVV.

Figure 1.15. Unc_screenshot 13
This material is about learning how to use VVVV.

Figure 1.16. Unc_screenshot 13

Figure 1.17. Lecloneur_screens
This material is about learning how to use VVVV.

Figure 1.18. Sebl

Figure 1.19. Lecloneur_screens
This material is about learning how to use VVVV.

Figure 1.20. Lorenz_2

Figure 1.21. Brozzmical_screen
This material is about learning how to use VVVV.

Figure 1.22. Lechloneur_screens

Figure 1.23. Lechloneur_screens
This material is about learning how to use VVVV

Figure 1.24. Vux_screenshot

Figure 1.25. Vux_screenshot
Figure 1.26. WirmachenBunt, Mutabor Design G

Figure 1.27. Stain_MIMPI
This material is about learning how to use VVVV.

Figure 1.28. Eletromeier_screens

Figure 1.29. Vux_screenshot

Created by XMLmind XSL-FO Converter.
This material is about learning how to use VVVV

Figure 1.30. Dottore_screensh

Figure 1.31. mr prudence_Isograms
This material is about learning how to use VVVV.
Chapter 2. User interface, language, menu system, the Inspektor and the use of help files

2.1. First steps, installation and useful tips

It is generally recommended to use the most up-to-date version of VVVV because the developers are constantly fixing the known bugs while extending the software. However, in case you find any annoying system errors you should go back to a previous version.

First of all, download the latest release of the software from http://vvvv.org/downloads. It is important to download the addon pack as well because this contains very useful extensions which are used in many patches.

**Figure 2.1. Downloading VVVV**

It is discouraged to use the built-in compression software of the operating system with the downloaded files, because it can leave some parts of the package unpacked. You should use Winrar, Winzip or 7zip instead. You need to copy the addon pack’s zipped content to the root directory of the VVVV library. When the above is done you need to run the crack.exe file which is in the VVVV library. Without running this, the software will not recognize the *.v4p extension as default. It also shows if any of the necessary software is missing that are needed to run VVVV.

**Figure 2.2. crack.exe**
There is only one last important step left to do to run the software: setting the Administrator mode. If this is not permitted, the program will still run but it will be useless. This setting exists since Windows Vista and can be activated in two ways:

- Authorizing the whole operation system to be Admin. For this you have to choose the Never notify option under User Accounts within the Control Panel. After restarting the computer, VVVV will run as expected.

  **Figure 2.3. Administrator mode**

- Setting only VVV as Admin. For this you need to right-click on the VVVV.exe file and under the Compatibility menu within Settings, choose Run this program as administrator. It is important to note that if you use this option you need to do this every time you install a new version of the software.

  **Figure 2.4. Running VVVV.exe in administrator mode**
When everything is set up, launch the VVVV.exe file.

2.2. The interface

After launching, you will not see a usual user interface because there aren't buttons or menus here. Even though it may seem only a grey window at the moment, in fact VVVV is very user-friendly. Clicking on the canvas reveals various menus.

Figure 2.5. VVVV after launching
Three different kinds of menus can be launched from the user interface:

- The first one is the node list. This can be launched by double-clicking the left mouse button. You can search here with typing or right-clicking. When you start typing, the menu will show only those nodes that start with that particular letter. When you right-click, the nodes will appear grouped by functions. This can be useful when you don't remember the exact name of the node that you are looking for, but you know what it is used for.

Figure 2.6. Two left clicks
Figure 2.7. Two left clicks + a right click

Figure 2.8. Two right clicks

- In the second menu you can browse the different message types. This can be launched by double-clicking the right mouse button.
• The main menu can be launched with a middle click or by right-clicking while holding down the Space button. In this menu you can find some familiar functions followed by their shortcuts.

**Figure 2.9. Main menu**
Finally let’s discuss some basic concepts. A program written in VVVV is called a patch and the little boxes that we connect within the program are the nodes. The patches are saved with a *.v4p extension.

These are essentially text files that are slightly similar to *.xml documents. They contain the positions, parameters and the connections of the nodes. Every node has inlets and outlets that are indicated by small grey boxes. Imagine it as if the functions of a conventional text-based programming language were boxed in and its variables and outgoing parameters were the inlets and outlets. The only difference is that here you do not need to declare function; you only have to refer to them.

Now double-click with the left mouse button, choose the lfo function and press enter. You can see five inlets on the top of the box. If you hover your mouse over them, the program shows the names and the variables of the inlets: Period: 1.00s, Pause: 0, Reverse: 0, Reset: 0, Phase: 0.00. The outlets are on the bottom of the box: Output: 0.xxxx~, Change: 0 (this will change to 1 every second), and you will see a number at the Cycles outlet.

The parameters and the potential hidden settings of the nodes can be seen by pressing Ctrl + i. This is the shortcut for the Herr Inspektor window. If you click on the lfo node you can see the parameters of its inlets and outlets. We will discuss the use of the Inspektor further in another lesson.

**Figure 2.10. Lfo + Inspektor**
If you want to know more about a node, how it works and what it can be used for, then click on it and press the F1 function button. This opens the node's help file. (In the case of the lfo node it’s called \textit{LFO (Animation) help}.) Fortunately, VVVV help files are very readable and informative.

If you want to use a patch that you found in a help file, but you don't want to re-create it, then you can copy it after selecting it, pressing Ctrl +c, and pasting it to your own patch with Ctrl +v.

\textbf{Figure 2.11. Lfo help file}
If you want to navigate inside a patch, you can use the mouse scroll or press and hold the right mouse button on an empty grey surface while moving the mouse. The whole patch will move inside its window.

2. Video Lesson
Chapter 3. Message types, inlets and outlets

In VVVV a message box is called an IOBox. There are several different kinds of message boxes, depending on their application. We distinguish number (Value), text (String), colour (Color), data (Node) and drop-down menu (Enumeration) boxes. Each IOBox parameter, i.e., the number of messages that they contain, the font they use, and much more, can be precisely adjusted in the Inspektor - as it is explained later in detail.

Figure 3.1. The Message menu

Let's discuss the most common types of messages, the numerical ones, first! In VVVV you can choose from a number of predefined variations if you double-right-click any free surface on the grey background. This will bring up the message menu, where you'll find a list containing Bang, Toggle, Integer, 2D Vector, 3D Vector and 4D Vector named functions which are all number-based messages.

A Bang megnyomáskor egy pillanatra felvillan, és kiküld egy 1-es üzenetet, majd visszatér 0-ra. Ezt például léptetéshez, törléshez és hasonlókhoz lehet használni.

The Bang and Toggle have similar functions. Both of them are a kind of switch which sends 0 by default. The Bang glows for a moment when pressed and sends the message 1 and then goes back to 0: this can be used for example to scroll, delete etc.

When the Toggle is pressed, the output switches to 1 from 0 and remains there, while pressing it again resets it to 0.

It is easy to tell them apart by looking at their edges. The Bang has rounded edges while the edges of the Toggle are square. It is worth bearing in mind that the messages in VVVV primarily act to right click and not to the left one, as usual. (Double left click changes the message box to keyboard input mode.)

Now, try the buttons! Double-right-click on the grey background and create a Toggle and a Bang. Under both of them (also with right click) create two simple number boxes. In order to see the change, the switches must be connected to the numbers. This can be done by connecting the outlets (the small grey squares found on the bottom of the objects) of the Bang and the Toggle to the inlets (on the top) of the number boxes. If you start switching the buttons now, the change will be visible.
The traditional number boxes which are easy to make with a double right click also have some predefined standard versions: these can be found in the message menu among the vector ending ones. These are mainly needed if an inlet needs more than one value; for example in the case of x-y-z coordinates. This can also be set manually in the Inspektor while even more numbers can be added.

The text box is primarily used for text input, because some nodes can only receive text-based messages. However, you can also write numbers, but you will not be able to connect them to any number-based inlet. The easiest way to try this is to choose the string from the message menu and put two simple number boxes next to it. If you click on the outlet of the string, you won't be able to connect it to the numbers. (Once you started a connection but changed your mind, just right-click on the background and the line will disappear.) If you click on the outlet of one of the number boxes, then you can see how the inlet of the other number box thickens, indicating that they can be connected, while the inlet of the string remains the same, showing that they cannot be connected.
The best way to try what we learned so far is for you to choose the text function with a double left click. The program offers all the nodes containing text. You need the very top one, Text (ex9). Hovering your mouse over the inlets of the box, you will see that there are many parameters to change, similarly to the ones found in the interface of text editors.

Figure 3.5. Text module

One of them is called Layer: [Supports: ex9 layer]. This means that the inlet is expecting an ex9 renderer, so to visualise the text module a render window is needed. To open one, you have to write renderer to the node menu and choose the top one called Renderer (ex9). Just click on the outlet of the text that will blink, indicating that it can be connected to the inlet of the Renderer. When you connect them, the text “vvvv” will appear in the system window. This is the default value of the Text object.

Figure 3.6. Hello Renderer
In order to edit it, click on the outlet of the String box and you will see which inlet of the Text you can connect it to, because the right one thickens. Right-click on the String and write something, for example “HELLO”. You could see earlier that amongst the inlets of the Text box there is one called size. If you right-click on this inlet and start dragging your mouse, you see the resolution of the text changing, and not its size. Write “300” after a simple right click and press enter.

**Figure 3.7. Resolutions 10 and 300**

In order to change the size of the text, you need to connect a transform box as well. Click on the node menu, type transform and choose the Transform 2d function. You can connect the outlet of the transform to the second inlet of the Text. If you hover over the inlets of the transform, you can see parameters like x-y position, scale, rotate etc. You will need the Scale x, Scale y functions. Make a number box and connect its outlet to both of the scales. The text disappears from the Renderer because the size of the scale is set to 0. Change it to 0.5 for instance.

**Figure 3.8. Scale X Y**
Translate x, y is responsible for the position of the text inside the transform node. To set the position of the text, connect the first two outlets of the Renderer (x and y) to the Translate x, y parameters.

**Figure 3.9. Moving text with mouse**

You can control the colours with the Color message type. Make two Color boxes, connect one of them to the Renderer and change its colour by right-clicking while dragging the mouse. Connect the other Color to the color inlet of the Text box to change the colour of the text.

**Figure 3.10. Hello in colour**

In order to control it with mouse position you need a box that transforms the numbers to colour. There are many kinds of nodes that do this, but use HSL (Color Join) for now. As the name implies, colours can be mixed with Hue, Saturation and Lightness parameters. If you connect its outlet to the inlet of the Color box which is connected the Text box, the text will change to white because the Lightness is set to 1 by default. So despite adjusting the Hue, the text will remain white. However, if you change the Lightness to 0.5, then you can get full colours. Next, you can make the colour depend on the position. All you need to do is to connect the x outlet of the Renderer box to the Hue inlet of the HSL (Color Join) box.

**Figure 3.11. Adjusting HSL with mouse position**
1. Video Lesson
Chapter 4. Komplexebb matematikai feladatok, fájlba írás és olvasás

First let’s look at the most often used basic operations: addition, subtraction, multiplication and division. Create one of each, as the picture shows below! The software recommends many types of basic operations in the node menu, but for now we only need the ones ending with value. Their usage is pretty straightforward. The Spectral Value node adds or multiplies vectors comprised of multiple numbers.

Figure 4.1. Basic mathematical operations

There are two ways to solve a formula. You can either put it together yourself using the operations it contains, or you can use the Expr module to define whole formulas.

Figure 4.2. Complex mathematical operations
Since it is often necessary to let several things interact for starting an operation or triggering and inlet, familiarity with logic gates is a must for the VVVV user. Logic gates are as follows: the outlet of the OR module change to 1 immediately if it receives a 1 on any of its inlets - pretty useful when an inlet (of a module reset for example) gets impulses from many places to implement an action. The outlet of AND only changes to 1 when all its inlets hold 1. The XOR is the exact opposite, as it only sends 1 if its inlets are all different. The NOT module inverts its inlet, meaning if it receives 1 it will output 0.

Figure 4.3. Using logic gates
If the result is not in the proper range, it can be adjusted with further multiplication, division or adding. The Map module makes this a whole lot easier. The range of the incoming data can be adjusted on the 2nd and the 3rd inlet of Map while the range of the outgoing data is adjusted on the 4th and 5th inlet.

**Figure 4.4. Resize**

![Diagram of Resize](image1)

In order to change between two values or to choose between two inlets of origin a Switch is needed. It is really easy to multiply the inlets and outlets of the gates with the help of the Inspektor. This can be seen in Figure 4.6.

**Figure 4.5. Inlet and outlet switches**

![Diagram of Inlet and Outlet Switches](image2)

**Figure 4.6. Switches with multiple inlets and outlets**

![Diagram of Multiple Inlets and Outlets](image3)
The easiest way to fade between two inlets is with *InputMorph*. This can be achieved with ordinary modules as well, as you can see in Figure 4.7.

**Figure 4.7. The InputMorph**
If your inlet changes but the transition between states is not good enough, you can smoothen it with filters like LinearFilter, Damper, DeNiro, Oscillator or Decay. LinearFilter creates a linear transition between two states with a given time. Damper does the same with a quick attack and a slow decay (similarly to a release of a spring) while the DeNiro has a slow attack and a short decay (just like a car). The Oscillator works similarly, but it has a release time, too. Decay is a kind of LinearFilter as well, but its attack and decay can be adjusted via its 2\textsuperscript{nd} and 3\textsuperscript{rd} outlet. Many more parameters can be adjusted with the ADSR envelope generator, which may be familiar from synthesisers.

**Figure 4.8. Filters**
Figure 4.9. Decay and ADSR
If your inlet consists of multiple values and you would like to smoothen it, then you need to use the B-Spline module. It is expecting a LinearSpread on its first inlet. With this you can set how many numbers you want going out. This means that if your source is made of, for instance, 12 numbers, and LinearSpread smoothen it using 100 numbers, then it will mediate 100 values. The module’s second inlet is expecting the control inlet while the third one is used for setting the level of smoothness. If the latter is set to 0, then the outcome will be square, if it is set to 1 it stays the same, while if it is set to 3, the result will be soft.

Figure 4.10. B-Spline
We mentioned vectors before, but it is time to look at them in detail! Every message type can consist of multiple messages and a number of methods for their composition and interpretation. The main modules which have been available for a long time, are the Vector (joint), Vector (split), Cons, and Stallone. There are four different kinds of Vectors: 2d, 3d, 4d, and Spread. The first three have a fixed number of inlets and outlets while the Spread’s inlets and outlets are adjustable. The Vector module works in a way that if you have multiple values on your inlets, then the first one of each will make the first row, the second of each the second row, and so on. To display the values that are received in the first inlet in the first row, followed by the numbers of the second inlet, et cetera, you can use the Cons, Stallone module – a more modular module, since you can set the number of its inlets and outlets, and even the listing method in the Inspektor.

The Zip and Unzip modules appeared only recently. They are much more effective when you work with more than 25 numbers at a time, so you should use these when listing many numbers. The number of inputs and outputs of these modules can be specified using the Inspektor.

**Figure 4.11. Vector types**
Figure 4.12. Vector, Zip, Cons and Stallone
To make a list of numbers from, for example, zero to one hundred, use the \textit{I} module. It works similarly to \textit{LinearSpread}: if you write 0 to its first inlet and 1 to its second (\textit{Width}), then you will get a linear list between -0.5 and 0.5. You can achieve the same result but with random values if you use the \textit{RandomSpread}. You can set the level of randomness with its \textit{RandomSeed} inlet. The \textit{CircularSpread} creates numbers along a circle that have \textit{x} and \textit{y} values. The \textit{x} and \textit{y} parameters, setting the position of the centre of the circle, and other parameters for the width and height, as well as for the filling, can be adjusted.

\textbf{Figure 4.13. I}

\textbf{Figure 4.14. LinearSpread and RandomSpread}
Komplexebb matematikai feladatok, fájlból írás és olvasás

Figure 4.15. CircularSpread
We will discuss reading and writing files in the last lesson.

1. Video Lesson
Chapter 5. Rendering types

Let’s look at the first render module that is in the node menu which is also the most often used, the Renderer EX9! All two- and three-dimensional modules can be displayed with this.

In order to try the module you need to make a quad DX9 first, and connect it to the render module. A white square will appear in the render module that needs to be resized for full screen. You can do that by changing scale x and scale y to 2 on the transform inlet. Quad is a simple slab that is capable of displaying pictures and videos. You can load images with the Filetexture module. If you connect the outlet of the module to the corresponding inlet, the clean slab will change to black (its default state) until the desired picture is selected on the first inlet of the Filetexture module.

If you know in advance that you will always need a full screen display that you will not necessarily want to resize, then the best choice is the FullscreenQuad module, since it renders full screen by default. Anyway, if you want to resize it later, it is possible with the transform inlet, just like in the case of the Quad.

Figure 5.1. Full Screen Quad Render

Both modules have color inlets as well, for setting the colour and transparency of the image. The Texture Transform inlet is also worth noting which is used for resizing textures. This can be interesting when your texture is smaller than the displayed image. In this case the software will offer functions that allow you to repeat the texture, stretch the last row of pixels etc. In the case of the Quad, these functions are available through connecting the Address to the Sampler State function. In the case of the FullscreenQuad module the tab before the last, called Texture Address Mode is there to do the same.

These are the basic functions of the render modules. We will talk about them in more detail in the following lessons.

Figure 5.2. Quad Render
The next module in line is the **SVG**, i.e. *Scalable Vector Graphics* which is an xml-based vector display. It was made for the web, and also contains a *Quad*, where transforming and colouring operates in a similar manner as the *EX9* type of *Quad*, but this one renders pixel-based pictures a little harder. If you search for modules related to SVG you would see that there are hardly any available. This program is mainly used for generating simple two-dimensional graphics and it has a great disadvantage: high CPU usage. You should not dig any deeper unless you need to save your patch as XML.

**Figure 5.3. SVG Render**
Another one is the TTY render that is used for displaying messages. It is mainly used for displaying error messages, but it can also be programmed with a TTY type Write module.

**Figure 5.4. TTY Render**

The GDI i.e. Graphic Device Interface render is also a two-dimensional renderer for line-based graphics. The related modules are: Point, Text, RoundRect, Circle, Line and the BézierLine. Unlike the others, this has no transform inlet, but all its parameters can be set inside the module, such as the x and y position, width, height, colour etc. In some rare modules, the line thickness can be adjusted as well.

**Figure 5.5. GDI Render**

Finally let’s talk about the SWF renderers that are used for displaying Flash images. There are two options for displaying these file types.

The first one is the Flash renderer that is suitable for displaying Flash files without external modules. You can load the file in this render module, set the screen size, loop points, as well as the start and end key frames.

**Figure 5.6. Flash Render**
The other option is the *Flash EX9* for which you must use an EX9 render module. It works like a *Quad* but it plays *SWF* files. For external control, apart from the variables mentioned above, you can use mouse position or mouse press which can be very useful.

**Figure 5.7. Flash EX9 Render**

![Flash EX9 Render](image)

The next module is the *HTML* string. You can use it to render *HTML* codes. It also lets you embed videos.

**Figure 5.8. HTML Render String**

![HTML Render String](image)

The last module in this chapter is the *HTML* render.

*HTML* render lets you display websites and get images or text from a website.
Figure 5.9. HTML Render

Chrome integration is forthcoming in the next versions.

1. Video Lesson
Chapter 6. Layers

Start by creating two Ex9 Quads and a render window. Connect a 2d Transform to each Quad.

As you have already experienced when we discussed numbers, only one parameter can be connected to one inlet, unless you have joined them using a vector. Images work in a similar way. If you want to display two or more Quads in the same render window, you will need to use a Group. You can adjust the number of inlets in the Inspektor.

Create a Group and connect two Quads to it. Only one square will appear because the two Quads are on the top of each other. In order to place them next to each other, write 0.5 in the X position of one of the Transforms and 0.5 in the other one.

There are two types of texture loading modules: one of them is the FileTexture which is used for loading still images. This should be familiar from the previous lecture. The other one is called FileStream (DShow9) and used for loading videos, sound and midi.

Figure 6.1. The FileTexture

FileTexture has the following inputs: play, loop, start and end point, jump, cue, speed and file name. The following outputs are useful: audio out, midi out, video out, length and current position. In order to be able to connect the video output to a Quad, it has to be transformed with a VideoTexture module, because this outlet is a DirectShow type originally and you will need a TextureOut.

On loading and starting a video you might not see anything happening: this may be due to the fact that the end time is not set at its input. You can achieve this by connecting the Duration outlet to the EndTime. This will not take effect immediately because of the resulting feedback loop, which is not supported by the system. The signal has to be delayed by 1 frame using the FrameDelay module. You will use this module often because it has texture and text types as well. When this is done, you can connect the inlet and the outlet and the video begins to play.

Figure 6.2. The Filestream
The Filestream can handle quite a few formats, but it can take time until you find the right one. If you know that you don’t need audio and you don’t want to change the start and end points of the video either, then you should use the MediaPlayer module. It has built-in FrameDelay and VideoTexture, too.

There is already some rudimentary integration of the VideoLAN project’s multimedia player, the VLC Player. This will bring significant changes because VLC can play almost any kind of file type and a lot of player functions can be reached from it. However, in this lesson, we do not dwell on it because this module is still in a beta stage and does not work perfectly.


You can also use only one Quad for many layers. In this case you need to control it using more parameters and Group is not necessary.

Figure 6.3. Textured layers

The following example shows how you can choose a number of images from Quad’s texture with an HSL module.

To make a transition between the layers, make a Quad and connect it to a render window. After that, connect an HSL Color Join module to the color inlet of the Quad. The next step is to connect a LinearSpread to the HSL module’s Hue and Alpha inlets and set them 0.5! Thus the value will be between 0 and 1. Change the Spread Count to 10 on the last outlet, so you have 10 values. Do the same on the other LinearSpread!
Figure 6.4. HSL and LinearSpread

Now you see a white square in the render window. This means that the Lightness is on 1: change it to 0.5! You get a reddish colour which can be changed by connecting a LinearSpread module to the Alpha inlet and change its Phase inlet. Alternatively, connect an LFO slowed down to 3 seconds.

With some texture, we will be able to see our work more clearly.

To generate the texture, create a Text Ex9 Texture woei module and connect it to the texture input of the Quad! The text “VVVV” appears on the render. In order to get 10 different texture layers from this Text module we will need an i module and a SpellValue. The i module makes integers between two given values, while the SpellValue generates letters from these numbers. Connect the two and write 65 and 75 to the first to inlets of i. Choose the ASCII function in SpellValue’s drop-down menu.

Figure 6.5. Text texture

You may now see the letters from A to Z, because in the ASCII character table the ABC starts at number 65. If you connect this to the correct inlet of Text, then every letter will have different colour. Resizing is needed in order to see only two letters. Connect the LinearSpread that is connected to the Alpha inlet onto a Map and set the Destination Minimum to minus 5!

Figure 6.6. Text layers
The only difference between the following example and the previous one is that it is shifted in space and not with HSL. However we are working with a FileTexture and a FileStream compared to a single source of texture. Similarly to numbers, you will need a Cons module in order to add them. As we only have two textures, the Quad has to repeat it five times to get the required ten textures.

**Figure 6.7. Transform with LinearSpread**

The render doesn’t normally have a perspective: a Camera Softimage module is needed for this. When working with several layers, it is really important to set the Windowed Depthbuffer Format of the renderer in the Inspektor to D24X8, because otherwise it does not render the screen correctly.

**Figure 6.8. Layers in Perspective**

Our last example is going to be a little more difficult than the previous ones: in this one we are going to re-size objects with mouse-distance and distort them on button press.

As a first step we need to make a 25x25 Quad net with the help of the Cross module (without this the squares can only be put diagonally).

The Cross does nothing else but gets the positions of module’s first inlet and repeats it with the values of the second inlet, i.e. if on the first inlet the value is 5 and on the second one is 3, then we will get a 5x3 matrix.
After creating the Quad-net, you can determine which square is the closest to the mouse with the Points2Vector. Connect the x and y values of the Cross module to the first two inlets of the Quad module and connect the x and y values of the Mouse module to the third and the fourth one! By doing so, you get a list on the third outlet, called Length. This is the distance we are looking for! If you connect this to a Map and write 0.1900 to the Destination Minimum and 0.12000 to the Maximum, and finally, connect the outlet to the x and y values of your transform module, it will start following your mouse.

**Figure 6.9. Cross2d and a 20x20 Quad**

![Cross2d and a 20x20 Quad](image)

The next one is a very spectacular distortion: the Attractor 2d. Similarly to Points2Vector it also expects the position of the squares on its first and second inlet and the position of the mouse on the third and fourth.

The remaining inlets are the Strength, Power and Radius. With the latter you can set the level of distortion (set this to 1.4200) while the first two is responsible for its strength. Create an Or and wire it to the left and right mouse click. Connect a Map to the right click and type -2 to its minimum outlet and 2 to its maximum! Multiply its outlet with the outlet of Or and connect it to the Strength. Write 1.290 to the Power.

**Figure 6.10. The Points2Vector controlled Quads**
The effect is ready which you can adjust later when you get more familiar with *Attractor 2d*’s behaviour.

**Figure 6.11. The Final Render**

1. **Video Lesson**
Chapter 7. Rendering 3D Objects

In this lesson we are going to have a look at three-dimensional renderers and their editing possibilities. Let’s start with the most often used 3D file loaders and the most important renderers!

You may know Scene, the Assimp file loader from other 3D software. This is an open source 3D object loader which can handle many kinds of 3D formats like .dae, .blend, .3Ds, .ase, .obj, .stl and so on. Hopefully, common 3D modelling features like bones and animation will be integrated shortly.

After loading a file, you will definitely need a Mesh Assimp module that converts the output of Scene to Mesh. There are many other modules for Scene like the Camera which reads the camera position saved in the object, Material which is responsible for colouring and texture data, and the Node which gives the transformation data of the object.

It is important to know that the Scene loader permanently optimizes the vertexes. This can cause problems when you save many shapes with the same vertex and would like to morph between them. This does not work as expected because the vertex numbers are not the same after loading.

Figure 7.1. A Scene file loader

The other file loader is the ColladaFile. It can only read .dae files but it has many functions. A lot of things that are missing from the previous loader are already integrated into this one: we can use bones, or movement that was saved with external software, as you can see in the example file. The skeleton can be controlled with a Kinect or any other method, but this is never easy because every technique requires different transformations and it is hard to get used to them.

The Mesh module is also needed with the ColladaFile module, just like with the previous loader. A normal Transform, the Skinning transform, colouring and texture routing have also been integrated into this module. It also has a Time inlet which is necessary when playing animations.

It is enough to connect a simple three-dimensional renderer to the Mesh module, the PhongDirection for example, but this can’t handle skeleton functions on its own. You need to use the Skinning module for that which has a Skinning transform inlet. In order to get the object’s original position you must connect the ordinary Transform outlet to the renderer’s transform inlet! It is even recommended to put a 3d Transform between the two, so you can easily change the original transformation.
The third and simplest 3D loader is the XFile which uses .x files (the DirectX 3D format). This format has neither an extra module, nor a Transform outlet. However, it is really easy to edit and create new files using a custom program called Writer.

Figure 7.4. 3d Render
Create a DrawFixed module, a renderer and a Camera Softimage and connect them! Change the Initial Distance of the Camera module to 1. (Don’t forget to set the DeptBuffer to D24X8, otherwise you will get an interesting sight.) In order to make a Mesh net choose the Grid EX9.Geometry module and connect it with your DrawFix. The Grid has two Resolution inlets (X and Y). Change them both to 20, for instance! When you create a Fill module and connect it to your DrawFix and select the WireFrame function on its right inlet, its resolution becomes visible.

**Figure 7.5. DepthBuffer setup**

![DepthBuffer setup](image)

**Figure 7.6. A Grid**

![A Grid](image)
Every Mesh can be divided into 3D positions. Create a Mesh Geometry Split module alongside the VertexBuffer Geometry Split module and connect all three! The Mesh module has two outlets: VertexBuffer and Indices. VertexBuffer normally has a Position XYZ and a Normal XYZ. You have to set the Texture Coordinate 0 to 2d, because texture coordinates will be needed for the upcoming exercises.

Create the Mesh Joint and the VertexBuffer Joint after that, which are the opposite of the previous example. You will mostly need the Position XYZ, because this is what has to be modified later on, so the rest (Normal XYZ and texture coordinates) can be connected between the VertexBuffers just like the indexes of the Meshes. You can split and merge the positions using the Vector 3d Split and the Joint. You can connect the X and Y values so the Mesh Joint can already be connected to DrawFixed.

Our aim now is to modify the Z positions. We could do this with Random spread but instead we are going to use an image for this. For this purpose we are going to get the necessary colour information with the Pipet Texture Simple module. This tool has three inlets: a texture where the source image must be loaded, and X and Y coordinates. Use the Plasma module for texture. This generates some plasma on which saturation, size and density can be set manually (while at it, bring the saturation down to 0). Connect this with Pipet and also connect the X and Y values of the Vector Split module to the other two inlets. This will query the colour specified by the X and Y coordinates of the Grid module from the texture.

Figure 7.7. Grid transformed with Pipet
When this is all done, you will get 400 colours on the outlet of Pipet, i.e. 20x20 values. If you connect this to a Color Split you will get the luminosity. You can get the Z coordinates with this. (It is recommended to resize the given data with Map, because there will be a huge difference between 0 and 1, so type in 0.2500 on the outgoing minimum and 0 to the outgoing maximum. The outgoing stream has to be inverted because otherwise the darker values will rise and the lighter ones will descend, which is the opposite of how waves look.)

The easiest way to make the plasma move is by adding the first and last outlet of an LFO while changing its timing to 10 seconds. Connecting this to a 2d Vector and then into the Offset inlet of the Plasma module will start the motion instantly.

1. Video Lesson
Chapter 8. Transformation and grouping

In this part we are going to get familiar with different transformations, grouping and basic physics.

You should have noticed that you can connect all the transforms to each other. You can define the position of an object with the first layer of transform and modify the whole patch with a second layer. This could be a solution in many situations but there are scenarios when we need to multiply our objects or modify only a part of them, in which cases this approach may not work.

Figure 8.1. Normal Transform

Make a Render with a Group and connect an AxisAndGrid to the first inlet of the Group and a PhongDirection to the second one. Connect a Box EX9Geometry to its inlet and a 3d Transform to the transform inlet. Create a Camera (Transform Softimage) and connect it to the View and Projection inlet of the Render. Make 10 random X and Y positions with RandomSpread so your boxes are visible in the 3D space. Zoom in or change the width of Random if needed, so the objects are clearly distinguishable. If you connect a Transform 3d and start rotating your elements, you will see that it moves all of them. Instead of Transform you can use Rotate, Scale, UniformScale, Translate modules, too. With these you will only be able to use the single transformation that you need.

Figure 8.2. Parental Transform
You can move the original Transform all at once in another way, too. For this, you need to connect a `* (Transform)` between the Phong and the original Transform and multiply its second inlet with another Transform. You can also rotate the individual objects one by one with multiplication. You can get all the transformations from the original Transform with a Decompose Vector node. If you want to rotate your objects around their own axis, then you simply need to connect the X Y Z coordinates to the X Y Z inlets of the transform and to Center X Y Z. This way all the objects will get their own origin. Once this is done, you can control the rotations with SetSlice or RandomSpread.

**Figure 8.3. Transform Multiplication**
Figure 8.4. Decompose vector

With the GetSlice node, we can select different items of a list containing multiple Transforms. It has three inlets. The first one is the Transform; the second one is the Bin Size, with which we can set the number of selections, and finally the Index which corresponds to the designated items. If we connect this to a Decompose module we get the precise data of the chosen transforms.

Figure 8.5. GetSlice Transform
We can give brand-new positions to our objects one by one if we connect a SetSlice (Transform) between the original Transform and Phong, because this module has two Transform inlets. The first one is the original and the second one is the new position. There is also a Bin Size and an Index inlet with which we can set to which elements the transformations are applied.

**Figure 8.6. SetSlice Transform**

![SetSlice Transform](image)

**Figure 8.7. Apply Transform**

![Apply Transform](image)
Another way to position elements is with \textit{ApplyTransform}. To move our objects, the $X Y Z$ coordinates have to be connected directly to the object and to the first inlet of a $\text{Transform}$. We can connect the $X Y$ and $Z$ outlets to a normal $\text{Transform 3d}$. In this case the module calculates the rotation and moves the object in space, but does not rotate it at the same (as before). We can use this module for repositioning or rotating $\text{Vertex}$ positions, for example.

**Figure 8.8. Spread Transform Multiplication**

Sometimes you need to multiply a $\text{Transform}$ containing many elements (this does not work with another layer of transform). You need to make a $\text{Node}$ message for this. You have to connect $\text{Transform}$, used for multiplication, to its left inlet and the $\text{SpreadCount}$ of the original to its right inlet which is the $\text{Bin Size}$. Finally we have to connect it to the previously used multiplication. As a result, it will multiply the objects.

Let's talk a little physics now!

There are three different kinds of physics available in VVVV:

- \texttt{Box2d} \url{http://en.wikipedia.org/wiki/Box2D},
- \texttt{ODE} \url{http://ode-wiki.org/wiki/index.php?title=Main_Page}
- \texttt{Bullet} \url{http://en.wikipedia.org/wiki/Bullet_(software)}

You have probably met with the engine of \texttt{Box2d} in various computer or mobile games (it is usually used in motorcycle and tank games). It has been integrated to VVVV, too. There is an external object called \texttt{box2d playground} made by TGD with superb user interface design. You can find it on the Contribution page here:

\url{http://vvvv.org/contribution/box2d-playground}

Először vegyük a Box2d-t, ezzel az Engine-vel már találkoztál különféle PC-s, illetve telefonsos játékokban (többen motoros és tankos játékokban használták). A VVVV- ben is integrálva lett, és ebből egy nagyon jól használható kezelőfelülettel külön letölthető kiegészítőt készített TGD, mégpedig a box2d playground-ot. Ezt megtalálhatoz a Contribution oldalán: \url{http://vvvv.org/contribution/box2d-playground}.

**Figure 8.9. BOX2D Interface**
When starting the patch you can see that the user interface appears in a DirectX render. You can make static and dynamic objects like squares and triangles with it. Pre-set elements are also available, like a grid or boxes lined up like text. Their settings can be modified in the CreateBoxTypo or the CreateBoxArray modules within the CreateObjects library. It is also possible to load and save sets of elements. If you click on the Load button, a default “skeleton” loads. On this example you can see the joining points of the specific elements. The fastest way to create this is by dragging elements on top of each other in Halt World mode and right-clicking the point where you want to put the joining points.

**Figure 8.10. BOX2D file load**
The Bullet physics comes with the addonpack, and after copying it, you can find examples in the \addonpack\girlpower\BulletPhysics\ library. Bullet is a really complex 3d physics engine that is used by many well-known companies in physical simulations. If you open the example patches, you see that they work a little differently than usual. They are not connected, not even with a Send and Receive module. You can make static or dynamic objects with CreateRigidBody and CreateSoftBody in the 3D space. Furthermore, these objects can make simple geometric forms (it has a built-in Bullet module) or you can load your own objects, because the aBVH module can convert the Mesh file to the correct format.

**Figure 8.11. Bullet in OpenCL**

![Bullet in OpenCL](image)

**Figure 8.12. Bullet BVH Mesh**

![Bullet BVH Mesh](image)
When creating physics with SoftWorld, you get the information required to control the modules in the final Render. You get the mesh forms from the SoftWorld with RigidBody. The transformations that you can connect to a PhongDirection are available from the GetRigidBodyDetails.

**Figure 8.13. Soft rigid**

1. **Video Lesson**
Chapter 9. Effects and Plug-ins

In this tutorial we are going to get familiar with the different integrated modules. If you have already installed the Add-on Pack, your version of VVVV already has plenty of new plug-ins and effects beyond the basic functions.

Figure 9.1. Effect list

Typing “FX” pulls up a list of the available effects. Most of them have been developed the UNC Company. Many effects, like holes, blurs, distortions, colour corrections should be familiar from popular image and video editing suits.

As a first step, we are going to make a drawing patch made from external plug-ins. You will be able to use this for masking and drawing in other patches. The patch itself is built from the Brush node and the Blend node which are responsible for the addition of images and the Buffer node which is a tool for temporary storage.

Make a Render first and connect a Group to it. Connect a FullScreenQuad to the first inlet of the Group, and Cursor to the second one. Create a MouseState Joint and connect it to the Cursor. Connect the MouseState X and Y and the left mouse button to the identical outlet of the Render, so that the Cursor will follow your mouse movement. You don’t see that happening yet, because there is nothing connected to the texture inlet of the FullScreenQuad, so the whole screen is white.

Figure 9.2. Brush plug-in
To resolve this, connect a newly made Brush. If you have a look at its inlets you can see that it has an X and Y position, for brush size and brush intensity. Connect the X Y positions of the Render module to the correct inlets of the Brush and set its size to 0.1500. Create a Blend EX9.Texture Mixer and connect the Brush to its first inlet.

The image to be added the second inlet must be connected to itself through a FrameDelay EX9.Texture. If you connect it to the FullscreenQuad, you will see nothing, so set the Blend Mode to Screen and the Opacity to 0 and then back to 1. Now you can draw continuously.

It might be strange what you see now, because this module draws a little cloud adding a narrow black outline to the brush. This can be corrected in the Brush module by setting the alpha of the Background Color to 0.

**Figure 9.3. Blend and Buffer**
Connect a Buffer EX9.Texture module between the Blend and the FrameDelay. Connect the left mouse button outlet of the Render through an Or to the Set inlet of the Buffer. In this way, if you connect the right mouse button to the Opacity inlet of the Blend through a NOT gate, the program clears your picture on a right mouse click and draws and stores it on pressing the left mouse button.

It is important to get a totally black picture after clearing the canvas, which implies that you have to make the Brush disappear for a moment. To do this, connect the left mouse button to its Show Brush inlet. Thus the program will only show the brush when the left mouse button is held down. To invert the colour of the mouse pointer as compared to the canvas, make a PipetSimple and connect the Blend outlet to the Pipet.

**Figure 9.4. Cursor and Pipet**
This module is expecting the mouse position on its other two inlets – but not between 1 and +1 but from -0.5 to +0.5, so the X and Y outlet of the Render module must be divided by two and connected to Pipet.

Now we are going to learn how to use additional effects in a Render patch.

**Figure 9.6. DX9 Texture**
Every Render module has a texture outlet which you can connect to a newer Quad or maybe to a Render. Normally the effects are all built the same way. If you right-click on an effect node, you can see that it is built similarly to the patches we saw before. The difference is that they use Shaders instead of Quads or other visualizers and the outlet is for a DX9 Texture node.

**Figure 9.7. Hidden Render without resolution setting**

When using DX9 it is important to set the Render resolution because if not, the program will use the actual window size for rendering. If you set it, then the window size will not matter anymore (even if you make it disappear with ALT+3) – the program will render in the size that you have set. This parameter can also be specified separately in the Inspektor.

**Figure 9.8. Hidden Render with resolution setting**
Figure 9.9. Resolution setup in DX9Texture

Figure 9.10. Alpha texture out

Figure 9.11. Sample effect
Thanks to the developers and users of VVVV there are many plug-ins and accessories available on the http://vvvv.org/contributions website. You can find many plug-ins, projects, device managers and various Shaders in alpha or beta state.

It is worth visiting the site frequently because a lot of people upload useful external patches addressing a wide range of problems. These are usually well documented, right there on the site or separately in the help patch, but if you find a bug or run into a problem, it is good to know that there may be more information on the site related to the patch. The forums are also very useful for finding solutions to specific problems.

Over time, most users collect a whole library of plug-ins that they use repeatedly. If you had enough of always having to copy your favourite plug-ins to the root library of your new patch, do the following.

**Figure 9.12. Contribution site**
Launch VVVV and press \textit{ALT}+R right after start-up or pull-up the menu with the middle mouse button and choose the \textit{Show Root} option. The \textit{Root.v4p} window appears. Note that all the windows that are open at this moment are copied to the root patch. In this case it is called \textit{0.v4p}. This has to be deleted, otherwise it will launch on every start-up. Of course this might be just what you want: to launch a patch on every start-up. In that case, copy the patch you want to launch on start-up to \textit{Root.v4p}.

\textbf{Figure 9.13. Root patch}
In order to load the plug-ins automatically, you have to find the String list in the Root window which is connected to the right inlet of Cons. You can choose there with a right click which library to load on start-up. You can save and quit after selecting them. However, it is not worth loading too many libraries at start-up. You should add them one by one when making a complex patch. If you copy a new plug-in to a pre-loaded library, it will load automatically while the program is running, so it is not necessary to restart it.

1. Video Lesson
Chapter 10. Audio Analysis

In this lesson we are going to get familiar with sound control.

Figure 10.1. AudioAnalysis

![Diagram showing AudioAnalysis module]

The easiest way to control sound is by using the AudioAnalysis module. This is a ready-made detection tool based on FFT (Fast Fourier Transform) which has many modules. This object makes a list of 256 numbers that correspond to different properties of the sound. AudioAnalysis has only one outlet and sends all the data through that in the following order: 0-3 low, medium and high frequencies per channel, 4-7 BeatDetect, 8-11 Decay changes in the former, 12-17 EnergyOut, 18-274 the whole FFT spectrum.

Figure 10.2. Recording device

![Diagram showing recording device settings]

Figure 10.3. Windows audio recording settings
In this module you can choose the sound source, its gain and stereo or mono mode. Not only physical audio input can be used with this module. In Windows 7 you can set the internal sound to be the input. This option is not exposed by default in Windows, but if you click on the background of the tab, and choose to show switched-off inlets, then it becomes visible. Now if you play a sound in another software, VVVV will also receive it.

If you only want to use only a part of the AudioAnalysis, then you have two choices. You can use the SplitAudioAnalysis which splits the signal into groups. Alternatively, you can use GetSlice or GetSpread, where you can set exactly what you want to use from the outgoing data.

Let’s make a patch based on FFT!

Make an AudioIn (Dshow9) module, switch it on and set the sampling to 22050 Hz, 512 Byte and 8 bit mono. Next, make a FFT (Dshow9) module and connect it to the audio input. It is good practice to set the Spread Count of the FFT to 64, 128 or 25. If you play some music or sound now, you will see how their values change between 0 and 1 on the first two outlets. Now make a Render with a Group, connect a Quad to it and a Transform 2d to that. Connect a Linear Spread below all those and set its width to 2. It is worth connecting the Spread Count to the same values the FFT, 128 for example.

**Figure 10.4. AudioIn and FFT**
If you want to change the sampling resolution, the width of the squares will change dynamically. The next step is setting the height. For the height of the Quad to start from -1 and reach +1, the FFT has to shift its Y centre and its Y position at the same time. Set the Y position to -1 and the Center Y to -0.5. If you connect the FFT to the Y inlet of the Scale module, it becomes visible. If you connect a Min module to the inlet, you can set the maximum height to 2, for example. Smooth the incoming signal with InputMorph. Connect the FFT to Input 1 and connect the outlet to the second inlet through a FrameDelay. This way you set the level of smoothing between 0 and 1. If you divide this value by 2, connect it to an InputMorph Color and set it to fade between two colours, then your colours will change, too.

The incoming sound can be drawn, together with a signal base. In order to implement such a configuration, make a Line EX9 and connect it to a Group. Normally, this module is expecting data between two positions, but if you choose Single Input, it will only be waiting for parameters through Vertices XYZ. Connect a Vector 3d into this inlet and connect the previously used Linear Spread to its X input. Connect a Map to its Y position which smoothens the FFT between -1 and +1. If you want to use a delayed line, as customary in music players, then connect a Decay between the Vector and the Map and set it according to your needs.

Now we are going to see how to select a segment of the data and calculate its average value so we only get data from only a specific part of the FFT.

**Figure 10.5. TagPoint plug-in**

First we have to work out how to highlight depending on mouse position. There is a designated module for that: the TagPoint. The positions of the points have to be connected to the first inlet, i.e. the LinearSpread through a Vector 3d, while leaving the 2nd and the 3rd inlet on 0. Its second inlet is the size of the Quads, i.e. the 3d Vector whose X value is 1.5 which was previously divided by the SpreadCount. Its Y value is 2, because this is the maximum Quad size. Its Z value is 0. The module is expecting the mouse position on its 3rd inlet. Unfortunately, we cannot simply connect the mouse to it, because we will have to delay the click with 1 frame and we cannot use the right click either, because we will need it for clearing the selection.
Create a Mouse System Window and a MouseState Joint and connect their inlets: X coordinate to X and Y coordinate to Y and left click to left click, through a FrameDelay. Now you can connect all those to the TagPoint. Connect the left and right mouse button to an Or and connect that to Reset Pin. Finally, connect a NIL module to the Reset To inlet. You need this to make the program not selecting anything when deleting the selection since the NIL module sends nothing as data. This has a reverse module as well, the AvoidNIL which sends a pre-set number when receiving nothing. Finally, if you connect a Sample And Hold (S+H) to the outlet and only open the gate when there is a click – i.e. by connecting an Or, then you are done with the selection implementation.

**Figure 10.6. Mouse and MouseState for TagPoint**

Make a SetSlice Color and connect it between the aforementioned Quad and InputMorph. Set a completely different colour and connect the S+H to its Index pin which is the last one. Now if you select something with your mouse your columns will change colour. The next thing to do is making the selection greyer, so that it is more visible. Make a Quad and a Transform 2d and connect them to the Group. In order to get the position of the Quads you need to connect the index input of GetSlice to the S+H, and its input to the LinearSpread. To get the average value of these, we use the Bounds module. If you connect this, you will get the exact centre, width, minimum and maximum values of the incoming data. Connect the centre to the X, the width to the Scale X and set the Scale Y to 2.

Again, the average value of the movement of the selected values is calculated with the Bounds module. Copy the previously made Quad Bounds part and connect it to the Group. Connect the centre and width values of the previous Bounds to the corresponding pin of the new Transform. Connect the other Bounds to the Y inlet through a Map, which is scaled between -1 and +1. Finally, connect the FFT of its GetSlice Input and resize the Scale Y to 0.1.

Now you can control anything with this Bounds module on the finished module or smoothen it with an InputMorph regardless of its previous value.

**Figure 10.7. FFT patch**
1. Video Lesson
Chapter 11. Video and tracking

In this lesson we are going to learn a few things about tracking and cameras.

There are many ways to use camera input in VVVV. The simplest one is the integrated VideoIn which is a Directshow compatible module. Unfortunately, not every video camera is compatible with Windows; – it is worth checking if the camera is compatible with the WDM specification.

Figure 11.1. VideoIn DirectShow

In the VideoIn module you can set the frame rate, the resolution and the format (MJPg, UYV2, UYVY, RGB8, RGB24, RGB32 are implemented). Even more parameters can be set with the Inspektor, like Brightness, Contrast, Exposure, Gain etc., but for these to work the camera itself must support these features. If you click on the Property Page tab in the menu linked to the camera driver, a window will appear. On the last two outlets you can see what kind of resolution and mode is available, as well as the current setting.

Similarly to Filestream, this module also needs a VideoTexture, because you can only connect it to a Quad or a TextureFX through that.

Figure 11.2. VideoIn Property page
Another, less common camera module is for the *uEye*. This camera is manufactured by *IDS* and it is used in industrial applications. It is a high-end camera offering a wide range of settings as well as a crystal clear picture. *Point Grey*, *Basler* and *Imaging Source* produce equally high quality cameras. Of all these, Point Grey is the most reasonably priced. It is worth investing in one of these, since a noise-free image and a good selection of parameters can help a lot with detection and tracking.

**Figure 11.3. VideoIn Ueye**

The *OpenCv VideoIn* camera input is not integrated in the program. Therefore, it has to be downloaded from the Contribution site: [http://vvvv.org/contribution/vvvv.packs.image](http://vvvv.org/contribution/vvvv.packs.image).

**Figure 11.4. VideoIn OpenCV DirectShow**
Figure 11.5. VideoIn OpenCV Ueye Property page
If you search for the VideoIn module after installing OpenCV, you will see three new module variations: CLEye, OpenCv, and OpenCv DirectShow. The CLEye is used for PSP Eye cameras. However, we will need the DirectShow version. Similarly to the original DirectShow VideoIn, you can also set its parameters in the pop-up window, unless you use the CaptureProperty (DirectShow) which you can configure elsewhere. It is worth having a look at the Help files because they introduce a lot of new functions like projector calibration or StructuredLight.

Figure 11.6. IDS Ueye
Figure 11.7. Imaging Source

Figure 11.8. Basler cameras
Figure 11.9. Point Grey Flea

Figure 11.10. Point Grey 3d cameras
The last one is the *Kinect/Primesense*. There are two ways of managing the *Kinect* camera. The first one is using the *Microsoft Kinect SDK* and its associated modules. The other one is the plug-in made by *OpenNi*. For the latter, you need to install *OpenNi, Nite* and *Driver*. Depending on your camera, there are two Driver variations to choose from. There is one for *Kinect* and another one for *Primesense* which is produced by *Asus* under the *Xtion* and *Xtion Pro* product names (only the pro version gives colour picture). The great advantages of the *Xtion* cameras are their size. They are tiny and can run off a USB port. On the other hand, they lack servos for positioning – but that is only used in special cases anyway, so it should not be a problem for most applications.

**Figure 11.11. Asus Xtion, Xtion Pro**
Every *Kinect* and *Primesense* driver and installer can be found in the library. Keep in mind that you cannot use the *Microsoft Driver* and the *OpenNi Driver* at the same time. You can always switch between them in the device manager. However, it is good to decide which plug-in you want to use in advance, because their incorrect installation can cause a lot of problems. For VVVV to recognize the device you have to install the correct *OpenNi, Nite* and *Driver*.

**Figure 11.12. Microsoft Kinect**

Once you have connected your camera to VVVV we can have a look at the tracking features.

The first one is *Contour* detection. Open the *Contour.v4p* file and press play. The image on the left side is a cleaned-up version of the incoming picture which can be interpreted by the *Contour* module. On the video view on the top you can see that the picture is very noisy and there is additional light caused by the infra LEDs. You can clean that up by subtracting and empty image (where nothing can be seen) for the feed with a *Buffer*. This is a basic level background subtraction. If the image is not good enough, you can amplify it with *Feedback* and reduce the noise with *Blur*. After that you only need to adjust brightness, contrast and set the saturation to 0.

The *Contour* module only understands *DirectShow* type videos coming from the *VideoIn* module, so the *DX9 Texture* has to be converted to this format with *AsVideo*. You can set the threshold and noise reduction after the conversion. On the outlets you get the positions needed for drawing the contours of the found points; the centre, width, direction and address parameters. You can use this method for multitouch operations or even drawing the contours of people.

**Figure 11.13. Contour Tracking**
The other possibility is using the Tracker module made by UNC. This can track colours and you can see it working in the Color.v4p file. The Tracker module has a relatively small amount of options to choose from. You can set the colour you want to track, the threshold and the smoothness. It does not recognize more than one colour and it is good practice to use only a single colour mark, because the program will put the point in the absolute centre of the marks.

**Figure 11.14. Color Tracking**

The third method for tracking is implemented in Primesense. Both developers made similar tracking modules, such as RGB, User/Player, Depth or Skeleton. Only OpenNi has Hand and Gesture tracking, and only the Microsoft module has Face recognition. Even so, they both need the Kinect module to work properly. It is possible to connect RGB if you want to see the camera view. However, this does not work with the Xtion because there is no camera image. The module names are pretty self-explanatory: the Depth module is for depth sensing and Skeleton is for skeleton tracking. You can set the depth range with a Lumakey when using Depth. The same applies to User as well, but here you get the users divided into colours.
Figure 11.15. Kinect Depth

The *OpenNi Skeleton* produces the *X Y Z* positions of the joints chosen from the list. It is often enough to concentrate only on the head and the arms, but it is also possible to get all the data.

Each library includes an example patch.

Figure 11.16. Kinect user detection

1. Video Lesson
Chapter 12. External controllers

In this lecture we are going to learn a few things about external controllers.

There are numerous ways to control VVVV with external devices. You can use a Midi controller, a Wacom Tablet, Wiimote, Joystick, Android phone or an iPhone.

Let’s see the Midi controllers first!

**Figure 12.1. Behringer BCR Midi interface**

Many companies manufacture Midi controllers, such as Akai, Behringer, Korg, Alesis or Doepfer. Fortunately, there are also many pre-made VVVV patches on the Contribution site which can be used with these products. The most popular ones are the Akai, as well as the Behringer BCR series, and also the Korg Nano controller.

We are going to have a detailed look at the Korg Nano Kontrol. This is a relatively cheap device which costs as little as 45 EUR. It is not too precise, but it is easy to carry around and it has many channels. It comes in three different versions: Kontrol has 9 sliders, 9 buttons and 9 switches. The Pad version has 16 pads and a touch surface, while the Key version has 25 keys.

**Figure 12.2. Korg Nano series**
You can configure which Midi device you want to use on which channel and with which controller in the MidiController module. Once this is done, you can set up the Controller with GetSlice. In more complex controllers you can do a special setup. If it has encoder type (endless) potentiometers, then you can set them to work linearly or logarithmically, and you can also specify their values. You can get familiar with the MidiController module in the attached KorgNanoKontrol.v4p file.

Figure 12.3. Wacom tablet

The next possible control method is using a Wacom Tablet in conjunction with the module called Tablet. There are many projects on the Internet that use this controller. It is good practice to build your own control surface for the tablet. You can do this by assigning different sections of the pen’s precise X and Y coordinates to different functions. This can be a simple slider, an X Y slider, a button or even a switch. If you have an older version of the Wacom tablet, you can print out the outlines and put it under the protective film which comes with the device. On newer versions, you can just simply use stickers on the surface to mark the different control areas.
You can use the buttons, the pressure level, the eraser and the angle as controllers. (Note that only the more expensive tablets like Intuos and similar ones support angle output.)

**Figure 12.4. Tablet Switches and Sliders**

Furthermore, the tablet control can be used for drawing and managing different effects. You can find two possible use cases amongst the example patches. One of them is a simple pressure sensitive drawing patch that you can find in the *Tube2d* library. The other one is the *Vux FluidSolverGPU* for tablets.

**Figure 12.6. Tablet Fluid patch**
A harmadik vezérlési lehetőség a Wiimote, ami a mindenki által ismert a Nintendo Wii eszközt használja. Ehhez a legtöbb hack-et Johnny Lee készítette, és nagy részük az eszköz elején található infrakra dését alapul, ami eredetileg arra szolgált, az eszköz térbeli helyzetének pozicionálását segítse a televízió tetején elhelyezett infraadó jelének befogásával. Ha viszont ehelyett egy gömbemre kötött 940 nm-es ledet irányítsuk az eszköz felé, akkor már is használható VVVV-s trackelésre is.

**Figure 12.7. Wiimote**

The third control option is to use a Wiimote which is a controller for the well-known Nintendo Wii. Johnny Lee made the most hacks for this device using the infrared receiver which is used for positioning in 3D space. If the LED is replaced with a 940nm one, the same device can be used for tracking in VVVV.

The Wii can only track up to 4 points, which is not a lot, but it can be very useful in certain occasions. It doesn’t even need any special installation. You only have to connect it through Bluetooth. When the computer asks, you just need to give it an empty passcode and it will install itself.
After setting it up, you have to find the *WiiMote* module. You can switch the *LEDs* on and off with the Led inlet and change them and vibrate the device with the Rumble inlet. You can even use the controller for remote control with the latter, if you like to experiment with things and willing to unsolder the *LEDs*. With the help of a transistor, you can also control other devices with it. Finally, the vibration is useful as a metronome or for giving a sign to an onstage actor.

**Figure 12.9. Wiimote plug-in**
Several WiiMotes can be used in VVVV by switching the Id inlet on in the Inspektor and specifying the target device.

Amongst the outlets there are navigation buttons, the A, B, 1, 2 buttons, the three system buttons, the accelerometer data, the angle and finally the infra positions. There are many other connection variations in the Help file to use in a variety of situations.

**Figure 12.10. Miscellaneous Wiimote settings**
Finally, we are going to have a look at the OSC-based Android controllers. There is only a beta version available, and it is only stable on ICS systems.

The client application is called Kontrolleur and you can download it from Google Play under the following link:

Figure 12.11. Kontrolleur interface

The current .apk is in the attached library. You can turn it on by pressing CTRL+K after making a numberbox, a slider or a number. The same numberbox will appear in the client application, so that the computer can be controlled by the phone or the phone by the computer. You can connect multiple numberboxes to the phone and if you name them in the Inspektor, the same names appear in the phone.
It is important to differentiate the incoming and outgoing ports, otherwise the data will get lost. If you are unaware of the exact IP address of your computer, just type 255 (the IP would be something like 192.168.1.255) in the last three fields. In this case the program will send the data to all the IP addresses on the network. Furthermore, the phone and additional parameters like multitouch positions, data from the magnetic sensor (used when docking) and orientation can be selected individually.

Figure 12.13. Kontrolleur setting
1. Video Lesson
Chapter 13. Arduino and some basic electronics

The manufacturers have been distributing IC-s that can be programmed to run software with all their inlets and outlets available to use. The problem with these is their price. They are far too expensive for the everyday user. The Arduino which was developed by the Italian Massimo Banzi, addresses this problem by simplifying the device and the language needed to program it. Thanks to the affordable pricing, it became very popular with media artists and hobby electrics enthusiasts.

The Arduino is essentially a simple computer that has many inlets and outlets, a relatively slow processor and low power consumption. Its most popular use cases are controlling motors, displays, lighting, speakers or even computers with switches, potentiometers (variable resistors), sensors or other devices. There is even a kind of Arduino especially built for fashion designers who work on intelligent clothes that can be sewn onto fabrics. Other applications include 3D printing, robotics or music.

There were only a few models available at the beginning: the normal sized Duemilanove, a Bluetooth version, and a Mini. The hardware design is Open Source, therefore anyone can develop it. Consequently, by now there are countless different versions available on the market. Furthermore, many companies distribute their own Arduino accessories. The device has a standard PIN layout (consistent through generations) so many different shields can be used across different versions. Even us, we can make one ourselves!

The Arduino can run in stand-alone mode, without continuous connection to a computer. The control program can be written to the memory of the Arduino using the Integrated Development Environment application, available from the official website (http://Arduino.cc/en/Main/Software). The IDE works with a simplified C programming language called AVR C. Again, the completed program can be uploaded through this development environment to the Arduino, where it is stored and run on the Atmel Atmega chip – a very primitive sort of CPU. Computer control is usually implemented through a serial port, since there is a USB-serial port translator on the Arduino which communicates directly with the Atmel Atmega IC. Serial port access is facilitated by specialised software tools. There are many options, but the easiest and most versatile one is Firmata (http://firmata.org/wiki/Main_Page). All node-based programming languages like Max-Msp, PureData or VVVV usually interface with this tool. You can easily configure your inlets and outlets in Firmata, according to what you want to use them for (PWM, Servo, Digital or Analogue). Its only disadvantage is that it can be really slow due to the many options and the constant communication. It can cause real problems, as when only a few messages in a second arrive at the computer.

In case you need speedy access to your messages, it is worth considering writing your own serial program that only does what you need. The examples which are included in the download can give a head start when writing your own programs, thanks to the many useful comments. When you get stuck, the forums can be enlightening, because someone else probably has already bumped into the same problem and hopefully it is already solved.

It is hard to get original Arduino products in Hungary – only the AvrDuino copy is available (http://avr.tavir.hu/modules.php?name=Content&pa=showpage&pid=80), so if you want to work with an original one, you have to order from abroad. The trustworthy European companies that distribute it are Farnell (http://farnell.com/Arduino), RS-Online (http://hu.rs-online.com/web/generalDisplay.html?id=Arduino) and Proto Pic (http://proto-pic.co.uk/categories/Arduino/). You can get your boards within a couple of days from RS-Online and Farnell, while it usually takes a week and a half for Proto Pic to ship their products. It is worth to keep delivery times in mind when you have a deadline.

Another option is buying your devices from eBay, but avoid cheap Chinese versions, because the PCB and the soldering is often really low quality, which results in a short life. The biggest manufacturer is Sparkfun (https://www.sparkfun.com/pages/Arduino_guide). They distribute their own Arduino, shields, sensors and other accessories. There are also many useful tutorials available on their site. Finally, you can even make your own Arduino, if you have an Atmel programmer. You can make one for as little as 2000 HUF if you are working with the right PCB manufacturer.

If you run the Arduino patch, you will see that you can choose on the first inlet whether the PIN should be an inlet or an outlet, the type on the second one and the serial port number on the last one (this can be checked in the Control Panel).
In the basic version of Arduino, the in- and outlets are situated in two rows. On the top row there are 12 digital in/outlets. The ones marked with a wave sign are PWM capable. This means that you can control LEDs or servo motors with them. The remaining six you can only switch on and off. On the bottom row, you can find 6 analogue inlets, 2 grounds, and finally the 5 V and 3.3 V power chords. If you want to make an LED blink, you can do that through a resistor with any of the outlets situated in the top row.

An LED has two legs: a shorter and a longer one. The shorter one is the ground and the longer one is the positive. Another way to distinguish them is by looking at the top of the LED, because one side is chipped, which is always the ground. Every LED has a reference voltage. It is important to respect these values because too little voltage will do nothing but a little too much (even 5V) can burn it out immediately.

You absolutely must use the correct resistors with LEDs. There is a very useful site where this can be easily calculated (http://led.linear1.org/1led.wiz). It even generates a precise circuit diagram for connecting multiple LEDs.

The opening voltage of the red LEDs are usually 2.4 V, while the other colours’ are between 3 V and 3.3 V. That is why it is important to know which type of LED you are dealing with. You can find datasheets on the Internet listing the correct values for different LEDs. The other important factor is the current: this is usually measured in milliamper (mA). The outlets of the Arduino are capable of 5 V and 40 mA, so LEDs over 40 mA should not be directly connected to them. If you want to connect an LED that has higher power consumption, you will have to use an external power supply and a transistor. You should definitely learn a bit more about it before getting started!

You need to use a 150 ohm resistor for a simple red LED and 100 ohm for the other colours. You have to use a resistor when using a switch because the Arduino's inlet is “floating” in the initial position. This means that it is not always clear whether a switch is on or off. The problem is resolved by connecting the initial position to the ground with a 10 kiloohm resistor. The following wiring diagram shows how.

It is interesting to use 10 kiloohm resistors with potentiometers. The two outer legs must to be connected to the ground and to the +5 V inlet. The middle one can be connected to any of the Analogue Reads. The wiring diagram below demonstrates this solution. You can even make a MIDI mixer based on this.

Egy egyszerű, piros LED-hez körülbelül 150 ohmos ellenállást kell használnod, a többi színhez pedig 100 ohmost.

Kapcsoló használata esetén szintén ellenállást kell használni, mivel alaphelyzetben az Arduino bemenete „lebeg”, vagyis nem minden esetben tudja eldönteni, hogy a kapcsolónk éppen milyen állásban van. Ezt úgy tudod kiküszöbölni, ha az alapállapotot egy 10 kohm-os ellenállással leviszed a földre. Ennek a bekötését a kép alapján tudod elkészíteni.

Potméterekhez szintén 10 kohmos ellenállást érdemes használni: itt a potméter két szélső lábát a földre, illetve a +5 voltos bemenetre kell raktóni, míg a középső egy tetszőleges Analogue Read-be, amelynek elkészítésében segít a lentebb látható rajz.

Ezek alapján már akár saját MIDI-keverőt is tudsz készíteni.

**Figure 13.1. Standard Arduino**
Figure 13.2. Standard Arduino back side

Figure 13.3. Arduino Bluetooth
Figure 13.4. Arduino Mini

Figure 13.5. Arduino Mega
Figure 13.6. Lilypad

Figure 13.7. Arduino Fio
Figure 13.8. Arduino Micro

Figure 13.9. Arduino Ethernet
Figure 13.10. Serial port

Figure 13.11. Arduino Game
Figure 13.12. Arduino Wifi

Figure 13.13. Arduino GSM shield
Figure 13.14. Lilypad MP3 player
Figure 13.15. Arduino Midi shield

Figure 13.16. Arduino Ethernet shield
Figure 13.17. Arduino Proto shield

Figure 13.18. Arduino Zigbee shield
Figure 13.19. Arduino breadboard shield
Figure 13.20. Arduino Motor shield
Figure 13.21. Lilypad circuit

Figure 13.22. Lilypad climate dress
Figure 13.23. Lilypad climate dress
Figure 13.24. Arduino robot
Figure 13.25. Arduino Fritz software

Figure 13.26. Arduino circuit
Figure 13.27. Single Led Calculator

LED calculator

This is the new version of the single LED series resistance calculator, good for when you have a single LED anode to 1.
The LED series/parallel wizard is suitable for those of you who need to do calculations involving more than one.

LED calculator: current limiting resistor value

5V

130 ohms, 1/2W

The wizard recommends a 1/2W or greater 130 ohm resistor. The color code for 130 ohms is brown-green-brown.

led.linear.org

Figure 13.28. Serial Led Calculator
Figure 13.29. Arduino interface
Figure 13.30. Arduino and Led
Figure 13.31. Arduino and Switch

Figure 13.32. Arduino and Potentiometer
1. Video Lesson
Chapter 14. Communication between computers and different control methods

You may remember the OSC control from the previous lesson, where we discussed the Kontrolleur. OSC is based on the UDP protocol, which is not a safe way to communicate. There is no guarantee of packet delivery over a UDP connection, so the sender does not get notified if a packet is lost or fails to be processed on the other side of the link. This enables a low protocol overhead so that the communication is faster, making OSC ideal for real-time audio/video synthesis. However, if such a trade-off is undesirable, the TCP communication should be used for setting up links over a TCP/IP (Internet) network.

There are many OSC capable controller applications on the Android Market and in the App Store. One of the best multi-platform ones is TouchOsc, but its editor only runs on Apple systems. OSC can even be used for communicating between computers because almost every programming language has an OSC decoder/encoder, TCP client/server and UDP client/server external library.

The OSC messages can be floating point or integer numbers and text messages. The latter must be indicated when sending the message. In VVVV this can be done under the Type Tags tab with the $s = \text{string}, f = \text{float}, i = \text{integer}$ settings. Failing to set the message type means risking that the receiver will not be able to decode the message. The address of the messages must also be set, which is used for separating them. For instance, if you want to send the X and Y position of the mouse, then you can write /mouse_x and /mouse_y in the address tab in the String message box.

It is also crucial to set the addresses on the receiving side as well; otherwise the program does not know how to separate the messages. The last essential settings are the UDP port and the IP address which have to be the same on both sides. If you are working with a constantly changing IP address, then you can use 255 as the last octet, in which case you are sending to the broadcast address of the subnet network. In the latter case, all the IP addresses on that network will get your messages. Alternatively, if you wish to share more information or a smaller image with Max/MSP, then you can use the Max/Jitter Matrix Nodes module, from the Contribution section of the website (http://vvvv.org/contribution/maxjitter-matrix-nodes). As the name indicates, it is capable of sending and receiving matrices from VVVV. The message will arrive as colour information at the VVVV side which can be decomposed with an RGB Split into number information. The next controller type is DMX. This is primarily used in light engineering to control dimmers and robot lights. You can control up to 512 channels with a single DMX controller. You will probably not need more than this, unless you want to control a lot of RGB LEDs or sophisticated robot lights.

Dimmers are needed to control traditional light bulbs. There are of different performances with one or more channels. They are mainly used in theatres. It is worth asking in advance how they are patched, so you will not have to deal with that on site. The robot lamps have their own dimmers, so their electricity supply is solved. However, they use a lot of channels because separate channels are needed for the servos, shutters, mirror- and colour changers and the other functions. There are robot lamps that use several hundred channels. Those are controlled with Artnet which is also a DMX-based communication protocol.

The cheapest and easiest way to make your own programmable lights is using high power LED modules and 12 V DMX PWM controllers. It is important to get familiar with the cooling characteristics of high power LEDs before buying them. It is also crucial to use the right voltage, otherwise they burn out immediately.

Until recent years, large light systems could only be controlled with expensive light desks which easily cost several thousand euros. Luckily, USB controlled DMX systems are pretty common nowadays. In certain situations highly sophisticated control mechanisms can be implemented with these and a laptop. The most important USB controller manufacturers are Ecue, Sunlight and Enttec. The latter has the best value for money because they include the software with the purchase. Enttec makes a wide range of USB products but only the pro category works correctly with VVVV. If you want to get a cheaper version of Enttec devices, it is worth having a look on eBay. There is an Italian company making Enttec compatible products (http://www.splabs.it/?portfolio=usb-dmx-converter).
You can find the necessary examples and drivers for controlling the Enttec among the attached files. The USB DMX is a kind of serial communication protocol, so you need to find out the installation port of your DMX. This can be found in the Control Panel. You can control the Enttec on the first inlet with one or more SetSlice(s) between 0 and 1.

And finally, the Artnet. Artnet is a modernized communication protocol which is based on DMX technology. Its most striking feature is that every device has its own IP address and they are controlled through conventional UTP (Ethernet) cables. Many devices can be controlled with the computer’s Ethernet port and a Switch. This is really cheap, since UTP cables and the switches are dirt-cheap consumer products.

A lot of robot lamps and robot projectors rely on Artnet, so you should decide your preferences before embarking on a project. If you have to control multiple lamps or robot lamps, then you should use Artnet, because in this case you will not have to deal with a lot of USB DMX converters. Artnet-DMX controllers are relatively expensive, although they can be found on eBay for around 100 – 150 EURs. The other advantage of Artnet compared to the old DMX standard is that you can connect multiple controllers to the switch, so there is no need to control the whole lighting system with only a single computer.

Végezetül következzen az Artnet. Az Artnet egy modernizált DMX-alapú kommunikációs forma. A legszemélyesebb sajátossága, hogy itt mindegyik eszköznek saját IP-címe van, a vezérlés pedig UTP-kábelken keresztül történik. Ennek köszönhetően a számítógép szokásos ethernet kimenetén keresztül, egy hagyományos Switch segítségével már lehetséges egyszerre több eszköz vezérlése is, ami jelentősen csökkenti az anyagköltséget, mivel az UTP-kábel és a Switch szinte filléres költség a többi megoldáshoz képest.

Ma már nagyon sok robotlámpát, illetve robotprojektor alapból Artnettel szerelnek, így mindeneképpen érdemes előre elődönteni, hogy a DMX vagy az Artnet-e a számodra legjobb megoldás. Ha nagyon sok lámpát, vagy sok csatornát használó robotlámpákat kell használni, akkor mindeneképpen Artnet a legmegfelelőbb, mert így nem kell egyszerre több USB DMX átalakítót használnod. Egyedül az Artnet-DMX konverter kerül kicsit több az eddigiekhez képest, de eBay-en találhatsz már akár 30-40000 forintért is 6-kimenetes változatokat. Az Artnet egy másik fontos előnye a régi DMX-hez képest, hogy a Switch-be akár több vezérlőt is köthetsz, így nem szükséges egyetlen gépről vezérelned a teljes világítási rendszert.

**Figure 14.1. Touch Osc Ipad Interface**
Figure 14.2. Touch Osc in Iphone

Figure 14.3. OSC Receiver

Figure 14.4. OSC Sender
Communication between computers and different control methods

Figure 14.5. UDP Sender

Figure 14.6. UDP Receiver

Figure 14.7. Sunlite USB DMX
Communication between computers and different control methods

Figure 14.8. Sunlite USB DMX
Figure 14.9. Enttec USB DMX
Communication between computers and different control methods

Figure 14.10. Enttec USB DMX clone
Communication between computers
and different control methods

Figure 14.11. Enttec USB DMX clone

Figure 14.12. Dimmer

Figure 14.13. Compulite Controller
Figure 14.14. High-End Hog Controller
Figure 14.15. DMX PWM converter

Figure 14.16. LED Lamp
Communication between computers and different control methods

Figure 14.17. DL3 Projector

Figure 14.18. Enttec USB DMX receiver
Communication between computers and different control methods

Figure 14.19. Enttec USB DMX interface
Figure 14.20. Cheap Artnet input
Figure 14.21. Cheap Artnet DMX output
Communication between computers and different control methods

Figure 14.22. Enttec Artnet Dmx converter

Figure 14.23. Artnet Sender
Communication between computers and different control methods

Figure 14.24. Artnet Receiver

Figure 14.25. Little Cat Midi software

1. Video Lesson
Chapter 15. User interface design

The best example of user interface design is the IRIS project which was published by Milo on the VVVV Contribution site (http://iris.nsynk.de/). This is a brand-new multitouch control surface for VVVV which can be used in a DirectX renderer. It does not need a node editor and it is strikingly similar to Touch Designer. In addition to the control surface, Boygrouping is also implemented in this system. With this you can choose whether you want to project from one computer or you want to synchronize multiple computers (the latter is essential in projector systems).

There are two user interface options in VVVV: Gui and Multitouch Gui. You can also make your own with Quad, Tagpoint, Pickpoint and other modules.

The first Gui control surface was published by Vux. The two control surfaces are very similar to each other, but the newer version has less features. You get a Transform on both nodes’ outlet, which positions the Quad. You can set their position on their inlets. These are the number of rows and columns, button size on the X and Y grid and finally, the values of the buttons. The last option is for recalling a saved setting.

You can choose from the following button types: ToggleButton, Slider, SliderXY and RotaryButton. There is also a RadioButton in the Gui version which is equivalent to a Bang. You can also make one with the help of Togedge and Framedelay. You just have to reset Togglebutton to zero after every mouse click.

You can also make a 3D user interface, if you want to.

There are two variations in VVVV: one of them is Button (3D Quad) which you can use with Quads positioned in space. The Quad's Transform position has to be connected to the first inlet, the mouse position to the second inlet and the Camera View Projection to the last one. Thanks to those, you will get where the mouse is above the Quads on the first outlet, that is, the first one on the second outlet (in case you have multiple Quads behind each other) and the mouse position on the last one.

The other spatial solution is the Button (3D Mesh) module. You can load 3D objects into this, which can even result in a collada file consisting of multiple Meshes. You can choose among its elements with the mouse position. In contrast with the Button, a Mesh has to be connected to this module. It is important to know that this module sends only numbers which can be leveraged to select different parts. It is essential to use a display module with this (PhongDirectional or a simple Constant).

You can use your user interface with mouse, Wacom tablet, multitouch monitor or a multitouch interface that you have built on your own. It is worth having a look at the second-hand market if you want to buy a multitouch monitor, especially Elo branded devices. This manufacturer makes the best shock proof monitors. Lately, other vendors started to make multitouch monitors as well, but even the prestigious HP cannot make a really useful one.

If you only want to work with selecting or MouseOver, it is probably the best to use TagPoints or PickPoints. You can select the points with the left mouse button when using TagPoints and invert the selection with the right one. PickPoints only works if you click when your mouse is over an object - similarly to the Button 3D Mesh.

To proceed, let's have a look at writing to a file, which is essential for recalling the latest state of your buttons!

You have two options. One of them is to save the data into a RAW file. This is the quickest solution but it cannot be edited in a word processor because it is stored as raw data.

The other method is saving your material as a text file. This can be edited independently from VVVV, but it's relatively slow so it's not the best solution when writing multiple files. Regardless of which method you use, if you want to save numbers, then you always have to convert them to RAW with AsRaw orAsString beforehand.

You can use the Write modules for writing to file. You always have to connect the data you want to save to the first inlet, while the file's name and location goes to the second one – the third one is conversion type (used for text) and the fourth one is always the writing action itself.

A file can be loaded automatically or manually. When loading manually you simply have to connect a button to the corresponding Read module and specify the path of the file. You can connect its outlet to the ValueInput
inlet of your button after converting it to number with AsValue. You have to connect the button that is used for loading to the SetValue PIN of the buttons with Framedelay. (Framedelay is needed here because the reading takes time and this change has to be delayed.)

For automatic loading, you have to connect an OR (spectral) module to the Hit outlet of the buttons, and a TogEdge to this one. The outlet of UpEdge has to be connected to a Write, while the outlet of DownEdge to a FrameDelay that is connected with another one. The first FrameDelay has to be connected to the Reader and the second one to the SetValue inlet of the buttons (because of the delay mentioned above). Now everything is saved after every click, and the values of the buttons also load.

If you want automatic loading on start-up, then you have to connect an OR module to the Read inlet of a Reader module by connecting a FrameDelay to the first inlet of the OR and an OnOpen module to the second one. Sometimes your program loads for a long time because of its size, so the OnOpen happens too quickly. In this case, it is worth delaying it with a Delay module.

**Figure 15.1. Toggle Button Gui**
Figure 15.2. Toggle Button Gui Multitouch

Figure 15.3. XY Slider Gui
Figure 15.4. XY Slider Gui Multitouch
Figure 15.5. Slider Gui
Figure 15.6. Slider Gui Multitouch
Figure 15.7. Rotary Slider GUI
Figure 15.8. Rotary Slider Gui Multitouch
Figure 15.9. Radio Button Gui
Figure 15.10. Button 3D Quad
Figure 15.11. Button 3D Mesh

Figure 15.12. PicPoints
Figure 15.13. TagPoints

Figure 15.14. Write Raw
Figure 15.15. Write String

Figure 15.16. Write and read automatically
1. Video Lesson

2. Node list

Source: http://vvvv.org/documentation/node-reference

- (Value) Subtracts one value from another

* (Value Spectral) Returns the product of the incoming spread
* (Transform) Returns a \times b; concatenates matrices

+ (Value) Returns the sum of the inputs

+ (Value Spectral) Returns the sum of the incoming spread

- (Value) Subtracts one value from another
/ (Complex) Returns the Division of the inputs

= (Value) Returns 1 when a is equal to b within the given tolerance

**ADSR (Animation)** Classic synthesizer envelope generator with attack time, decay time, sustain level and release time
**AND (Boolean)** Slicewise operation: returns true (1) if all inputs are true (>0.5)

![Diagram of AND (Boolean)](image1)

**AND (Boolean Spectral)** ANDs the incoming spread. only true (1) if all slices are >0.5

![Diagram of AND (Boolean Spectral)](image2)

**ApplyTransform (Transform)** Applies the given transform to the points

![Diagram of ApplyTransform (Transform)](image3)

**ApplyTransform (Transform Vector)** Applies the given transform to the points

![Diagram of ApplyTransform (Transform Vector)](image4)

**Arduino (Devices StandardFirmata 2.x)**
AsString (Raw) Decodes a sequence of bytes into a string using the specified encoding

AsString (Value) Converts a value with a given valuesubtype to a string

AsValue (Raw) Interprets a sequence of bytes as a value
**AsValue (String)** Converts a string to a numerical value. In case the string is no valid number, the default will be used. The node will strip all non-numeric characters from the beginning and the end of the string.

**AudioAnalysis (DShow9)** Module to analyze the audio input with FFT, beat detector and beat tracker.

**AudioIn (DShow9)** Gets an audio signal from the sound card.
AvoidNIL (String) Replaces an empty string spread with a default string

B-Spline (Value) Retrieves a point on a B-spline curve

Blend (EX9.Texture Mixer) Blend two textures
Blur (EX9.Texture Filter) Fast texture blur

Bounds (Spectral) Returns minimum, maximum and the average of the incoming spread

Buffer (Spreads) Inserts the input at the given index and returns the whole Buffer as spread
**Button (3d Mesh)** Checks if the cursor is over a mesh (in 3d space).

**Camera (Transform Softimage)** Move your camera with keyboard and mouse like in the 3d modeller "Softimage"
Change (String) Outputs 1 when the input string has changed in this frame. Outputs 0 if the input was equal to the one in the last frame.

CircularSpread (Spreads) Creates a spread of values forming an oval.

Cons (EX9.Texture)

Cons (Spreads) Concatenates all input spreads to one output spread

Contour (FreeFrame DShow9)
Count (String) Returns the number of slices in the given spread of strings

Count (Value) Returns the number of slices in the given spread of values

Cross (2d) Generates all combination pairs of the two inputs at the output
**Cross (3d)** Generates all combination triplets of the inputs at the outputs

**CubeMap (EX9.Effect)**

**Cursor (DX9)** Draws a cross-shaped cursor into the ex9 renderer. It also can draw a selection rectangle
**Damper (Animation)** Applies a force to the input value which prevents the output to reach the new value for a given time.

**Decay (Animation)** Follows the input value with a constant speed.

**Delay (Animation)** Delays incoming values for a certain time.
DeNiro (Animation) Acceleration based filter which allows super-smooth movements

Dir (File Advanced)

DMX (Network Artnet Receiver) Receives a DMX universe from a remote device
**DMX (Network ArtNet Sender)** Sends a DMX universe to a remote device

**DrawFixed (EX9.Effect)** Fixed function pipeline rendering of a mesh

**DX9Texture (EX9.Texture)** Allows to use an EX9 Renderer to render into a texture
Expr (Value Advanced) Auto compiled version of Expr (Value)

FFT (DShow9) Gets a spread with a FFT analysis

FileStream (DShow9) Reads a DirectShow stream from disk
FileStream (EX9.Texture VLC) Fully spreadeble video/image to texture player based on LibVlc

FileStream2 (DShow9) Reads a DirectShow stream from disk
FileTexture (EX9.Texture) Allows image files to be used as textures in the DX window

Fill (EX9.RenderState) Set fill mode to solid, wireframe or point

Flash (EX9) Renders the Surface of a SWF File to a Direct3D Texture
FlatDirectional (EX9.Effect) Basic vertex based flat lightning with directional light

Frac (Value) Splits a value into an integer part and a fractional part. The integer is always the next smaller number. The sum of the outputs equals the input.

FrameDelay (Animation) Delays the input value one calculation frame.
FrameDelay (EX9.Texture) Framedelay for textures

FrameDifference (Animation) Calculates the difference between the current input value and the input value from the last frame

GetSlice (String) Gets all slices specified in the index input from the input spread

GouraudDirectional (EX9.Effect) Basic vertex based lightning with directional light
Grid (EX9.Geometry)

Group (EX9) Group layers to be rendered one after another

HSL (Color Join) Creates a colour out of the given hue, saturation, lightness and alpha values
**HSL (Color Split)** Splits a colour into hue, saturation, lightness and alpha values

**HTTP (Network Get)** Gets data from given urls on the web

**HTTP (Network Post)** Posts data to given urls on the web
I (Spreads)

InputMorph (Value) Linearly interpolates between multiple value inputs

Inspektor (VVVV) Herr Inspektor, wer glaums denn, dass ich bin!?

Keyboard (System Global)
Kinect (Devices Microsoft)

Kinect (Devices OpenNI) Provides access to a Kinect through the OpenNI API

Light (EX9 Direction) Helper to adjust a directional light within your 3d scene
Line (EX9)

LinearFilter (Animation) Ramps from one value to another in a given time in a linear fashion

Map (Value) Maps the value in the given range to a proportional value in the given output range
**Max (Value)** Returns the maximum of the inputs (use this as logical or)

**MediaPlayer (EX9.Texture Source)** Simple video player

**Mesh (EX9.Geometry Assimp)**
Mesh (EX9.Geometry Collada) Returns a D3D9 mesh consisting of all meshes specified by index

Mesh (EX9.Geometry Join) Creates a mesh from a VertexBuffer and a spread of indices

MidiNote (Devices) Handles midi inputs
**Min (Value)** Returns the minimum of the inputs

**Mouse (System Window)**

**MTSlider (2d GUI)** A spread of slider groups
MTSlider (GUI Multitouch) A spread of sliders for multi touch

NIL (Spreads) Outputs an empty value spread

OnOpen (VVVV) Bangs once, when the patch is loaded
**OR (Boolean)** Slicewise operation: returns true (1) if one of the inputs is true (>0.5)

**OSCDecoder (Network)** Decodes an OSC packet into its arguments

**OSCEncoder (Network)** Generates an OSC packet from a given address and arguments

**Oscillator (Animation)** Pendulum (swinging) style oscillator
**PickPoints (3D Vector)** Pick points of a point cloud via mouse within the ex9 renderer. The module operates in projection space, so you have to transform the point cloud into that space first.

**Pipet (EX9.Texture Simple)** Get the color of the specified pixel.

Points2Vector (2d) Returns the center point and the angle of the two given points

Quad (DX9) Renders a rectangle in the dx window

Quad (SVG) Renders a rectangle into a Renderer (SVG)
Queue (Spreads) Inserts the input at index 0 and drops the oldest slice in a FIFO fashion

R (Value) Receive numeric values from a S node in the patch

RandomSpread (Spreads) Creates a spread of values randomly filling a range

Reader (Raw) Returns the contents of a file or a spread of files as a spread of raw data
Reader (String) Returns the content of a file or a spread of files as a spread of strings

Renderer (EX9) DirectX9 Render Window

Renderer (Flash) Flash Renderer Window
Renderer (GDI) Graphic Device Interface Window

Renderer (HTML String) Renders a HTML string
Renderer (HTML URL) Renders URLs as HTML

Renderer (TTY) Teletypewriter window
**Resample (Spreads)** Resample the input spread to a new size by applying various techniques

**Reverse (Spreads)** Reverses the order of the slices in the spread

**S (Value)** Send numeric values to a R node in the patch
**S+H (Animation) Sample and Hold**

**Scene (Assimp) Assimp loader**

**Select (Value) Select the slices which form the new spread**

**Separate (String) Splits a given string at the selected separator into slices**
**SetSlice (Spreads)** Replace individual slices of the spread with the given input.

**Stallone (Spreads)** Stallone is a very flexible node, which allows you to merge slices from any number of inputs and distribute them to any number of slices on any number of outputs.

**Switch (Value Input)**
Switch (Value Output)

TagPoints (3D Vector) Tag points of a point cloud via mouse within the ex9 renderer

Text (EX9) Draws flat Text
Text (EX9.Texture) Renders text on textures

TogEdge (Animation) Detects upwards and downwards flanks in the input
Transform (Transform 2d) Transforms the incoming transformation in 2d

Transform (Transform 3d) Transforms the incoming transformation in 3d

UDP (Network Client) is sending UDP packets
UDP (Network Server) is waiting for UDP packets

Unzip (Value) Unzips a spread into multiple spreads

Vector (2d Join) Builds a Vector out of its ingredients (the catesian coordinates)

Vector (2d Split) Returns cartesian coordinates of a Vector
VertexBuffer (EX9.Geometry Join) Creates a vertex buffer

VertexBuffer (EX9.Geometry Split) Deconstructs a vertex buffer
**VideoIn (DShow9)** Gets a video signal from a directshow compatible device

**VideoTexture (EX9.Texture VMR9)** Renders video onto a texture
Writer (Raw) Writes raw data into a file

Writer (String) Writes a string to a text file

Zip (Value) Zips spreads together